

Stefan Sokołowski

MATEMATYKA DYSKRETNA

wykład 10

**Inst. Informatyki Stosowanej, ANS
Elbląg, 2024/2025**

Drzewa — kod Prüfera

Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .

Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .

Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x

Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y

Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$

Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.

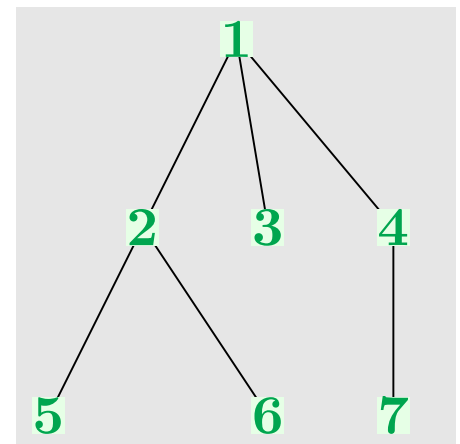
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD:

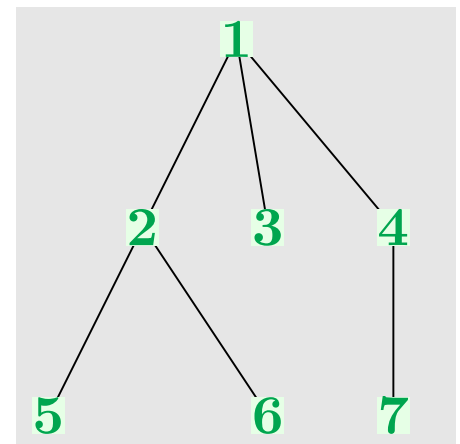
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD:

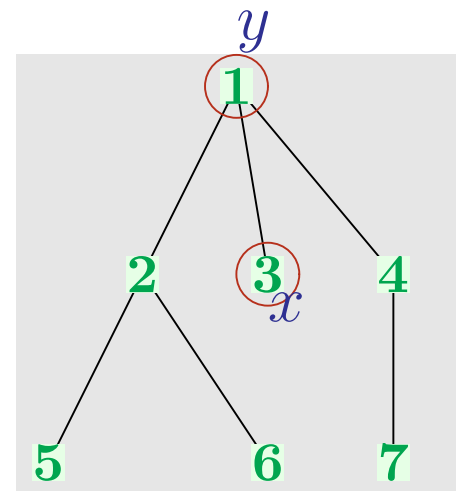
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD:

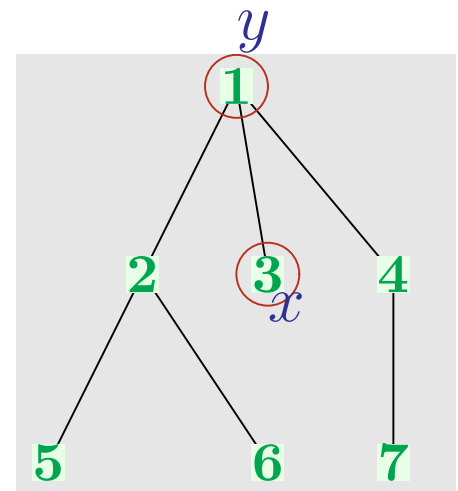
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. **do kodu dopisać wierzchołek y** ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD:

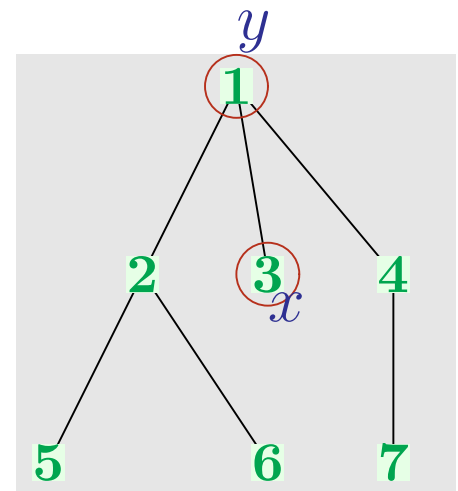
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1

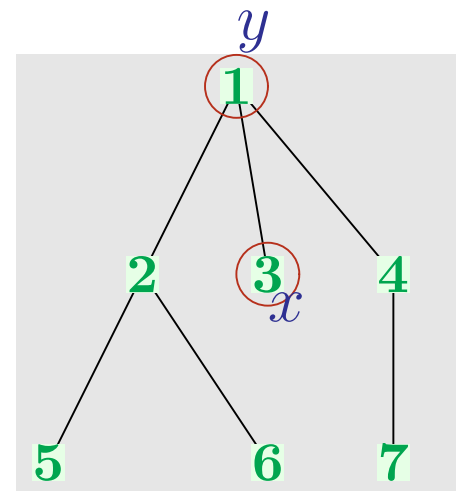
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. **usunąć z drzewa liść x i krawędź $\{x, y\}$;**
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1

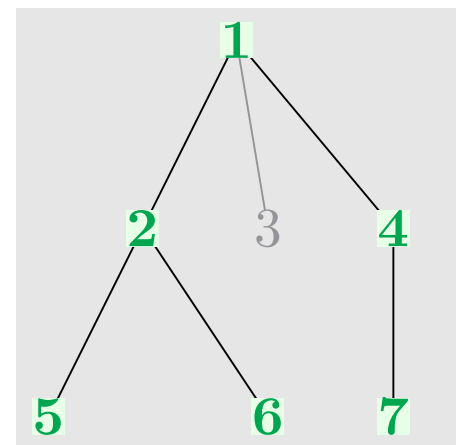
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. **usunąć z drzewa liść x i krawędź $\{x, y\}$;**
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1

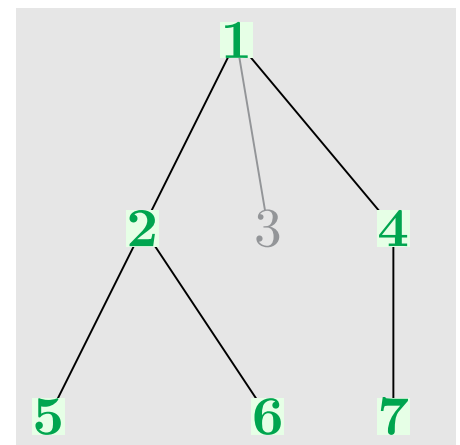
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1

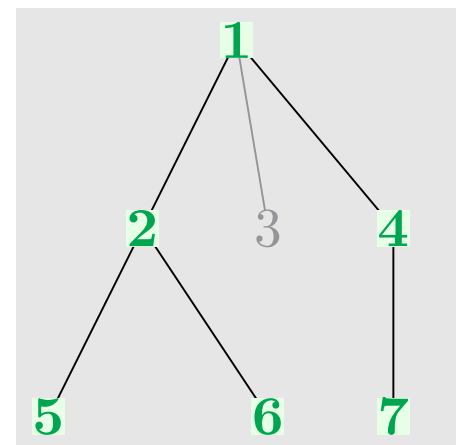
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1

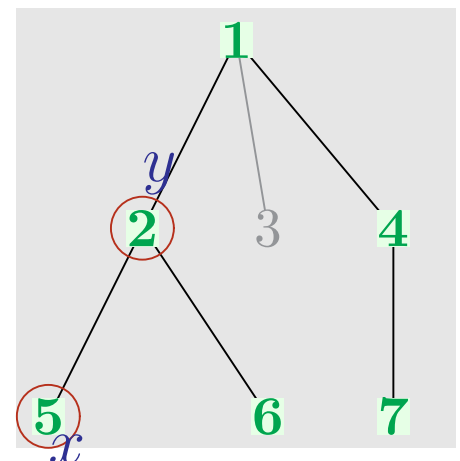
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1

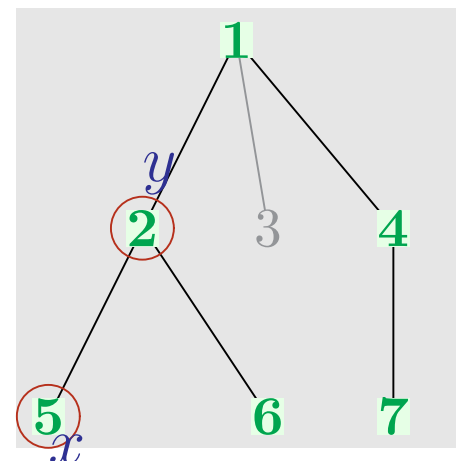
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1

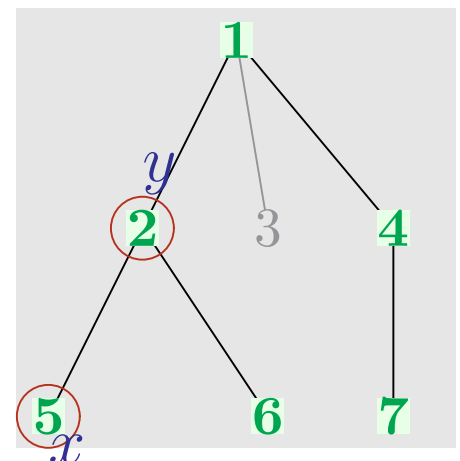
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. **do kodu dopisać wierzchołek y** ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2

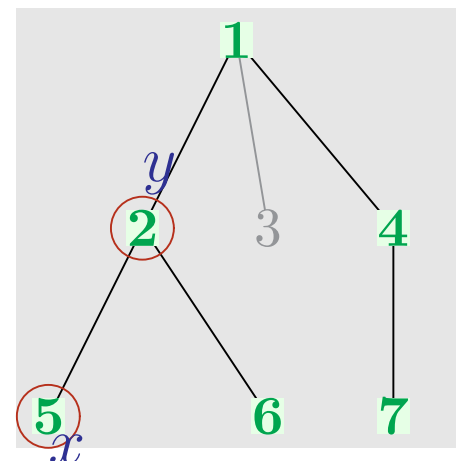
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. **usunąć z drzewa liść x i krawędź $\{x, y\}$;**
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2

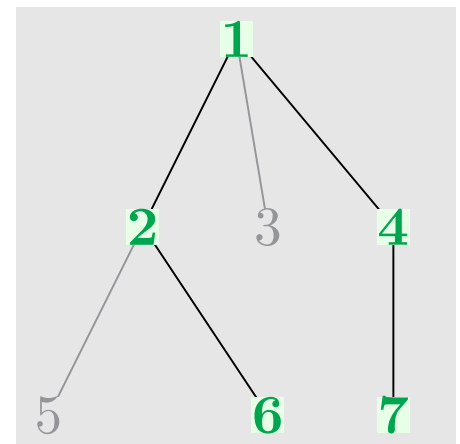
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. **usunąć z drzewa liść x i krawędź $\{x, y\}$** ;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2

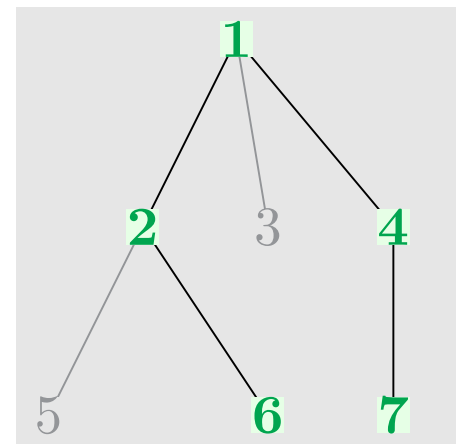
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2

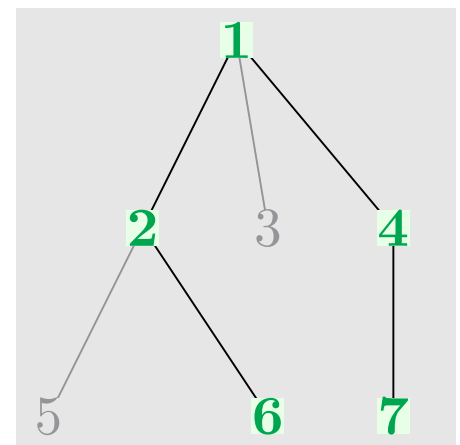
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2

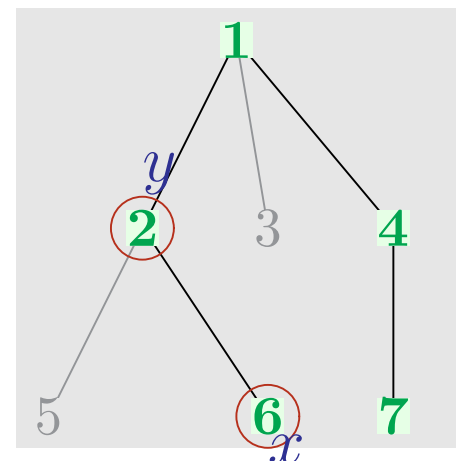
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2

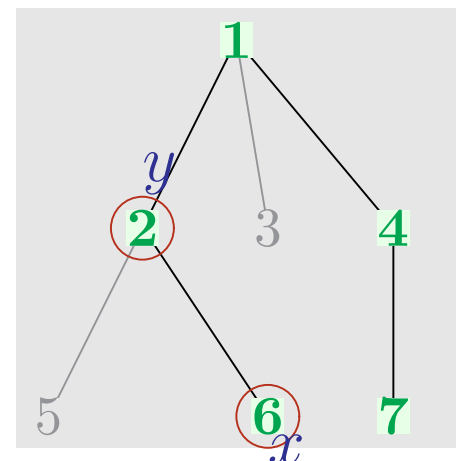
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. **do kodu dopisać wierzchołek y** ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2

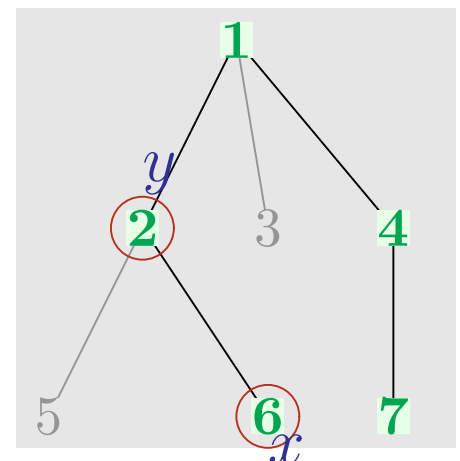
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. **do kodu dopisać wierzchołek y** ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2

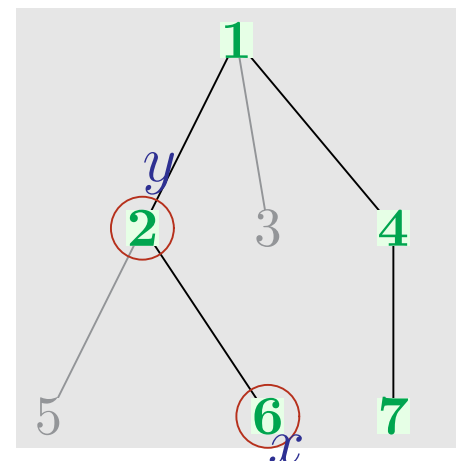
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. **usunąć z drzewa liść x i krawędź $\{x, y\}$;**
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2

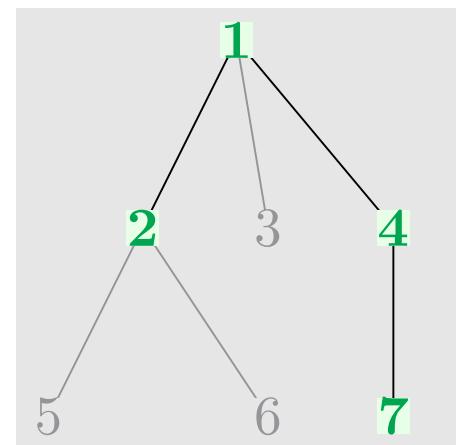
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. **usunąć z drzewa liść x i krawędź $\{x, y\}$;**
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2

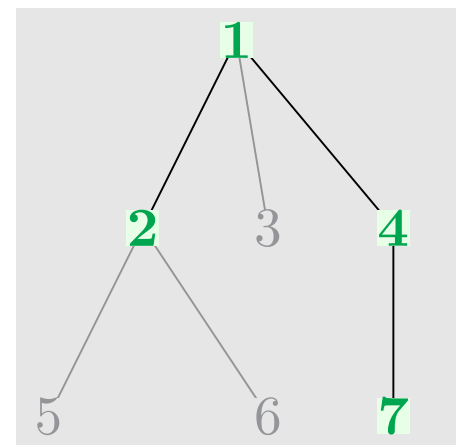
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2

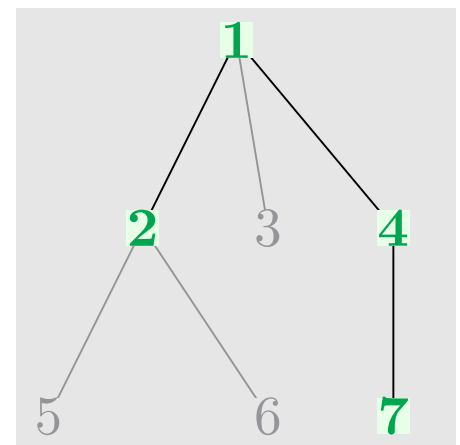
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2

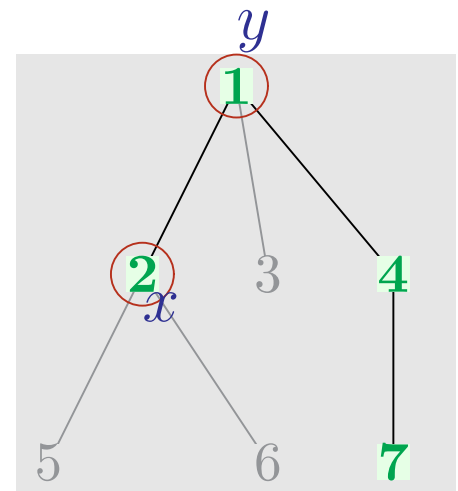
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2

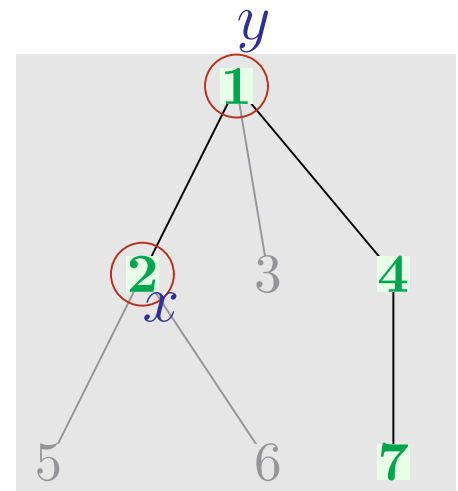
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. **do kodu dopisać wierzchołek y** ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2

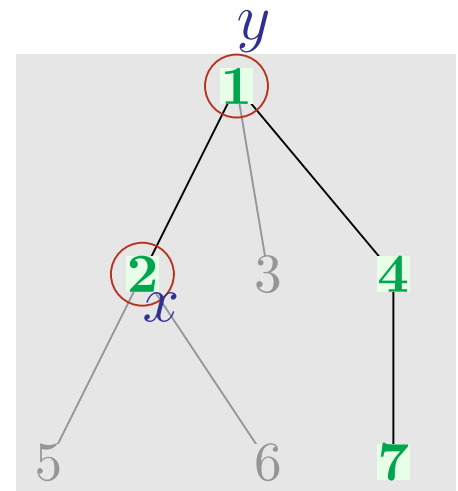
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. **do kodu dopisać wierzchołek y** ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2,1

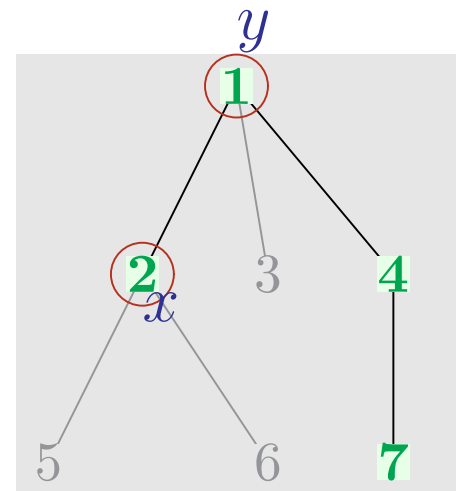
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. **usunąć z drzewa liść x i krawędź $\{x, y\}$;**
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2,1

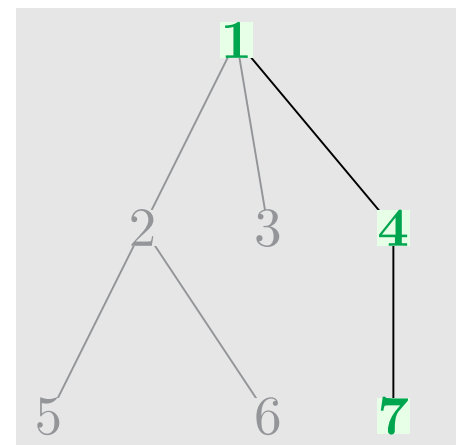
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. **usunąć z drzewa liść x i krawędź $\{x, y\}$;**
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2,1

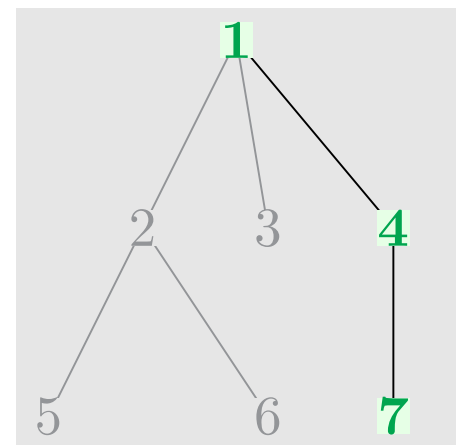
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2,1

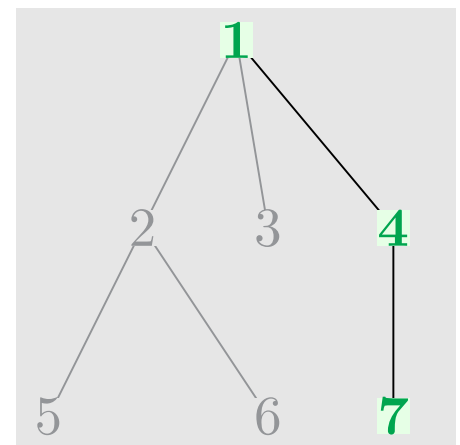
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2,1

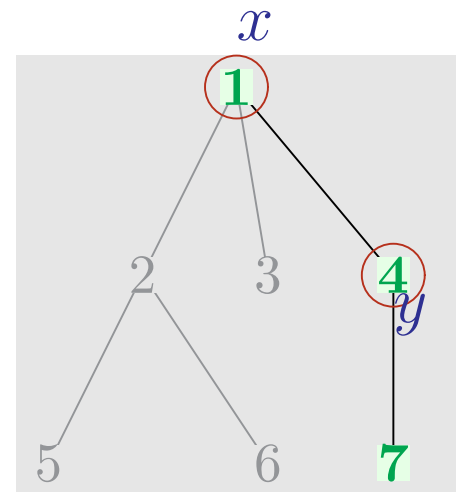
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2,1

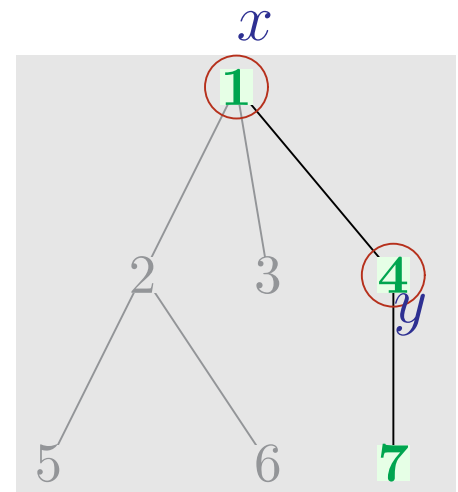
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. **do kodu dopisać wierzchołek y** ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2,1

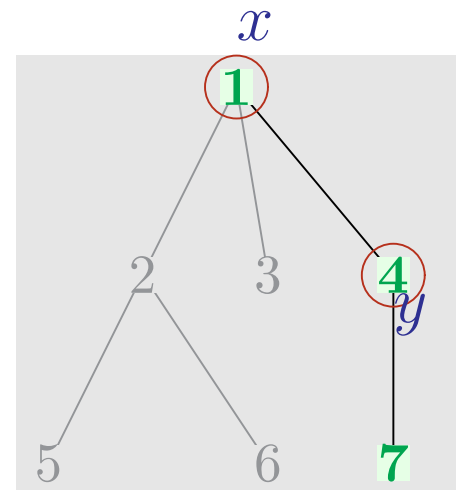
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. **do kodu dopisać wierzchołek y** ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2,1,4

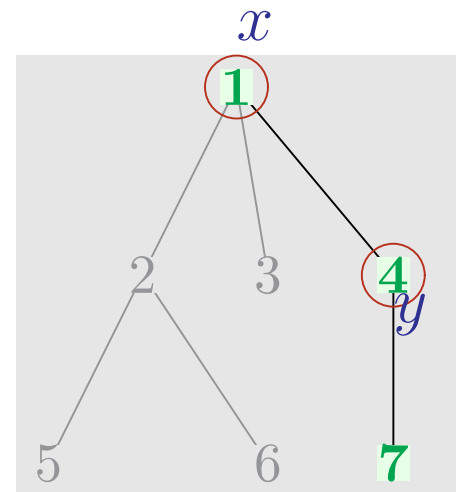
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. **usunąć z drzewa liść x i krawędź $\{x, y\}$;**
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2,1,4

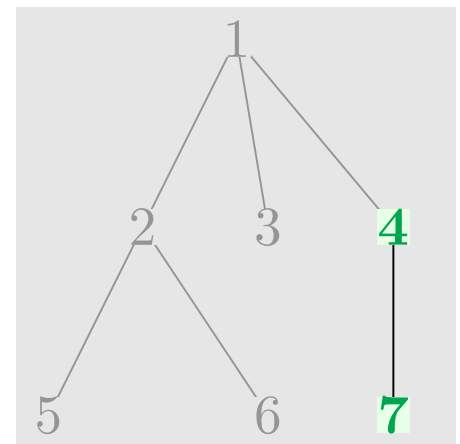
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. **usunąć z drzewa liść x i krawędź $\{x, y\}$;**
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2,1,4

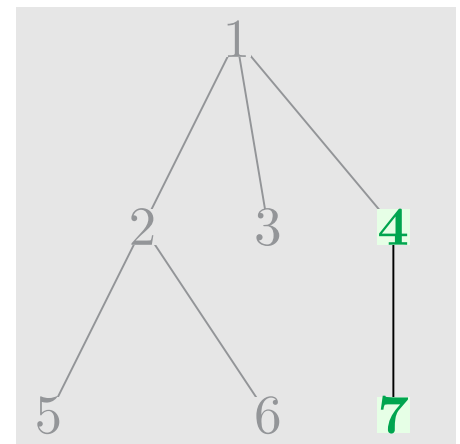
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2,1,4

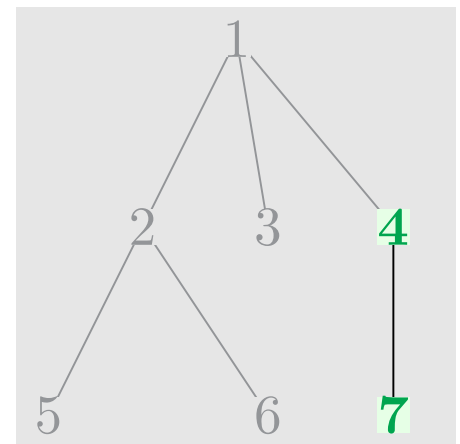
Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Kodowanie:

1. znaleźć w drzewie liść x o najmniejszym numerze i wierzchołek y połączony krawędzią z x ;
2. do kodu dopisać wierzchołek y ;
3. usunąć z drzewa liść x i krawędź $\{x, y\}$;
4. jeśli w drzewie pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1.



KOD: 1,2,2,1,4

Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

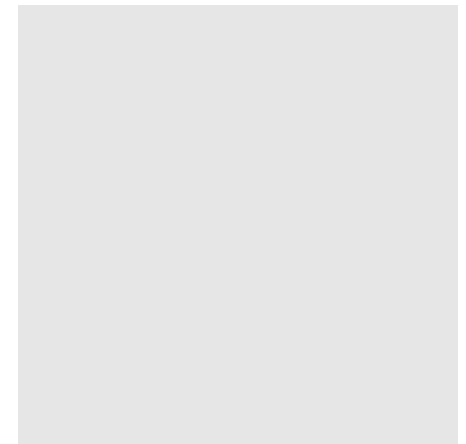
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 1,2,2,1,4

Lista: 1,2,3,4,5,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

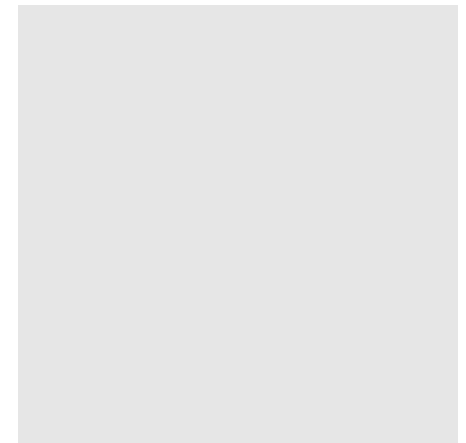
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. **znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;**
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 1,2,2,1,4

Lista: 1,2,3,4,5,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

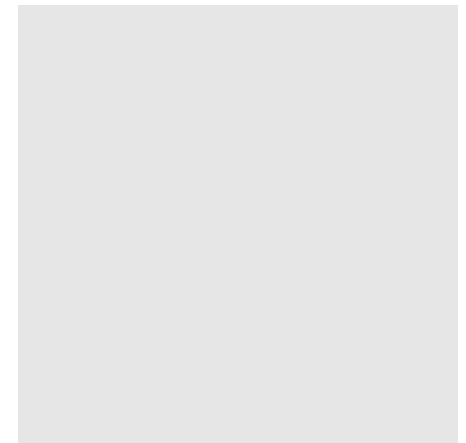
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. **znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;**
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 1,2,2,1,4

Lista: 1,2,③,4,5,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

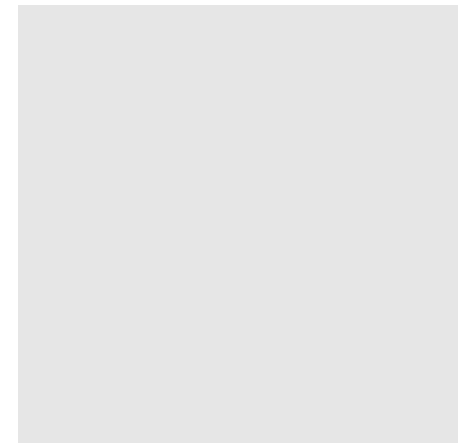
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 1,2,2,1,4

Lista: 1,2,3,4,5,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

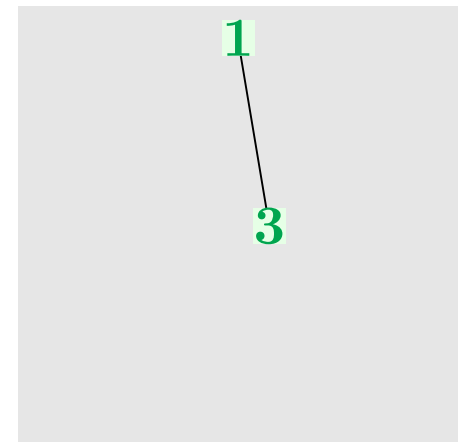
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: ①,2,2,1,4

Lista: 1,2,③,4,5,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

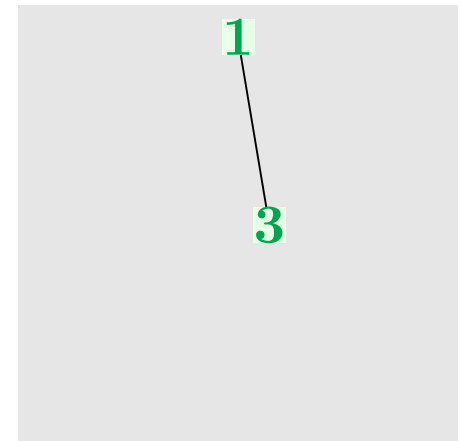
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. **usunąć k z listy oraz ℓ z kodu;**
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: ①,2,2,1,4

Lista: 1,2,③,4,5,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

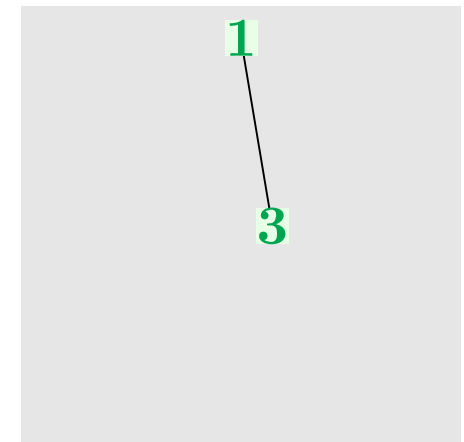
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. **usunąć k z listy oraz ℓ z kodu;**
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 2,2,1,4

Lista: 1,2,4,5,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

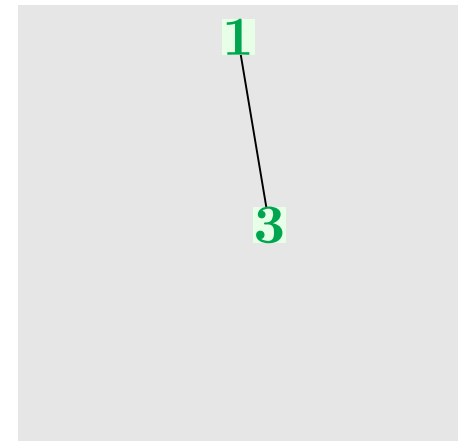
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. **jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;**
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 2,2,1,4

Lista: 1,2,4,5,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

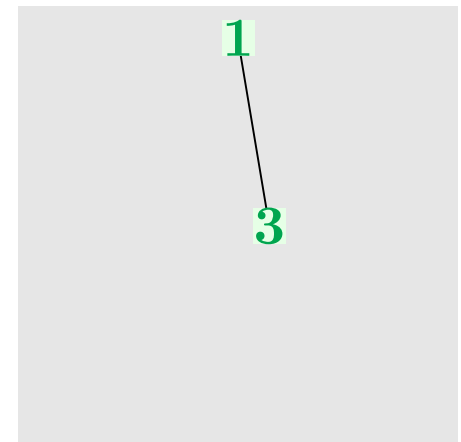
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 2,2,1,4

Lista: 1,2,4,5,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

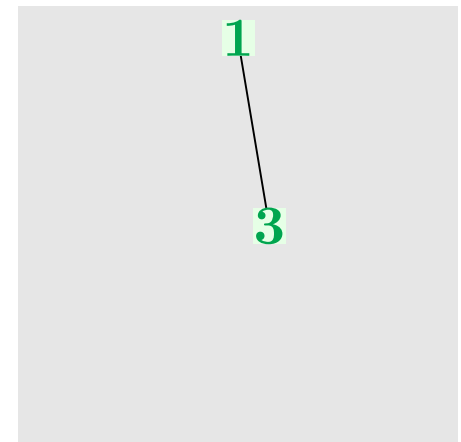
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. **znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;**
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 2,2,1,4

Lista: 1,2,4,5,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

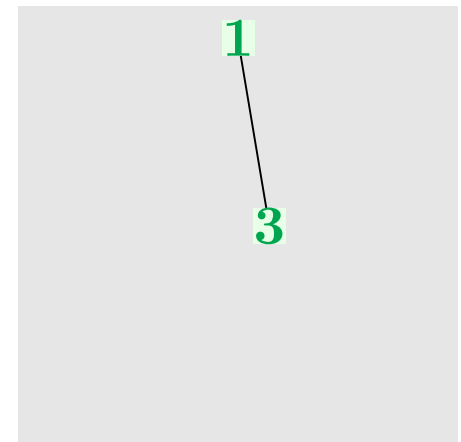
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 2,2,1,4

Lista: 1,2,4,5,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

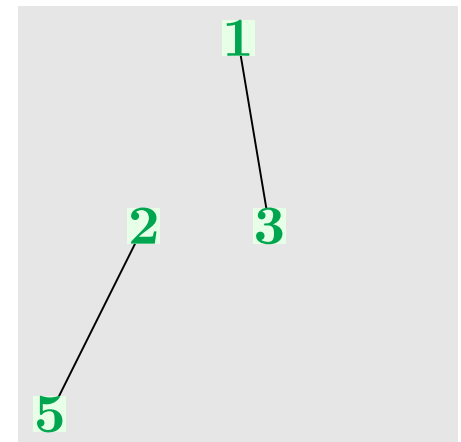
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: ②,2,1,4

Lista: 1,2,4,⑤,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

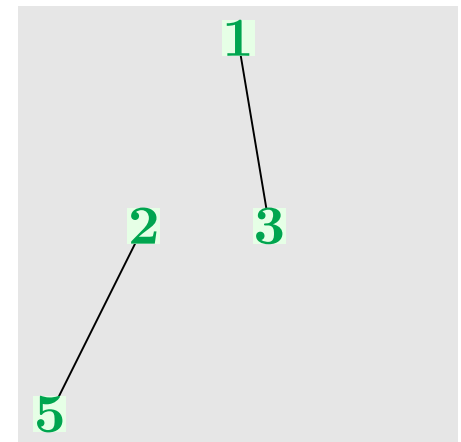
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. **usunąć k z listy oraz ℓ z kodu;**
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: ②,2,1,4

Lista: 1,2,4,⑤,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

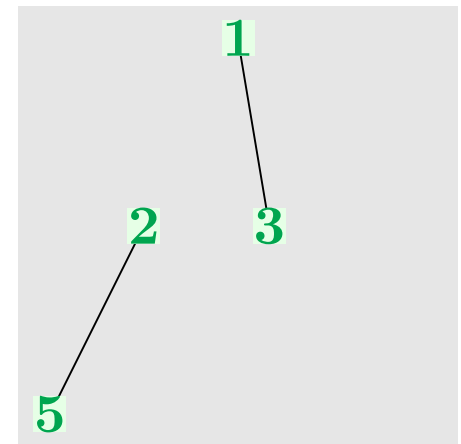
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. **usunąć k z listy oraz ℓ z kodu;**
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 2,1,4

Lista: 1,2,4,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

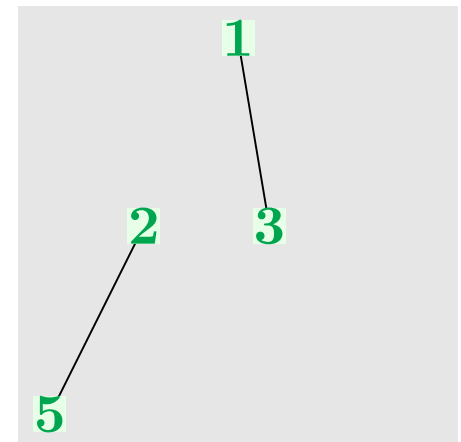
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. **jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;**
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 2,1,4

Lista: 1,2,4,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

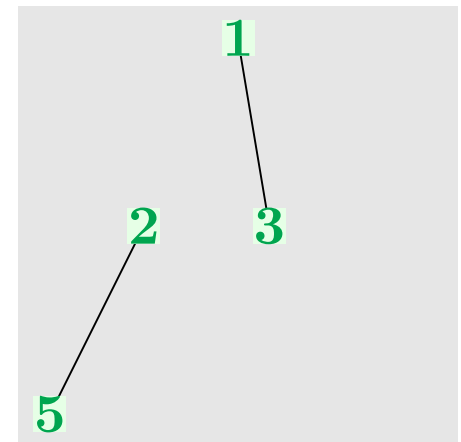
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. **znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;**
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 2,1,4

Lista: 1,2,4,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

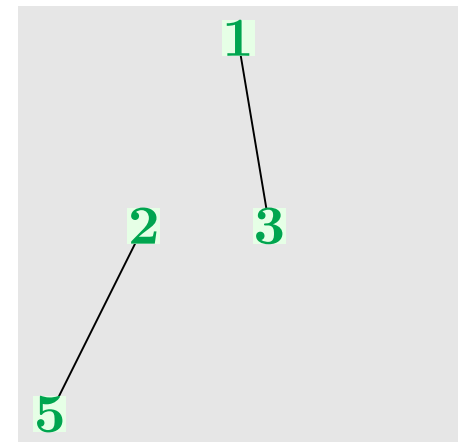
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. **znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;**
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 2,1,4

Lista: 1,2,4,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

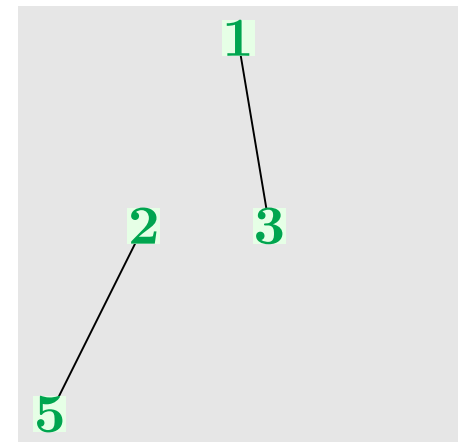
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 2,1,4

Lista: 1,2,4,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

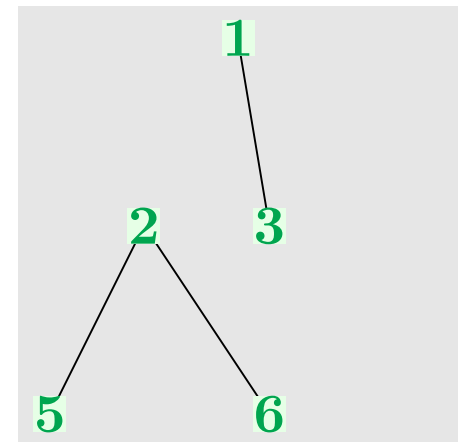
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 2,1,4

Lista: 1,2,4,6,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

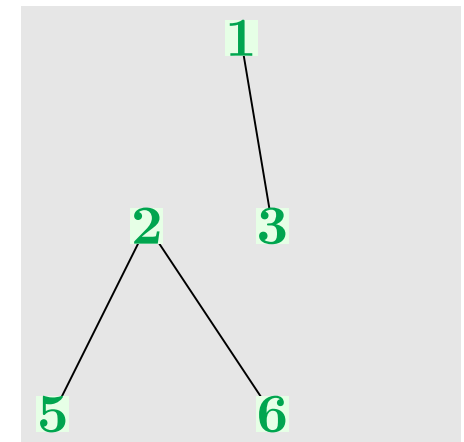
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. **usunąć k z listy oraz ℓ z kodu;**
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: ②,1,4

Lista: 1,2,4,⑥,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

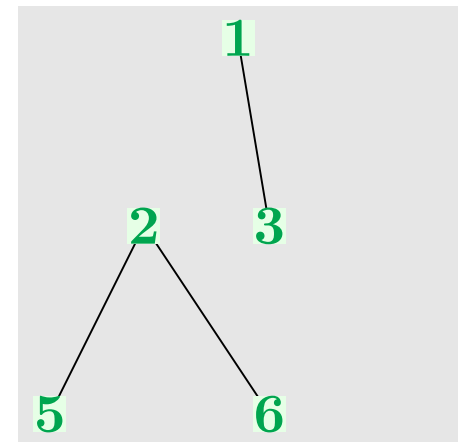
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. **usunąć k z listy oraz ℓ z kodu;**
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 1,4

Lista: 1,2,4,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

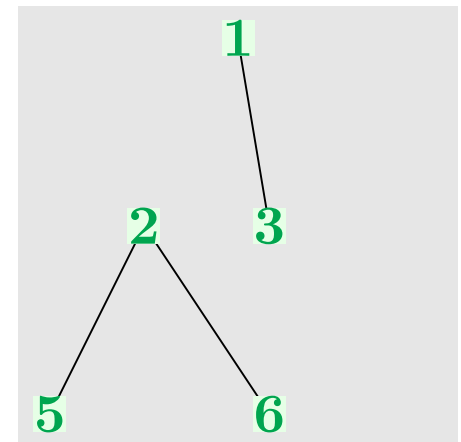
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. **jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;**
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 1,4

Lista: 1,2,4,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

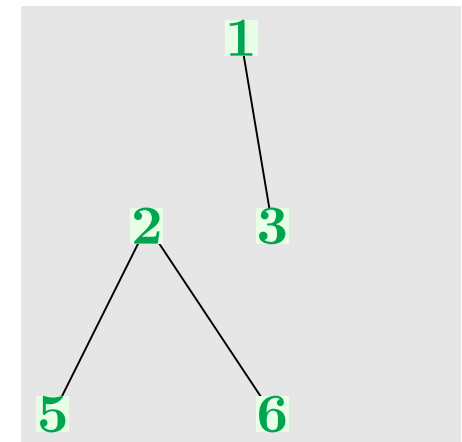
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. **znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;**
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 1,4

Lista: 1,2,4,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

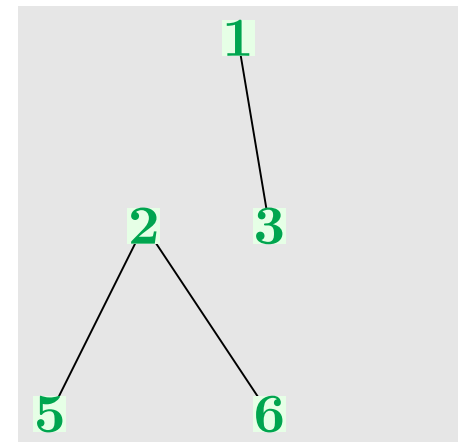
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. **znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;**
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 1,4

Lista: 1, 2, 4, 7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

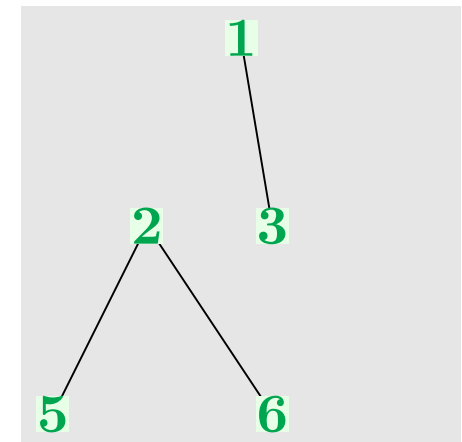
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 1,4

Lista: 1,2,4,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

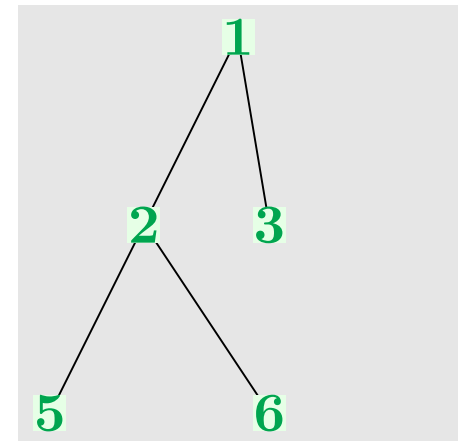
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 1,4

Lista: 1,2,4,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

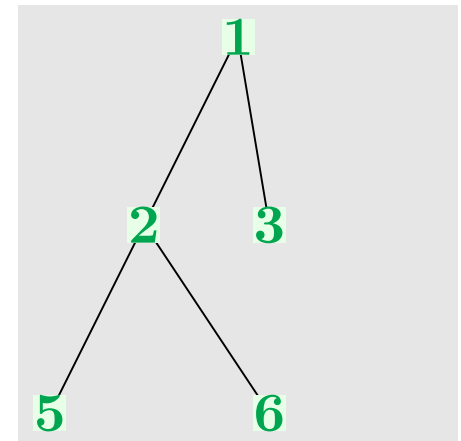
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. **usunąć k z listy oraz ℓ z kodu;**
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 1,4

Lista: 1,2,4,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

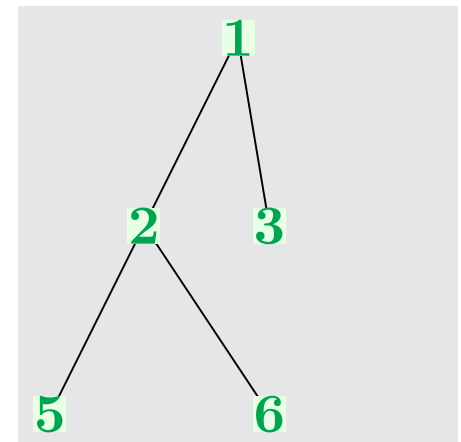
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. **usunąć k z listy oraz ℓ z kodu;**
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 4

Lista: 1,4,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

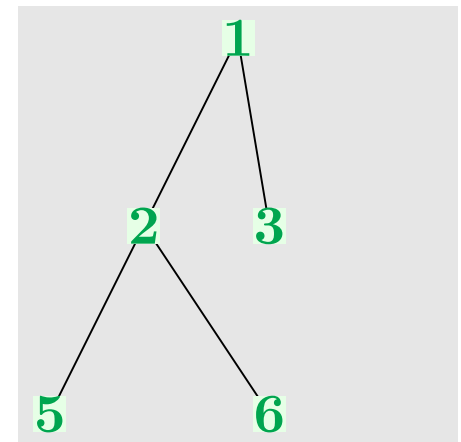
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. **jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;**
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 4

Lista: 1,4,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

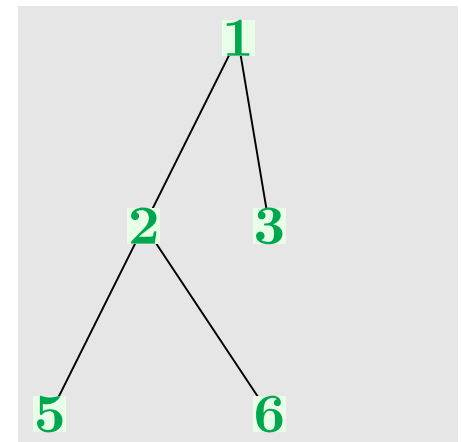
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. **znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;**
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 4

Lista: 1,4,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

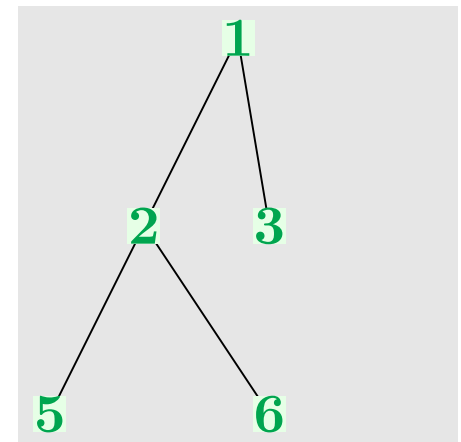
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 4

Lista: ①, 4, 7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

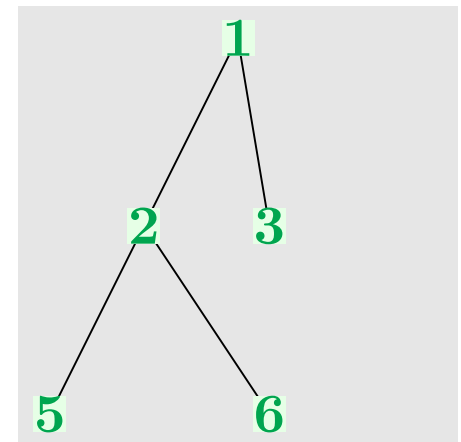
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: 4

Lista: ①, 4, 7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

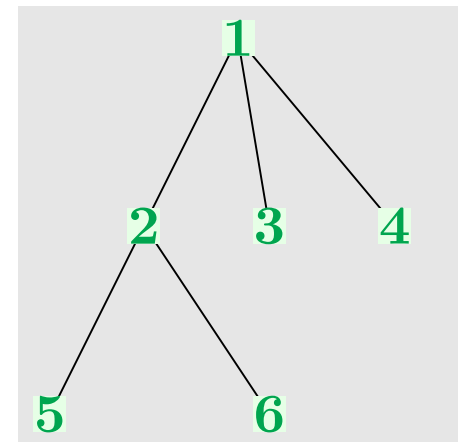
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: ④

Lista: ①, 4, 7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

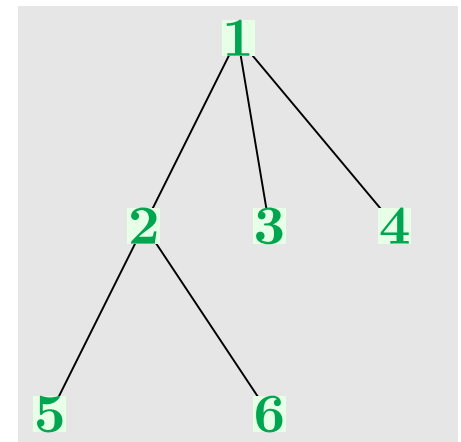
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. **usunąć k z listy oraz ℓ z kodu;**
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD: ④

Lista: ①, 4, 7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

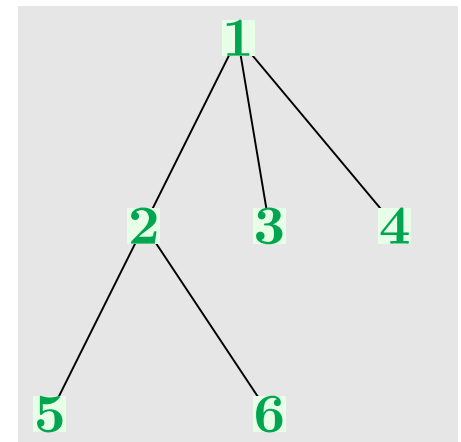
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. **usunąć k z listy oraz ℓ z kodu;**
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD:

Lista: 4,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

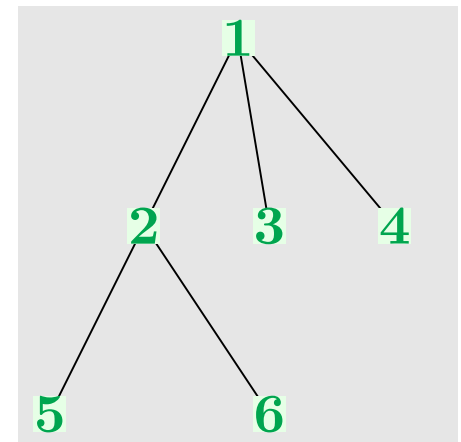
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. **jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;**
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD:

Lista: 4,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

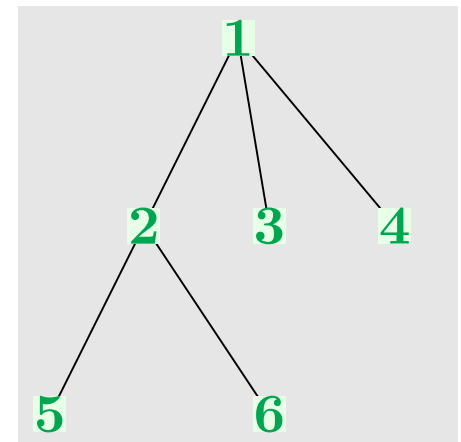
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. **połączyć krawędzią dwa pozostałe na liście wierzchołki.**

KOD:

Lista: 4,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

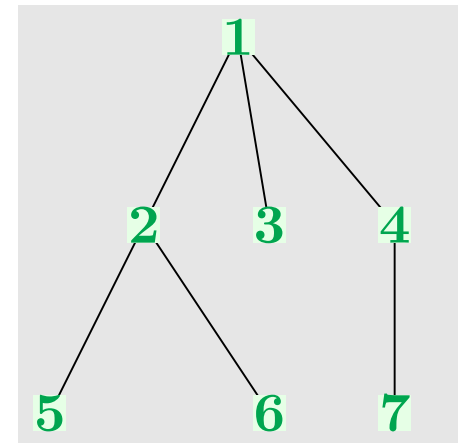
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. **połączyć krawędzią dwa pozostałe na liście wierzchołki.**

KOD:

Lista: 4,7



Drzewa — kod Prüfera

Niech $V = \{1, 2, \dots, n\}$.

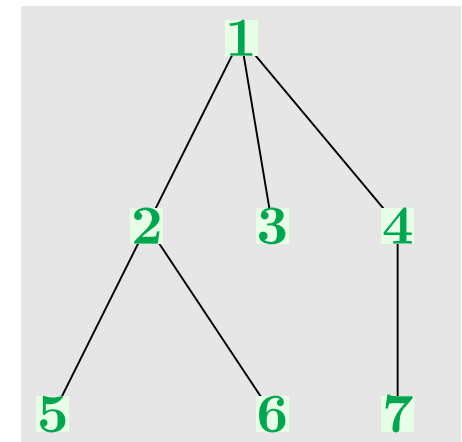
- **Kodowanie:** każde drzewo o $n > 2$ wierzchołkach V (i jakichś krawędziach) można **jednoznacznie** zapisać w postaci ciągu $n-2$ liczb ze zbioru V .
- **Odkodowywanie:** każdy ciąg długości $n-2$ liczb ze zbioru V **jednoznacznie** wyznacza drzewo o n wierzchołkach V .
- Powyższe kodowanie i odkodowywanie są swoimi wzajemnymi odwrotnościami.

Odkodowywanie:

1. znaleźć na liście najmniejszy numer k wierzchołka nie występujący w kodzie;
2. połączyć krawędzią wierzchołek k z pierwszym wierzchołkiem ℓ z kodu;
3. usunąć k z listy oraz ℓ z kodu;
4. jeśli na liście pozostały więcej niż 2 wierzchołki, to wrócić do pkt.1;
5. połączyć krawędzią dwa pozostałe na liście wierzchołki.

KOD:

Lista: 4,7



Drzewo spinające grafu

Drzewo spinające grafu

Definicja:

Drzewo **spinające** grafu to dowolny jego podgraf

- zawierający wszystkie jego wierzchołki i
- będący drzewem.

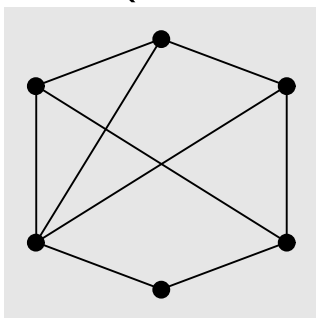
Drzewo spinające grafu

Definicja:

Drzewo **spinające** grafu to dowolny jego podgraf

- zawierający wszystkie jego wierzchołki i
- będący drzewem.

Przykład: (różne drzewa spinające tego samego grafu)



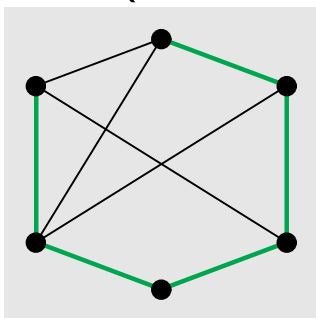
Drzewo spinające grafu

Definicja:

Drzewo **spinające** grafu to dowolny jego podgraf

- zawierający wszystkie jego wierzchołki i
- będący drzewem.

Przykład: (różne drzewa spinające tego samego grafu)



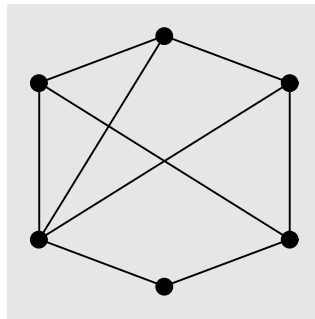
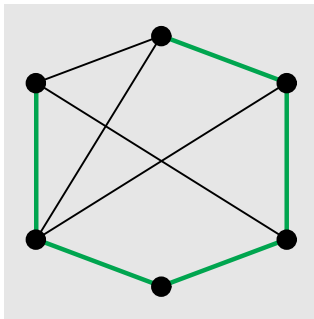
Drzewo spinające grafu

Definicja:

Drzewo **spinające** grafu to dowolny jego podgraf

- zawierający wszystkie jego wierzchołki i
- będący drzewem.

Przykład: (różne drzewa spinające tego samego grafu)



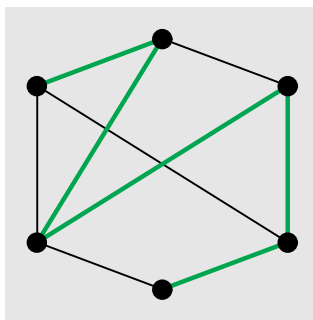
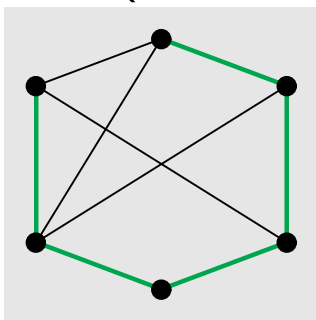
Drzewo spinające grafu

Definicja:

Drzewo **spinające** grafu to dowolny jego podgraf

- zawierający wszystkie jego wierzchołki i
- będący drzewem.

Przykład: (różne drzewa spinające tego samego grafu)



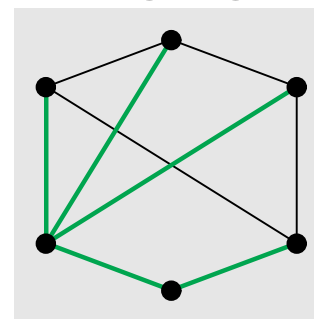
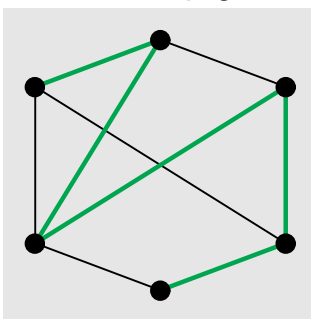
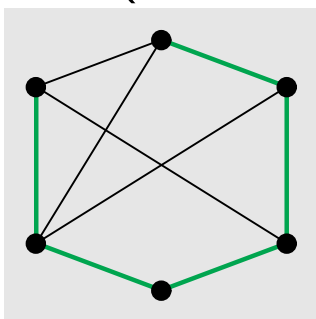
Drzewo spinające grafu

Definicja:

Drzewo **spinające** grafu to dowolny jego podgraf

- zawierający wszystkie jego wierzchołki i
- będący drzewem.

Przykład: (różne drzewa spinające tego samego grafu)



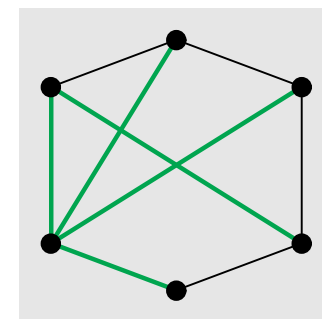
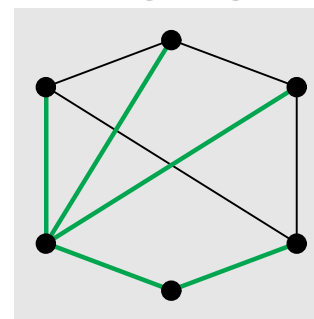
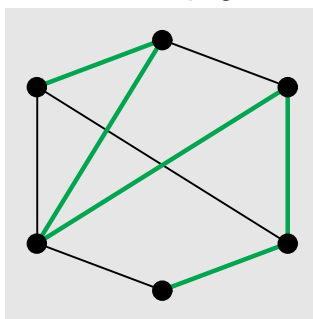
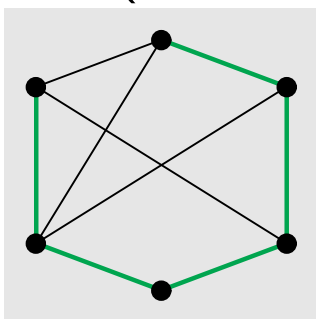
Drzewo spinające grafu

Definicja:

Drzewo **spinające** grafu to dowolny jego podgraf

- zawierający wszystkie jego wierzchołki i
- będący drzewem.

Przykład: (różne drzewa spinające tego samego grafu)



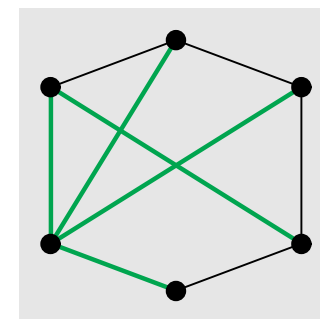
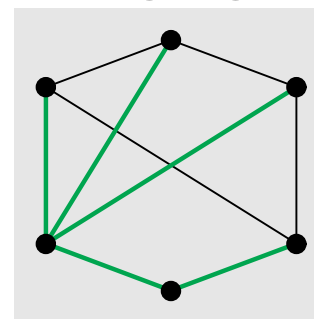
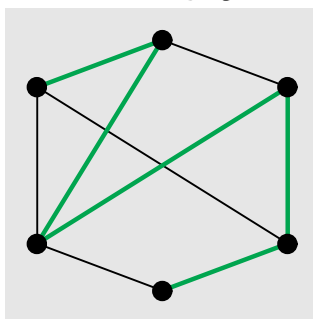
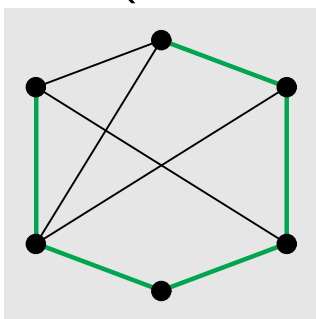
Drzewo spinające grafu

Definicja:

Drzewo **spinające** grafu to dowolny jego podgraf

- zawierający wszystkie jego wierzchołki i
- będący drzewem.

Przykład: (różne drzewa spinające tego samego grafu)



Twierdzenie: (prosty wniosek)

Drzewo spinające grafu $G = \langle V, E \rangle$ ma

- $|V|$ wierzchołków i
- $|V| - 1$ krawędzi.

Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego —
przeszukiwanie w głąb

Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego —
przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$.

Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi

Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny

Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

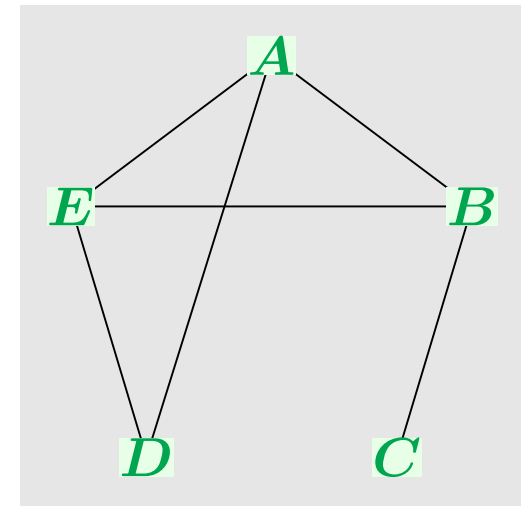
- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **jest** p , to zakończyć przeszukiwanie.

Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **jest** p , to zakończyć przeszukiwanie.

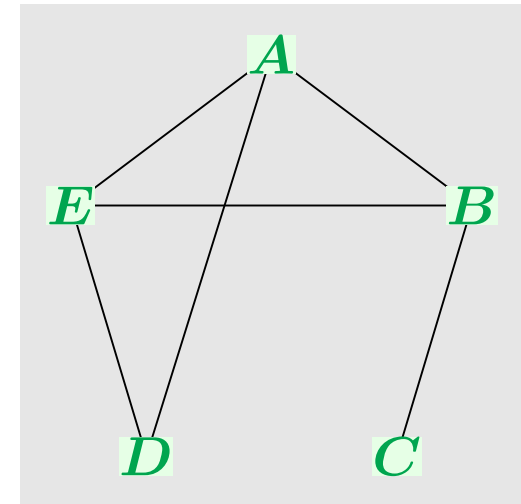


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **jest** p , to zakończyć przeszukiwanie.

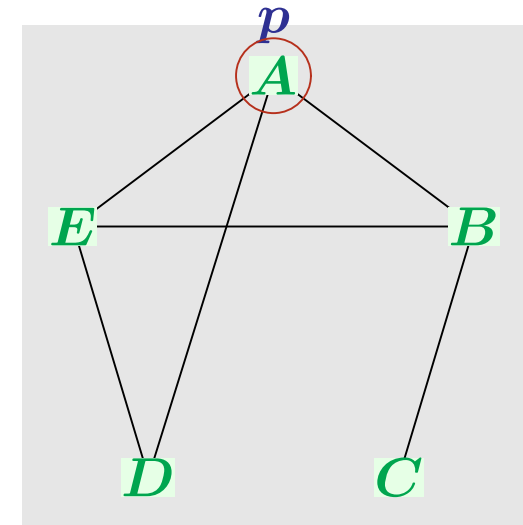


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **jest** p , to zakończyć przeszukiwanie.

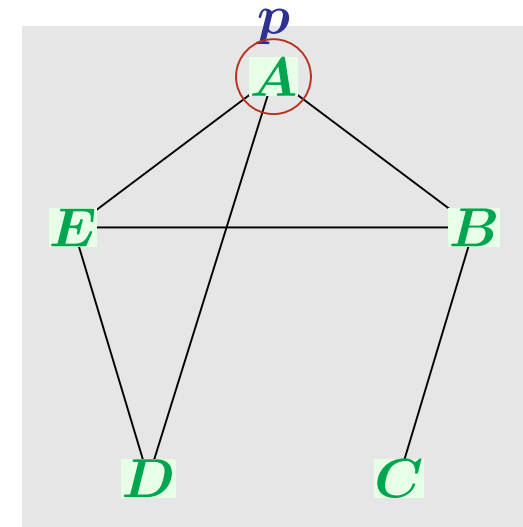


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **jest** p , to zakończyć przeszukiwanie.

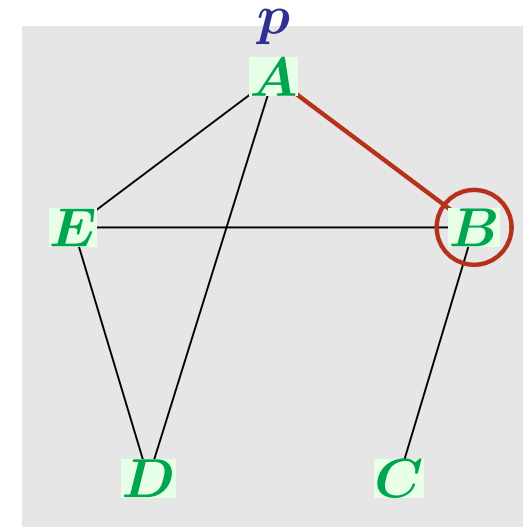


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **jest** p , to zakończyć przeszukiwanie.

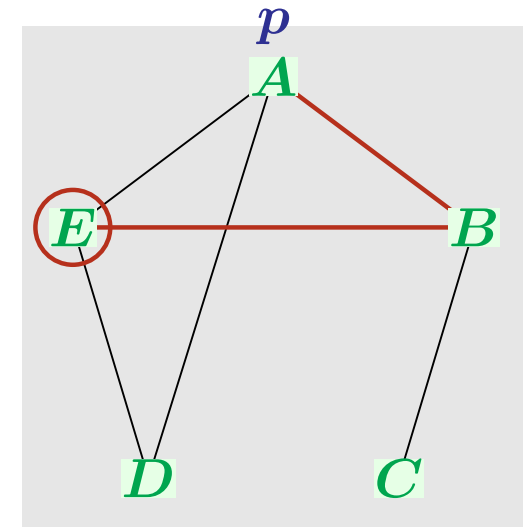


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **jest** p , to zakończyć przeszukiwanie.

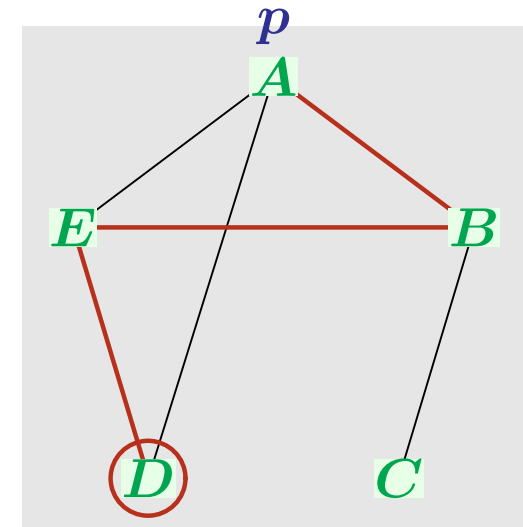


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **jest** p , to zakończyć przeszukiwanie.

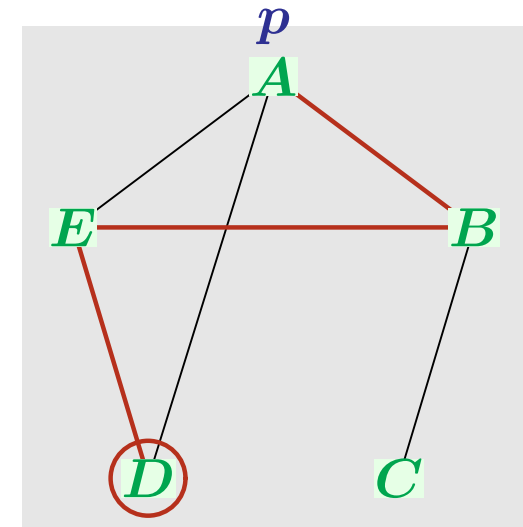


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **jest** p , to zakończyć przeszukiwanie.

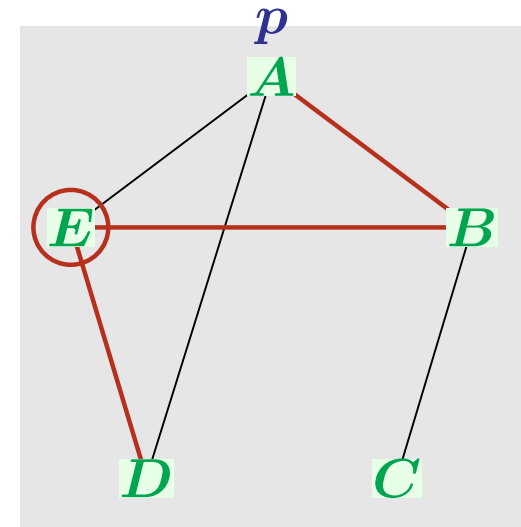


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **jest** p , to zakończyć przeszukiwanie.

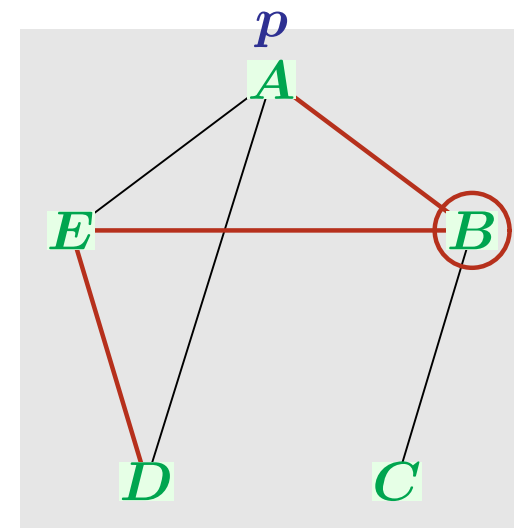


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **jest** p , to zakończyć przeszukiwanie.

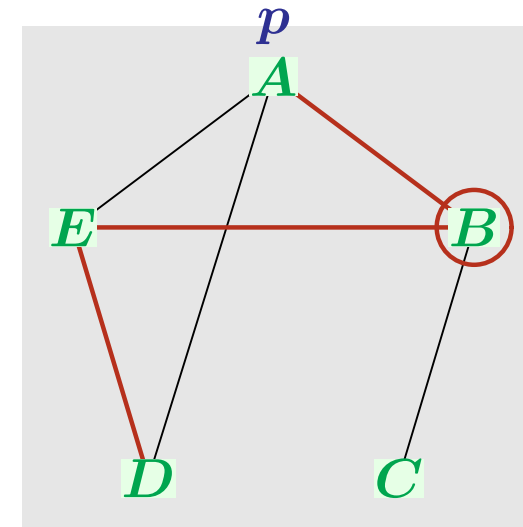


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **jest** p , to zakończyć przeszukiwanie.

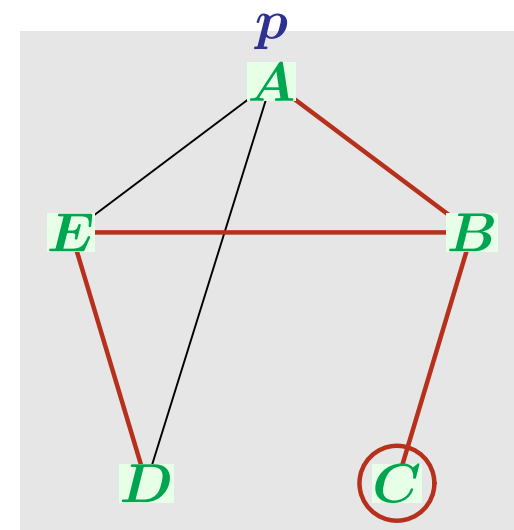


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem jest p , to zakończyć przeszukiwanie.

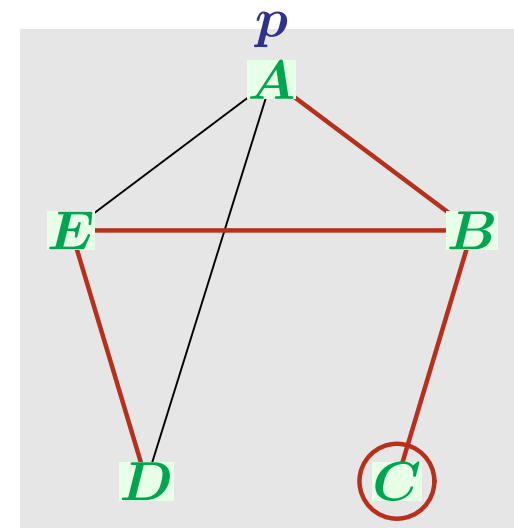


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem jest p , to zakończyć przeszukiwanie.

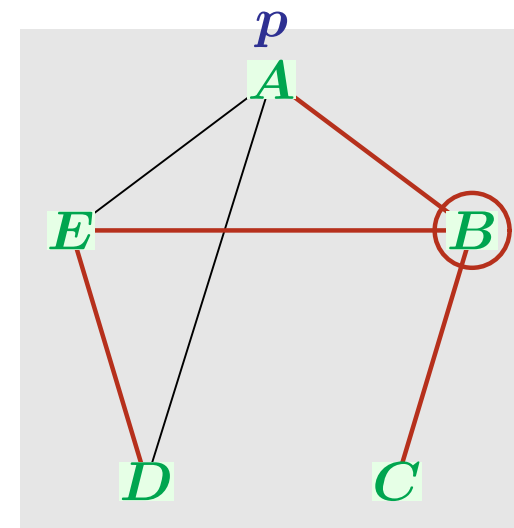


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem jest p , to zakończyć przeszukiwanie.

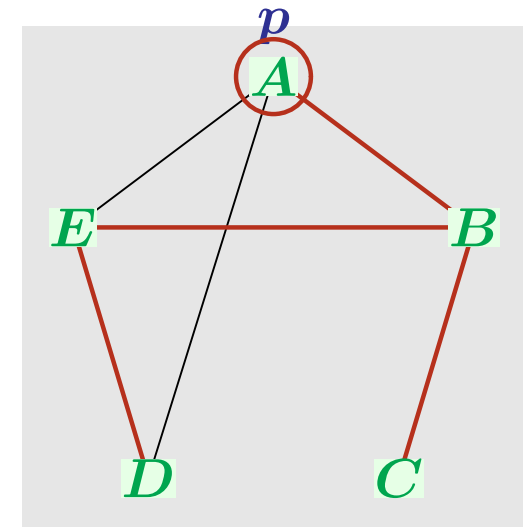


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem jest p , to zakończyć przeszukiwanie.

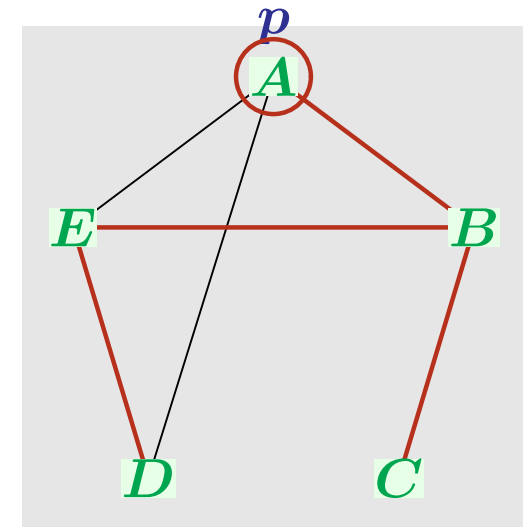


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **jest** p , to zakończyć przeszukiwanie.

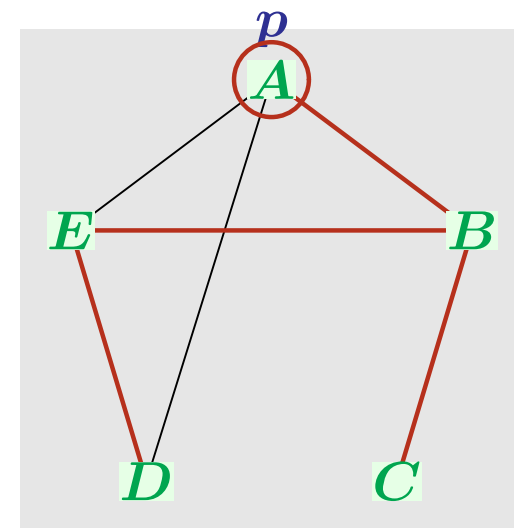


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem jest p , to zakończyć przeszukiwanie.

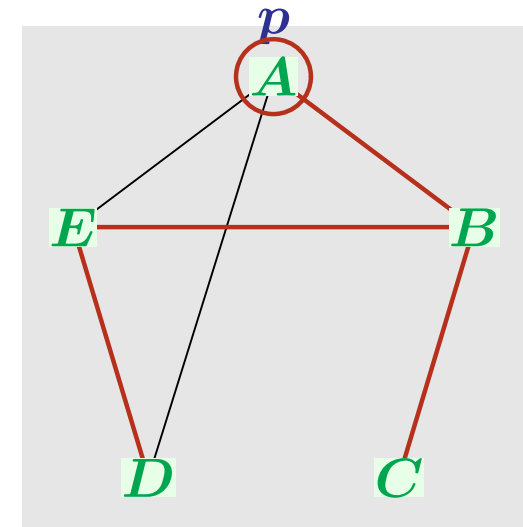


Drzewo spinające grafu

Znajdowanie drzewa spinającego dla grafu spójnego — przeszukiwanie w głąb

Zacząć od dowolnego wierzchołka $p \in V$. W trakcie całego przeszukiwania:

- jeśli z aktualnego wierzchołka prowadzi jakaś krawędź **do wierzchołka dotąd nie odwiedzonego**, to pójść po tej krawędzi
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem **nie jest** p , to cofnąć się do wierzchołka poprzedzającego aktualny
- jeśli nie ma takiej krawędzi a aktualnym wierzchołkiem jest p , to zakończyć przeszukiwanie.



Przeszukiwanie wszystkich wierzchołków grafu przydaje się w wielu algorytmach stosowanych w informatyce.

Problem trwałych małżeństw

— *stable matching problem*

Problem trwałych małżeństw

— *stable matching problem*

Porada minimalna:

nie poślubiaj nieznanomego(-ej)

Problem trwałych małżeństw

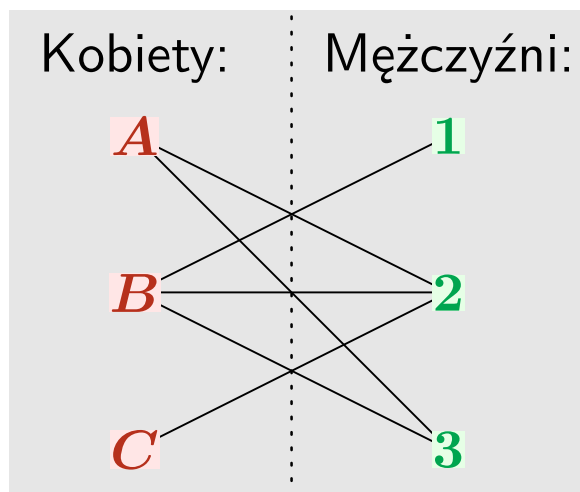
— *stable matching problem*

Porada minimalna:

nie poślubiaj nieznajomego(-ej)

Przykład:

Kto kogo zna:



Problem trwałych małżeństw

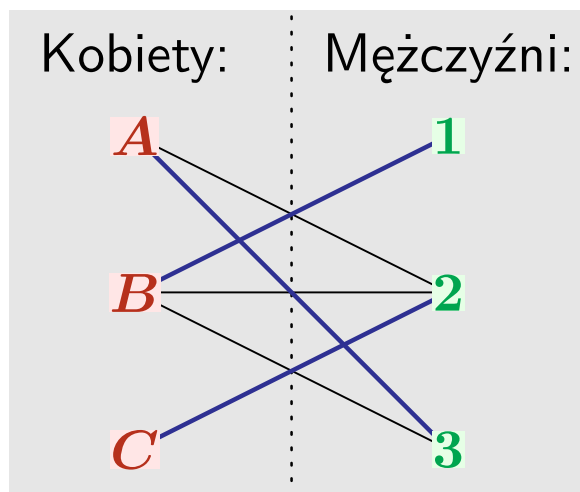
— *stable matching problem*

Porada minimalna:

nie poślubiaj nieznanego(-ej)

Przykład:

Kto kogo zna:



← np. **to** zgodne z poradą

Problem trwałych małżeństw

— *stable matching problem*

... A w terminach grafowych:

Problem trwałych małżeństw

— *stable matching problem*

... A w terminach grafowych:

Definicja:

Niech $G = \langle V_1 \cup V_2, E \rangle$ będzie grafem dwudzielnym ($V_1 \cap V_2 = \emptyset$ i każda krawędź łączy wierzchołek z V_1 z wierzchołkiem z V_2).

Problem trwałych małżeństw

— *stable matching problem*

... A w terminach grafowych:

Definicja:

Niech $G = \langle V_1 \cup V_2, E \rangle$ będzie grafem dwudzielnym ($V_1 \cap V_2 = \emptyset$ i każda krawędź łączy wierzchołek z V_1 z wierzchołkiem z V_2).

Skojarzeniem zbioru V_1 w zbiór V_2 nazywamy dowolny zbiór krawędzi $M \subseteq E$, w którym każde dwie krawędzie $e_1, e_2 \in M$ są rozłączne: $e_1 \cap e_2 = \emptyset$.

Problem trwałych małżeństw

— *stable matching problem*

... A w terminach grafowych:

Definicja:

Niech $G = \langle V_1 \cup V_2, E \rangle$ będzie grafem dwudzielnym ($V_1 \cap V_2 = \emptyset$ i każda krawędź łączy wierzchołek z V_1 z wierzchołkiem z V_2).

Skojarzeniem zbioru V_1 w zbiór V_2 nazywamy dowolny zbiór krawędzi $M \subseteq E$, w którym każde dwie krawędzie $e_1, e_2 \in M$ są rozłączne: $e_1 \cap e_2 = \emptyset$.

Pełnym skojarzeniem zbioru V_1 w zbiór V_2 nazywamy takie skojarzenie M , w którym każdy wierzchołek $v \in V_1$ należy do jakieś krawędzi $e \in M$.

Problem trwałych małżeństw

— *stable matching problem*

... A w terminach grafowych:

Definicja:

Niech $G = \langle V_1 \cup V_2, E \rangle$ będzie grafem dwudzielnym ($V_1 \cap V_2 = \emptyset$ i każda krawędź łączy wierzchołek z V_1 z wierzchołkiem z V_2).

Skojarzeniem zbioru V_1 w zbiór V_2 nazywamy dowolny zbiór krawędzi $M \subseteq E$, w którym każde dwie krawędzie $e_1, e_2 \in M$ są rozłączne: $e_1 \cap e_2 = \emptyset$.

Pełnym skojarzeniem zbioru V_1 w zbiór V_2 nazywamy takie skojarzenie M , w którym każdy wierzchołek $v \in V_1$ należy do jakieś krawędzi $e \in M$.

Uwaga: nie zakładamy, że każdy $v \in V_2$ należy do jakiejs krawędzi z M .

Problem trwałych małżeństw

— *stable matching problem*

... A w terminach grafowych:

Definicja:

Niech $G = \langle V_1 \cup V_2, E \rangle$ będzie grafem dwudzielnym ($V_1 \cap V_2 = \emptyset$ i każda krawędź łączy wierzchołek z V_1 z wierzchołkiem z V_2).

Skojarzeniem zbioru V_1 w zbiór V_2 nazywamy dowolny zbiór krawędzi $M \subseteq E$, w którym każde dwie krawędzie $e_1, e_2 \in M$ są rozłączne: $e_1 \cap e_2 = \emptyset$.

Pełnym skojarzeniem zbioru V_1 w zbiór V_2 nazywamy takie skojarzenie M , w którym każdy wierzchołek $v \in V_1$ należy do jakiejś krawędzi $e \in M$.

Uwaga: nie zakładamy, że każdy $v \in V_2$ należy do jakiejś krawędzi z M .

Definicja: (pomocnicza)

Sąsiedztwem zbioru wierzchołków $S \subseteq V_1$ nazywamy zbiór

$$N_G(S) = \{v_2 \in V_2 \mid \exists v_1 \in S \{v_1, v_2\} \in E\}$$

(czyli zbiór tych wierzchołków w V_2 , do których dochodzi jakaś krawędź z S).

Problem trwałych małżeństw

— *stable matching problem*

... A w terminach grafowych:

Definicja:

Niech $G = \langle V_1 \cup V_2, E \rangle$ będzie grafem dwudzielnym ($V_1 \cap V_2 = \emptyset$ i każda krawędź łączy wierzchołek z V_1 z wierzchołkiem z V_2).

Skojarzeniem zbioru V_1 w zbiór V_2 nazywamy dowolny zbiór krawędzi $M \subseteq E$, w którym każde dwie krawędzie $e_1, e_2 \in M$ są rozłączne: $e_1 \cap e_2 = \emptyset$.

Pełnym skojarzeniem zbioru V_1 w zbiór V_2 nazywamy takie skojarzenie M , w którym każdy wierzchołek $v \in V_1$ należy do jakiejś krawędzi $e \in M$.

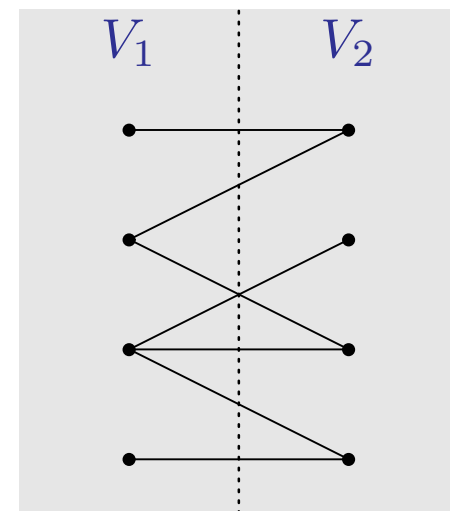
Uwaga: nie zakładamy, że każdy $v \in V_2$ należy do jakiejś krawędzi z M .

Definicja: (pomocnicza)

Sąsiedztwem zbioru wierzchołków $S \subseteq V_1$ nazywamy zbiór

$$N_G(S) = \{v_2 \in V_2 \mid \exists v_1 \in S \{v_1, v_2\} \in E\}$$

(czyli zbiór tych wierzchołków w V_2 , do których dochodzi jakaś krawędź z S).



Problem trwałych małżeństw

— *stable matching problem*

... A w terminach grafowych:

Definicja:

Niech $G = \langle V_1 \cup V_2, E \rangle$ będzie grafem dwudzielnym ($V_1 \cap V_2 = \emptyset$ i każda krawędź łączy wierzchołek z V_1 z wierzchołkiem z V_2).

Skojarzeniem zbioru V_1 w zbiór V_2 nazywamy dowolny zbiór krawędzi $M \subseteq E$, w którym każde dwie krawędzie $e_1, e_2 \in M$ są rozłączne: $e_1 \cap e_2 = \emptyset$.

Pełnym skojarzeniem zbioru V_1 w zbiór V_2 nazywamy takie skojarzenie M , w którym każdy wierzchołek $v \in V_1$ należy do jakieś krawędzi $e \in M$.

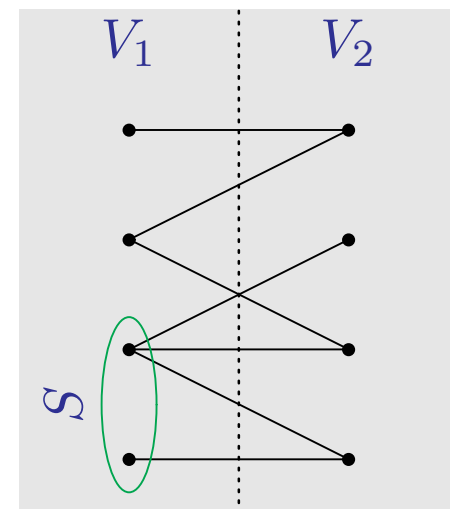
Uwaga: nie zakładamy, że każdy $v \in V_2$ należy do jakiejs krawędzi z M .

Definicja: (pomocnicza)

Sąsiedztwem zbioru wierzchołków $S \subseteq V_1$ nazywamy zbiór

$$N_G(S) = \{v_2 \in V_2 \mid \exists v_1 \in S \{v_1, v_2\} \in E\}$$

(czyli zbiór tych wierzchołków w V_2 , do których dochodzi jakaś krawędź z S).



Problem trwałych małżeństw

— *stable matching problem*

... A w terminach grafowych:

Definicja:

Niech $G = \langle V_1 \cup V_2, E \rangle$ będzie grafem dwudzielnym ($V_1 \cap V_2 = \emptyset$ i każda krawędź łączy wierzchołek z V_1 z wierzchołkiem z V_2).

Skojarzeniem zbioru V_1 w zbiór V_2 nazywamy dowolny zbiór krawędzi $M \subseteq E$, w którym każde dwie krawędzie $e_1, e_2 \in M$ są rozłączne: $e_1 \cap e_2 = \emptyset$.

Pełnym skojarzeniem zbioru V_1 w zbiór V_2 nazywamy takie skojarzenie M , w którym każdy wierzchołek $v \in V_1$ należy do jakieś krawędzi $e \in M$.

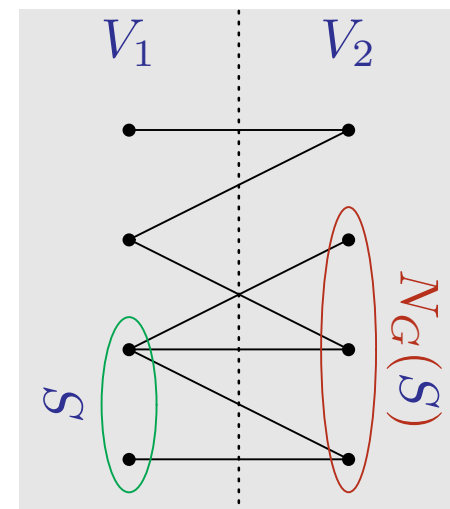
Uwaga: nie zakładamy, że każdy $v \in V_2$ należy do jakiejs krawędzi z M .

Definicja: (pomocnicza)

Sąsiedztwem zbioru wierzchołków $S \subseteq V_1$ nazywamy zbiór

$$N_G(S) = \{v_2 \in V_2 \mid \exists v_1 \in S \{v_1, v_2\} \in E\}$$

(czyli zbiór tych wierzchołków w V_2 , do których dochodzi jakaś krawędź z S).



Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
(**WH**) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
(**WH**) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic)

Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
(**WH**) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic)

$\Rightarrow$: oczywiste.

Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
(**WH**) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic)

$\Rightarrow$: oczywiste.

$\Leftarrow$: indukcja po $|V_1|$.

Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w V_2 $\iff$
(WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic)

$\Rightarrow$: oczywiste.

$\Leftarrow$: indukcja po $|V_1|$.

Dla $|V_1| = 1$ — oczywiste. Załóżmy, że $|V_1| \geq 2$.

Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w V_2 $\iff$
(WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic)

$\Rightarrow$: oczywiste.

$\Leftarrow$: indukcja po $|V_1|$.

Dla $|V_1| = 1$ — oczywiste. Załóżmy, że $|V_1| \geq 2$.

Prawa strona nierówności (WH) implikuje, że z każdego wierzchołka w V_1 wychodzi krawędź.

Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w V_2 $\iff$
 (WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic)

$\Rightarrow$: oczywiste.

$\Leftarrow$: indukcja po $|V_1|$.

Dla $|V_1| = 1$ — oczywiste. Załóżmy, że $|V_1| \geq 2$.

Prawa strona nierówności (WH) implikuje, że z każdego wierzchołka w V_1 wychodzi krawędź.

Przypadek $\forall S \subsetneq V_1 \quad |N_G(S)| \geq |S| + 1$:

Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
 (WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic)

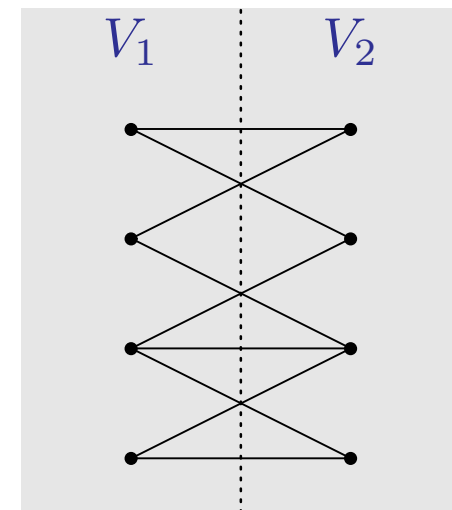
$\Rightarrow$: oczywiste.

$\Leftarrow$: indukcja po $|V_1|$.

Dla $|V_1| = 1$ — oczywiste. Załóżmy, że $|V_1| \geq 2$.

Prawa strona nierówności (WH) implikuje, że z każdego wierzchołka w V_1 wychodzi krawędź.

Przypadek $\forall S \subsetneq V_1 \quad |N_G(S)| \geq |S| + 1$:



Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
 (WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic)

$\Rightarrow$: oczywiste.

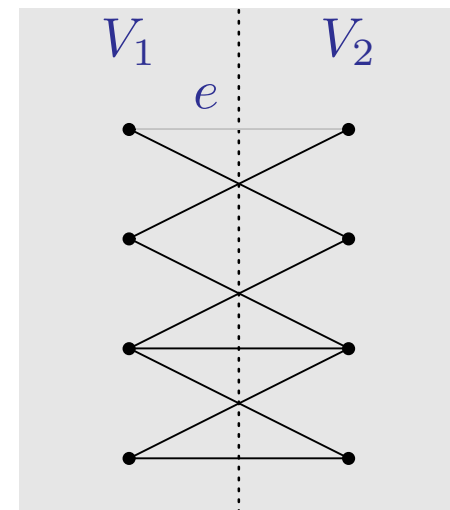
$\Leftarrow$: indukcja po $|V_1|$.

Dla $|V_1| = 1$ — oczywiste. Załóżmy, że $|V_1| \geq 2$.

Prawa strona nierówności (WH) implikuje, że z każdego wierzchołka w V_1 wychodzi krawędź.

Przypadek $\forall S \subsetneq V_1 \quad |N_G(S)| \geq |S| + 1$:

Usuńmy z grafu jedną krawędź e .



Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
 (WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic)

$\Rightarrow$: oczywiste.

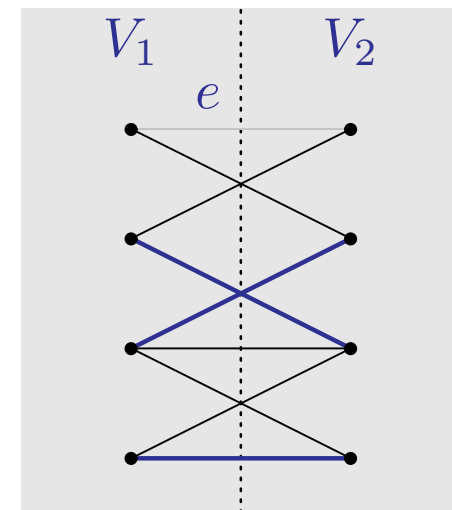
$\Leftarrow$: indukcja po $|V_1|$.

Dla $|V_1| = 1$ — oczywiste. Załóżmy, że $|V_1| \geq 2$.

Prawa strona nierówności (WH) implikuje, że z każdego wierzchołka w V_1 wychodzi krawędź.

Przypadek $\forall S \subsetneq V_1 \quad |N_G(S)| \geq |S| + 1$:

Usuńmy z grafu jedną krawędź e . Dla nowego grafu nierówność (WH) jest spełniona, więc z zał.ind. istnieje dla niego pełne skojarzenie M' .



Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
 (WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic)

$\Rightarrow$: oczywiste.

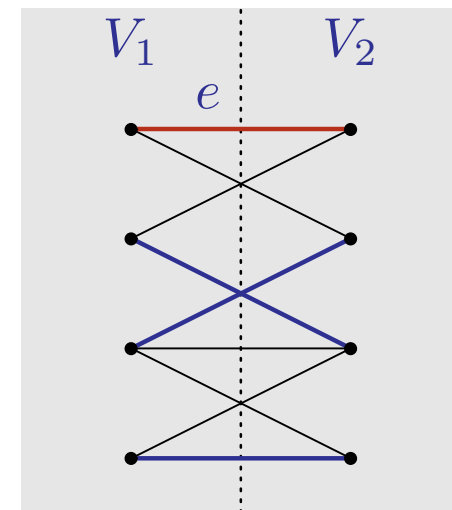
$\Leftarrow$: indukcja po $|V_1|$.

Dla $|V_1| = 1$ — oczywiste. Załóżmy, że $|V_1| \geq 2$.

Prawa strona nierówności (WH) implikuje, że z każdego wierzchołka w V_1 wychodzi krawędź.

Przypadek $\forall S \subsetneq V_1 \quad |N_G(S)| \geq |S| + 1$:

Usuńmy z grafu jedną krawędź e . Dla nowego grafu nierówność (WH) jest spełniona, więc z zał.ind. istnieje dla niego pełne skojarzenie M' . Pełne skojarzenie dla oryginalnego grafu: $M' \cup \{e\}$.



Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
(**WH**) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic — dokończenie)

Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
 (WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic — dokończenie)

$\Leftarrow$: indukcja po $|V_1|$, $|V_1| \geq 2$.

Przypadek ist. $\exists_{S' \subsetneq V_1} |N_G(S')| = |S'|$:

Problem trwałych małżeństw

— *stable matching problem*

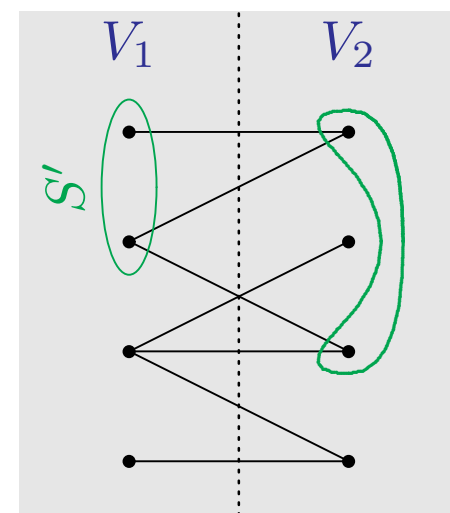
Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
 (WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic — dokończenie)

$\Leftarrow$: indukcja po $|V_1|$, $|V_1| \geq 2$.

Przypadek ist. $\exists_{S' \subsetneq V_1} |N_G(S')| = |S'|$:



Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
 (WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

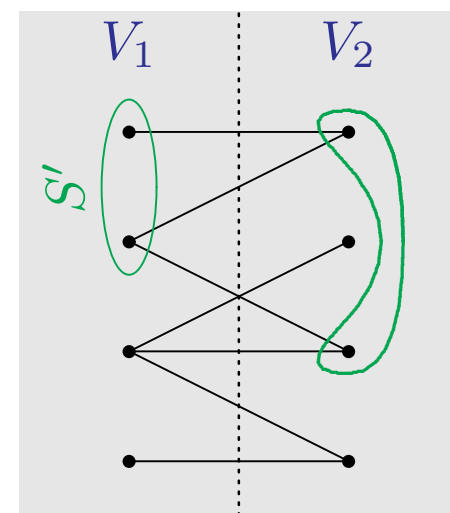
Dowód: (nieformalny szkic — dokończenie)

$\Leftarrow$: indukcja po $|V_1|$, $|V_1| \geq 2$.

Przypadek ist. $\exists_{S' \subsetneq V_1} |N_G(S')| = |S'|$:

Graf da się teraz rozbić na dwa grafy:

wierzchołki z S' i z $N_G(S')$ i wszystkie krawędzie między nimi



Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
 (WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

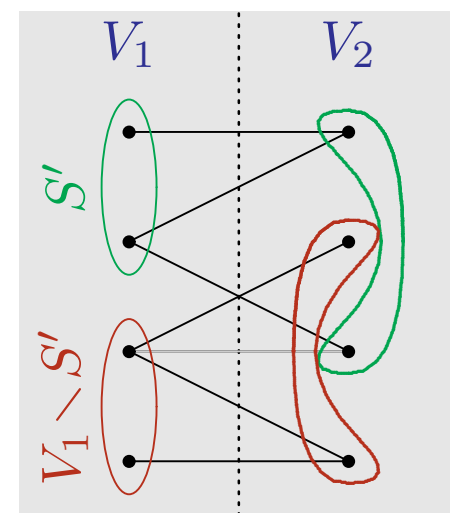
Dowód: (nieformalny szkic — dokończenie)

$\Leftarrow$: indukcja po $|V_1|$, $|V_1| \geq 2$.

Przypadek ist. $\exists_{S' \subsetneq V_1} |N_G(S')| = |S'|$:

Graf da się teraz rozbić na dwa grafy:

wierzchołki z S' i z $N_G(S')$ i wszystkie krawędzie między nimi,
 wierzchołki z $V_1 \setminus S'$ i z $V_2 \setminus N_G(S')$ i krawędzie „niekolizyjne”



Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
 (WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic — dokończenie)

$\Leftarrow$: indukcja po $|V_1|$, $|V_1| \geq 2$.

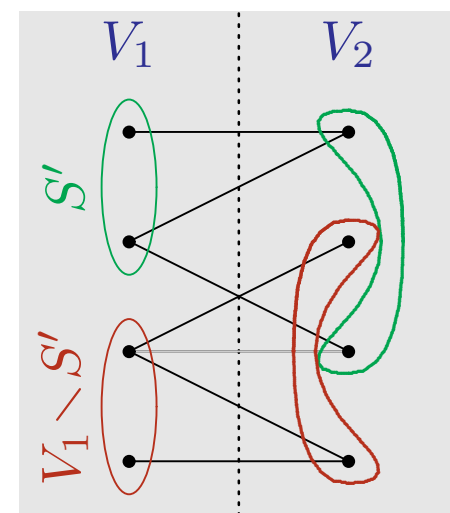
Przypadek ist. $\exists_{S' \subsetneq V_1} |N_G(S')| = |S'|$:

Graf da się teraz rozbić na dwa grafy:

wierzchołki z S' i z $N_G(S')$ i wszystkie krawędzie między nimi,

wierzchołki z $V_1 \setminus S'$ i z $V_2 \setminus N_G(S')$ i krawędzie „niekolizyjne”.

W każdym z tych grafów istnieje pełne skojarzenie.



Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
 (WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic — dokończenie)

$\Leftarrow$: indukcja po $|V_1|$, $|V_1| \geq 2$.

Przypadek ist. $\exists_{S' \subsetneq V_1} |N_G(S')| = |S'|$:

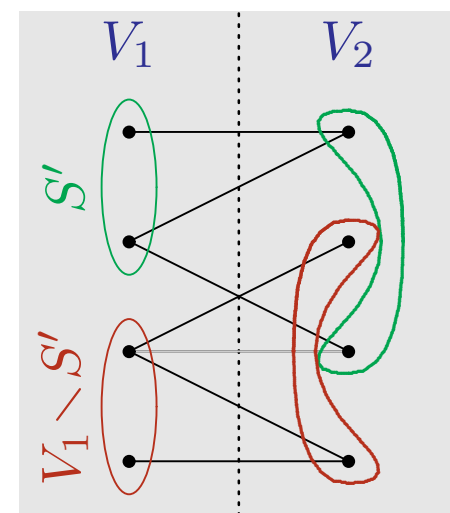
Graf da się teraz rozbić na dwa grafy:

wierzchołki z S' i z $N_G(S')$ i wszystkie krawędzie między nimi,

wierzchołki z $V_1 \setminus S'$ i z $V_2 \setminus N_G(S')$ i krawędzie „niekolizyjne”.

W każdym z tych grafów istnieje pełne skojarzenie.

Z nich robimy pełne skojarzenie dla całego grafu.



□

Problem trwałych małżeństw

— *stable matching problem*

Twierdzenie: (Hall, 1935)

W grafie dwudzielnym j.w. istnieje pełne skojarzenie V_1 w $V_2 \iff$
 (WH) dla każdego podzbioru $S \subseteq V_1$ zachodzi $|N_G(S)| \geq |S|$

Dowód: (nieformalny szkic — dokończenie)

$\Leftarrow$: indukcja po $|V_1|$, $|V_1| \geq 2$.

Przypadek ist. $\exists_{S' \subsetneq V_1} |N_G(S')| = |S'|$:

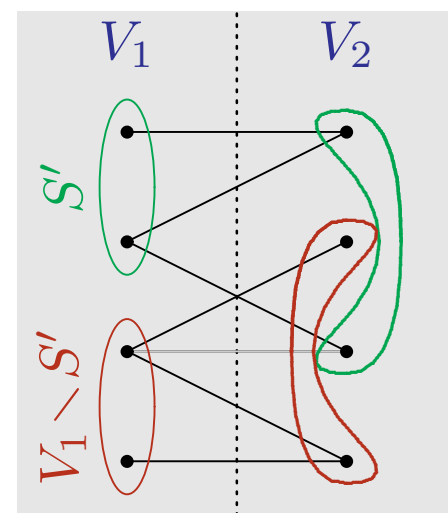
Graf da się teraz rozbić na dwa grafy:

wierzchołki z S' i z $N_G(S')$ i wszystkie krawędzie między nimi,

wierzchołki z $V_1 \setminus S'$ i z $V_2 \setminus N_G(S')$ i krawędzie „niekolizyjne”.

W każdym z tych grafów istnieje pełne skojarzenie.

Z nich robimy pełne skojarzenie dla całego grafu.



□

Twierdzenie: (wniosek z tw. Halla)

W grafie dwudzielnym regularnym (czyli z tym samym stopniem wszystkich wierzchołków) istnieje pełne skojarzenie.

Zastosowanie: Kwadraty łacińskie

Definicja:

Kwadrat łaciński stopnia $n \in \mathbb{N}$ to macierz $n \times n$ wypełniona liczbami ze zbioru $P_n = \{1, 2, \dots, n\}$ w taki sposób, że każdy wiersz i każda kolumna jest permutacją P_n

Zastosowanie: Kwadraty łacińskie

Definicja:

Kwadrat łaciński stopnia $n \in \mathbb{N}$ to macierz $n \times n$ wypełniona liczbami ze zbioru $P_n = \{1, 2, \dots, n\}$ w taki sposób, że każdy wiersz i każda kolumna jest permutacją P_n (czyli w każdym wierszu i każdej kolumnie każda liczba ze zbioru P_n występuje dokładnie raz).

Zastosowanie: Kwadraty łacińskie

Definicja:

Kwadrat łaciński stopnia $n \in \mathbb{N}$ to macierz $n \times n$ wypełniona liczbami ze zbioru $P_n = \{1, 2, \dots, n\}$ w taki sposób, że każdy wiersz i każda kolumna jest permutacją P_n (czyli w każdym wierszu i każdej kolumnie każda liczba ze zbioru P_n występuje dokładnie raz).

Przykład:

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

OK.

Zastosowanie: Kwadraty łacińskie

Definicja:

Kwadrat łaciński stopnia $n \in \mathbb{N}$ to macierz $n \times n$ wypełniona liczbami ze zbioru $P_n = \{1, 2, \dots, n\}$ w taki sposób, że każdy wiersz i każda kolumna jest permutacją P_n (czyli w każdym wierszu i każdej kolumnie każda liczba ze zbioru P_n występuje dokładnie raz).

Przykład:

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

OK.

OK.

Zastosowanie: Kwadraty łacińskie

Definicja:

Kwadrat łaciński stopnia $n \in \mathbb{N}$ to macierz $n \times n$ wypełniona liczbami ze zbioru $P_n = \{1, 2, \dots, n\}$ w taki sposób, że każdy wiersz i każda kolumna jest permutacją P_n (czyli w każdym wierszu i każdej kolumnie każda liczba ze zbioru P_n występuje dokładnie raz).

Przykład:

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

OK.

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

OK.

$$\begin{bmatrix} 1 & 2 & 4 & 3 \\ 2 & 1 & 3 & 4 \\ 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

OK.

Zastosowanie: Kwadraty łacińskie

Definicja:

Kwadrat łaciński stopnia $n \in \mathbb{N}$ to macierz $n \times n$ wypełniona liczbami ze zbioru $P_n = \{1, 2, \dots, n\}$ w taki sposób, że każdy wiersz i każda kolumna jest permutacją P_n (czyli w każdym wierszu i każdej kolumnie każda liczba ze zbioru P_n występuje dokładnie raz).

Przykład:

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

OK.

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

OK.

$$\begin{bmatrix} 1 & 2 & 4 & 3 \\ 2 & 1 & 3 & 4 \\ 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

OK.

$$\begin{bmatrix} 1 & 2 & 4 & 3 \\ 2 & 4 & 3 & 1 \\ 3 & 2 & 1 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

NIE!

Zastosowanie: Kwadraty łacińskie

Jak utworzyć kwadrat łaciński 4×4 ?

Zastosowanie: Kwadraty łacińskie

Jak utworzyć kwadrat łaciński 4×4 ?

Krok 1 —

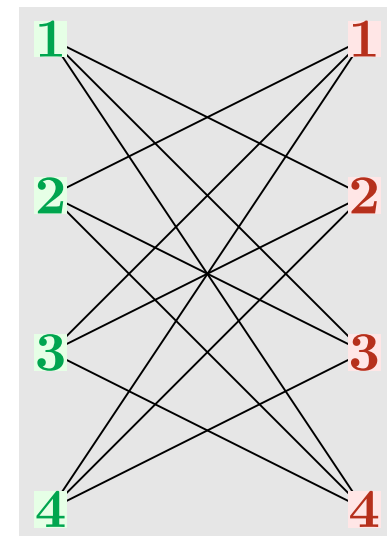
graf dwudzielny z krawędziami między **różnymi** liczbami

Zastosowanie: Kwadraty łacińskie

Jak utworzyć kwadrat łaciński 4×4 ?

Krok 1 —

graf dwudzielny z krawędziami między **różnymi** liczbami

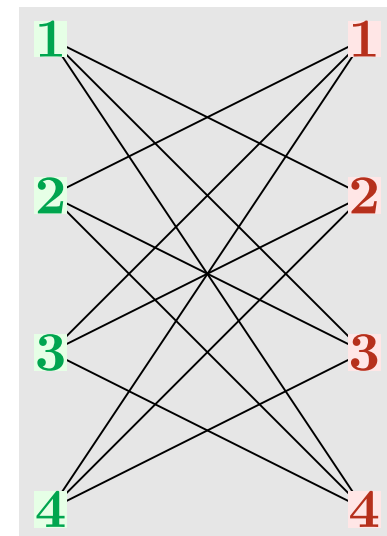


Zastosowanie: Kwadraty łacińskie

Jak utworzyć kwadrat łaciński 4×4 ?

Krok 1 —

graf dwudzielny z krawędziami między **różnymi** liczbami
znaleźć w nim pełne skojarzenie

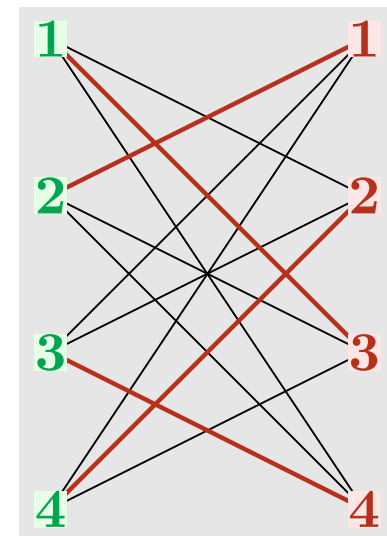


Zastosowanie: Kwadraty łacińskie

Jak utworzyć kwadrat łaciński 4×4 ?

Krok 1 —

graf dwudzielny z krawędziami między **różnymi** liczbami
znaleźć w nim pełne skojarzenie



Zastosowanie: Kwadraty łacińskie

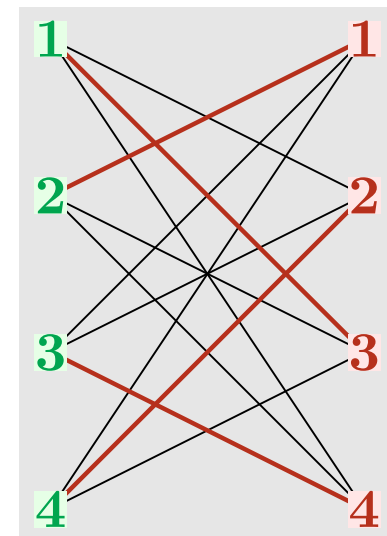
Jak utworzyć kwadrat łaciński 4×4 ?

Krok 1 —

graf dwudzielny z krawędziami między **różnymi** liczbami
znaleźć w nim pełne skojarzenie

pierwsze dwie kolumny kwadratu wg tego skojarzenia:

$$\begin{bmatrix} 1 & 3 & \dots \\ 2 & 1 & \dots \\ 3 & 4 & \dots \\ 4 & 2 & \dots \end{bmatrix}$$



Zastosowanie: Kwadraty łacińskie

Jak utworzyć kwadrat łaciński 4×4 ?

Krok 1 —

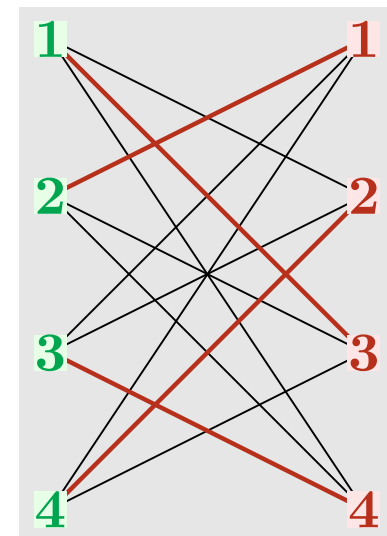
graf dwudzielny z krawędziami między **różnymi** liczbami
znaleźć w nim pełne skojarzenie

pierwsze dwie kolumny kwadratu wg tego skojarzenia:

$$\begin{bmatrix} 1 & 3 & \dots \\ 2 & 1 & \dots \\ 3 & 4 & \dots \\ 4 & 2 & \dots \end{bmatrix}$$

Krok 2 —

graf dwudzielny; po lewej wiersze macierzy, po prawej liczby



Zastosowanie: Kwadraty łacińskie

Jak utworzyć kwadrat łaciński 4×4 ?

Krok 1 —

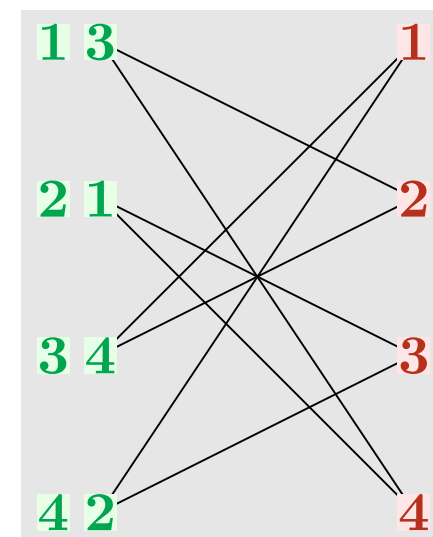
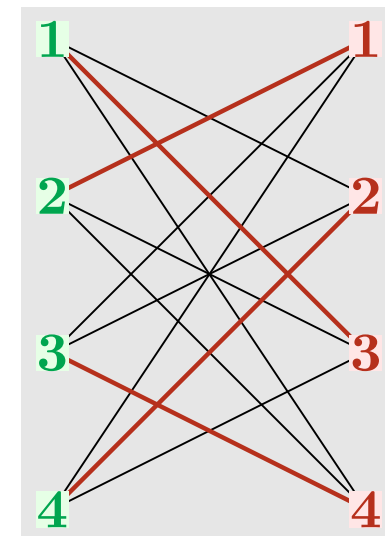
graf dwudzielny z krawędziami między **różnymi** liczbami
znaleźć w nim pełne skojarzenie

pierwsze dwie kolumny kwadratu wg tego skojarzenia:

$$\begin{bmatrix} 1 & 3 & \dots \\ 2 & 1 & \dots \\ 3 & 4 & \dots \\ 4 & 2 & \dots \end{bmatrix}$$

Krok 2 —

graf dwudzielny; po lewej wiersze macierzy, po prawej liczby



Zastosowanie: Kwadraty łacińskie

Jak utworzyć kwadrat łaciński 4×4 ?

Krok 1 —

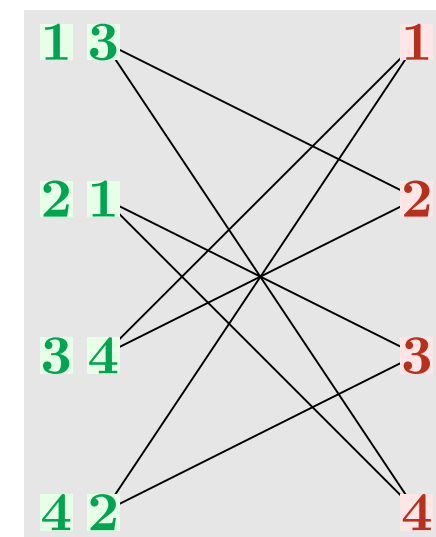
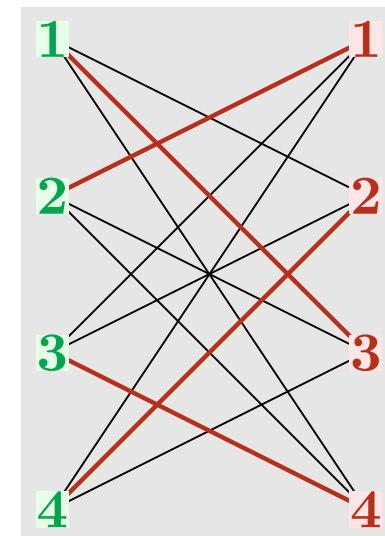
graf dwudzielny z krawędziami między **różnymi** liczbami
znaleźć w nim pełne skojarzenie

pierwsze dwie kolumny kwadratu wg tego skojarzenia:

$$\begin{bmatrix} 1 & 3 & \dots \\ 2 & 1 & \dots \\ 3 & 4 & \dots \\ 4 & 2 & \dots \end{bmatrix}$$

Krok 2 —

graf dwudzielny; po lewej wiersze macierzy, po prawej liczby
znaleźć w nim pełne skojarzenie



Zastosowanie: Kwadraty łacińskie

Jak utworzyć kwadrat łaciński 4×4 ?

Krok 1 —

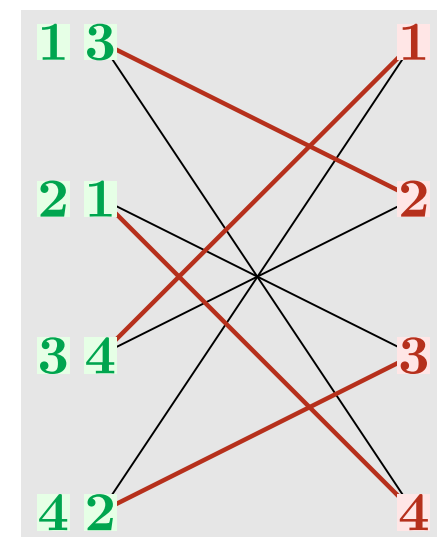
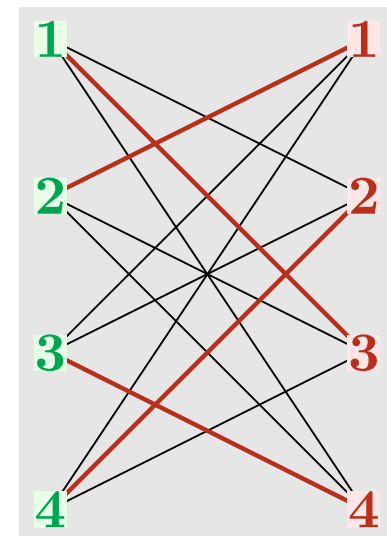
graf dwudzielny z krawędziami między **różnymi** liczbami
znaleźć w nim pełne skojarzenie

pierwsze dwie kolumny kwadratu wg tego skojarzenia:

$$\begin{bmatrix} 1 & 3 & \dots \\ 2 & 1 & \dots \\ 3 & 4 & \dots \\ 4 & 2 & \dots \end{bmatrix}$$

Krok 2 —

graf dwudzielny; po lewej wiersze macierzy, po prawej liczby
znaleźć w nim pełne skojarzenie



Zastosowanie: Kwadraty łacińskie

Jak utworzyć kwadrat łaciński 4×4 ?

Krok 1 —

graf dwudzielny z krawędziami między **różnymi** liczbami
znaleźć w nim pełne skojarzenie

pierwsze dwie kolumny kwadratu wg tego skojarzenia:

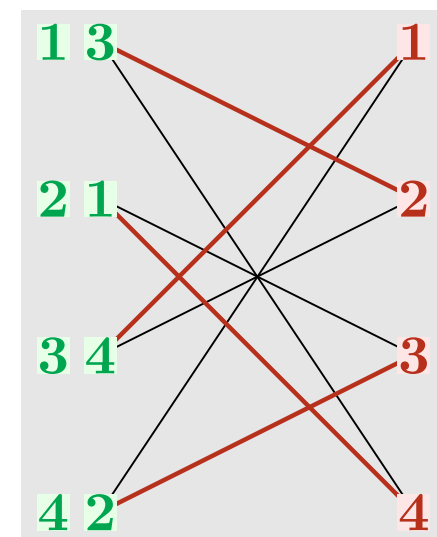
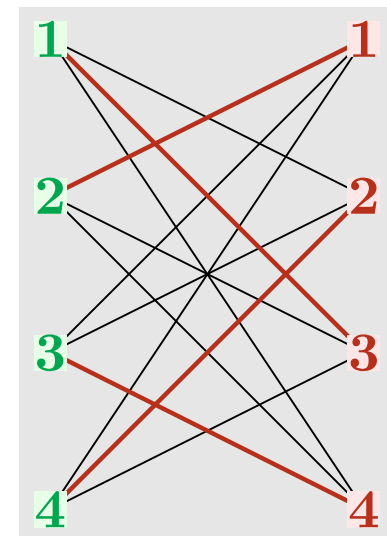
$$\begin{bmatrix} 1 & 3 & \dots \\ 2 & 1 & \dots \\ 3 & 4 & \dots \\ 4 & 2 & \dots \end{bmatrix}$$

Krok 2 —

graf dwudzielny; po lewej wiersze macierzy, po prawej liczby
znaleźć w nim pełne skojarzenie

pierwsze trzy kolumny kwadratu wg tego skojarzenia:

$$\begin{bmatrix} 1 & 3 & 2 & \dots \\ 2 & 1 & 4 & \dots \\ 3 & 4 & 1 & \dots \\ 4 & 2 & 3 & \dots \end{bmatrix}$$



Zastosowanie: Kwadraty łacińskie

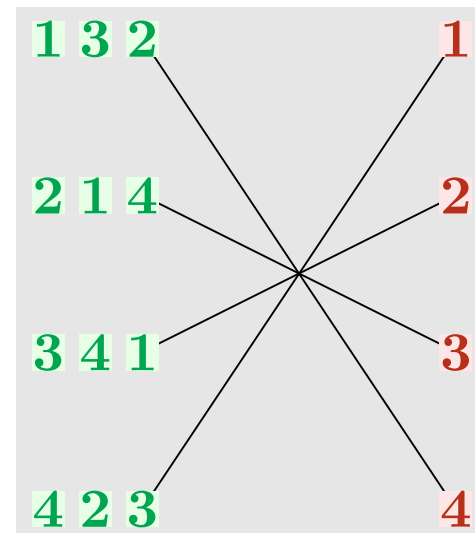
Krok 3 —

graf dwudzielny; po lewej wiersze macierzy, po prawej liczby

Zastosowanie: Kwadraty łącińskie

Krok 3 —

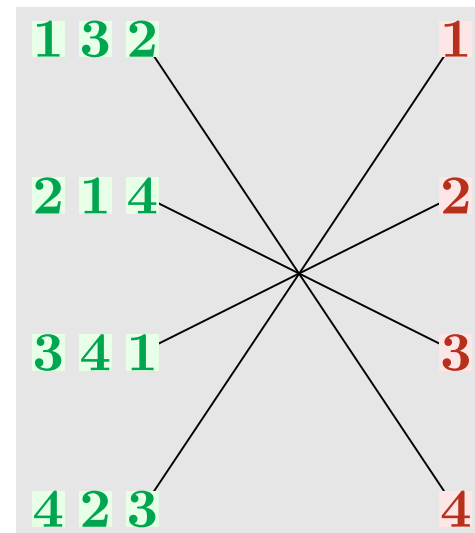
graf dwudzielny; po lewej wiersze macierzy, po prawej liczby



Zastosowanie: Kwadraty łacińskie

Krok 3 —

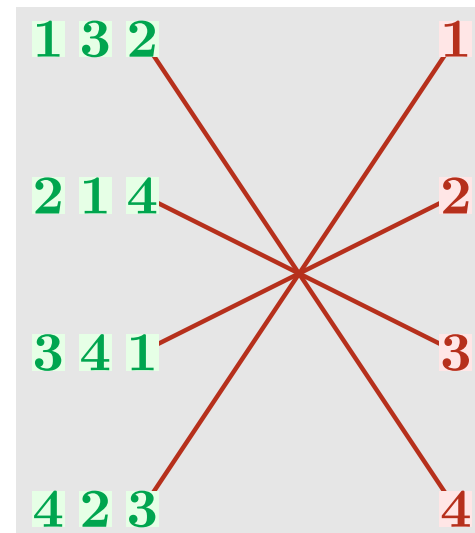
graf dwudzielny; po lewej wiersze macierzy, po prawej liczby
znaleźć w nim pełne skojarzenie (nie ma już wyboru...)



Zastosowanie: Kwadraty łącińskie

Krok 3 —

graf dwudzielny; po lewej wiersze macierzy, po prawej liczby
znaleźć w nim pełne skojarzenie (nie ma już wyboru...)



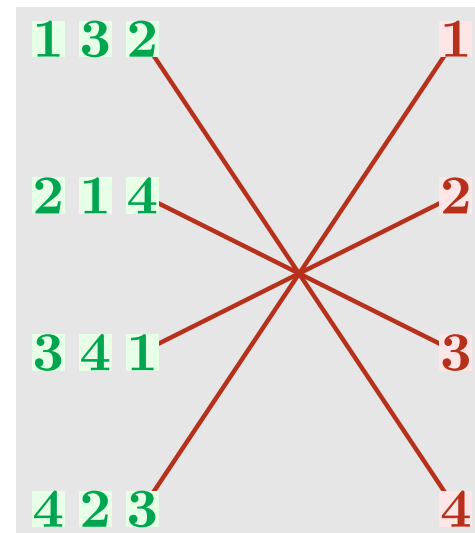
Zastosowanie: Kwadraty łacińskie

Krok 3 —

graf dwudzielny; po lewej wiersze macierzy, po prawej liczby
znaleźć w nim pełne skojarzenie (nie ma już wyboru...)

cały kwadrat:

$$\begin{bmatrix} 1 & 3 & 2 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ 4 & 2 & 3 & 1 \end{bmatrix}$$



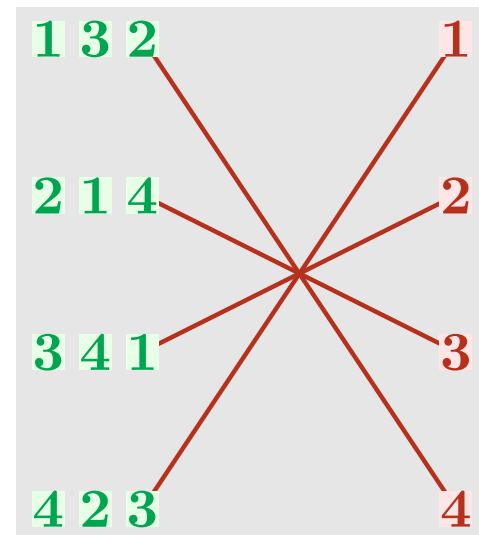
Zastosowanie: Kwadraty łacińskie

Krok 3 —

graf dwudzielny; po lewej wiersze macierzy, po prawej liczby
znaleźć w nim pełne skojarzenie (nie ma już wyboru...)

cały kwadrat:

$$\begin{bmatrix} 1 & 3 & 2 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ 4 & 2 & 3 & 1 \end{bmatrix}$$



Oczywiście tak można działać również dla kwadratów większych niż 4×4 .

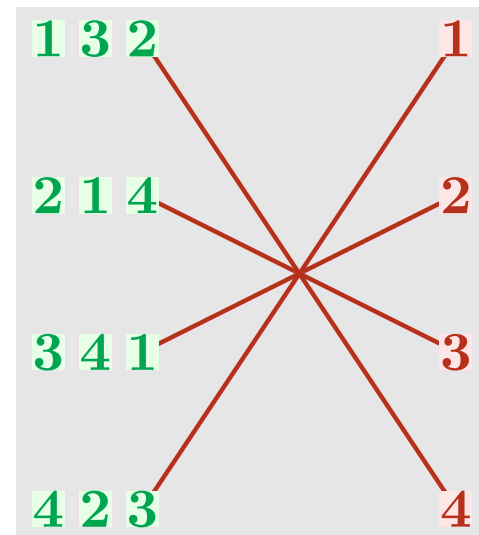
Zastosowanie: Kwadraty łącińskie

Krok 3 —

graf dwudzielny; po lewej wiersze macierzy, po prawej liczby
znaleźć w nim pełne skojarzenie (nie ma już wyboru...)

cały kwadrat:

$$\begin{bmatrix} 1 & 3 & 2 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ 4 & 2 & 3 & 1 \end{bmatrix}$$



Oczywiście tak można działać również dla kwadratów większych niż 4×4 .

Istotna jest możliwość znalezienia pełnego skojarzenia w każdym kroku.

Grafy hamiltonowskie

Z poprzedniego wykładu:

Definicja:

Graf G jest eulerowski $\iff$ istnieje cykl zawierający *wszystkie* jego krawędzie.

Grafy hamiltonowskie

Z poprzedniego wykładu:

Definicja:

Graf G jest **eulerowski** $\iff$ istnieje cykl zawierający *wszystkie* jego krawędzie.

Definicja: (nowa)

Graf G jest **hamiltonowski** $\iff$ istnieje cykl zawierający *wszystkie* jego wierzchołki

Grafy hamiltonowskie

Z poprzedniego wykładu:

Definicja:

Graf G jest **eulerowski** $\iff$ istnieje cykl zawierający *wszystkie* jego krawędzie.

Definicja: (nowa)

Graf G jest **hamiltonowski** $\iff$ istnieje cykl zawierający *wszystkie* jego wierzchołki, każdy po 1 razie (z wyjątkiem początkowego i końcowego).

Grafy hamiltonowskie

Z poprzedniego wykładu:

Definicja:

Graf G jest **eulerowski** $\iff$ istnieje cykl zawierający *wszystkie* jego krawędzie.

Definicja: (nowa)

Graf G jest **hamiltonowski** $\iff$ istnieje cykl zawierający *wszystkie* jego wierzchołki, każdy po 1 razie (z wyjątkiem początkowego i końcowego).

Grafy hamiltonowskie

Z poprzedniego wykładu:

Definicja:

Graf G jest **eulerowski** $\iff$ istnieje cykl zawierający *wszystkie* jego krawędzie.

Definicja: (nowa)

Graf G jest **hamiltonowski** $\iff$ istnieje cykl zawierający *wszystkie* jego wierzchołki, każdy po 1 razie (z wyjątkiem początkowego i końcowego).

Grafy hamiltonowskie

Z poprzedniego wykładu:

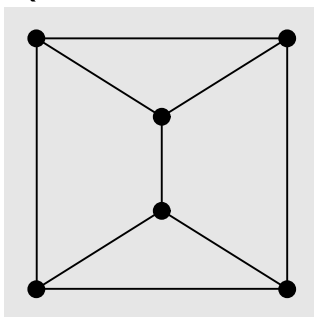
Definicja:

Graf G jest **eulerowski** $\iff$ istnieje cykl zawierający *wszystkie* jego krawędzie.

Definicja: (nowa)

Graf G jest **hamiltonowski** $\iff$ istnieje cykl zawierający *wszystkie* jego wierzchołki, każdy po 1 razie (z wyjątkiem początkowego i końcowego).

Przykład: (grafy hamiltonowskie)



Grafy hamiltonowskie

Z poprzedniego wykładu:

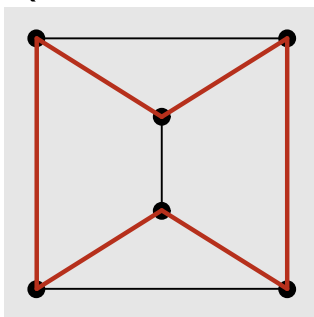
Definicja:

Graf G jest **eulerowski** $\iff$ istnieje cykl zawierający *wszystkie* jego krawędzie.

Definicja: (nowa)

Graf G jest **hamiltonowski** $\iff$ istnieje cykl zawierający *wszystkie* jego wierzchołki, każdy po 1 razie (z wyjątkiem początkowego i końcowego).

Przykład: (grafy hamiltonowskie)



Grafy hamiltonowskie

Z poprzedniego wykładu:

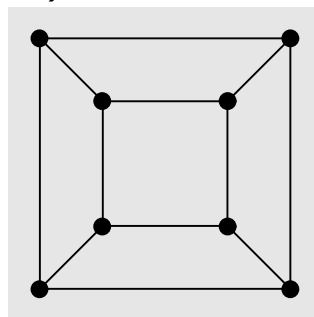
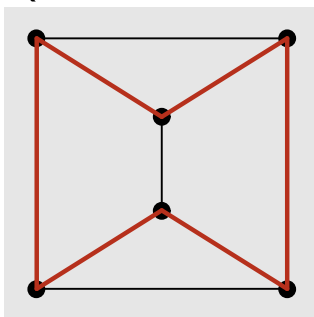
Definicja:

Graf G jest **eulerowski** $\iff$ istnieje cykl zawierający *wszystkie* jego krawędzie.

Definicja: (nowa)

Graf G jest **hamiltonowski** $\iff$ istnieje cykl zawierający *wszystkie* jego wierzchołki, każdy po 1 razie (z wyjątkiem początkowego i końcowego).

Przykład: (grafy hamiltonowskie)



Grafy hamiltonowskie

Z poprzedniego wykładu:

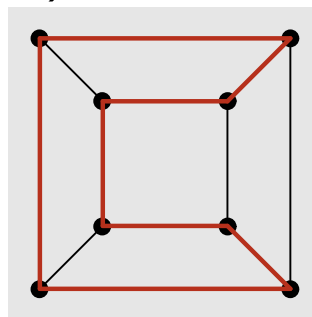
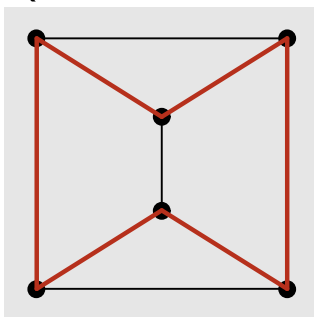
Definicja:

Graf G jest **eulerowski** $\iff$ istnieje cykl zawierający *wszystkie* jego krawędzie.

Definicja: (nowa)

Graf G jest **hamiltonowski** $\iff$ istnieje cykl zawierający *wszystkie* jego wierzchołki, każdy po 1 razie (z wyjątkiem początkowego i końcowego).

Przykład: (grafy hamiltonowskie)



Grafy hamiltonowskie

Z poprzedniego wykładu:

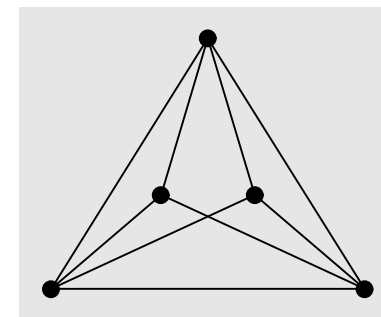
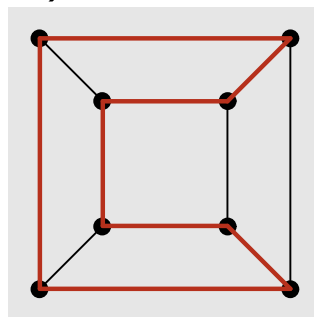
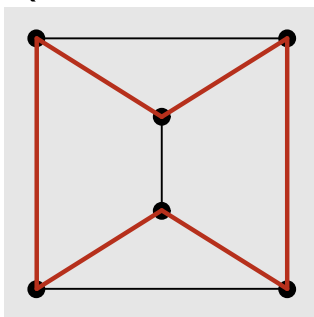
Definicja:

Graf G jest **eulerowski** $\iff$ istnieje cykl zawierający *wszystkie* jego krawędzie.

Definicja: (nowa)

Graf G jest **hamiltonowski** $\iff$ istnieje cykl zawierający *wszystkie* jego wierzchołki, każdy po 1 razie (z wyjątkiem początkowego i końcowego).

Przykład: (grafy hamiltonowskie)



Grafy hamiltonowskie

Z poprzedniego wykładu:

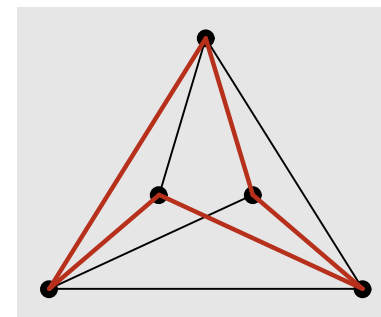
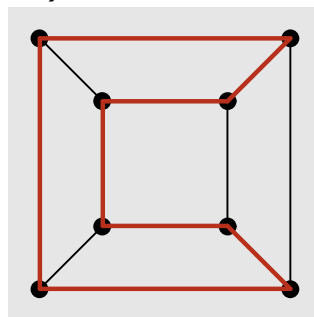
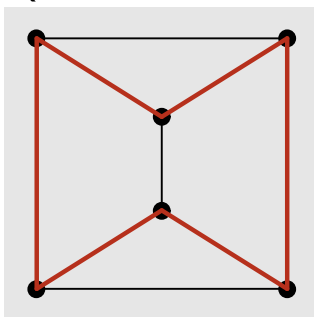
Definicja:

Graf G jest **eulerowski** $\iff$ istnieje cykl zawierający *wszystkie* jego krawędzie.

Definicja: (nowa)

Graf G jest **hamiltonowski** $\iff$ istnieje cykl zawierający *wszystkie* jego wierzchołki, każdy po 1 razie (z wyjątkiem początkowego i końcowego).

Przykład: (grafy hamiltonowskie)



Grafy hamiltonowskie

Z poprzedniego wykładu:

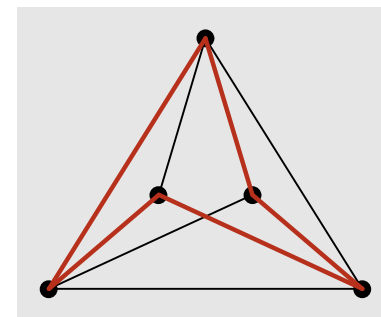
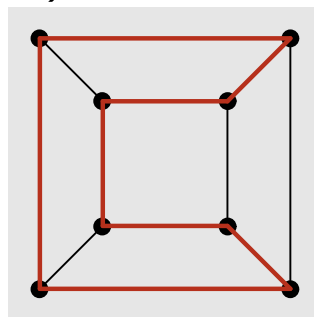
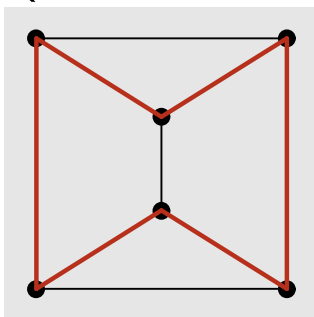
Definicja:

Graf G jest **eulerowski** $\iff$ istnieje cykl zawierający *wszystkie* jego krawędzie.

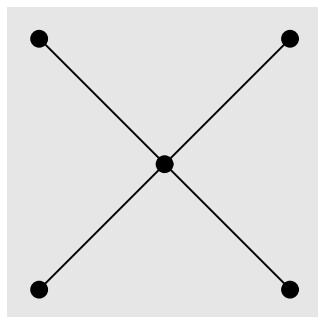
Definicja: (nowa)

Graf G jest **hamiltonowski** $\iff$ istnieje cykl zawierający *wszystkie* jego wierzchołki, każdy po 1 razie (z wyjątkiem początkowego i końcowego).

Przykład: (grafy hamiltonowskie)



Przykład: (grafy niehamiltonowskie)



Grafy hamiltonowskie

Z poprzedniego wykładu:

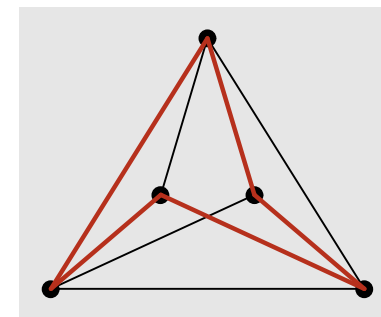
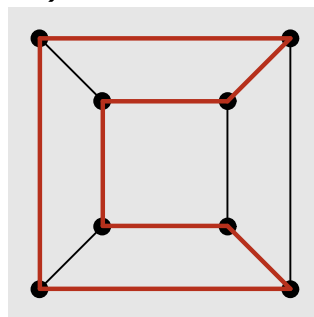
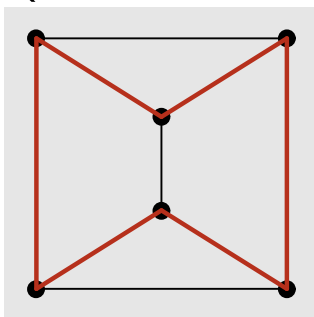
Definicja:

Graf G jest **eulerowski** $\iff$ istnieje cykl zawierający *wszystkie* jego krawędzie.

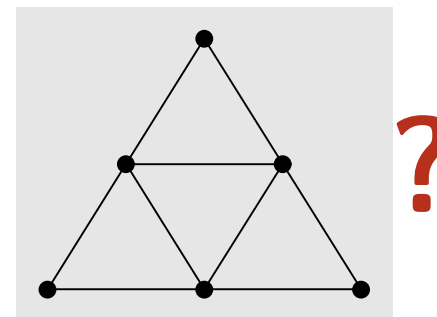
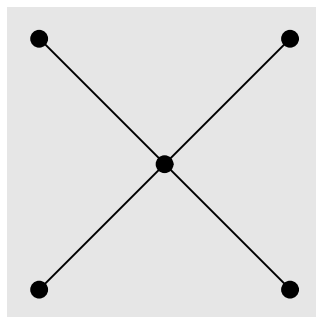
Definicja: (nowa)

Graf G jest **hamiltonowski** $\iff$ istnieje cykl zawierający *wszystkie* jego wierzchołki, każdy po 1 razie (z wyjątkiem początkowego i końcowego).

Przykład: (grafy hamiltonowskie)



Przykład: (grafy niehamiltonowskie)



Grafy hamiltonowskie

Problem komiwojażera:

— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu



Grafy hamiltonowskie

Problem komiwojażera:

— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze



Grafy hamiltonowskie

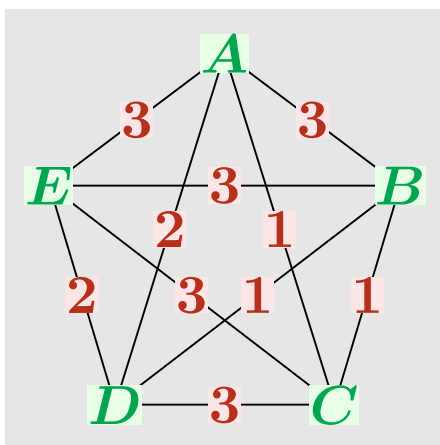
Problem komiwojażera:

— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze

Przykład:



Grafy hamiltonowskie

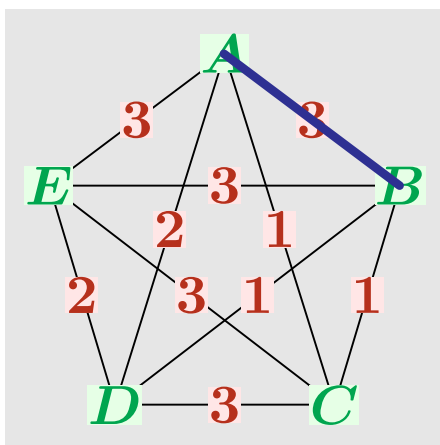
Problem komiwojażera:

— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze

Przykład:



Grafy hamiltonowskie

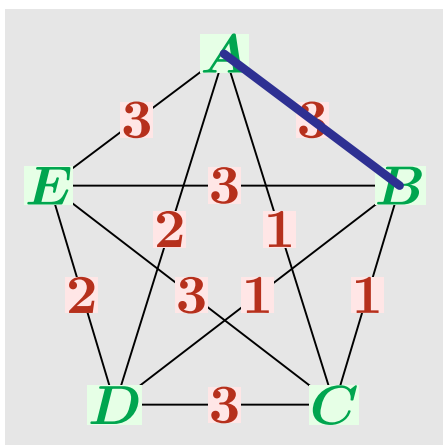
Problem komiwojażera:

— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze

Przykład:



droga wagi 3



Grafy hamiltonowskie

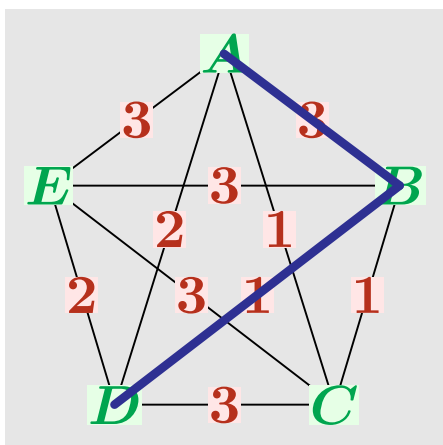
Problem komiwojażera:

— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze

Przykład:



droga wagi 10



Grafy hamiltonowskie

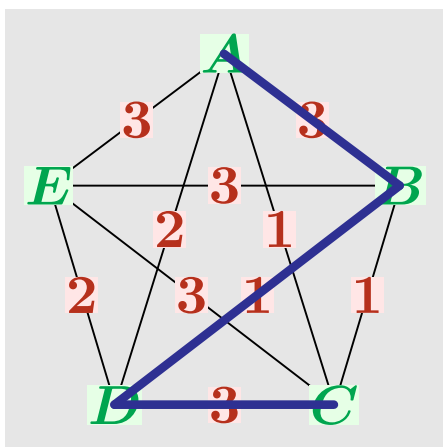
Problem komiwojażera:

— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze

Przykład:



droga wagi 12



Grafy hamiltonowskie

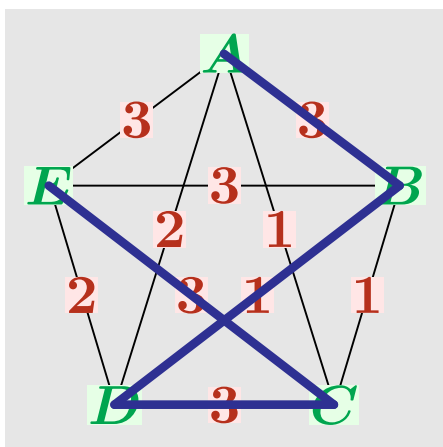
Problem komiwojażera:

— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze

Przykład:



droga wagi 10



Grafy hamiltonowskie

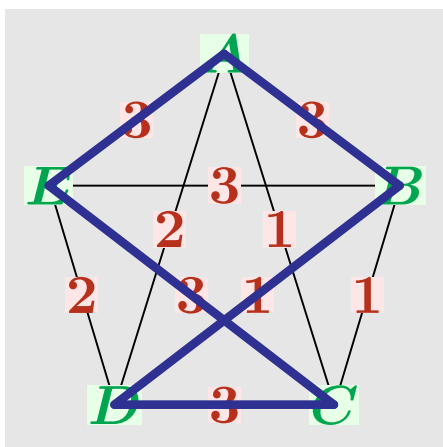
Problem komiwojażera:

— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze

Przykład:



droga wagi 13



Grafy hamiltonowskie

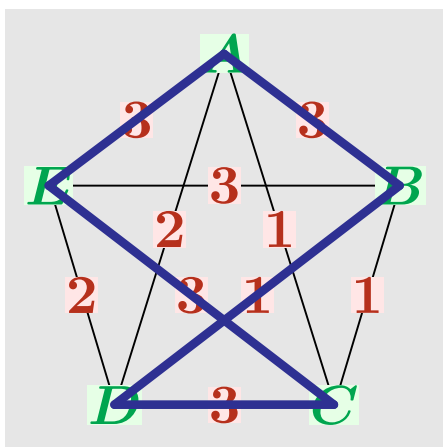
Problem komiwojażera:

— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze

Przykład:



cykl wagi 13



Grafy hamiltonowskie

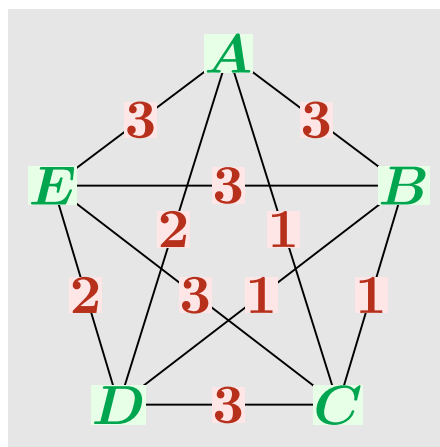
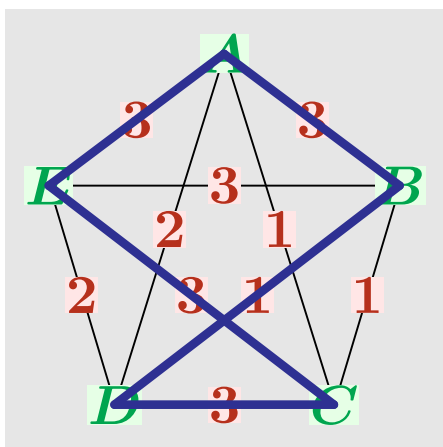
Problem komiwojażera:

— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze

Przykład:



cykl wagi 13



Grafy hamiltonowskie

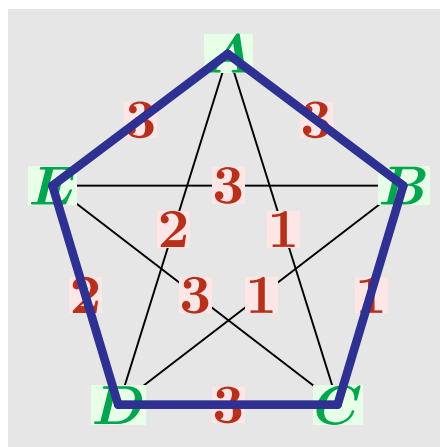
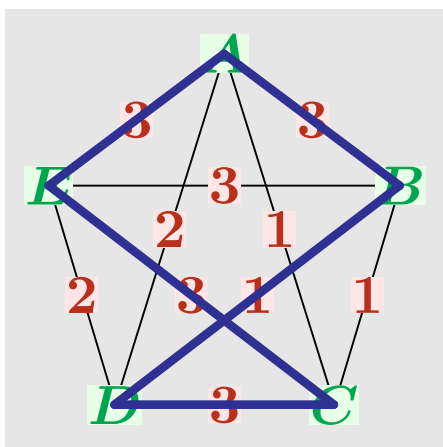
Problem komiwojażera:

— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze

Przykład:



cykl wagi 13



Grafy hamiltonowskie

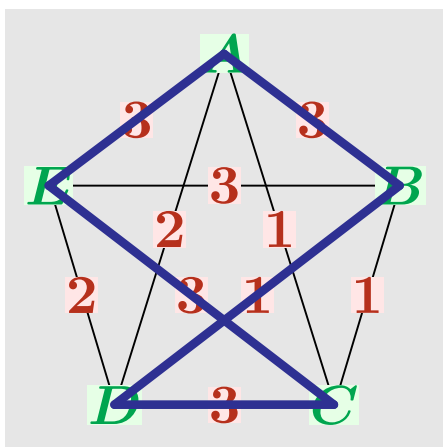
Problem komiwojażera:

— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

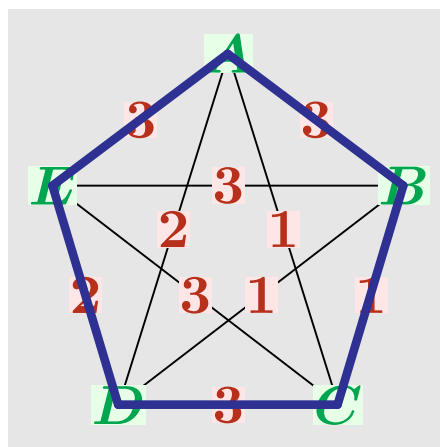
to znaczy:

— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze

Przykład:



cykl wagi 13



cykl wagi 12



Grafy hamiltonowskie

Problem komiwojażera:

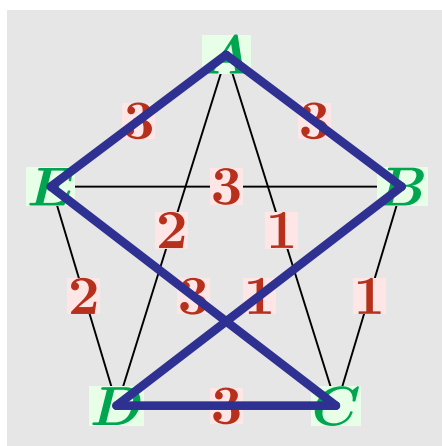
— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

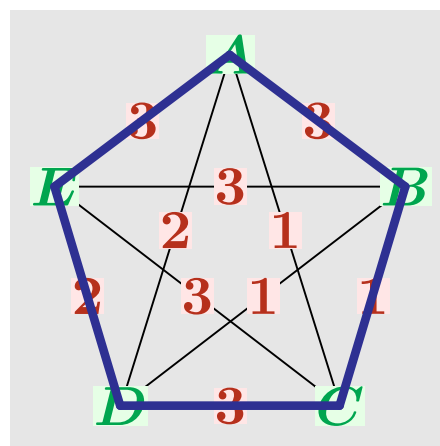
— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze



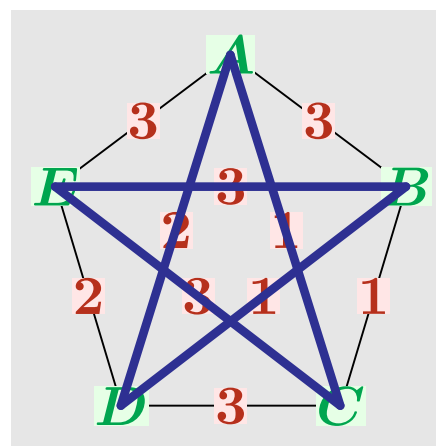
Przykład:



cykl wagi 13



cykl wagi 12



Grafy hamiltonowskie

Problem komiwojażera:

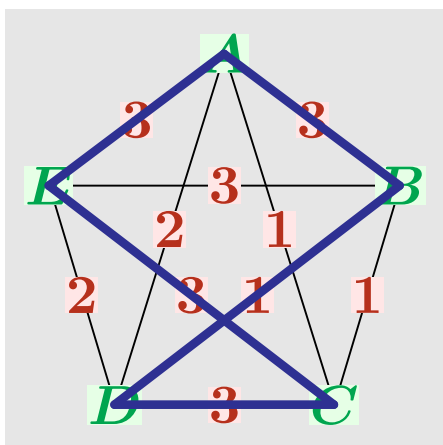
— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

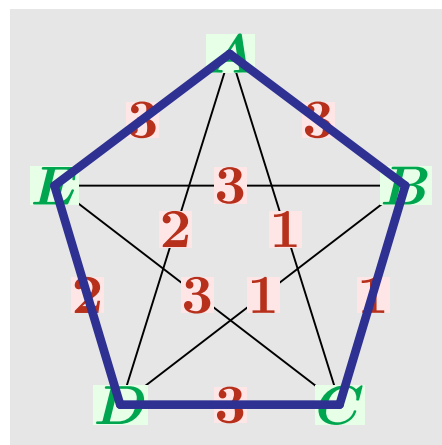
— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze



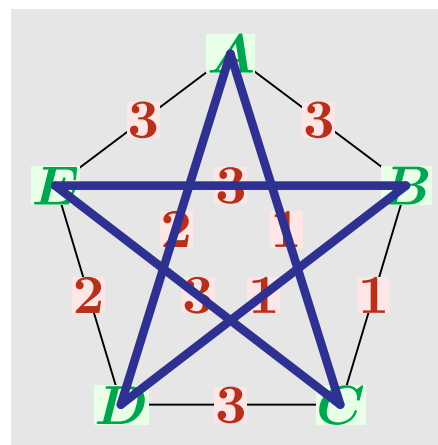
Przykład:



cykl wagi 13



cykl wagi 12



cykl wagi 10

Grafy hamiltonowskie

Problem komiwojażera:

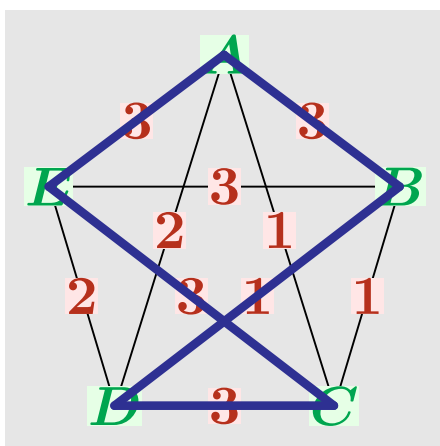
— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

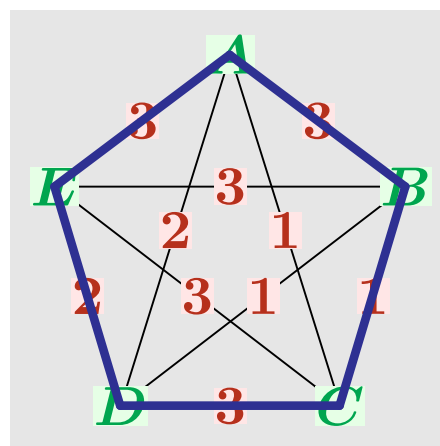
— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze



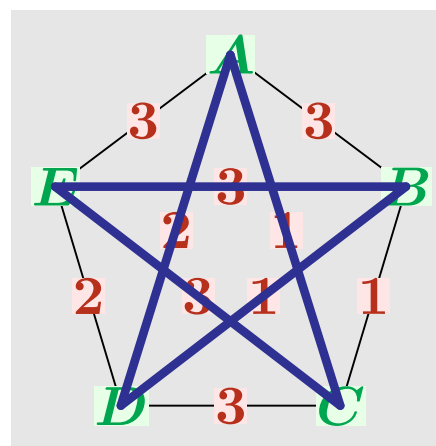
Przykład:



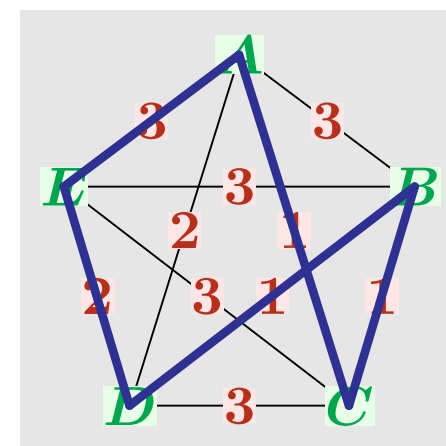
cykl wagi 13



cykl wagi 12



cykl wagi 10



Grafy hamiltonowskie

Problem komiwojażera:

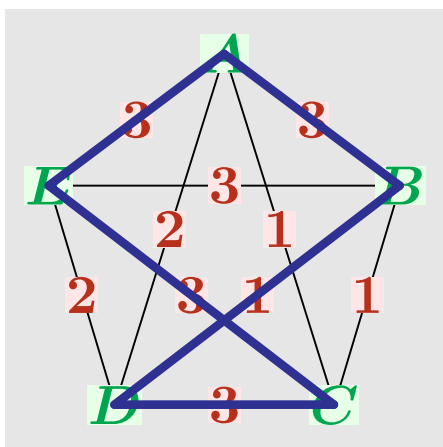
— znaleźć najkrótszą drogę odwiedzającą wszystkich odbiorców i powracającą do punktu startu

to znaczy:

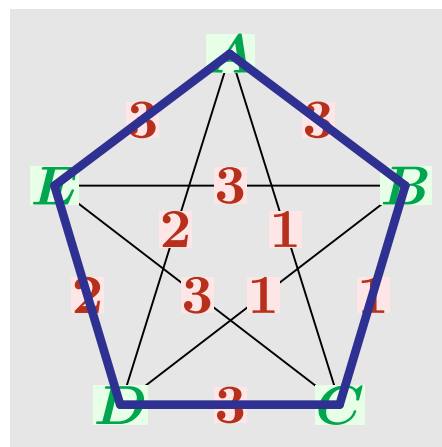
— w grafie *ważonym* znaleźć cykl Hamiltona o możliwie niskiej wadze



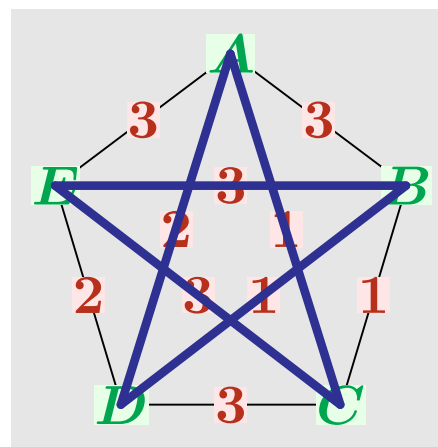
Przykład:



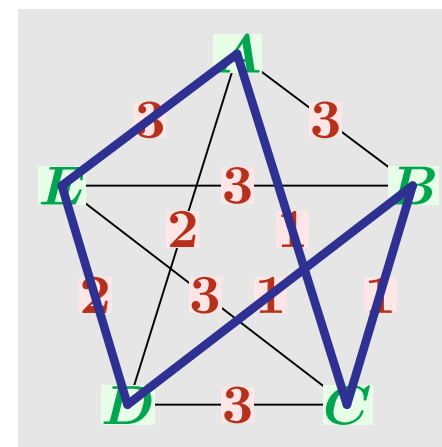
cykl wagi 13



cykl wagi 12



cykl wagi 10



cykl wagi 8

Grafy — kolorowanie krawędzi

Definicja:

Graf $G = \langle V, E \rangle$ jest **krawędziowo k -kolorowalny** ($k \in \mathbb{N}$) $\iff$ istnieje funkcja $f : E \rightarrow \{1, 2, \dots, k\}$ (krawędzie w zbiór k „kolorów”) taka, że

$$\forall_{e_1, e_2 \in E} \quad e_1 \cap e_2 \neq \emptyset \Rightarrow f(e_1) \neq f(e_2)$$

Grafy — kolorowanie krawędzi

Definicja:

Graf $G = \langle V, E \rangle$ jest **krawędziowo k -kolorowalny** ($k \in \mathbb{N}$) $\iff$ istnieje funkcja $f : E \rightarrow \{1, 2, \dots, k\}$ (krawędzie w zbiór k „kolorów”) taka, że

$$\forall_{e_1, e_2 \in E} \quad e_1 \cap e_2 \neq \emptyset \Rightarrow f(e_1) \neq f(e_2)$$

czyli każdym dwóm sąsiednim krawędziom przypisuje inny kolor.

Grafy — kolorowanie krawędzi

Definicja:

Graf $G = \langle V, E \rangle$ jest **krawędziowo k -kolorowalny** ($k \in \mathbb{N}$) $\iff$ istnieje funkcja $f : E \rightarrow \{1, 2, \dots, k\}$ (krawędzie w zbiór k „kolorów”) taka, że

$$\forall e_1, e_2 \in E \quad e_1 \cap e_2 \neq \emptyset \Rightarrow f(e_1) \neq f(e_2)$$

czyli każdym dwóm sąsiednim krawędziom przypisuje inny kolor.

Definicja:

Indeks chromatyczny grafu $\chi_e(G)$ —
najmniejsza liczba k taka, że graf G jest krawędziowo k -kolorowalny.

Grafy — kolorowanie krawędzi

Definicja:

Graf $G = \langle V, E \rangle$ jest **krawędziowo k -kolorowalny** ($k \in \mathbb{N}$) $\iff$ istnieje funkcja $f : E \rightarrow \{1, 2, \dots, k\}$ (krawędzie w zbiór k „kolorów”) taka, że

$$\forall e_1, e_2 \in E \quad e_1 \cap e_2 \neq \emptyset \Rightarrow f(e_1) \neq f(e_2)$$

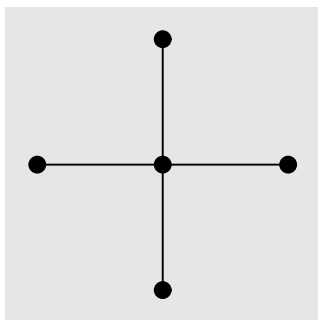
czyli każdym dwóm sąsiednim krawędziom przypisuje inny kolor.

Definicja:

Indeks chromatyczny grafu $\chi_e(G)$ —
najmniejsza liczba k taka, że graf G jest krawędziowo k -kolorowalny.

Przykład:

G_0 :



Grafy — kolorowanie krawędzi

Definicja:

Graf $G = \langle V, E \rangle$ jest **krawędziowo k -kolorowalny** ($k \in \mathbb{N}$) $\iff$ istnieje funkcja $f : E \rightarrow \{1, 2, \dots, k\}$ (krawędzie w zbiór k „kolorów”) taka, że

$$\forall e_1, e_2 \in E \quad e_1 \cap e_2 \neq \emptyset \Rightarrow f(e_1) \neq f(e_2)$$

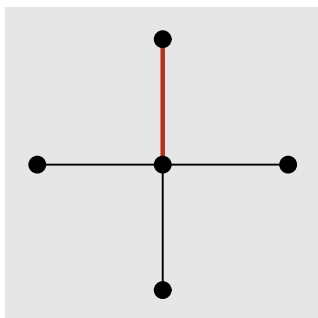
czyli każdym dwóm sąsiednim krawędziom przypisuje inny kolor.

Definicja:

Indeks chromatyczny grafu $\chi_e(G)$ —
najmniejsza liczba k taka, że graf G jest krawędziowo k -kolorowalny.

Przykład:

G_0 :



Grafy — kolorowanie krawędzi

Definicja:

Graf $G = \langle V, E \rangle$ jest **krawędziowo k -kolorowalny** ($k \in \mathbb{N}$) $\iff$ istnieje funkcja $f : E \rightarrow \{1, 2, \dots, k\}$ (krawędzie w zbiór k „kolorów”) taka, że

$$\forall e_1, e_2 \in E \quad e_1 \cap e_2 \neq \emptyset \Rightarrow f(e_1) \neq f(e_2)$$

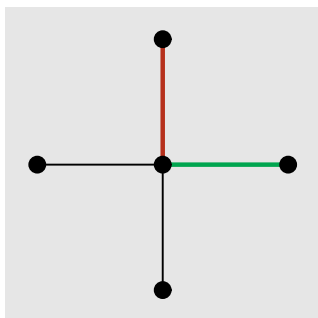
czyli każdym dwóm sąsiednim krawędziom przypisuje inny kolor.

Definicja:

Indeks chromatyczny grafu $\chi_e(G)$ —
najmniejsza liczba k taka, że graf G jest krawędziowo k -kolorowalny.

Przykład:

G_0 :



Grafy — kolorowanie krawędzi

Definicja:

Graf $G = \langle V, E \rangle$ jest **krawędziowo k -kolorowalny** ($k \in \mathbb{N}$) $\iff$ istnieje funkcja $f : E \rightarrow \{1, 2, \dots, k\}$ (krawędzie w zbiór k „kolorów”) taka, że

$$\forall e_1, e_2 \in E \quad e_1 \cap e_2 \neq \emptyset \Rightarrow f(e_1) \neq f(e_2)$$

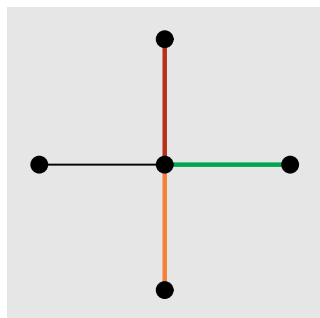
czyli każdym dwóm sąsiednim krawędziom przypisuje inny kolor.

Definicja:

Indeks chromatyczny grafu $\chi_e(G)$ —
najmniejsza liczba k taka, że graf G jest krawędziowo k -kolorowalny.

Przykład:

G_0 :



Grafy — kolorowanie krawędzi

Definicja:

Graf $G = \langle V, E \rangle$ jest **krawędziowo k -kolorowalny** ($k \in \mathbb{N}$) $\iff$ istnieje funkcja $f : E \rightarrow \{1, 2, \dots, k\}$ (krawędzie w zbiór k „kolorów”) taka, że

$$\forall e_1, e_2 \in E \quad e_1 \cap e_2 \neq \emptyset \Rightarrow f(e_1) \neq f(e_2)$$

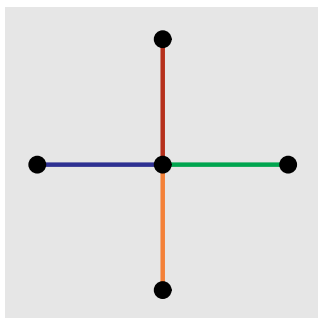
czyli każdym dwóm sąsiednim krawędziom przypisuje inny kolor.

Definicja:

Indeks chromatyczny grafu $\chi_e(G)$ —
najmniejsza liczba k taka, że graf G jest krawędziowo k -kolorowalny.

Przykład:

G_0 :



Grafy — kolorowanie krawędzi

Definicja:

Graf $G = \langle V, E \rangle$ jest **krawędziowo k -kolorowalny** ($k \in \mathbb{N}$) $\iff$ istnieje funkcja $f : E \rightarrow \{1, 2, \dots, k\}$ (krawędzie w zbiór k „kolorów”) taka, że

$$\forall e_1, e_2 \in E \quad e_1 \cap e_2 \neq \emptyset \Rightarrow f(e_1) \neq f(e_2)$$

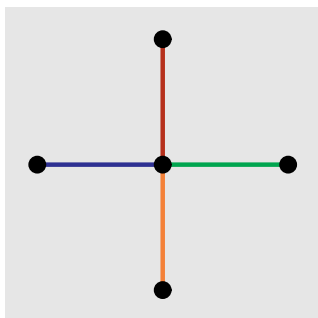
czyli każdym dwóm sąsiednim krawędziom przypisuje inny kolor.

Definicja:

Indeks chromatyczny grafu $\chi_e(G)$ —
najmniejsza liczba k taka, że graf G jest krawędziowo k -kolorowalny.

Przykład:

G_0 :



$$\chi_e(G_0) = 4$$

Grafy — kolorowanie krawędzi

Definicja:

Graf $G = \langle V, E \rangle$ jest **krawędziowo k -kolorowalny** ($k \in \mathbb{N}$) $\iff$ istnieje funkcja $f : E \rightarrow \{1, 2, \dots, k\}$ (krawędzie w zbiór k „kolorów”) taka, że

$$\forall e_1, e_2 \in E \quad e_1 \cap e_2 \neq \emptyset \Rightarrow f(e_1) \neq f(e_2)$$

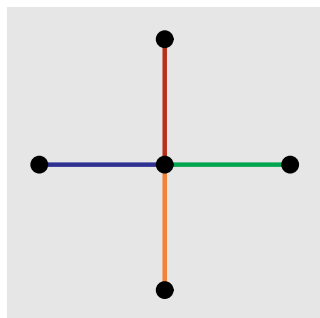
czyli każdym dwóm sąsiednim krawędziom przypisuje inny kolor.

Definicja:

Indeks chromatyczny grafu $\chi_e(G)$ — najmniejsza liczba k taka, że graf G jest krawędziowo k -kolorowalny.

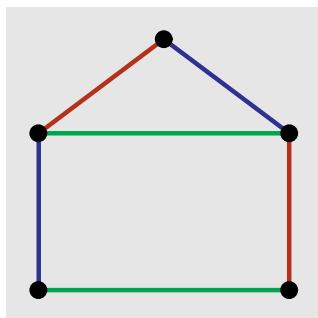
Przykład:

G_0 :



$$\chi_e(G_0) = 4$$

G_1 :



Grafy — kolorowanie krawędzi

Definicja:

Graf $G = \langle V, E \rangle$ jest **krawędziowo k -kolorowalny** ($k \in \mathbb{N}$) $\iff$ istnieje funkcja $f : E \rightarrow \{1, 2, \dots, k\}$ (krawędzie w zbiór k „kolorów”) taka, że

$$\forall e_1, e_2 \in E \quad e_1 \cap e_2 \neq \emptyset \Rightarrow f(e_1) \neq f(e_2)$$

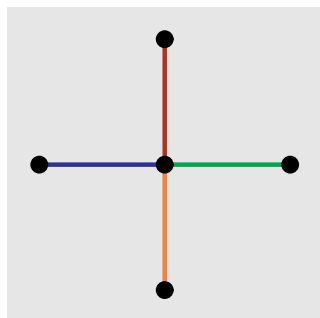
czyli każdym dwóm sąsiednim krawędziom przypisuje inny kolor.

Definicja:

Indeks chromatyczny grafu $\chi_e(G)$ — najmniejsza liczba k taka, że graf G jest krawędziowo k -kolorowalny.

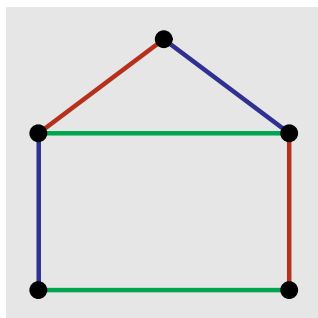
Przykład:

G_0 :



$$\chi_e(G_0) = 4$$

G_1 :



$$\chi_e(G_1) = 3$$

Grafy — kolorowanie krawędzi

Definicja:

Graf $G = \langle V, E \rangle$ jest **krawędziowo k -kolorowalny** ($k \in \mathbb{N}$) $\iff$ istnieje funkcja $f : E \rightarrow \{1, 2, \dots, k\}$ (krawędzie w zbiór k „kolorów”) taka, że

$$\forall e_1, e_2 \in E \quad e_1 \cap e_2 \neq \emptyset \Rightarrow f(e_1) \neq f(e_2)$$

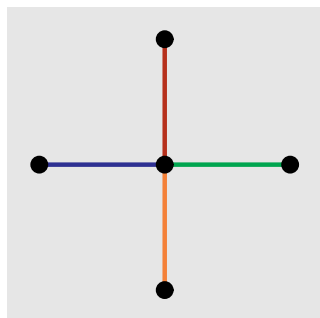
czyli każdym dwóm sąsiednim krawędziom przypisuje inny kolor.

Definicja:

Indeks chromatyczny grafu $\chi_e(G)$ — najmniejsza liczba k taka, że graf G jest krawędziowo k -kolorowalny.

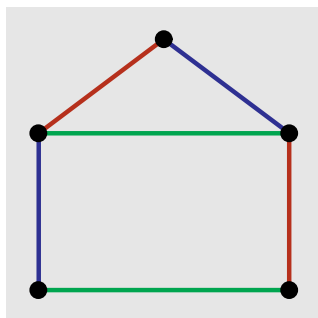
Przykład:

G_0 :



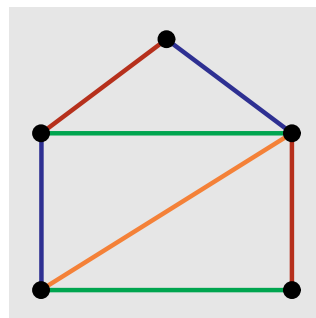
$$\chi_e(G_0) = 4$$

G_1 :



$$\chi_e(G_1) = 3$$

G_2 :



$$\chi_e(G_2) = 4$$

Grafy — kolorowanie krawędzi

Definicja:

Graf $G = \langle V, E \rangle$ jest **krawędziowo k -kolorowalny** ($k \in \mathbb{N}$) $\iff$ istnieje funkcja $f : E \rightarrow \{1, 2, \dots, k\}$ (krawędzie w zbiór k „kolorów”) taka, że

$$\forall e_1, e_2 \in E \quad e_1 \cap e_2 \neq \emptyset \Rightarrow f(e_1) \neq f(e_2)$$

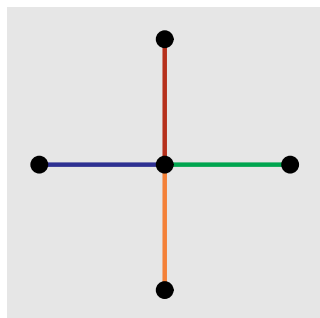
czyli każdym dwóm sąsiednim krawędziom przypisuje inny kolor.

Definicja:

Indeks chromatyczny grafu $\chi_e(G)$ — najmniejsza liczba k taka, że graf G jest krawędziowo k -kolorowalny.

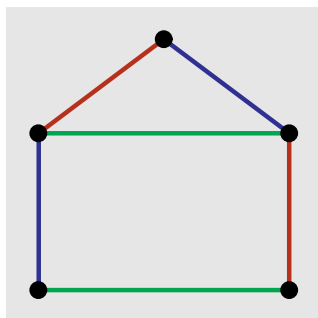
Przykład:

G_0 :



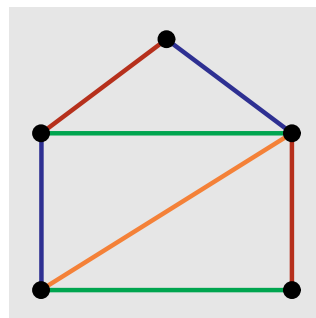
$$\chi_e(G_0) = 4$$

G_1 :



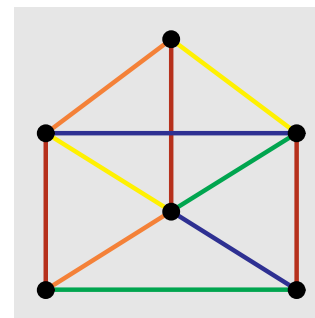
$$\chi_e(G_1) = 3$$

G_2 :



$$\chi_e(G_2) = 4$$

G_3 :



$$\chi_e(G_3) = 5$$

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n - 1 & \text{jeśli } 2 \mid n \end{cases}$$

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n - 1 & \text{jeśli } 2 \mid n \end{cases}$$

Dowód: (szkic)

Przypadek $2 \nmid n$

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n - 1 & \text{jeśli } 2 \mid n \end{cases}$$

Dowód: (szkic)

Przypadek $2 \nmid n$:

Dla $n = 3$ — oczywiste.

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n - 1 & \text{jeśli } 2 \mid n \end{cases}$$

Dowód: (szkic)

Przypadek $2 \nmid n$:

Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:
rozpatrzmy n -kąt foremny

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n - 1 & \text{jeśli } 2 \mid n \end{cases}$$

Dowód: (szkic)

Przypadek $2 \nmid n$:

Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku.

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

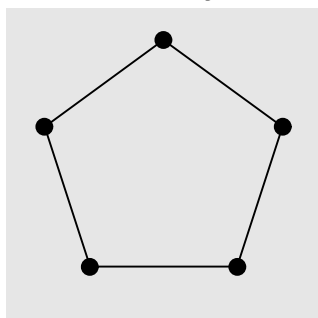
Dowód: (szkic)

Przypadek $2 \nmid n$:

Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n - 1 & \text{jeśli } 2 \mid n \end{cases}$$

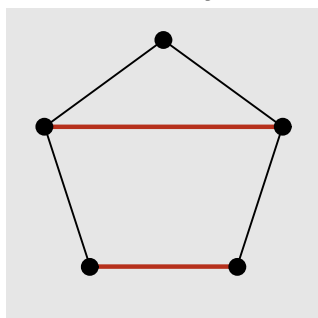
Dowód: (szkic)

Przypadek $2 \nmid n$:

Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

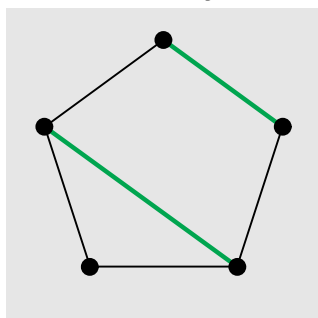
Dowód: (szkic)

Przypadek $2 \nmid n$:

Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

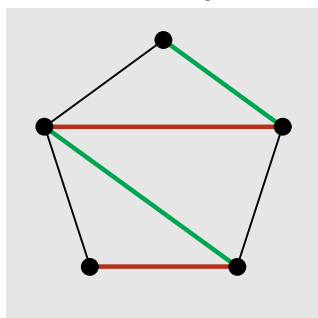
Dowód: (szkic)

Przypadek $2 \nmid n$:

Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

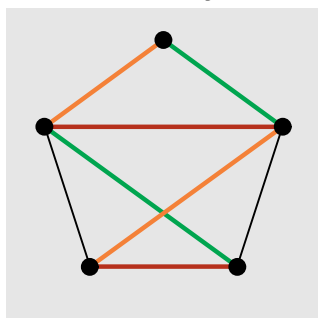
Dowód: (szkic)

Przypadek $2 \nmid n$:

Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

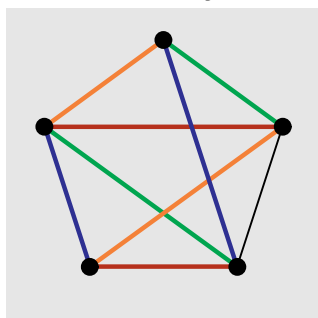
Dowód: (szkic)

Przypadek $2 \nmid n$:

Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

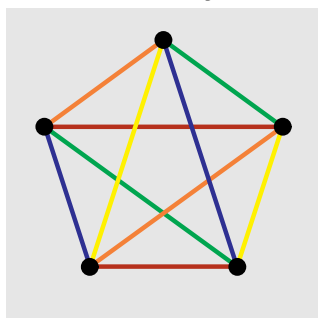
Dowód: (szkic)

Przypadek $2 \nmid n$:

Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

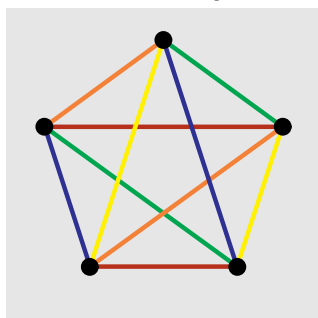
Dowód: (szkic)

Przypadek $2 \nmid n$:

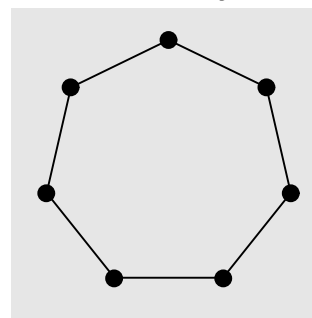
Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



dla 7-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

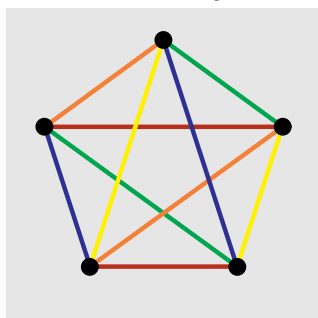
Dowód: (szkic)

Przypadek $2 \nmid n$:

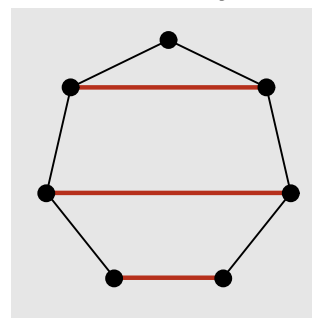
Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



dla 7-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

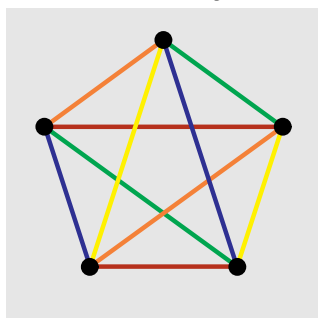
Dowód: (szkic)

Przypadek $2 \nmid n$:

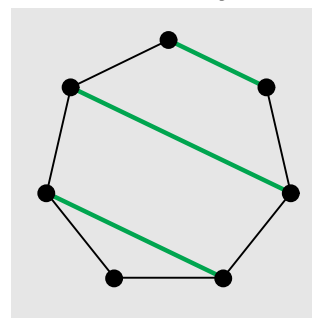
Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



dla 7-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

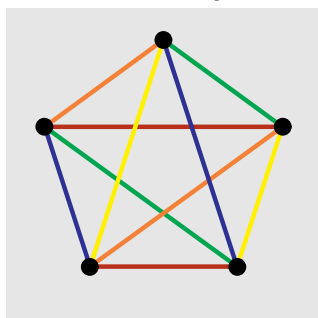
Dowód: (szkic)

Przypadek $2 \nmid n$:

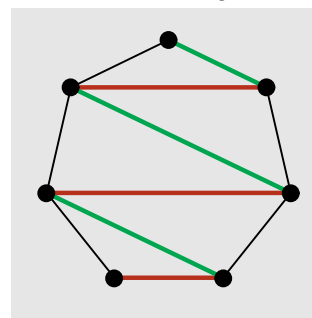
Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



dla 7-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

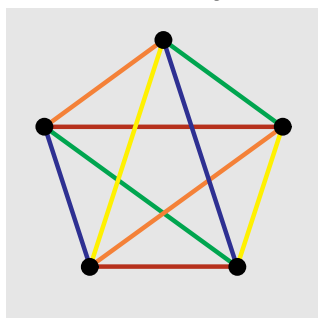
Dowód: (szkic)

Przypadek $2 \nmid n$:

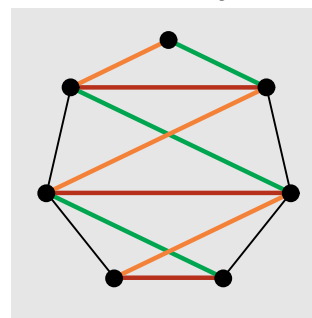
Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



dla 7-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

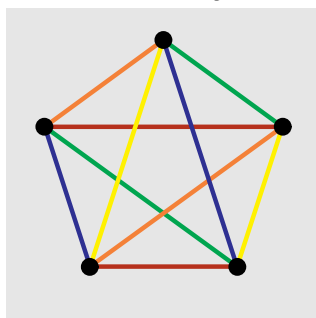
Dowód: (szkic)

Przypadek $2 \nmid n$:

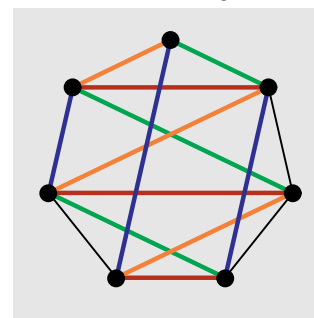
Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



dla 7-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

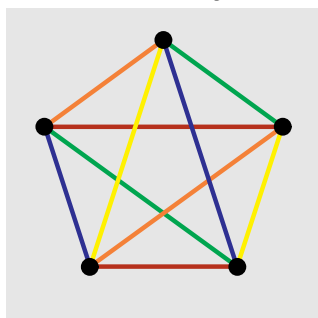
Dowód: (szkic)

Przypadek $2 \nmid n$:

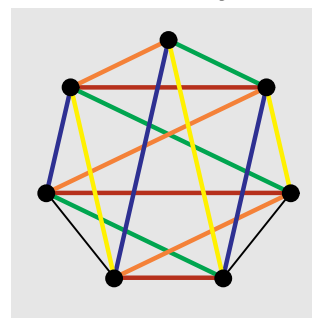
Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



dla 7-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

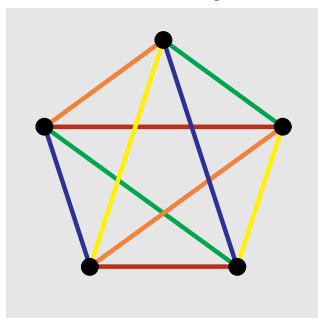
Dowód: (szkic)

Przypadek $2 \nmid n$:

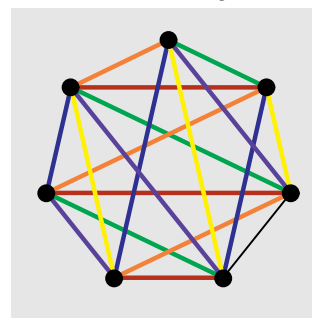
Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



dla 7-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

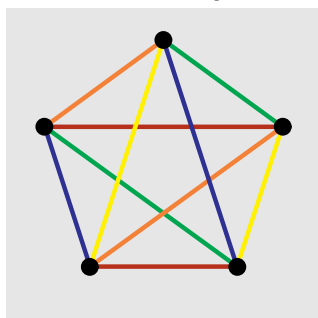
Dowód: (szkic)

Przypadek $2 \nmid n$:

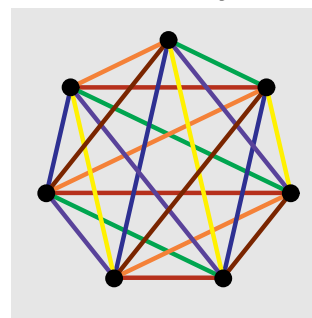
Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



dla 7-kąta:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

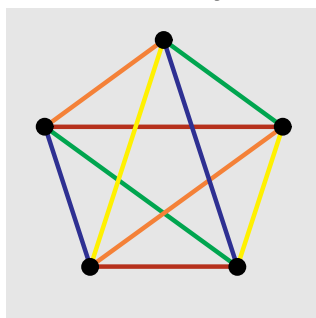
Dowód: (szkic)

Przypadek $2 \nmid n$:

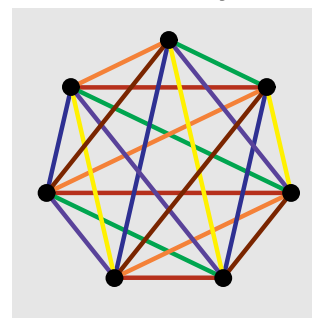
Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



dla 7-kąta:



Wszystkie równoległe krawędzie należy pomalować tym samym kolorem.

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

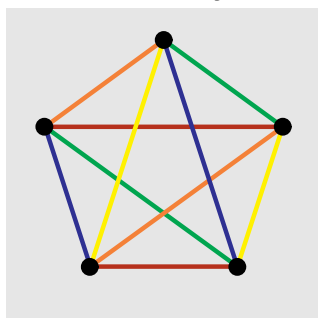
Dowód: (szkic)

Przypadek $2 \nmid n$:

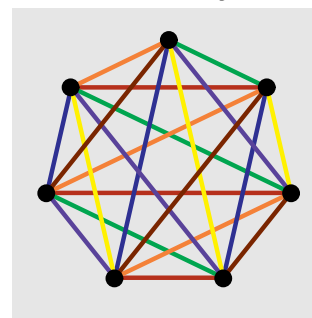
Dla $n = 3$ — oczywiste. Dalej dla $n \geq 5$:

rozpatrzmy n -kąt foremny; każda jego przekątna jest równoległa do któregoś jego boku. Np.:

dla 5-kąta:



dla 7-kąta:



Wszystkie równoległe krawędzie należy pomalować tym samym kolorem. Kolorów będzie n .

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n - 1 & \text{jeśli } 2 \mid n \end{cases}$$

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n - 1 & \text{jeśli } 2 \mid n \end{cases}$$

Dowód: (szkic — dokończenie)

Przypadek $2 \mid n$

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n - 1 & \text{jeśli } 2 \mid n \end{cases}$$

Dowód: (szkic — dokończenie)

Przypadek $2 \mid n$:

Dla $n = 2$ — oczywiste. Dalej dla $n \geq 4$

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n - 1 & \text{jeśli } 2 \mid n \end{cases}$$

Dowód: (szkic — dokończenie)

Przypadek $2 \mid n$:

Dla $n = 2$ — oczywiste. Dalej dla $n \geq 4$:

rozpatrzmy ostrosłup, którego podstawą jest $(n-1)$ -kąt foremny

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

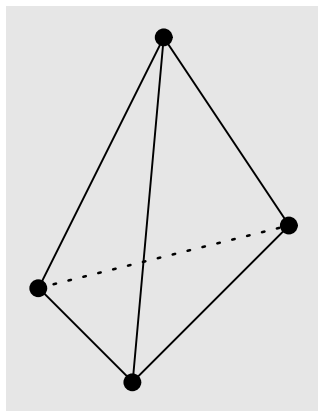
Dowód: (szkic — dokończenie)

Przypadek $2 \mid n$:

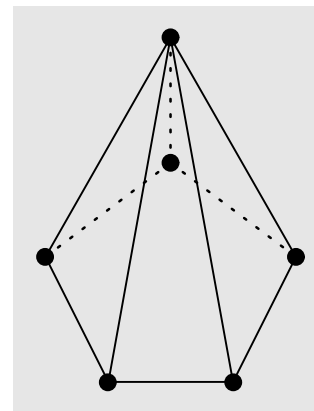
Dla $n = 2$ — oczywiste. Dalej dla $n \geq 4$:

rozpatrzmy ostrosłup, którego podstawą jest $(n-1)$ -kąt foremny:

dla $n = 4$:



dla $n = 6$:



Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

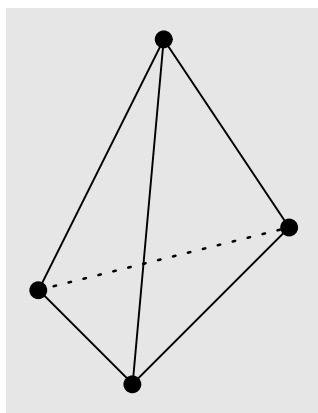
Dowód: (szkic — dokończenie)

Przypadek $2 \mid n$:

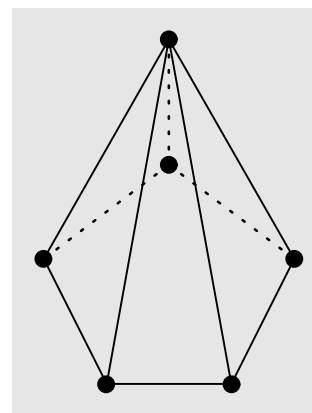
Dla $n = 2$ — oczywiste. Dalej dla $n \geq 4$:

rozpatrzmy ostrosłup, którego podstawą jest $(n-1)$ -kąt foremny:

dla $n = 4$:



dla $n = 6$:



krawędzie podstawy pomalowane są $n-1$ kolorami;

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

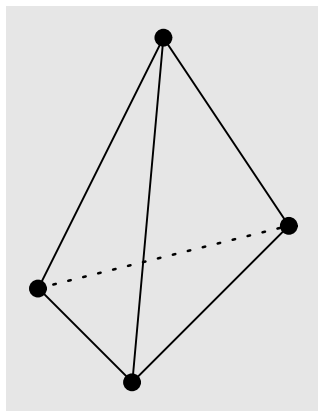
Dowód: (szkic — dokończenie)

Przypadek $2 \mid n$:

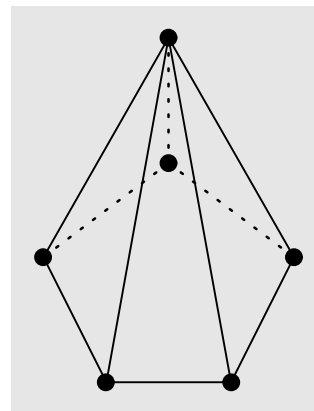
Dla $n = 2$ — oczywiste. Dalej dla $n \geq 4$:

rozpatrzmy ostrosłup, którego podstawą jest $(n-1)$ -kąt foremny:

dla $n = 4$:



dla $n = 6$:



krawędzie podstawy pomalowane są $n-1$ kolorami;

krawędziom wychodzącym z wierzchołków podstawy brakuje po jednym kolorze;

Grafy — kolorowanie krawędzi

K_n — graf pełny o n wierzchołkach

Twierdzenie:

$$\text{Jeśli } n \geq 2 \text{ to } \chi_e(K_n) = \begin{cases} n & \text{jeśli } 2 \nmid n \\ n-1 & \text{jeśli } 2 \mid n \end{cases}$$

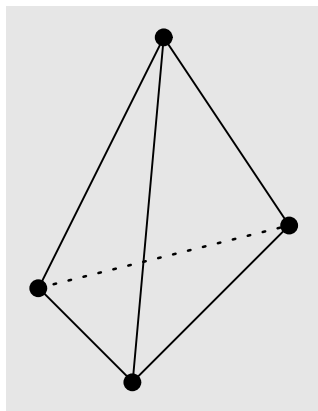
Dowód: (szkic — dokończenie)

Przypadek $2 \mid n$:

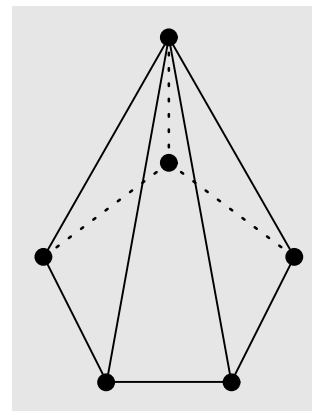
Dla $n = 2$ — oczywiste. Dalej dla $n \geq 4$:

rozpatrzmy ostrosłup, którego podstawą jest $(n-1)$ -kąt foremny:

dla $n = 4$:



dla $n = 6$:



krawędzie podstawy pomalowane są $n-1$ kolorami;

krawędziom wychodzącym z wierzchołków podstawy brakuje po jednym kolorze;
tym kolorem należy pomalować boczne krawędzie ostrosłupa.



Grafy skierowane

Z poprzedniego wykładu:

Definicja:

Graf (nieskierowany)

— para $G = \langle V, E \rangle$

- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy zbiór** wierzchołków.

Grafy skierowane

Z poprzedniego wykładu:

Definicja:

Graf (nieskierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy zbiór** wierzchołków.

Definicja: (nowa)
Graf (skierowany)

Grafy skierowane

Z poprzedniego wykładu:

Definicja:

Graf (nieskierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy zbiór** wierzchołków.

Definicja: (nowa)

Graf (skierowany)

— para $G = \langle V, E \rangle$

Grafy skierowane

Z poprzedniego wykładu:

Definicja:

Graf (nieskierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy zbiór** wierzchołków.

Definicja: (nowa)

Graf (skierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**

Grafy skierowane

Z poprzedniego wykładu:

Definicja:

Graf (nieskierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy zbiór** wierzchołków.

Definicja: (nowa)

Graf (skierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**

Grafy skierowane

Z poprzedniego wykładu:

Definicja:

Graf (nieskierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy zbiór** wierzchołków.

Definicja: (nowa)

Graf (skierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy ciąg** (czyli **para**) wierzchołków.

Grafy skierowane

Z poprzedniego wykładu:

Definicja:

Graf (nieskierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy zbiór** wierzchołków.

Definicja: (nowa)

Graf (skierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy ciąg** (czyli **para**) wierzchołków.

Krawędzie w grafie **skierowanym** to jakby ulice jednokierunkowe; krawędź (v_0, v_1)

Grafy skierowane

Z poprzedniego wykładu:

Definicja:

Graf (nieskierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy zbiór** wierzchołków.

Definicja: (nowa)

Graf (skierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy ciąg** (czyli **para**) wierzchołków.

Krawędzie w grafie **skierowanym** to jakby ulice jednokierunkowe; krawędź

$$(v_0, v_1) \text{ — } v_0 \longrightarrow v_1$$

Grafy skierowane

Z poprzedniego wykładu:

Definicja:

Graf (nieskierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy zbiór** wierzchołków.

Definicja: (nowa)

Graf (skierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy ciąg** (czyli **para**) wierzchołków.

Krawędzie w grafie **skierowanym** to jakby ulice jednokierunkowe; krawędź (v_0, v_1) — $v_0 \longrightarrow v_1$ — prowadzi od v_0 do v_1

Grafy skierowane

Z poprzedniego wykładu:

Definicja:

Graf (nieskierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy zbiór** wierzchołków.

Definicja: (nowa)

Graf (skierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy ciąg** (czyli **para**) wierzchołków.

Krawędzie w grafie **skierowanym** to jakby ulice jednokierunkowe; krawędź (v_0, v_1) — $v_0 \rightarrow v_1$ — prowadzi od v_0 do v_1

Dopuszczalne są *pętle*

(v, v)

Grafy skierowane

Z poprzedniego wykładu:

Definicja:

Graf (nieskierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy zbiór** wierzchołków.

Definicja: (nowa)

Graf (skierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy ciąg** (czyli **para**) wierzchołków.

Krawędzie w grafie **skierowanym** to jakby ulice jednokierunkowe; krawędź (v_0, v_1) — $v_0 \rightarrow v_1$ — prowadzi od v_0 do v_1

Dopuszczalne są *pętle*

(v, v) — 

Grafy skierowane

Z poprzedniego wykładu:

Definicja:

Graf (nieskierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy zbiór** wierzchołków.

Definicja: (nowa)

Graf (skierowany)

- para $G = \langle V, E \rangle$
- V — dowolny zbiór **wierzchołków**;
- E — zbiór **krawędzi**; pojedyncza krawędź to **dwuelementowy ciąg** (czyli **para**) wierzchołków.

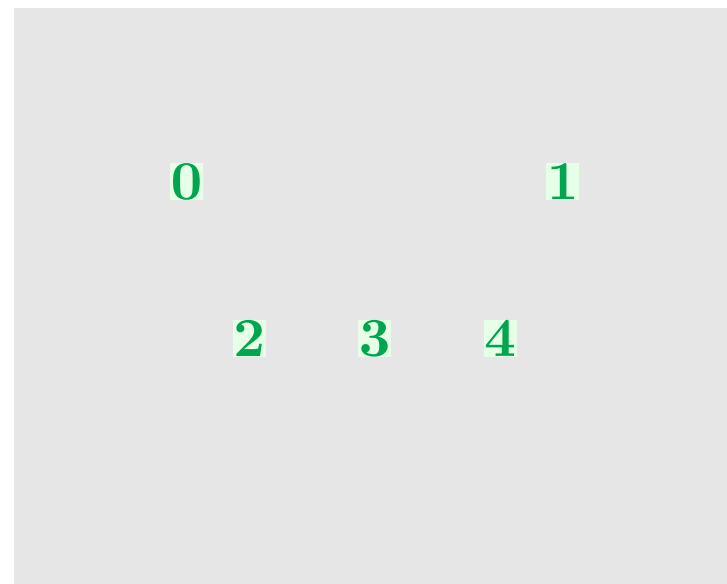
Krawędzie w grafie **skierowanym** to jakby ulice jednokierunkowe; krawędź (v_0, v_1) — $v_0 \rightarrow v_1$ — prowadzi od v_0 do v_1

Dopuszczalne są *pętle*

(v, v) —  — krawędzie z początkiem równym końcowi

Grafy skierowane

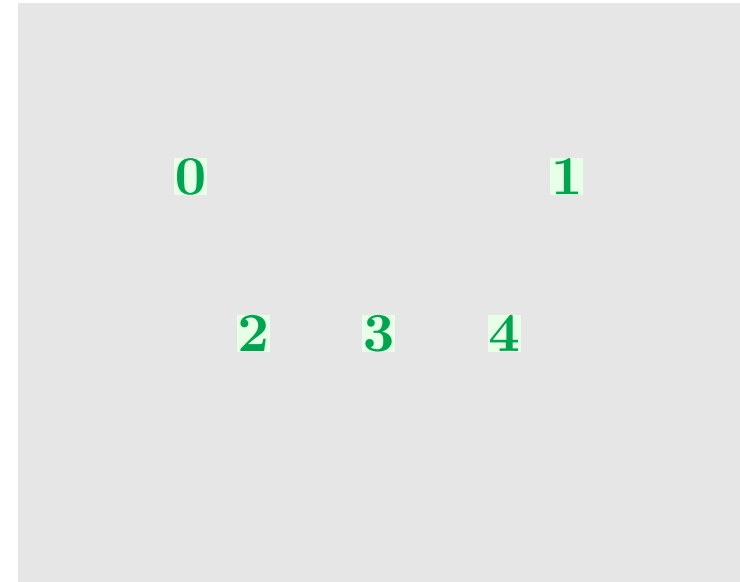
$$V = \{0, 1, 2, 3, 4\}$$



Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

$$E = \left\{ \right.$$

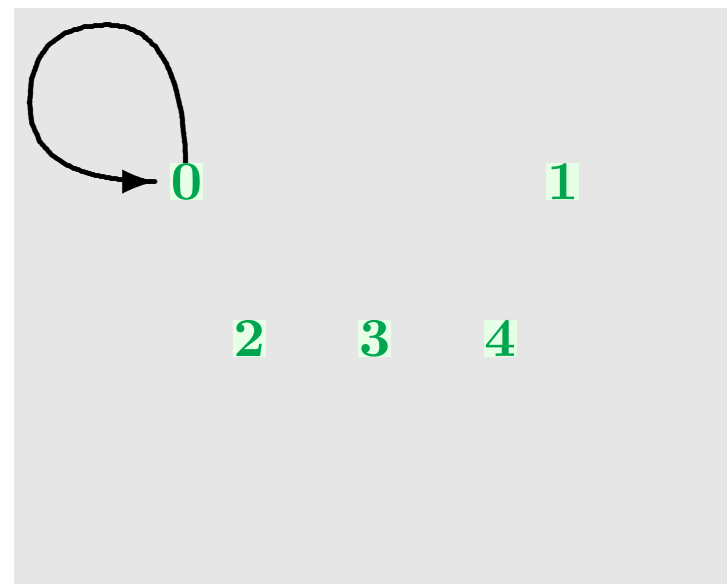


Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

$$E = \left\{ (0, 0) \right\}$$

}

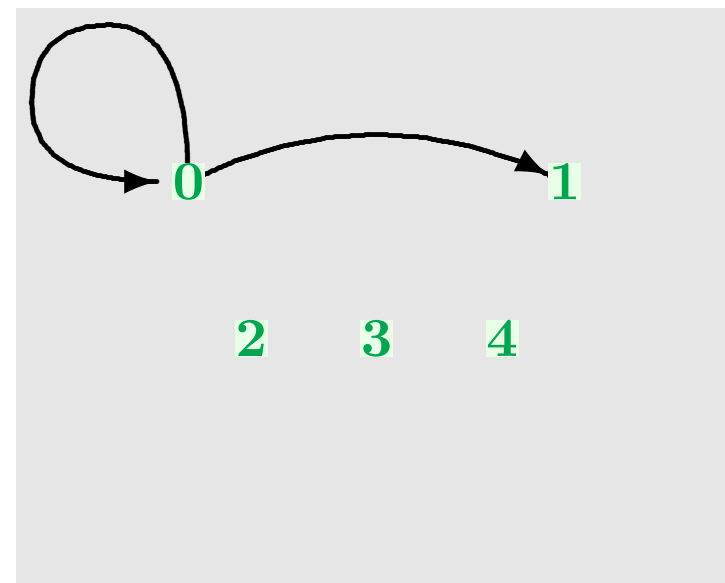


Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

$$E = \left\{ (0, 0), (0, 1) \right\}$$

}

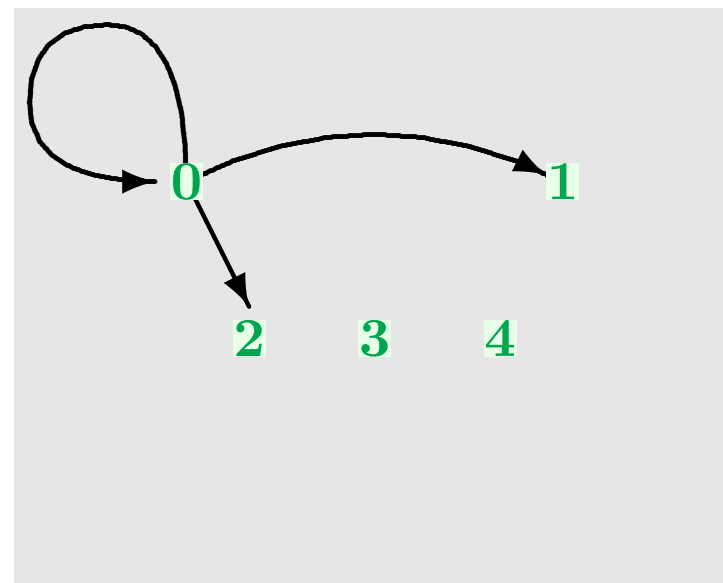


Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

$$E = \left\{ (0, 0), (0, 1), (0, 2) \right\}$$

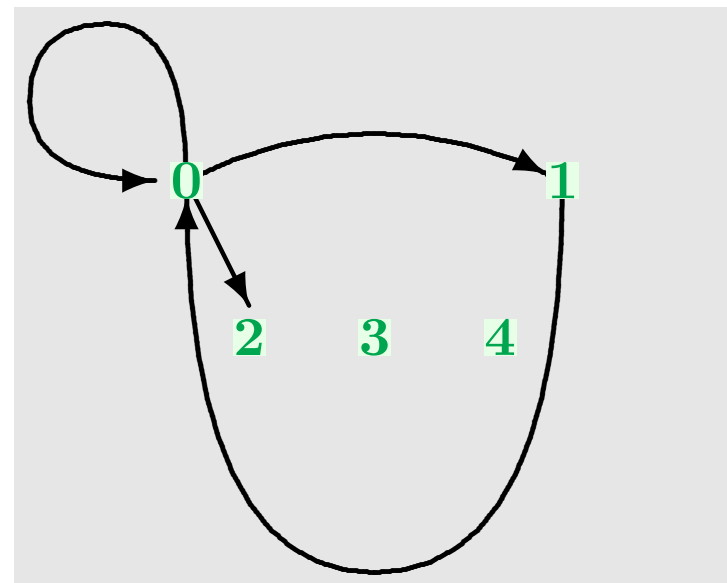
}



Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

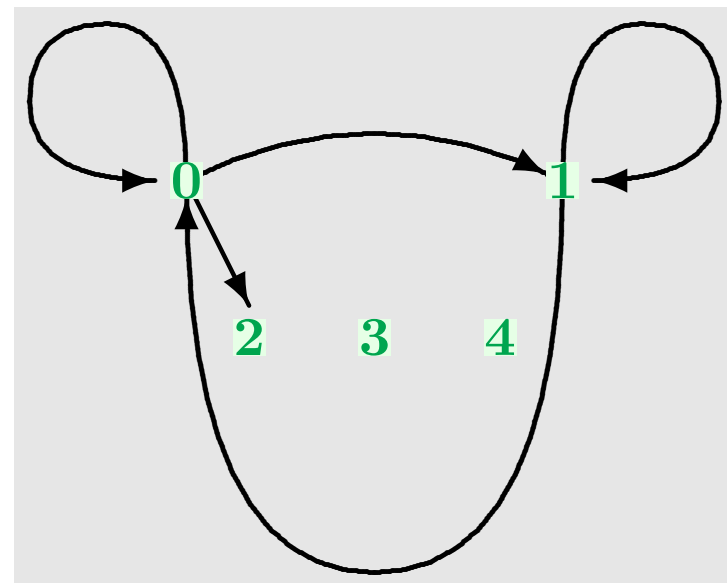
$$E = \left\{ (0, 0), (0, 1), (0, 2), (1, 0) \right\}$$



Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

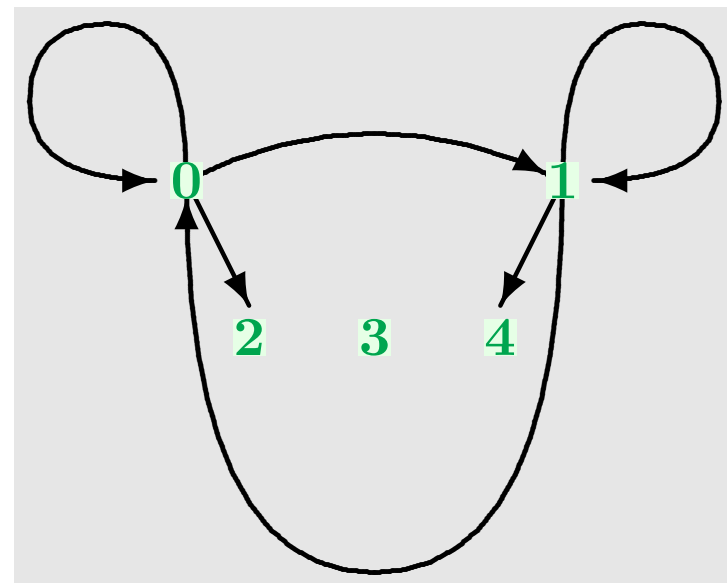
$$E = \left\{ (0, 0), (0, 1), (0, 2), (1, 0), (1, 1) \right\}$$



Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

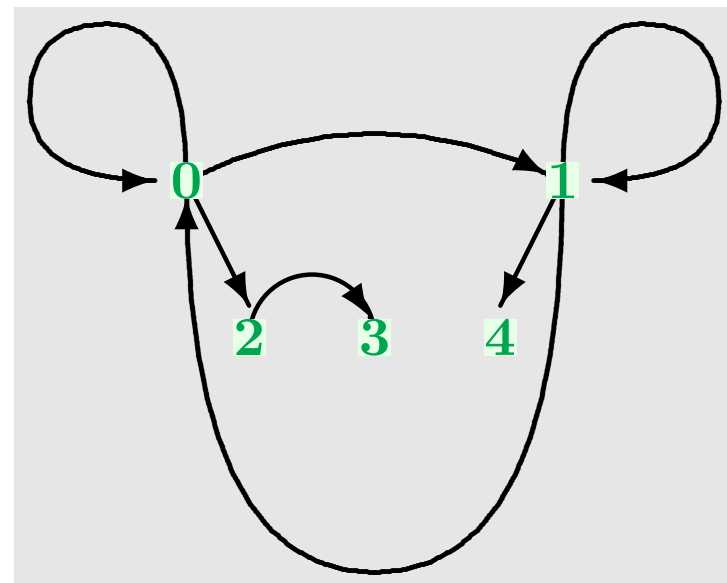
$$E = \left\{ (0, 0), (0, 1), (0, 2), (1, 0), (1, 1), \right. \\ \left. (1, 4) \right\}$$



Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

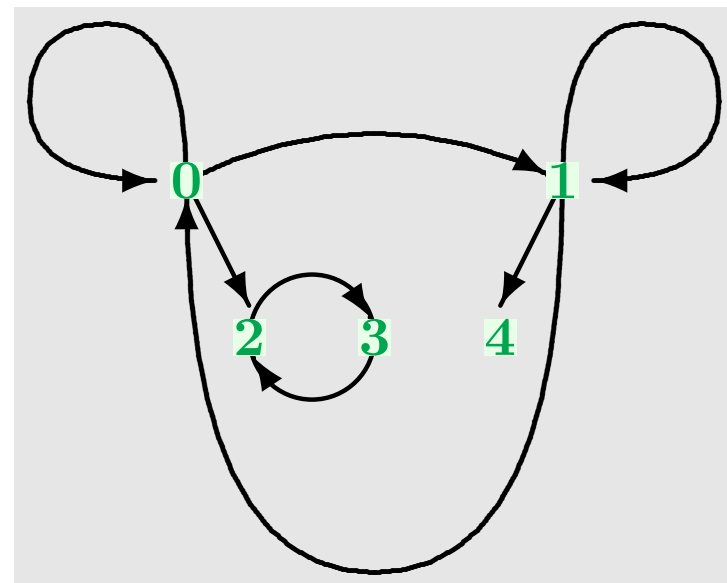
$$E = \left\{ (0, 0), (0, 1), (0, 2), (1, 0), (1, 1), \right. \\ \left. (1, 4), (2, 3) \right\}$$



Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

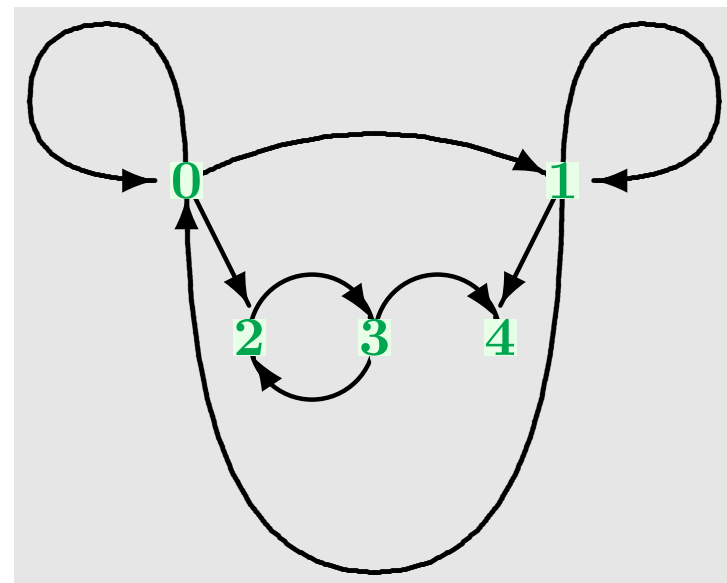
$$E = \left\{ (0, 0), (0, 1), (0, 2), (1, 0), (1, 1), \right. \\ \left. (1, 4), (2, 3), (3, 2) \right\}$$



Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

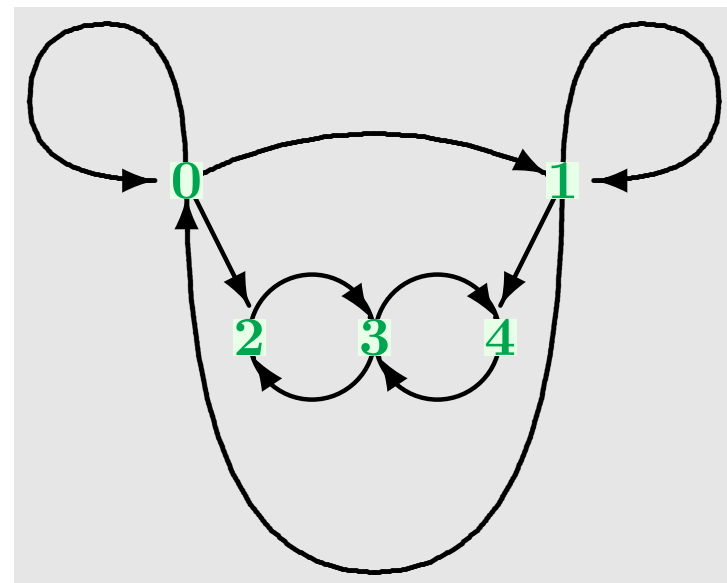
$$E = \left\{ (0, 0), (0, 1), (0, 2), (1, 0), (1, 1), \right. \\ \left. (1, 4), (2, 3), (3, 2), (3, 4) \right\}$$



Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

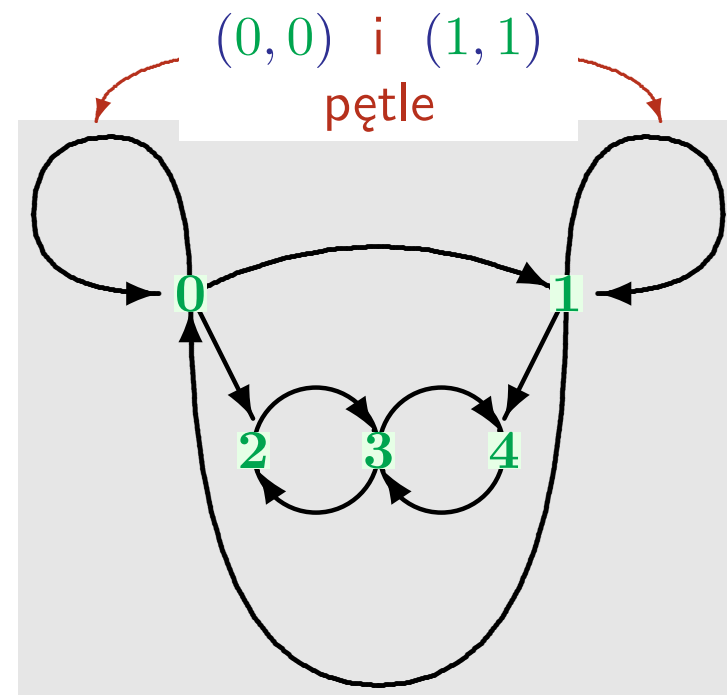
$$E = \left\{ (0, 0), (0, 1), (0, 2), (1, 0), (1, 1), \right. \\ \left. (1, 4), (2, 3), (3, 2), (3, 4), (4, 3) \right\}$$



Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

$$E = \left\{ \begin{array}{l} (0, 0), (0, 1), (0, 2), (1, 0), (1, 1), \\ (1, 4), (2, 3), (3, 2), (3, 4), (4, 3) \end{array} \right\}$$

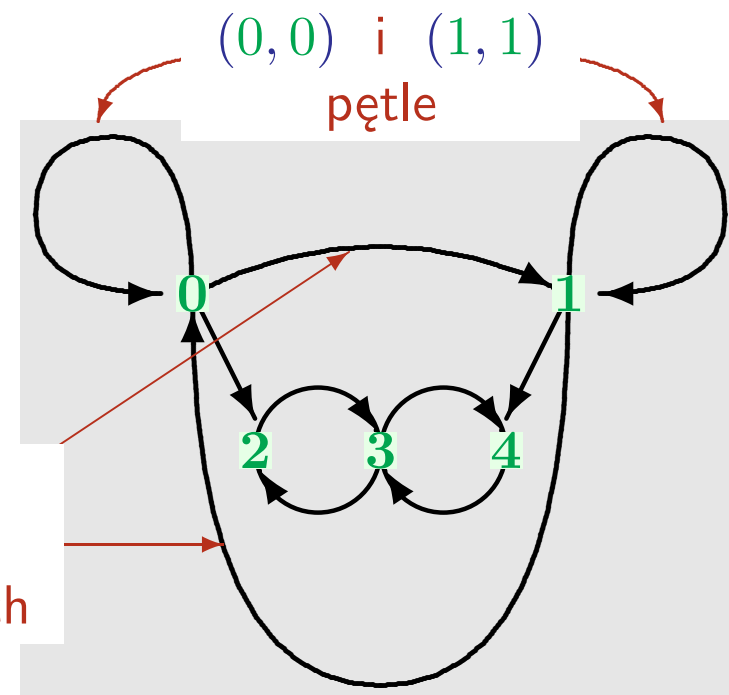


Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

$$E = \left\{ \begin{array}{l} (0, 0), (0, 1), (0, 2), (1, 0), (1, 1), \\ (1, 4), (2, 3), (3, 2), (3, 4), (4, 3) \end{array} \right\}$$

$(0, 1)$ i $(1, 0)$
— para krawędzi
przeciwnie skierowanych

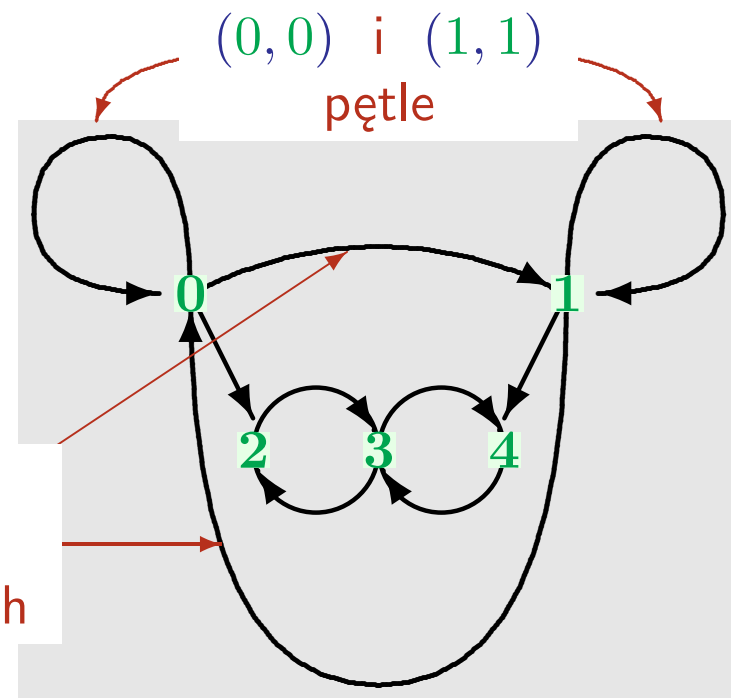


Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

$$E = \left\{ \begin{array}{l} (0, 0), (0, 1), (0, 2), (1, 0), (1, 1), \\ (1, 4), (2, 3), (3, 2), (3, 4), (4, 3) \end{array} \right\}$$

$(0, 1)$ i $(1, 0)$
— para krawędzi
przeciwnie skierowanych



Definicja: (wersje skierowane)

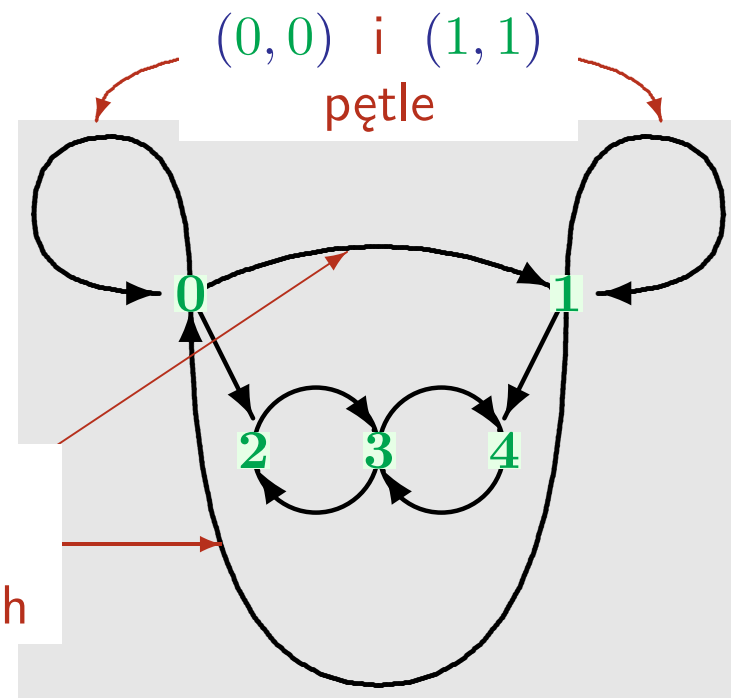
- **stopień wyjścia** wierzchołka $v \in V$ — $|\{u \in V \mid (v, u) \in E\}|$ (liczba wychodzących krawędzi)

Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

$$E = \left\{ \begin{array}{l} (0, 0), (0, 1), (0, 2), (1, 0), (1, 1), \\ (1, 4), (2, 3), (3, 2), (3, 4), (4, 3) \end{array} \right\}$$

$(0, 1)$ i $(1, 0)$
— para krawędzi
przeciwnie skierowanych



Definicja: (wersje skierowane)

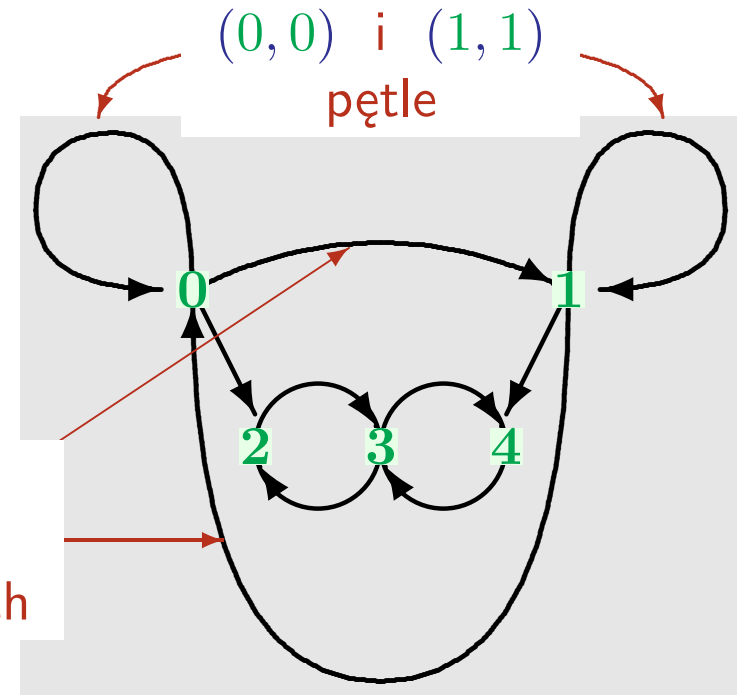
- **stopień wyjścia** wierzchołka $v \in V$ — $|\{u \in V \mid (v, u) \in E\}|$ (liczba wychodzących krawędzi)
- **stopień wejścia** wierzchołka $v \in V$ — $|\{u \in V \mid (u, v) \in E\}|$ (liczba wchodzących krawędzi)

Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

$$E = \left\{ \begin{array}{l} (0, 0), (0, 1), (0, 2), (1, 0), (1, 1), \\ (1, 4), (2, 3), (3, 2), (3, 4), (4, 3) \end{array} \right\}$$

$(0, 1)$ i $(1, 0)$
— para krawędzi
przeciwnie skierowanych



Definicja: (wersje skierowane)

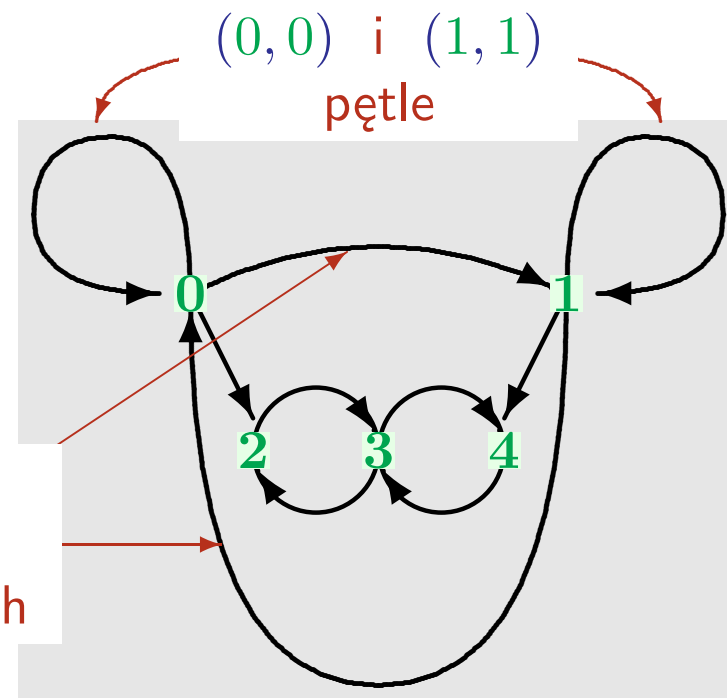
- **stopień wyjścia** wierzchołka $v \in V$ — $|\{u \in V \mid (v, u) \in E\}|$ (liczba wychodzących krawędzi)
- **stopień wejścia** wierzchołka $v \in V$ — $|\{u \in V \mid (u, v) \in E\}|$ (liczba wchodzących krawędzi)
- **droga** — ciąg $(v_0, v_1, \dots, v_d)$ wierzchołków taki że $(v_{i-1}, v_i) \in E$ dla $i \in \{1, 2, \dots, d\}$

Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

$$E = \left\{ \begin{array}{l} (0, 0), (0, 1), (0, 2), (1, 0), (1, 1), \\ (1, 4), (2, 3), (3, 2), (3, 4), (4, 3) \end{array} \right\}$$

$(0, 1)$ i $(1, 0)$
— para krawędzi
przeciwnie skierowanych



Definicja: (wersje skierowane)

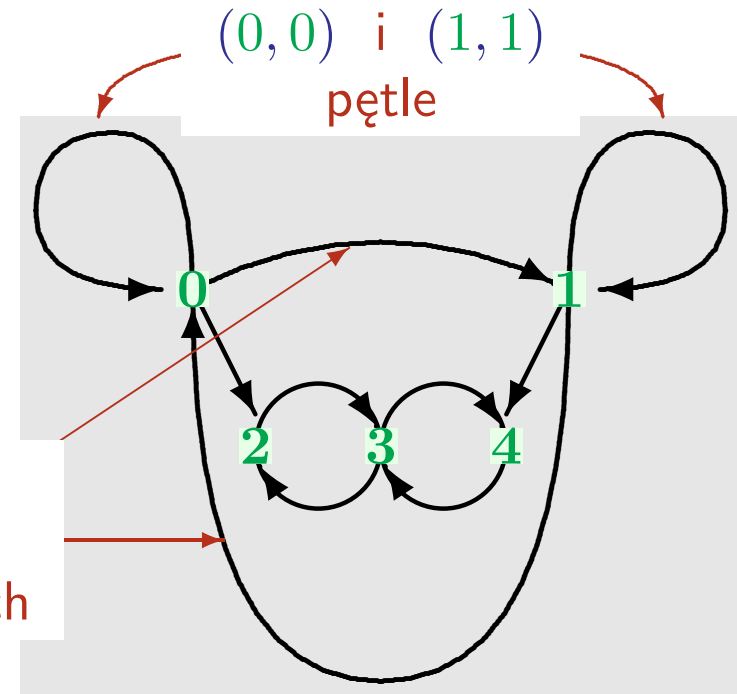
- **stopień wyjścia** wierzchołka $v \in V$ — $|\{u \in V \mid (v, u) \in E\}|$ (liczba wychodzących krawędzi)
- **stopień wejścia** wierzchołka $v \in V$ — $|\{u \in V \mid (u, v) \in E\}|$ (liczba wchodzących krawędzi)
- **droga** — ciąg $(v_0, v_1, \dots, v_d)$ wierzchołków taki że $(v_{i-1}, v_i) \in E$ dla $i \in \{1, 2, \dots, d\}$
- droga jest **zamknięta** $\iff v_d = v_0$

Grafy skierowane

$$V = \{0, 1, 2, 3, 4\}$$

$$E = \left\{ \begin{array}{l} (0, 0), (0, 1), (0, 2), (1, 0), (1, 1), \\ (1, 4), (2, 3), (3, 2), (3, 4), (4, 3) \end{array} \right\}$$

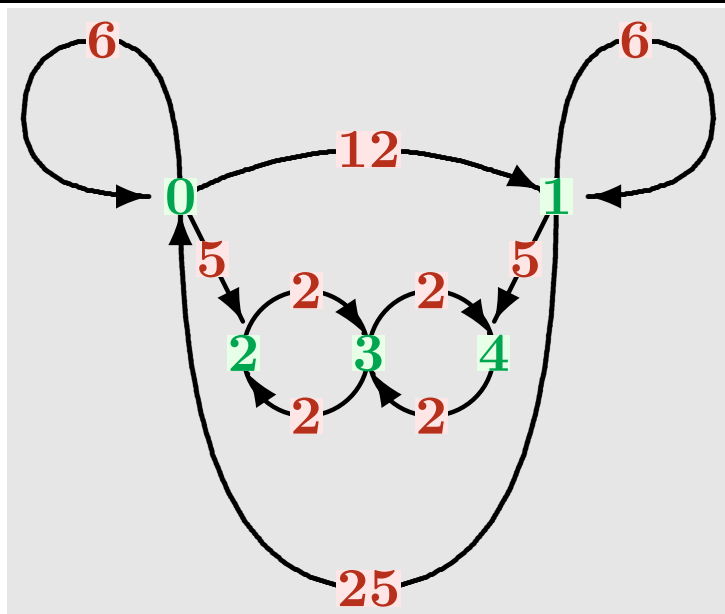
$(0, 1)$ i $(1, 0)$
— para krawędzi
przeciwnie skierowanych



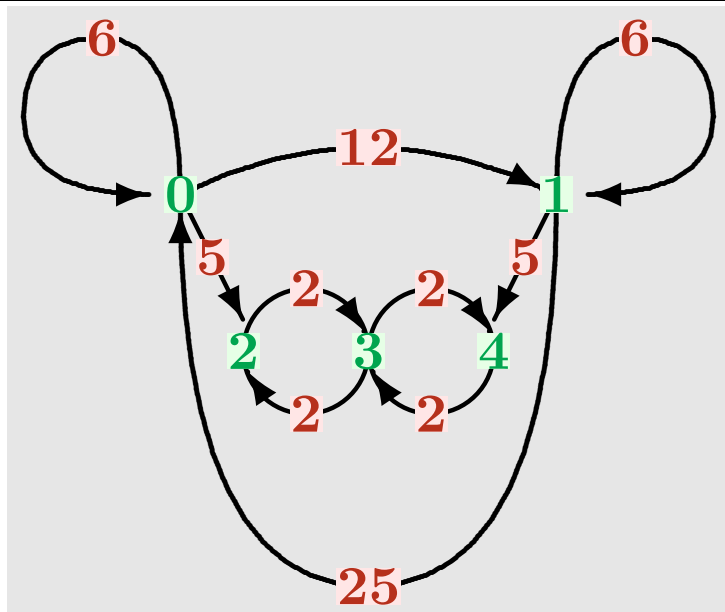
Definicja: (wersje skierowane)

- **stopień wyjścia** wierzchołka $v \in V$ — $|\{u \in V \mid (v, u) \in E\}|$ (liczba wychodzących krawędzi)
- **stopień wejścia** wierzchołka $v \in V$ — $|\{u \in V \mid (u, v) \in E\}|$ (liczba wchodzących krawędzi)
- **droga** — ciąg $(v_0, v_1, \dots, v_d)$ wierzchołków taki że $(v_{i-1}, v_i) \in E$ dla $i \in \{1, 2, \dots, d\}$
- droga jest **zamknięta** $\iff v_d = v_0$
- droga jest **prosta** $\iff$ krawędzie się w niej nie powtarzają

Reprezentacje skierowanego grafu ważonego



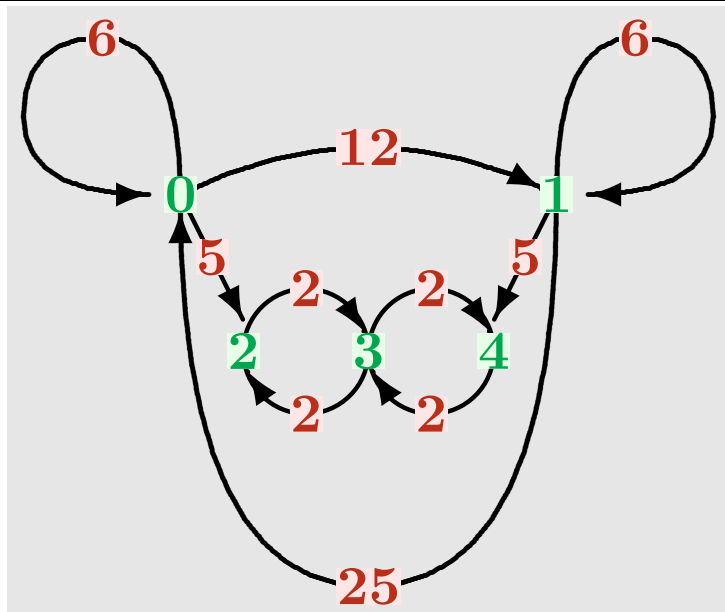
Reprezentacje skierowanego grafu ważonego



Macierz sąsiedztwa:

	0	1	2	3	4
0	6	12	5		
1	25	6			5
2				2	
3			2		2
4				2	

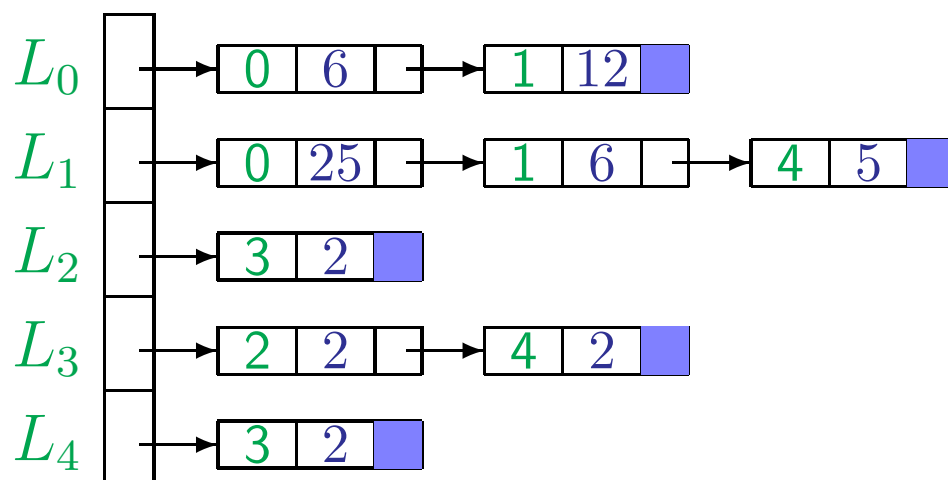
Reprezentacje skierowanego grafu ważonego



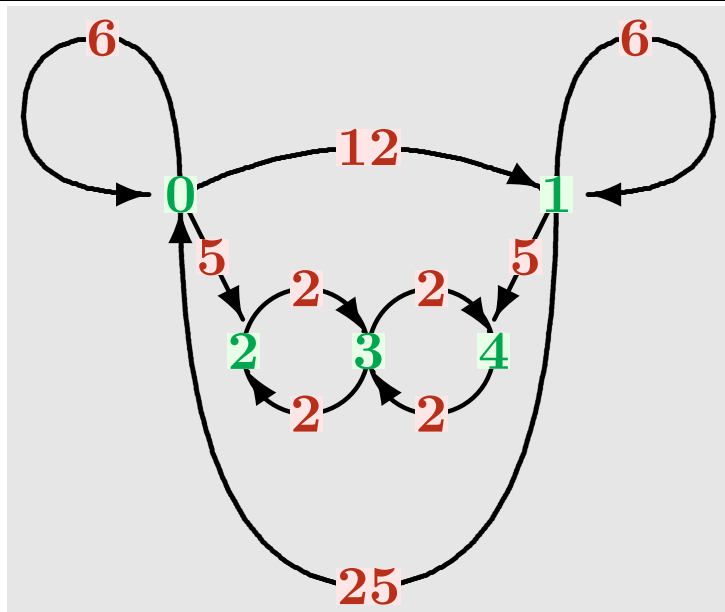
Macierz sąsiedztwa:

	0	1	2	3	4
0	6	12	5		
1	25	6			5
2				2	
3			2		2
4				2	

Listy sąsiedztwa:



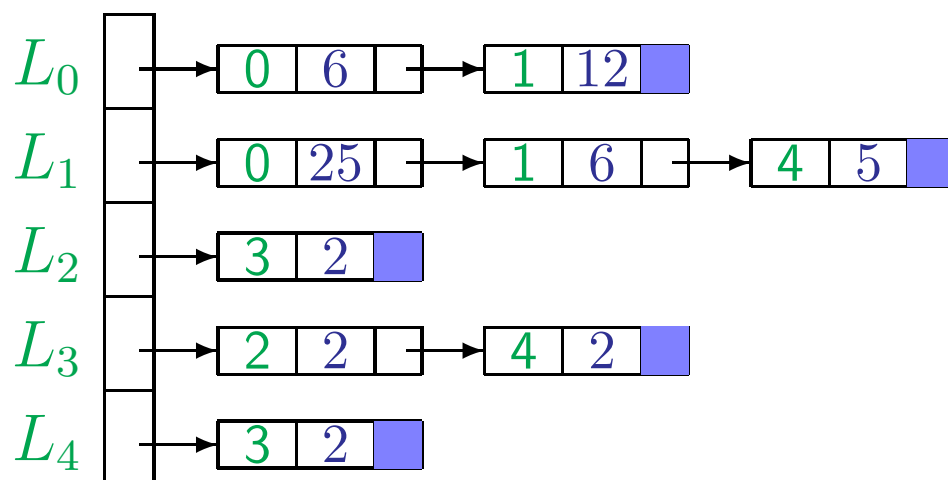
Reprezentacje skierowanego grafu ważonego



Macierz sąsiedztwa:

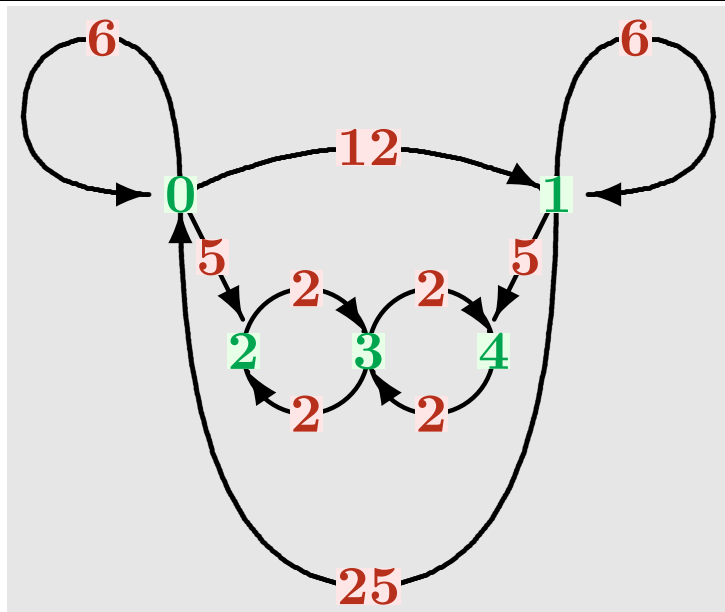
	0	1	2	3	4
0	6	12	5		
1	25	6			5
2				2	
3			2		2
4				2	

Listy sąsiedztwa:



- macierz nie musi być symetryczna

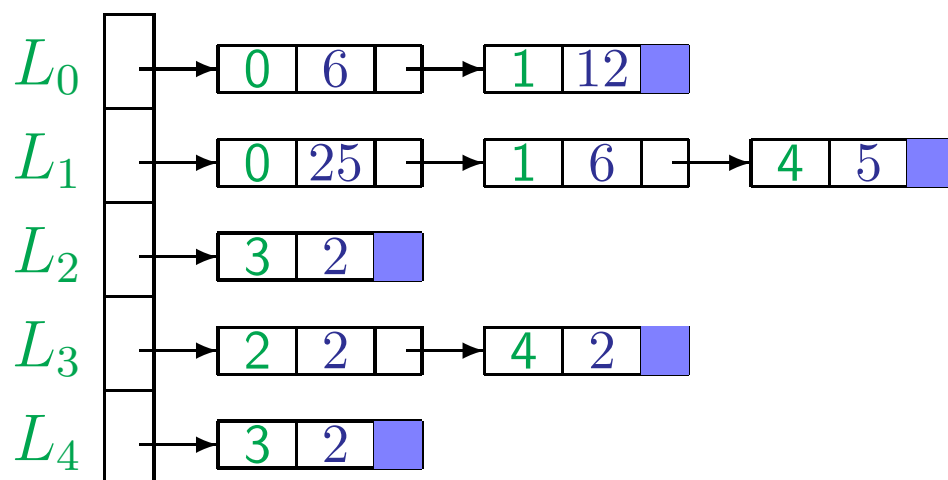
Reprezentacje skierowanego grafu ważonego



Macierz sąsiedztwa:

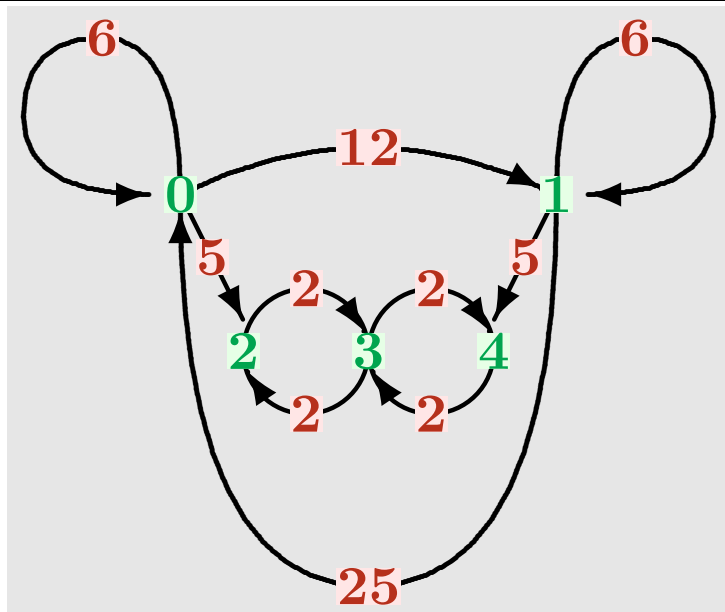
	0	1	2	3	4
0	6	12	5		
1	25	6			5
2				2	
3			2		2
4				2	

Listy sąsiedztwa:



- macierz nie musi być symetryczna
- na liście L_v występuje $u \nRightarrow$ na liście L_u występuje v

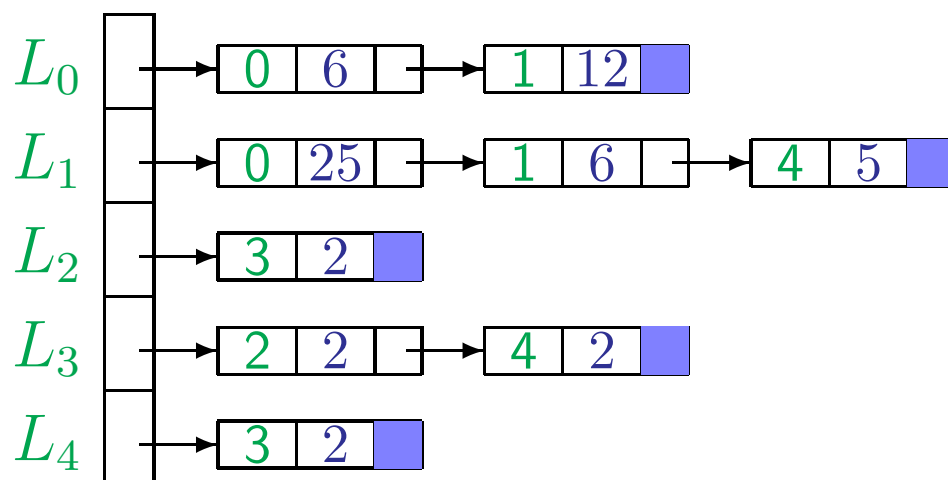
Reprezentacje skierowanego grafu ważonego



Macierz sąsiedztwa:

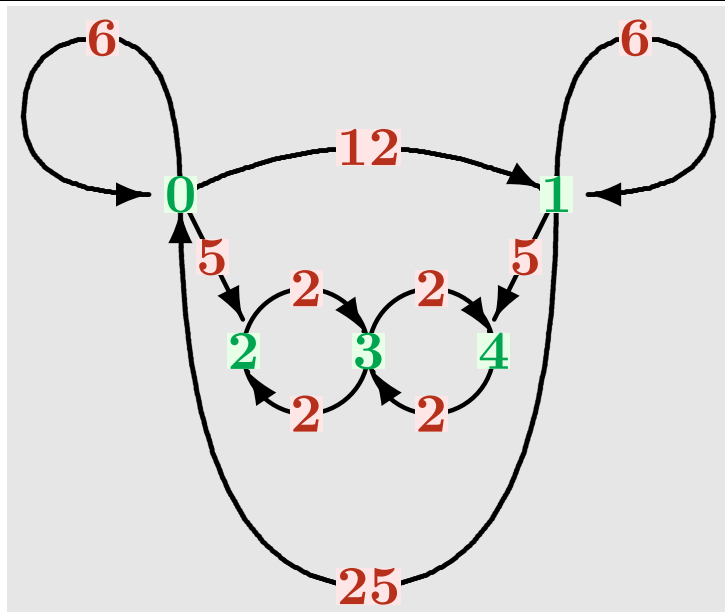
	0	1	2	3	4
0	6	12	5		
1	25	6			5
2				2	
3			2		2
4				2	

Listy sąsiedztwa:



- macierz nie musi być symetryczna
- na liście L_v występuje $u \nRightarrow$ na liście L_u występuje v
- na przekątnej macierzy — pętle

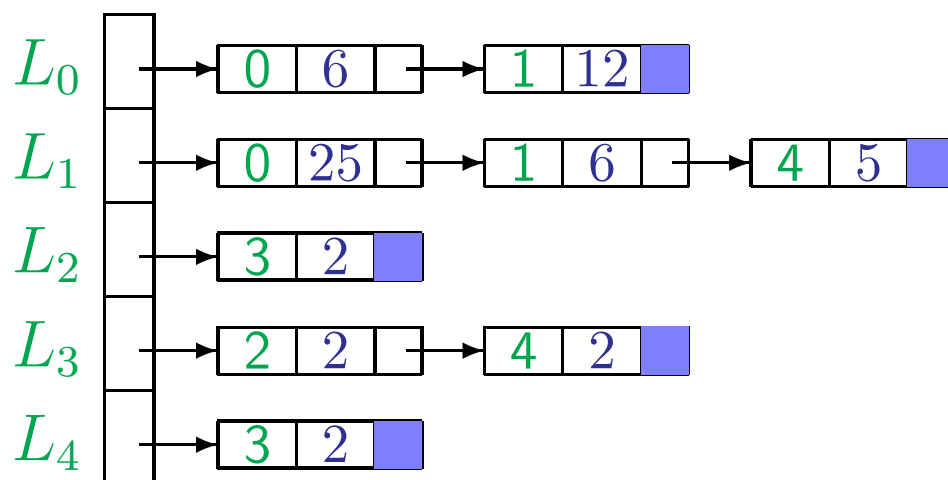
Reprezentacje skierowanego grafu ważonego



Macierz sąsiedztwa:

	0	1	2	3	4
0	6	12	5		
1	25	6			5
2				2	
3			2		2
4				2	

Listy sąsiedztwa:



- macierz nie musi być symetryczna
- na liście L_v występuje $u \nRightarrow$ na liście L_u występuje v
- na przekątnej macierzy — pętle
- v na liście L_v — pętla

Sortowanie topologiczne

Sortowanie topologiczne

Definicja:

Graf jest **acykliczny**, jeśli nie zawiera cykli, czyli zamkniętych dróg prostych o długości > 1 .

Sortowanie topologiczne

Definicja:

Graf jest **acykliczny**, jeśli nie zawiera cykli, czyli zamkniętych dróg prostych o długości > 1 .

Definicja:

Topologiczny porządek w skierowanym grafie acyklicznym $G = \langle V, E \rangle$ — porządek liniowy $\leq$ na V taki, że

$$\forall_{(v,w) \in E} v \leq w$$

Sortowanie topologiczne

Definicja:

Graf jest **acykliczny**, jeśli nie zawiera cykli, czyli zamkniętych dróg prostych o długości > 1 .

Definicja:

Topologiczny porządek w skierowanym grafie acyklicznym $G = \langle V, E \rangle$ — porządek liniowy $\leq$ na V taki, że

$$\forall_{(v,w) \in E} v \leq w$$

czyli wszystkie strzałki w grafie prowadzą od wierzchołków mniejszych do większych.

Sortowanie topologiczne

Definicja:

Graf jest **acykliczny**, jeśli nie zawiera cykli, czyli zamkniętych dróg prostych o długości > 1 .

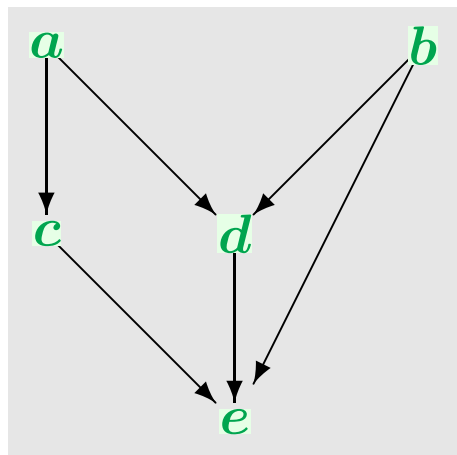
Definicja:

Topologiczny porządek w skierowanym grafie acyklicznym $G = \langle V, E \rangle$ — porządek liniowy $\leq$ na V taki, że

$$\forall_{(v,w) \in E} v \leq w$$

czyli wszystkie strzałki w grafie prowadzą od wierzchołków mniejszych do większych.

Przykład:



Sortowanie topologiczne

Definicja:

Graf jest **acykliczny**, jeśli nie zawiera cykli, czyli zamkniętych dróg prostych o długości > 1 .

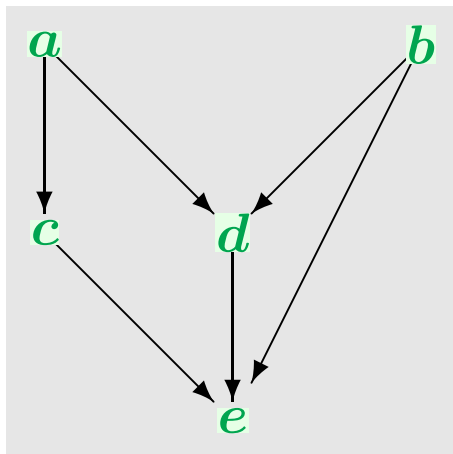
Definicja:

Topologiczny porządek w skierowanym grafie acyklicznym $G = \langle V, E \rangle$ — porządek liniowy $\leq$ na V taki, że

$$\forall_{(v,w) \in E} v \leq w$$

czyli wszystkie strzałki w grafie prowadzą od wierzchołków mniejszych do większych.

Przykład:



przykładowe porządki topologiczne w tym grafie:

$$a \leq b \leq c \leq d \leq e$$

Sortowanie topologiczne

Definicja:

Graf jest **acykliczny**, jeśli nie zawiera cykli, czyli zamkniętych dróg prostych o długości > 1 .

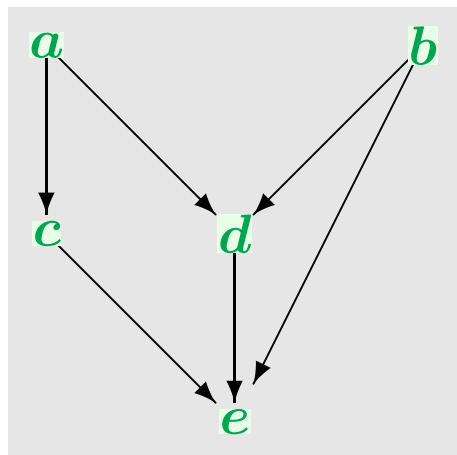
Definicja:

Topologiczny porządek w skierowanym grafie acyklicznym $G = \langle V, E \rangle$ — porządek liniowy $\leq$ na V taki, że

$$\forall_{(v,w) \in E} v \leq w$$

czyli wszystkie strzałki w grafie prowadzą od wierzchołków mniejszych do większych.

Przykład:



przykładowe porządki topologiczne w tym grafie:

$$a \leq b \leq c \leq d \leq e$$

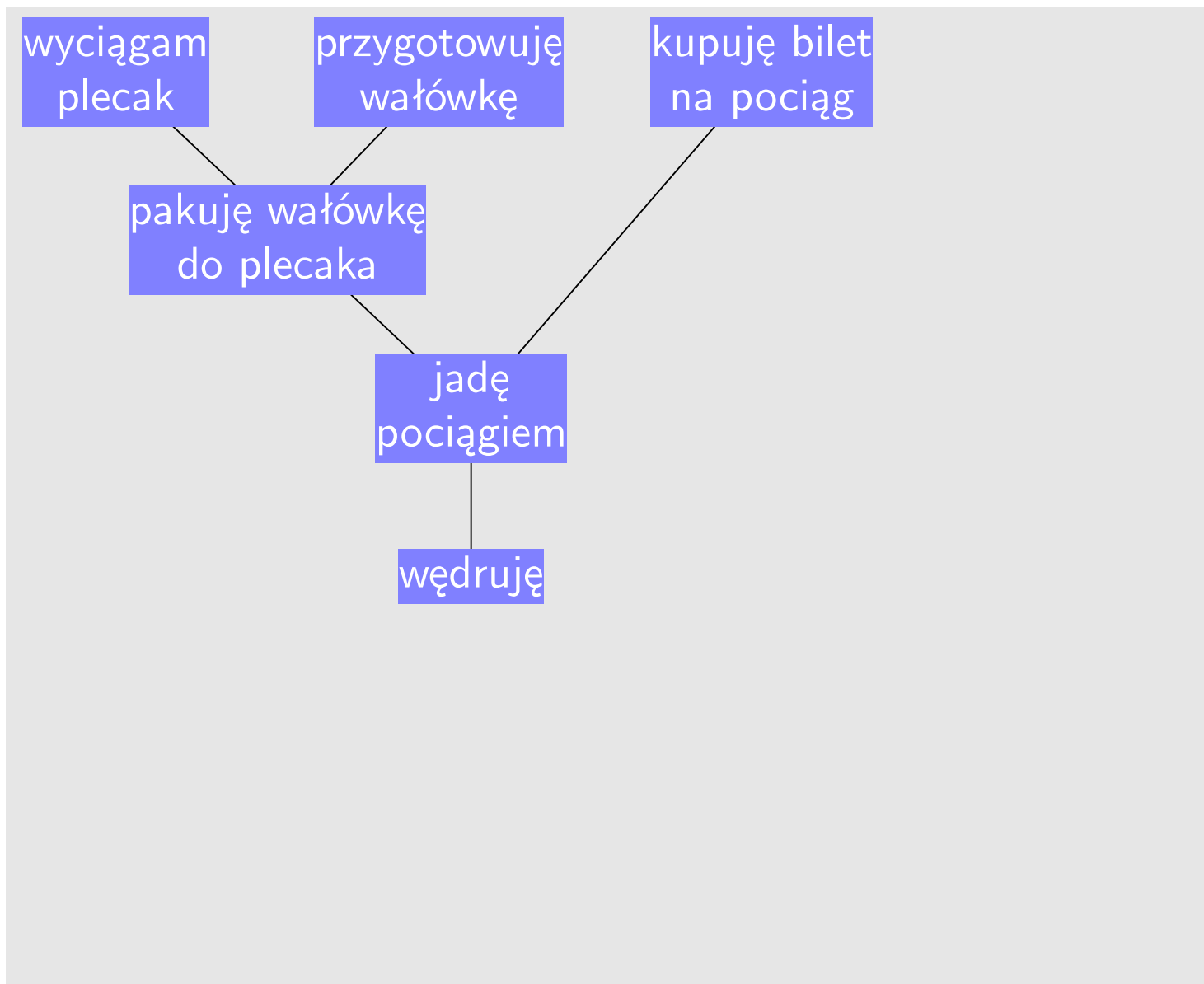
$$b \leq a \leq d \leq c \leq e$$

Sortowanie topologiczne

Plan wędrówki

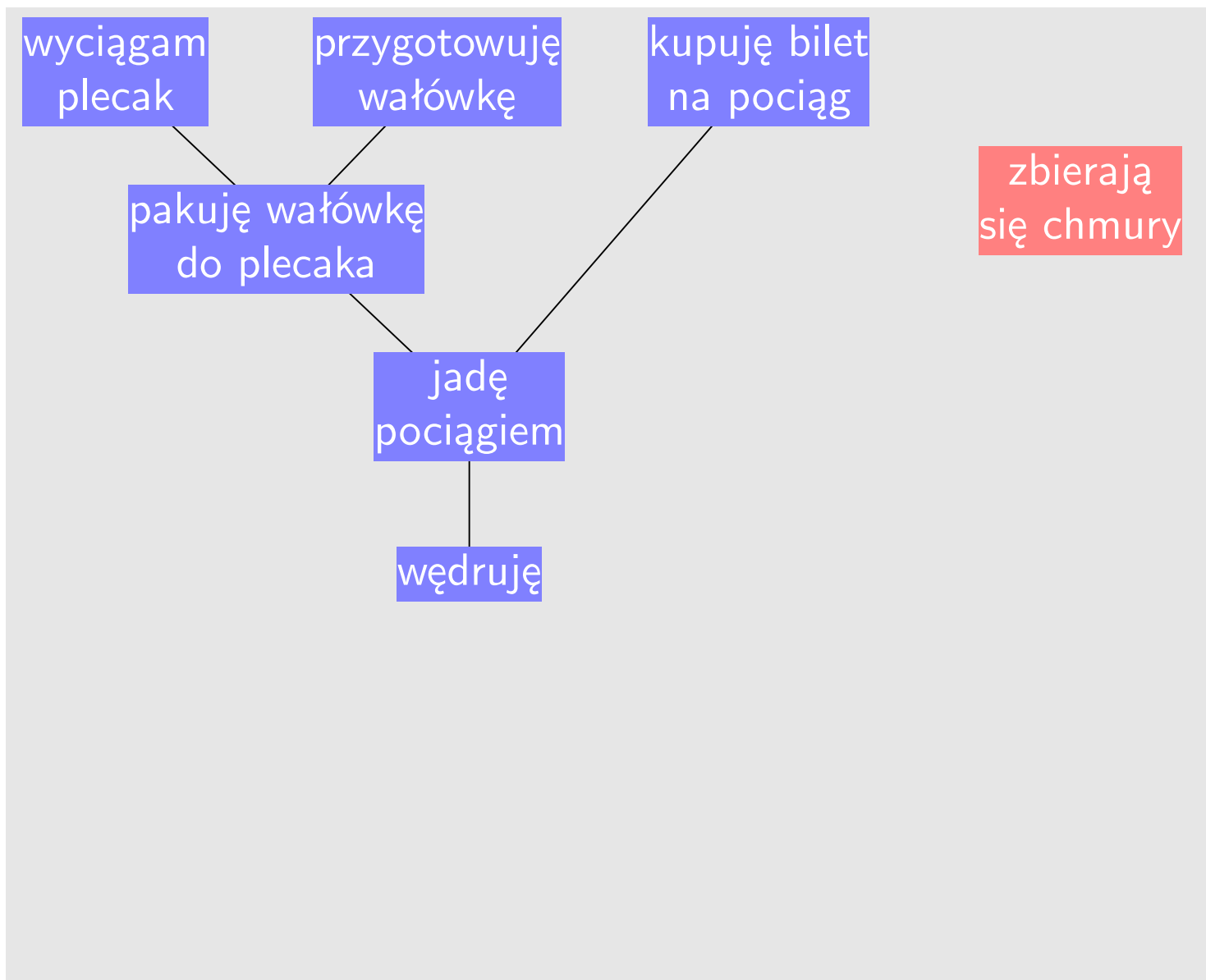
Sortowanie topologiczne

Plan wędrowki:



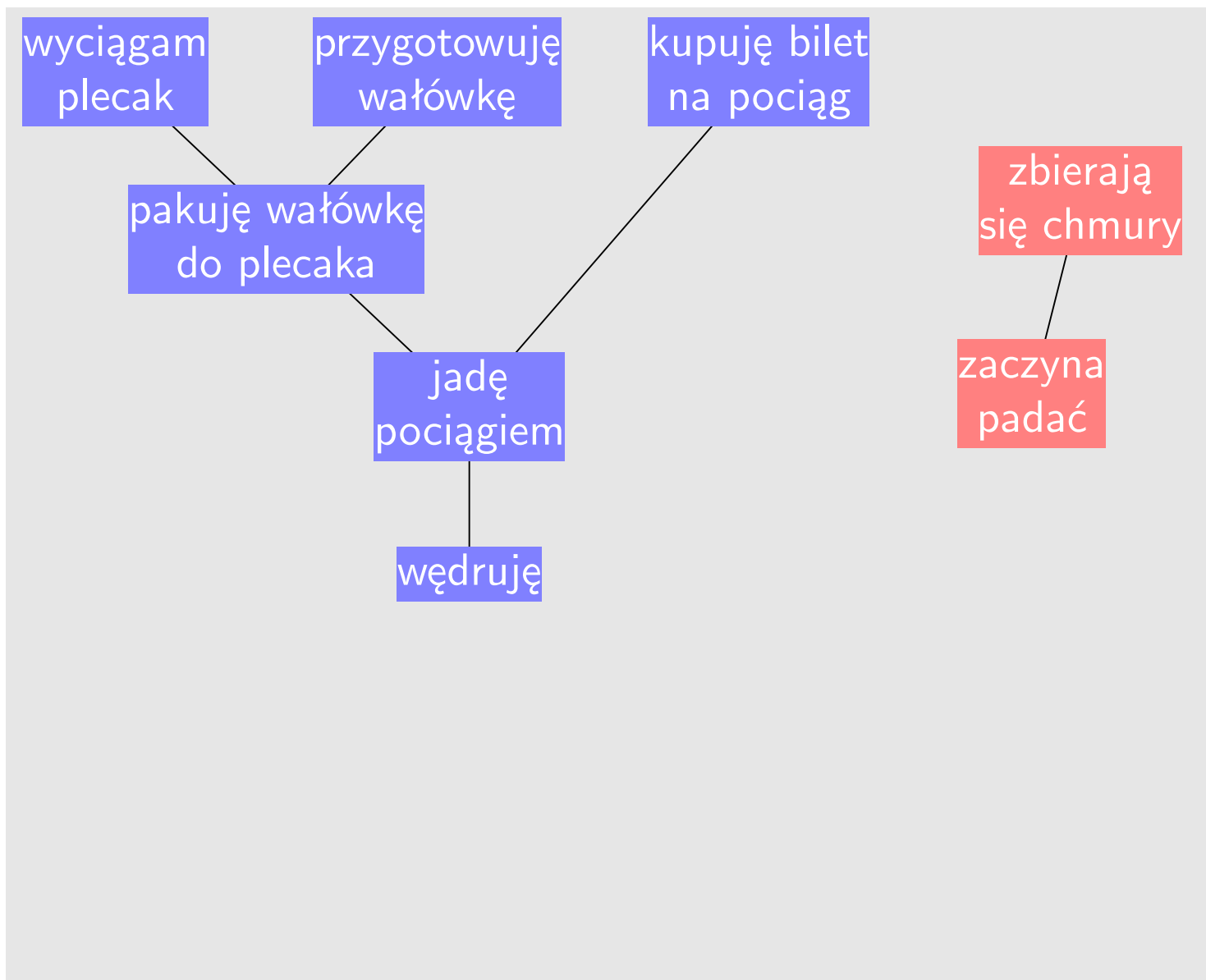
Sortowanie topologiczne

Plan wędrowki:



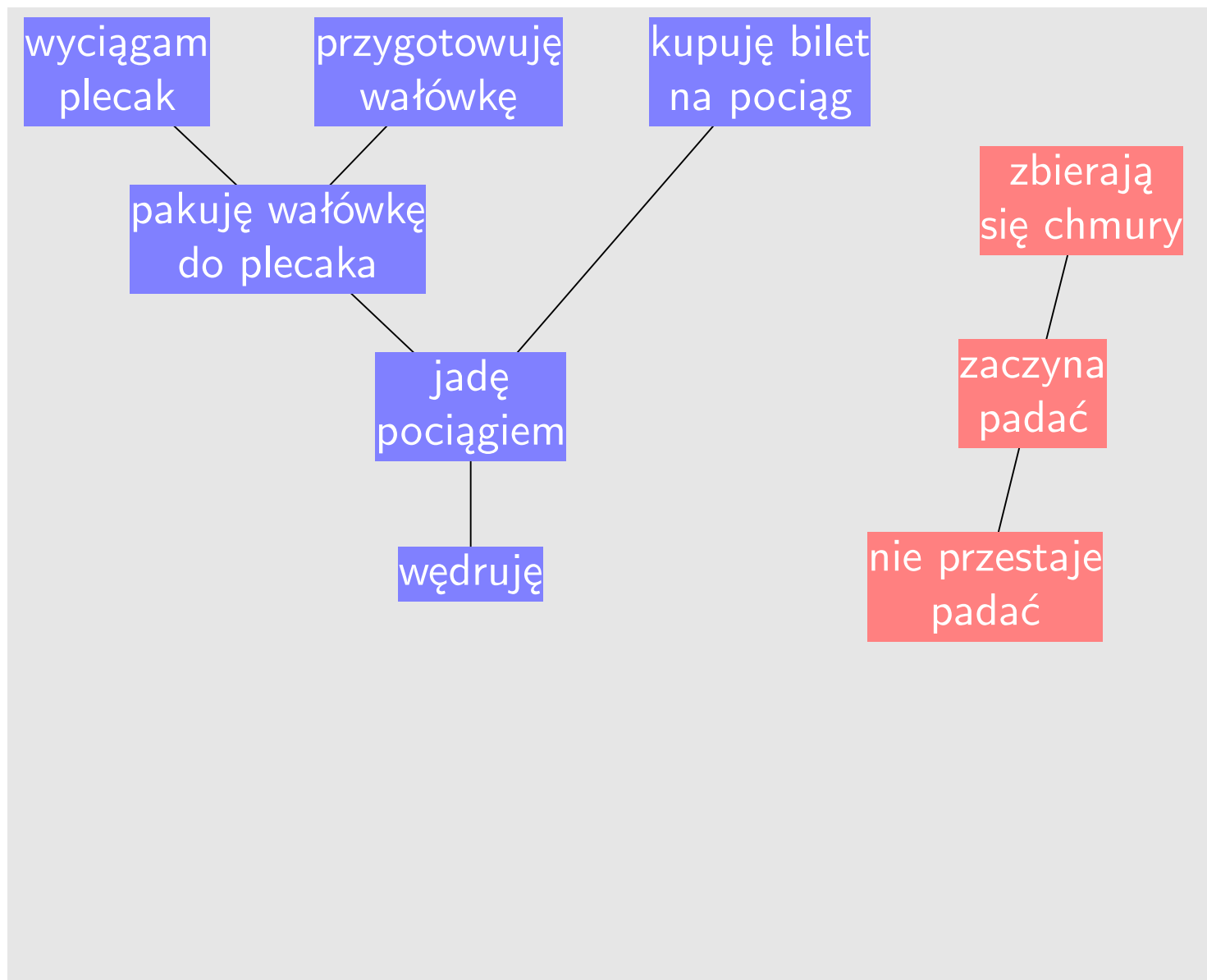
Sortowanie topologiczne

Plan wędrowki:



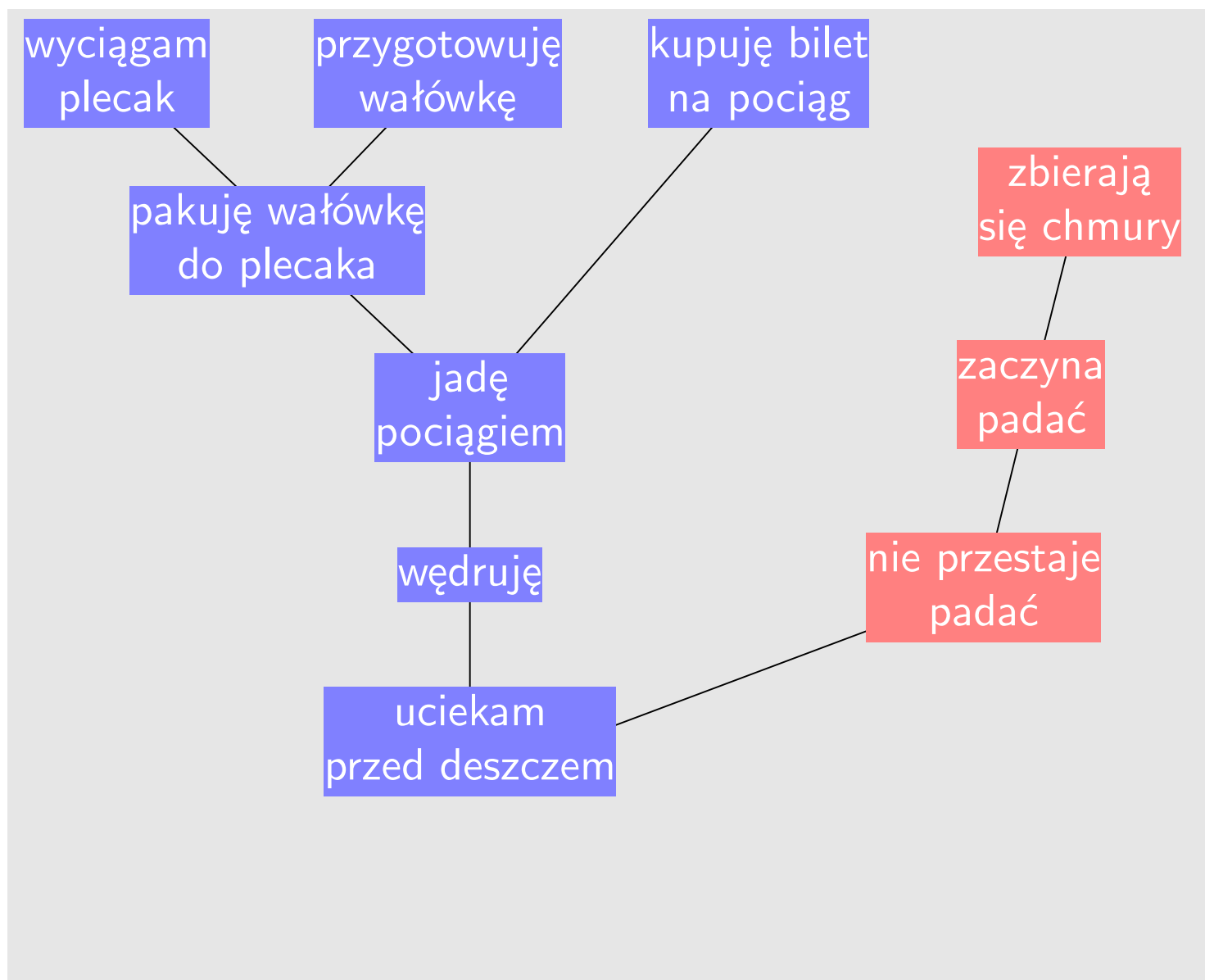
Sortowanie topologiczne

Plan wędrowki:



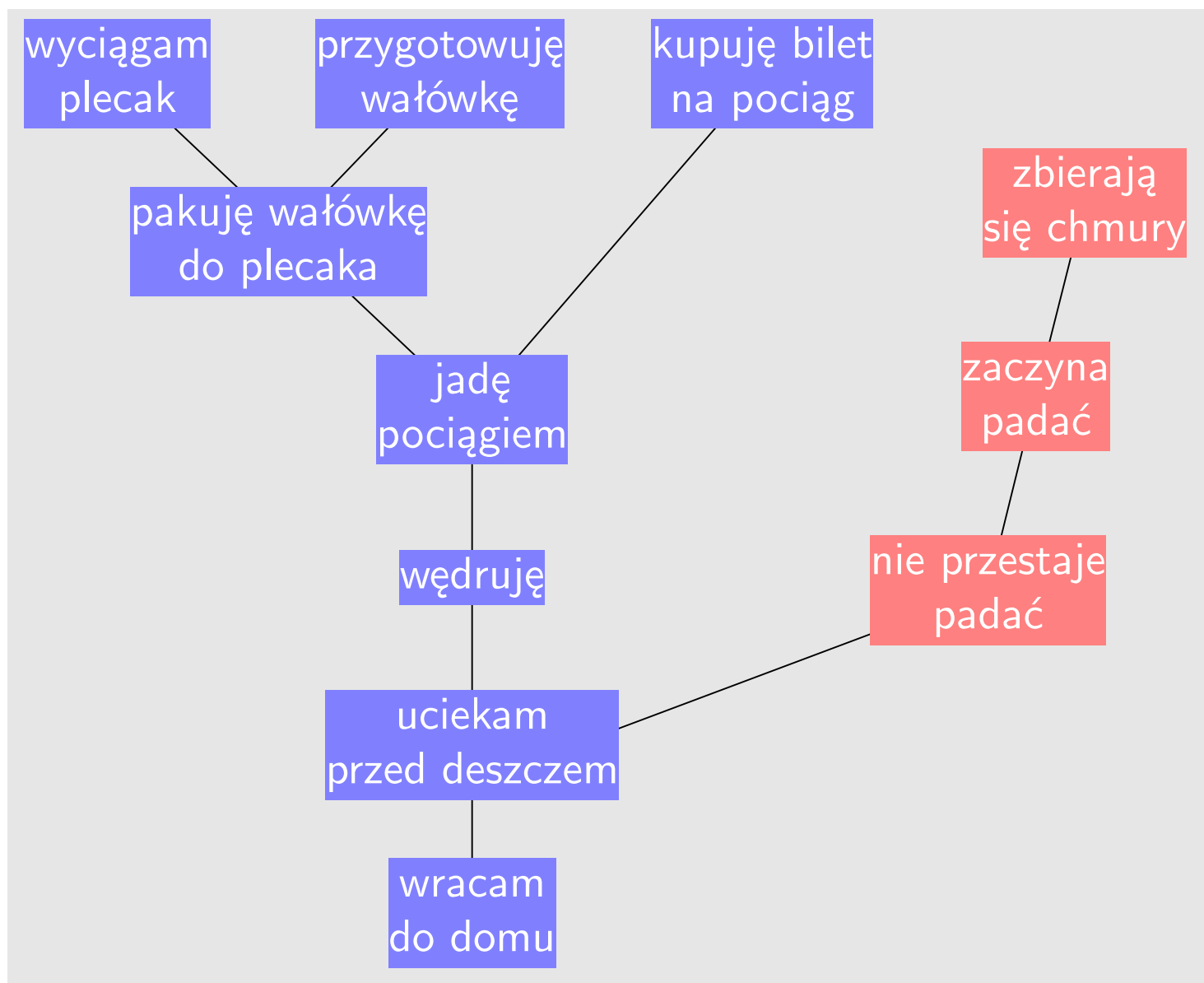
Sortowanie topologiczne

Plan wędrowki:



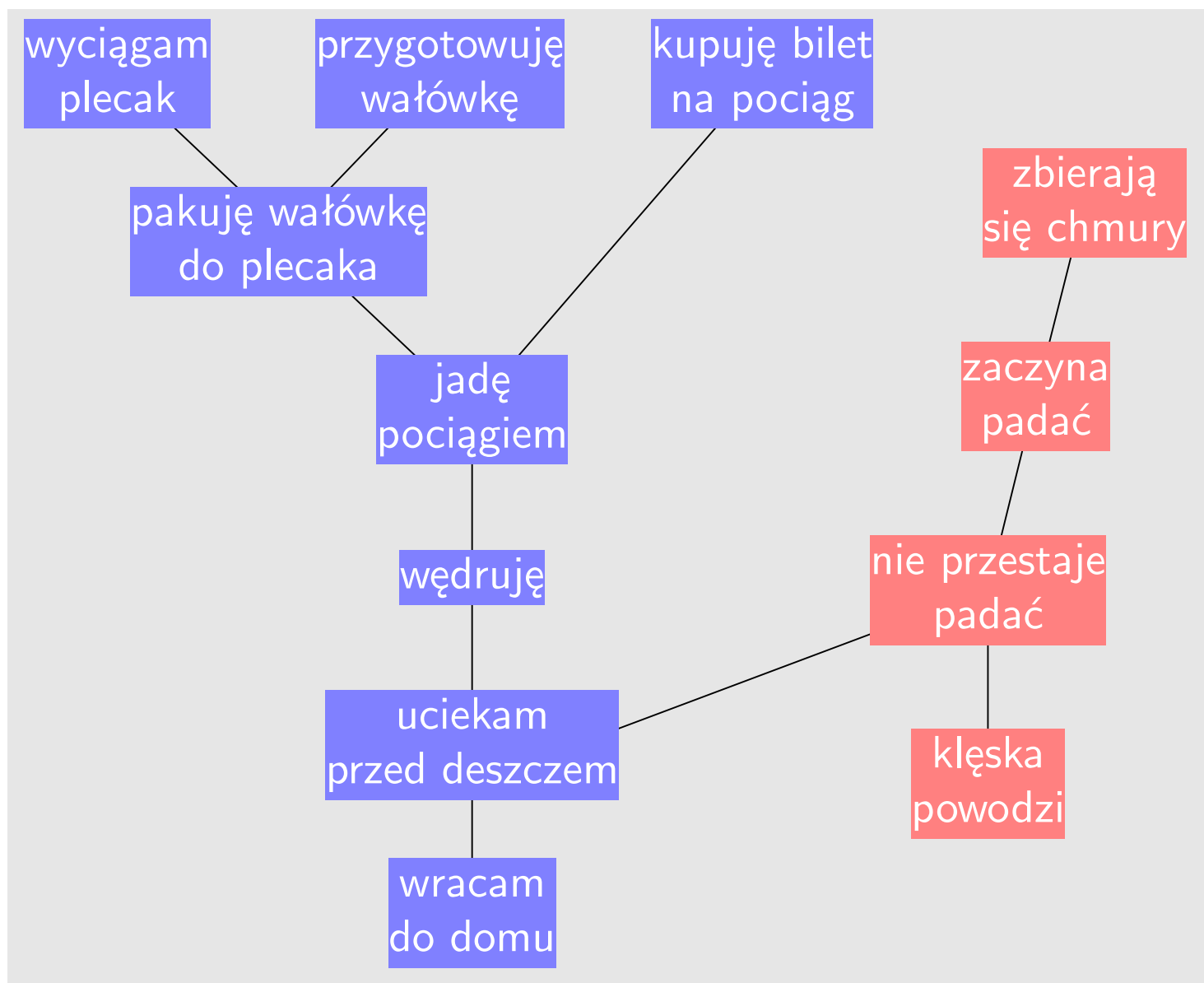
Sortowanie topologiczne

Plan wędrowki:



Sortowanie topologiczne

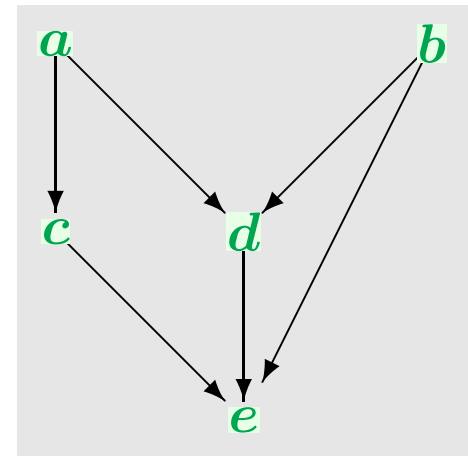
Plan wędrowki:



Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



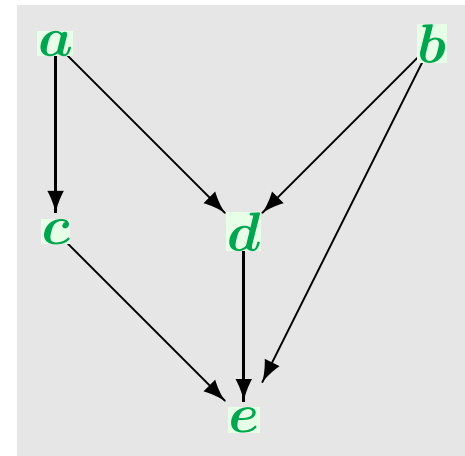
S :

L :

Sortowanie topologiczne

Algorytm sortowania topologicznego:

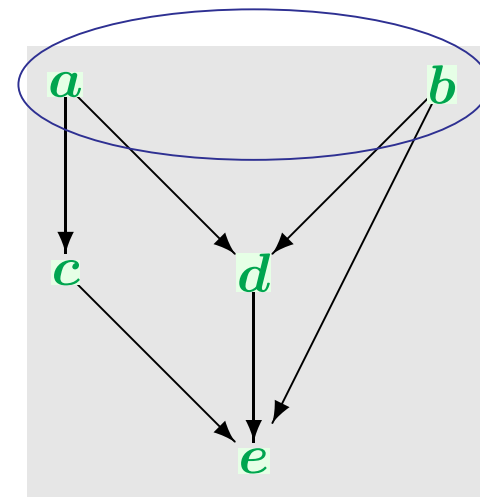
1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3

 $S :$ $L :$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



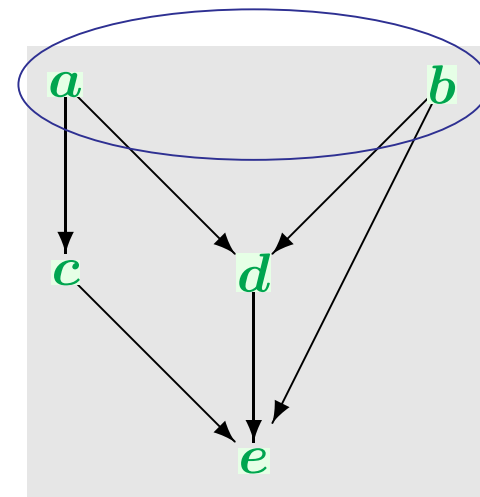
$S : a, b$

$L :$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



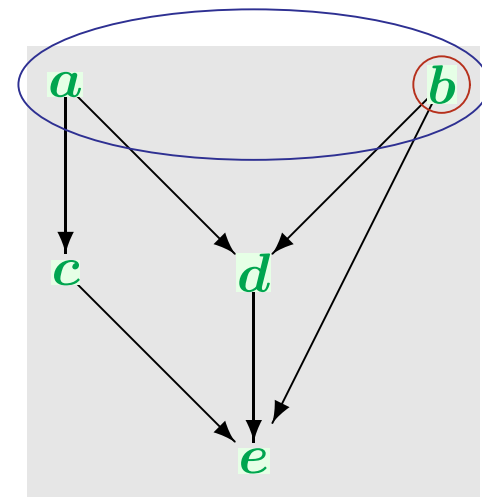
$S : a, b$

$L :$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



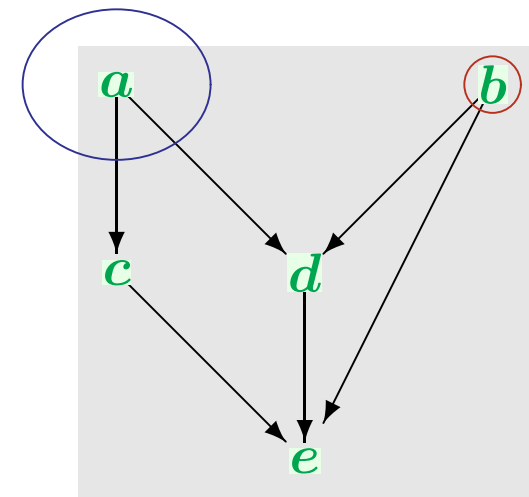
$S : a, b$

$L :$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



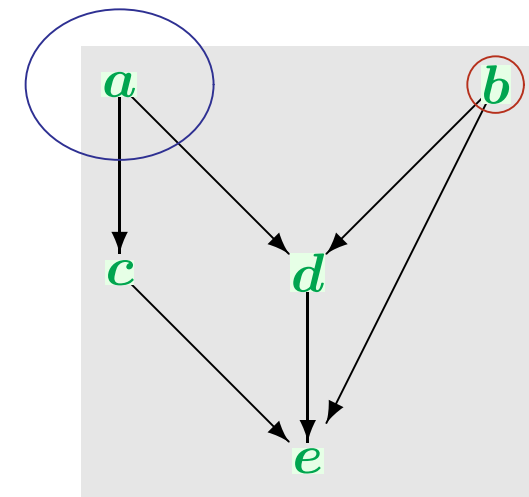
$S : a$

$L : b$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



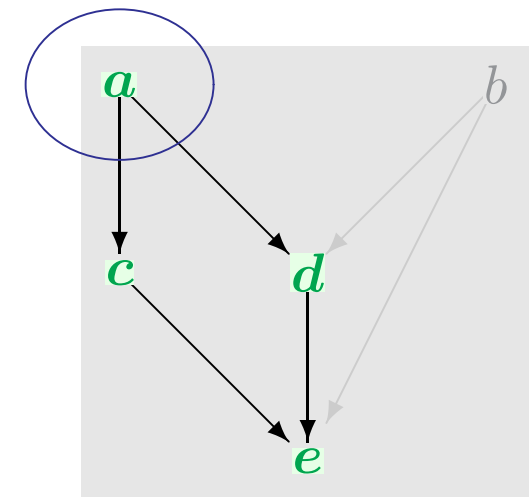
$S : a$

$L : b$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



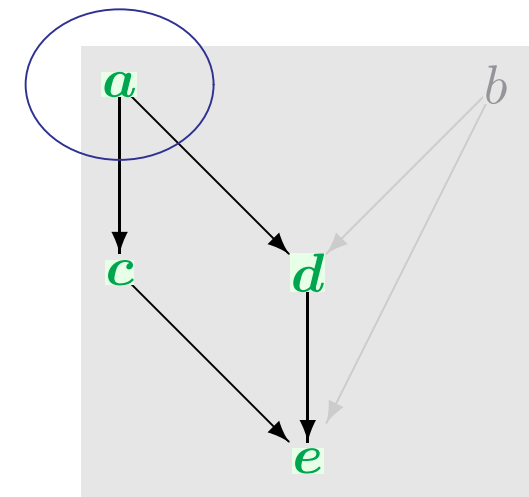
$S : a$

$L : b$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. **powrócić do 3**



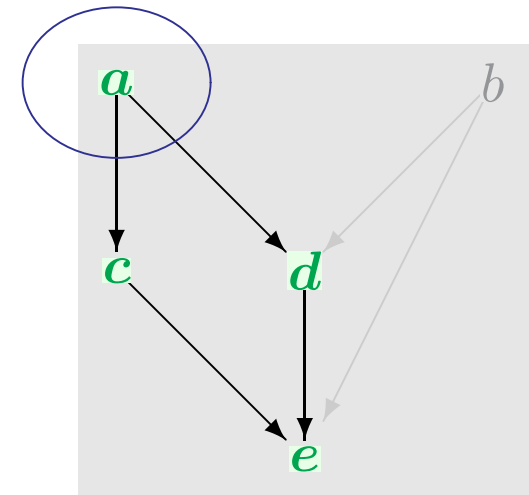
$S : a$

$L : b$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



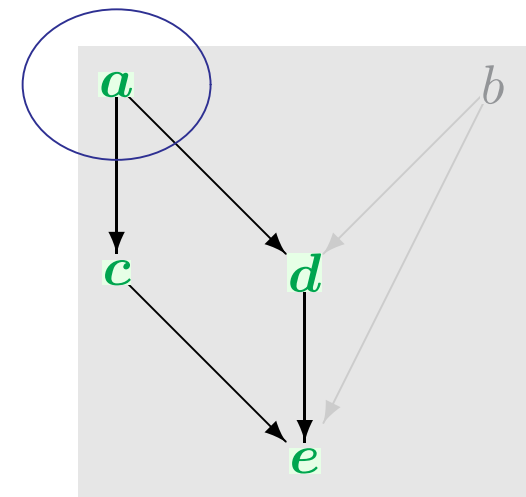
$S : a$

$L : b$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



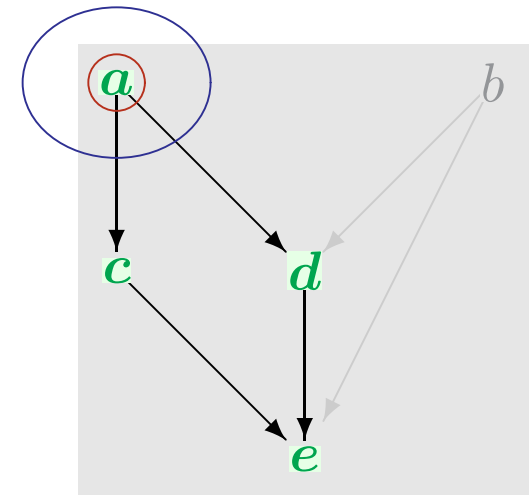
$S : a$

$L : b$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



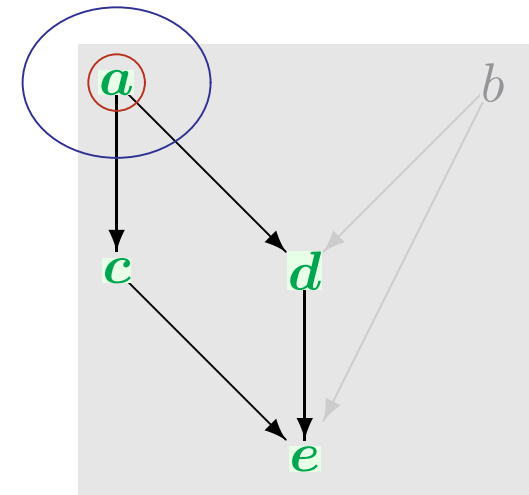
$S :$

$L : b, a$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



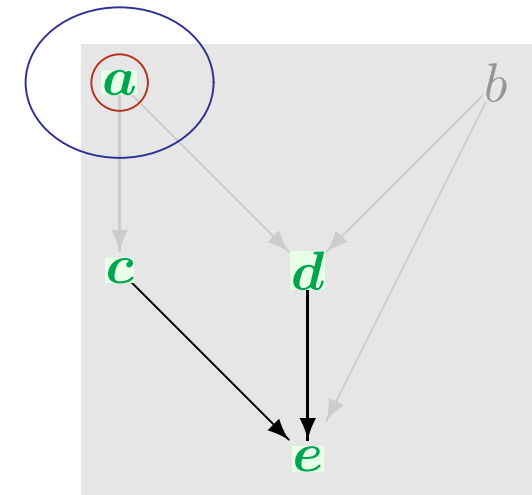
$S :$

$L : b, a$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



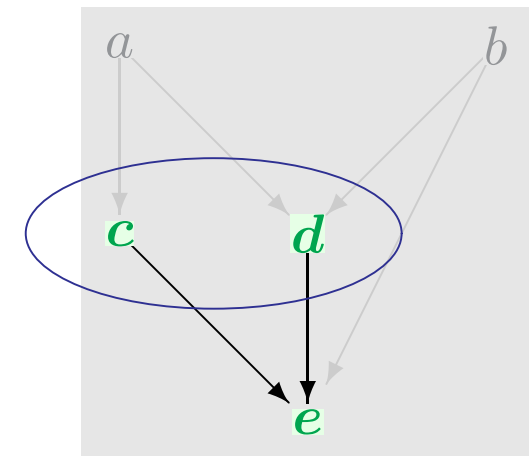
$S :$

$L : b, a$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

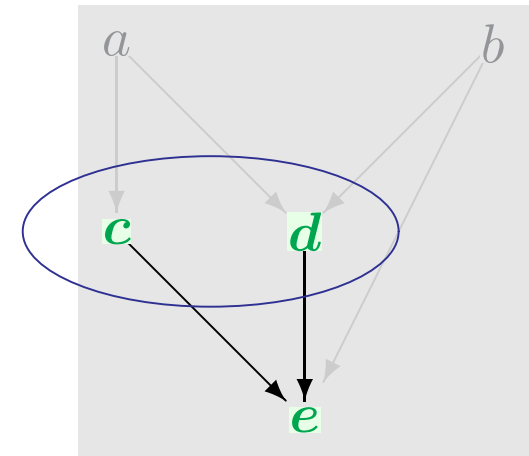
1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3

 $S : c, d$ $L : b, a$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



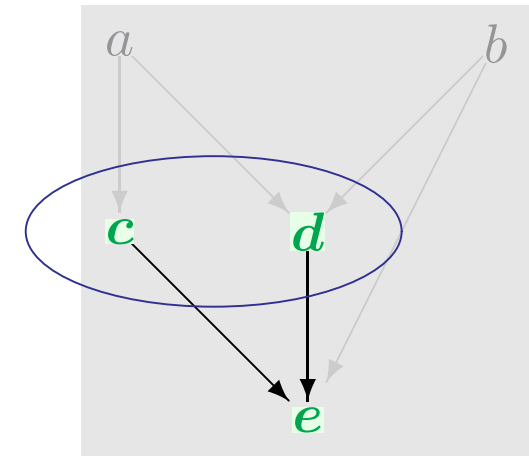
$S : c, d$

$L : b, a$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



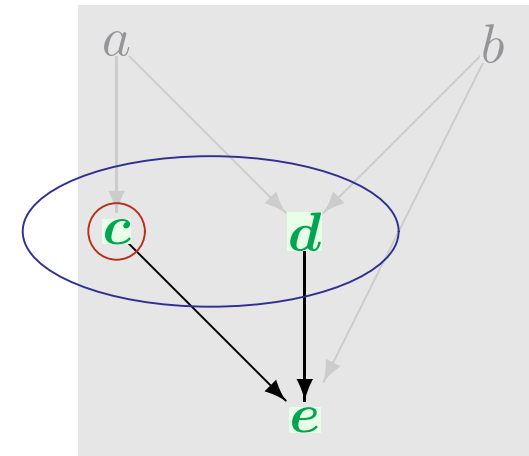
$S : c, d$

$L : b, a$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



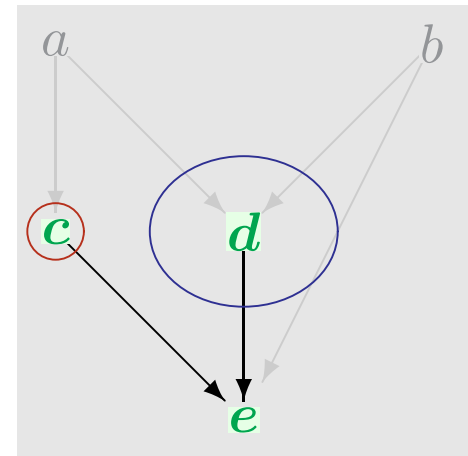
$S : c, d$

$L : b, a$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



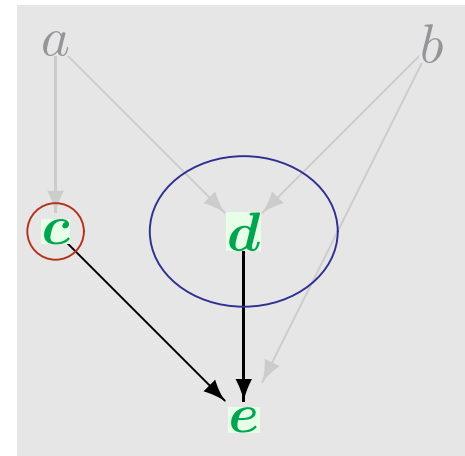
$S : d$

$L : b, a, c$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



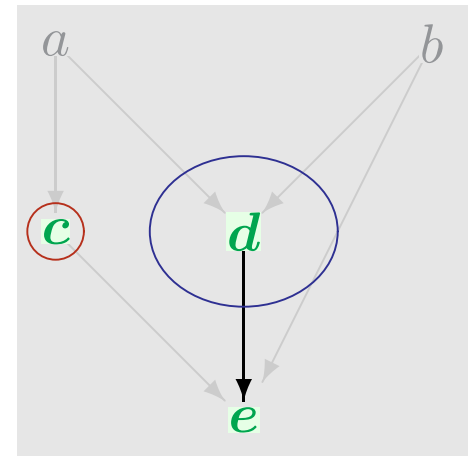
$S : d$

$L : b, a, c$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



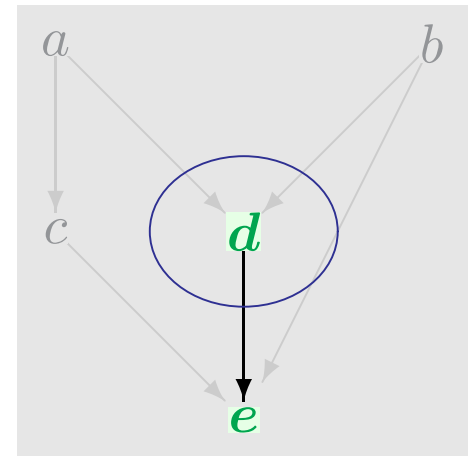
$S : d$

$L : b, a, c$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



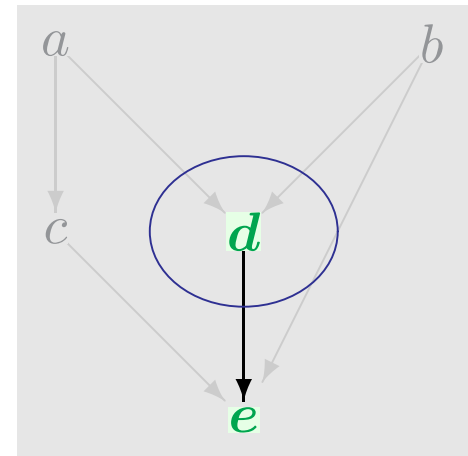
$S : d$

$L : b, a, c$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



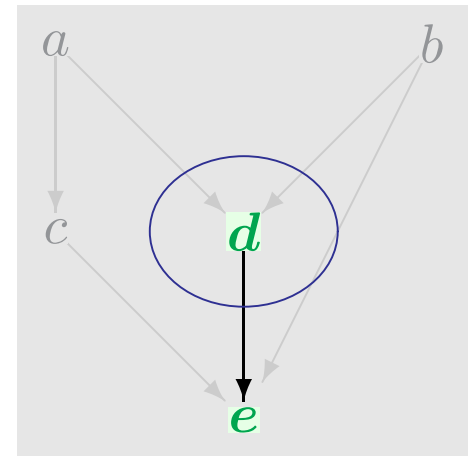
$S : d$

$L : b, a, c$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



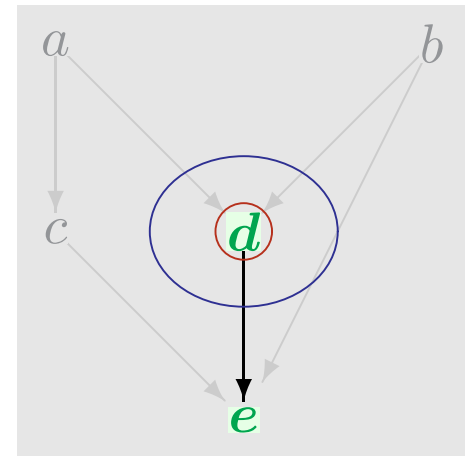
$S : d$

$L : b, a, c$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



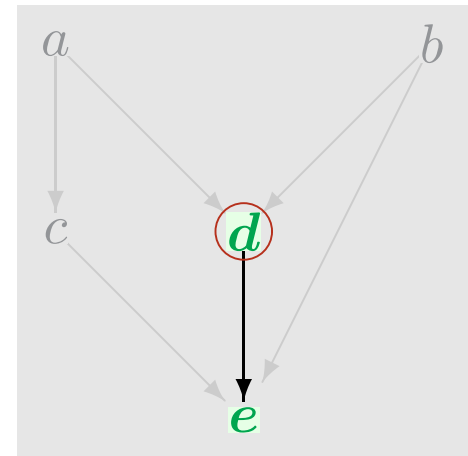
$S : d$

$L : b, a, c$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



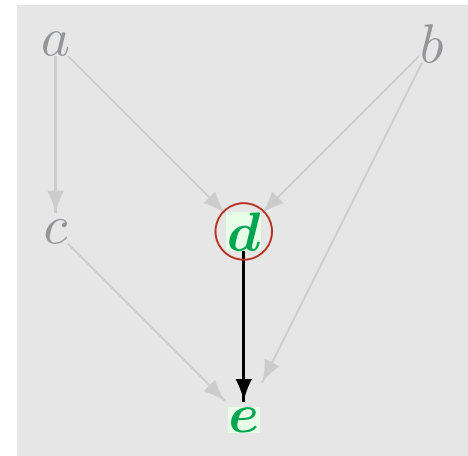
$S :$

$L : b, a, c, d$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



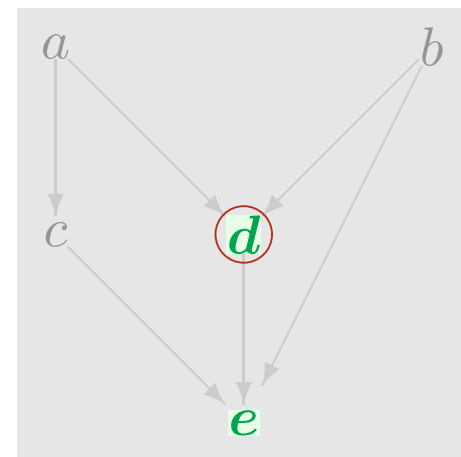
$S :$

$L : b, a, c, d$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



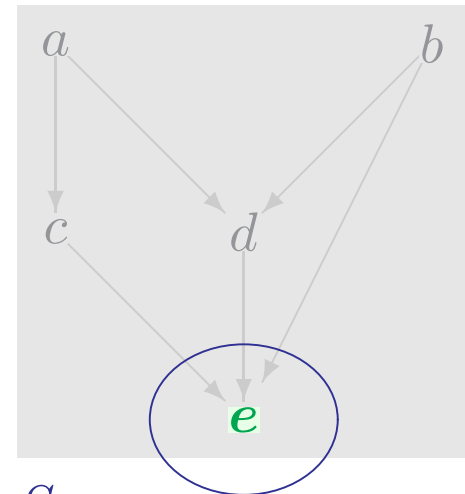
$S :$

$L : b, a, c, d$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



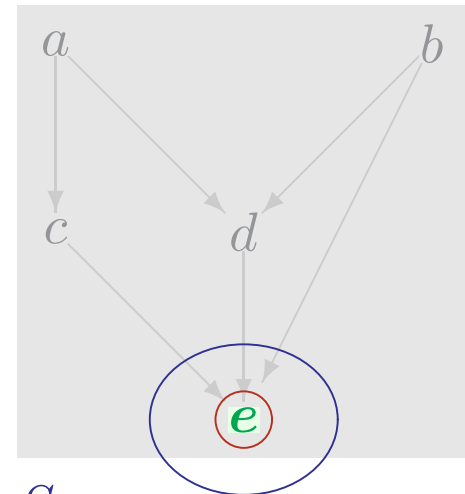
$S : e$

$L : b, a, c, d$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



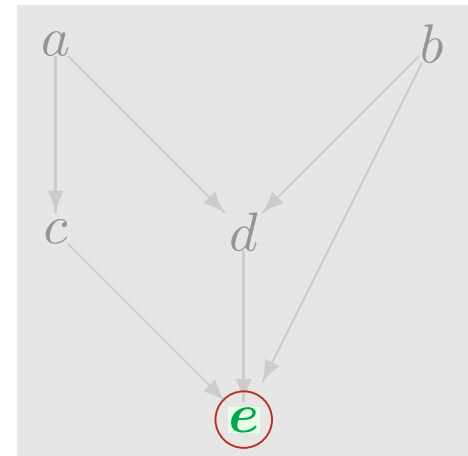
$S : e$

$L : b, a, c, d$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



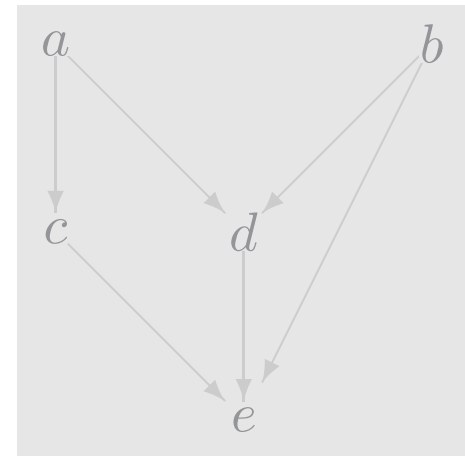
$S :$

$L : b, a, c, d, e$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



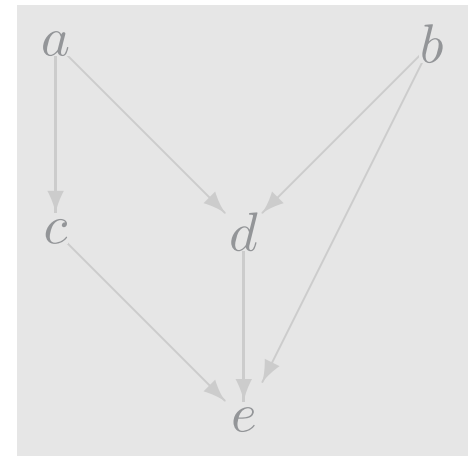
$S :$

$L : b, a, c, d, e$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



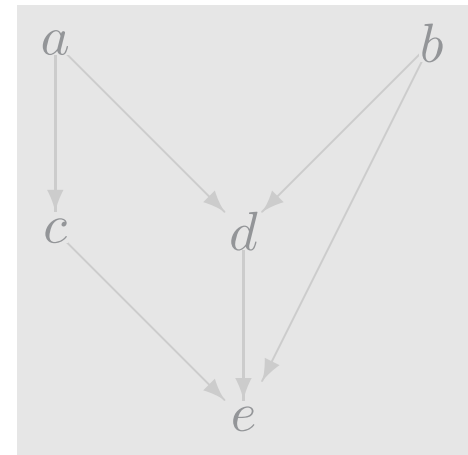
$S :$

$L : b, a, c, d, e$

Sortowanie topologiczne

Algorytm sortowania topologicznego:

1. lista $L \leftarrow$ pusta
2. zbiór $S \leftarrow$ wierzchołki, do których nie dochodzą żadne strzałki
3. jeśli $S = \emptyset$, to zakończyć; lista L zawiera uporządkowane topologicznie wierzchołki grafu
4. wybrać jakiś wierzchołek $v \in S$, przenieść go z S na koniec L
5. dla każdej krawędzi wychodzącej z v idącej do jakiegoś w
 - usunąć tę krawędź z grafu
 - jeśli do w nie dochodzi żadna krawędź, to dodać w do S
6. powrócić do 3



$S :$

$L : b, a, c, d, e$

Wynik: $b \leq a \leq c \leq d \leq e$