Politechnika Wrocławska

Struktury danych i złożoność obliczeniowa
Wykład 5.

Prof. dr hab. inż. Jan Magott

Algorytmy grafowe:

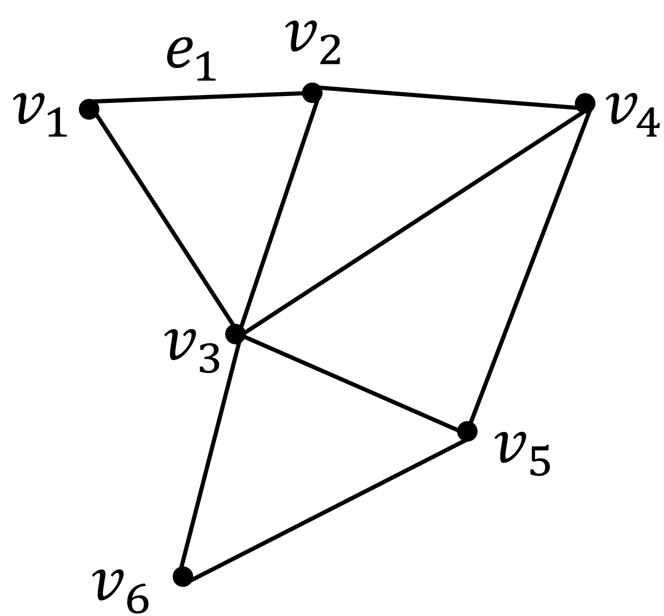
- podstawowe pojęcia,
- reprezentacja grafów,
- metody przeszukiwania,
- minimalne drzewa rozpinające,
- problemy ścieżkowe.

Algorytmy grafowe: podstawowe pojęcia

Graf nieskierowany jest parą uporządkowaną $G = (V, E)$, gdzie $V = \{v_1, v_2, \dots, v_n\}$ jest zbiorem wierzchołków (ang. vertices),

$E = \{e_1, e_2, \dots, e_m\}$ jest zbiorem krawędzi (ang. edges),

$e_i = \{v_1, v_2\}$ jest łukiem będącym zbiorem dwuelementowym wierzchołków.



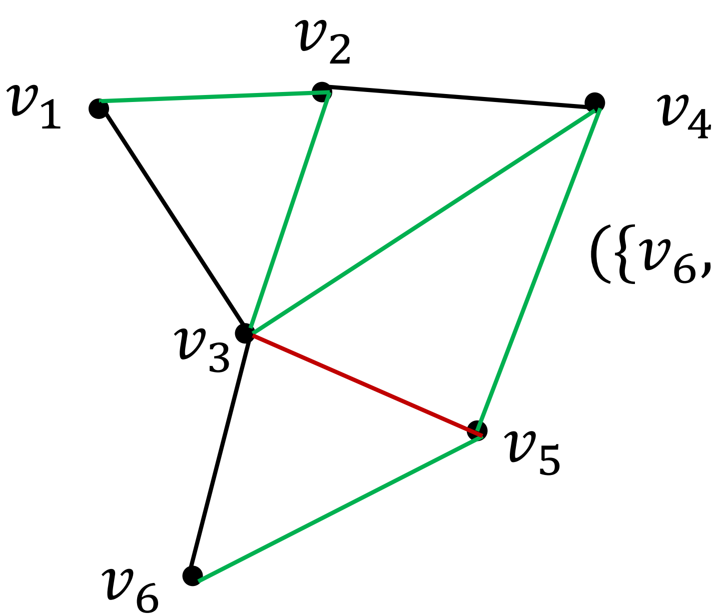
$e_1 = \{v_1, v_2\}$

Nie ma pętli $\{v_i, v_i\}$.

Algorytmy grafowe: podstawowe pojęcia

Droga (ścieżka) w grafie nieskierowanym jest ciągiem krawędzi $(\{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{k-2}, v_{k-1}\}, \{v_{k-1}, v_k\})$, który można wyrazić **ciągiem** wierzchołków

$(v_1, v_2, v_3, \dots, v_{k-2}, v_{k-1}, v_k)$.



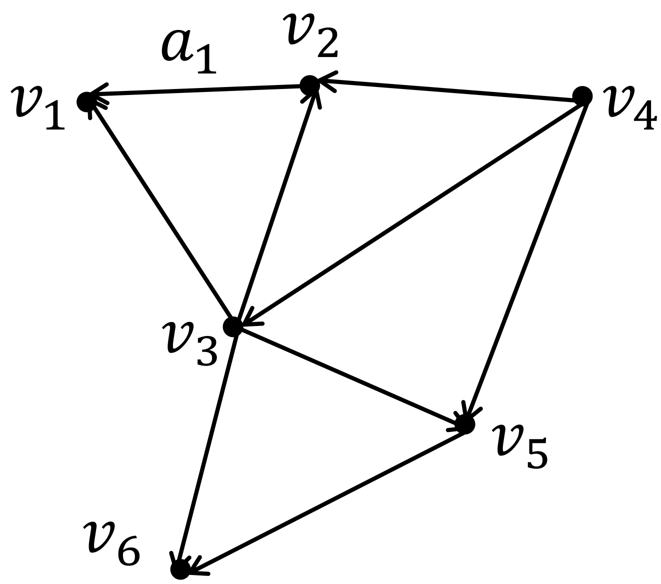
$(\{v_6, v_5\}, \{v_5, v_4\}, \{v_4, v_3\}, \{v_3, v_2\}, \{v_2, v_1\})$

$(v_6, v_5, v_4, v_3, v_2, v_1)$

Algorytmy grafowe: podstawowe pojęcia

Graf skierowany jest parą uporządkowaną $G = (V, A)$, gdzie $V = \{v_1, v_2, \dots, v_n\}$ jest zbiorem wierzchołków (ang. vertices), $A = \{a_1, a_2, \dots, a_m\}$ jest zbiorem łuków (ang. arcs),

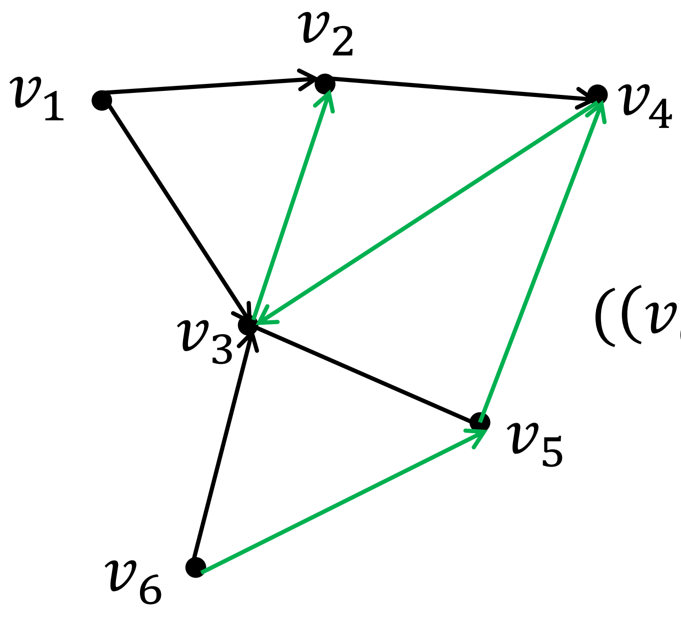
$a_i = (v_j, v_k)$ lub $a_i =< v_j, v_k >$ jest łukiem będącym parą uporządkowaną wierzchołków.



$a_1 = (v_2, v_1)$

Algorytmy grafowe: podstawowe pojęcia

Droga (ścieżka) w grafie skierowanym jest ciągiem łuków $((v_1, v_2), (v_2, v_3), \dots, (v_{k-2}, v_{k-1}), (v_{k-1}, v_k))$, który można wyrazić ciągiem wierzchołków $(v_1, v_2, v_3, \dots, v_{k-2}, v_{k-1}, v_k)$.



$$((v_6, v_5), (v_5, v_4), (v_4, v_3), (v_3, v_2))$$

$$(v_6, v_5, v_4, v_3, v_2)$$

Algorytmy grafowe: podstawowe pojęcia

Rozmiary grafu:

- Liczba wierzchołków n,
- Liczba krawędzi (łuków) m.

Algorytmy grafowe: podstawowe pojęcia

Graf nieskierowany z wagami krawędzi jest trójką uporządkowaną $G = (V, E, W)$, gdzie:

R jest zbiorem liczb rzeczywistych,

$W: E \rightarrow R$ jest funkcją wagi krawędzi.

Graf nieskierowany z wagami wierzchołków jest trójką uporządkowaną $G = (V, E, W)$, gdzie

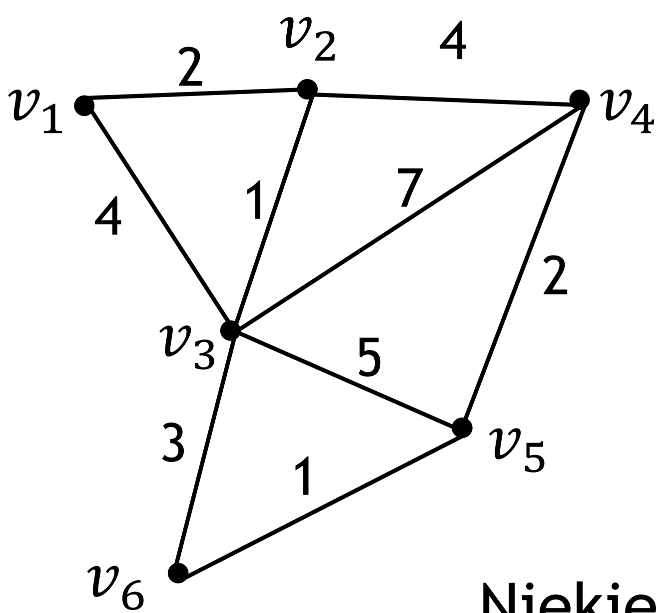
$W: V \rightarrow R$ jest funkcją wagi wierzchołków.

Graf skierowany z wagami łuków jest trójką uporządkowaną $G = (V, A, W)$, gdzie:

$W: A \rightarrow R$ jest funkcją wagi łuku.

Algorytmy grafowe: podstawowe pojęcia

Przykład grafu nieskierowanego z wagami krawędzi i jego macierz wag.



$$\begin{array}{c c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} &
\begin{bmatrix}
\infty & 2 & 4 & \infty & \infty & \infty \\
2 & \infty & 1 & 4 & \infty & \infty \\
4 & 1 & \infty & 7 & 5 & 3 \\
\infty & 4 & 7 & \infty & 2 & \infty \\
\infty & \infty & 5 & 2 & \infty & 1 \\
\infty & \infty & 3 & \infty & 1 & \infty
\end{bmatrix}
\end{array}$$

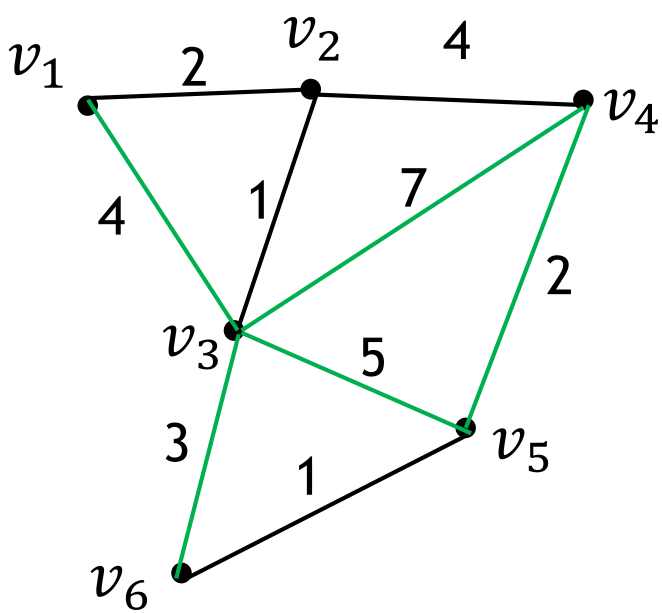
Niekiedy na przekątnej wpisywane są inne wartości wyróżnione.

Algorytmy grafowe: podstawowe pojęcia

Wagą (długością) drogi w grafie nieskierowanym jest suma wag krawędzi tej drogi.

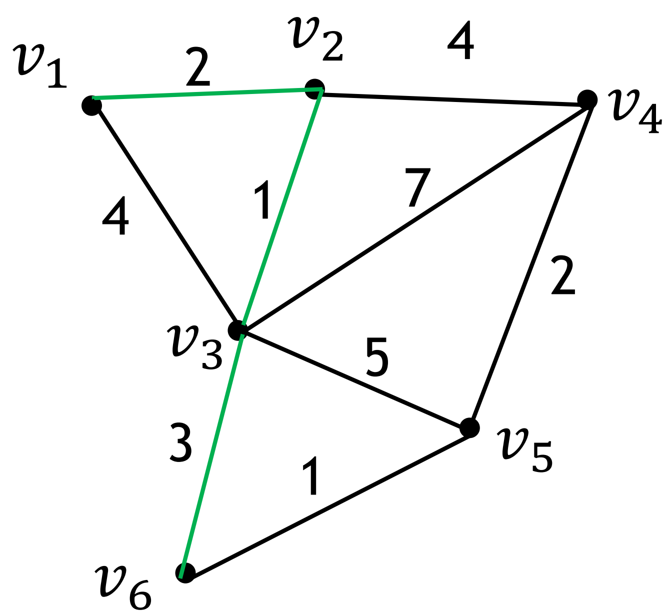
Wagą (długością) drogi w grafie skierowanym jest suma wag łuków tej drogi.

Algorytmy grafowe: podstawowe pojęcia



Waga drogi między wierzchołkami v_1 a v_6 równa 21

Algorytmy grafowe: podstawowe pojęcia



Waga drogi między wierzchołkami v_1 a v_6 równa 6

Algorytmy grafowe: podstawowe pojęcia

Graf nieskierowany jest **spójnym**, jeśli istnieje droga między każdą parą jego wierzchołków.

Graf skierowany jest **spójnym**, jeśli jego wersja nieskierowana jest grafem spójnym.

Drzewo nieskierowane jest grafem nieskierowanym spójnym i acyklicznym.

$|E| = |V| - 1.$

Dołączenie krawędzi do drzewa nieskierowanego tworzy cykl. Usunięcie krawędzi z drzewa nieskierowanego powoduje jego niespójność.

Algorytmy grafowe: podstawowe pojęcia

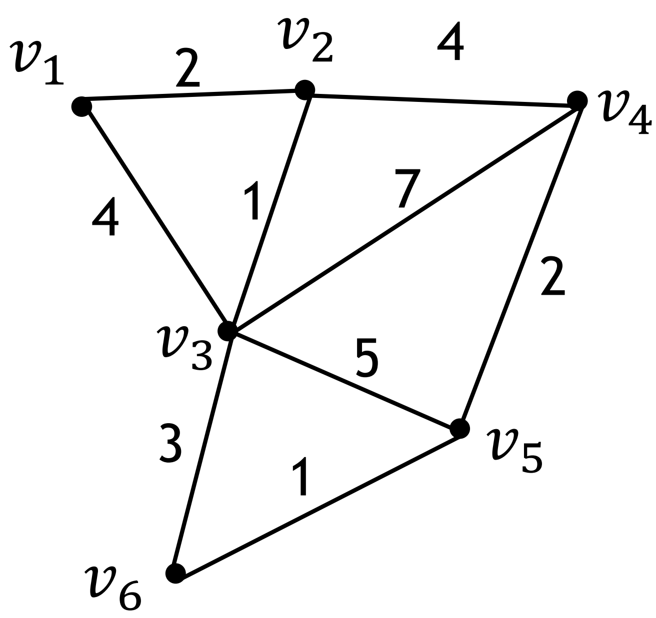
Podgrafem grafu nieskierowanego $G = (V, E)$ jest taki graf, że $V' \subset V$ i $E' \subset E$.

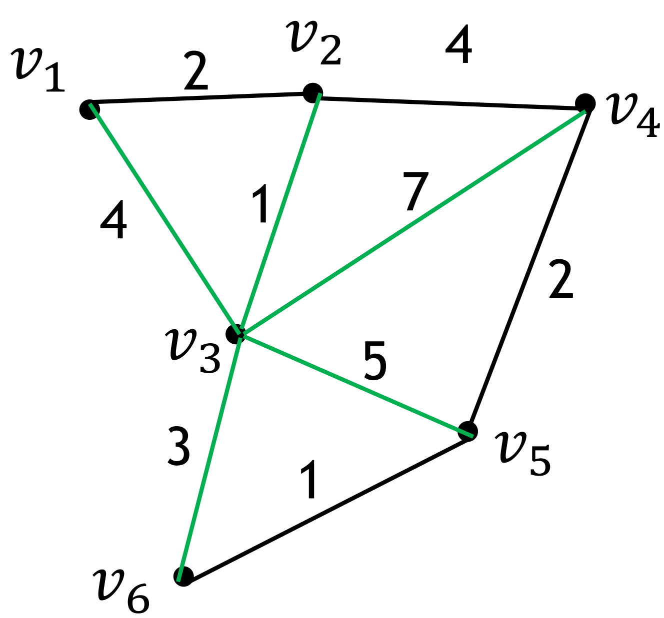
Drzewo rozpinające nieskierowanego grafu spójnego $G = (V, E)$ jest podgrafem $S = (V^S, E^S)$ spójnym będącym drzewem takim, że $V^S = V$.

Minimalne drzewo rozpinające grafu nieskierowanego z wagami jest drzewem rozpinającym o minimalnej sumie wag.

Algorytmy grafowe: podstawowe pojęcia

Przykład drzewa rozpinającego nieskierowanego grafu spójnego

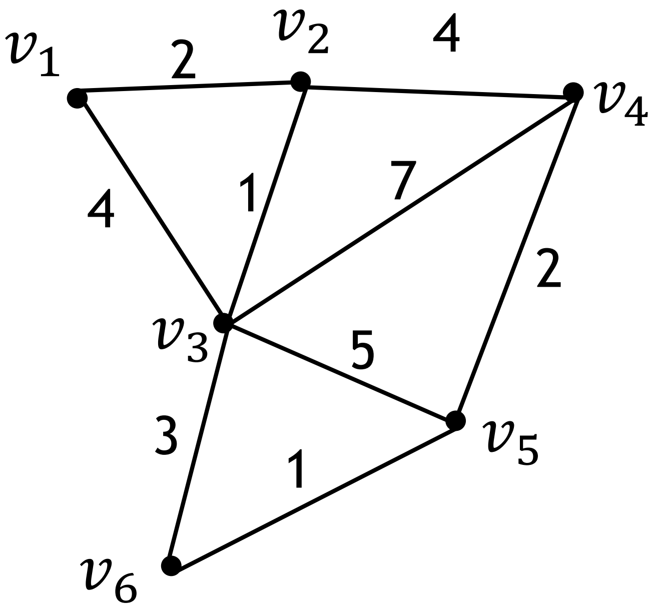


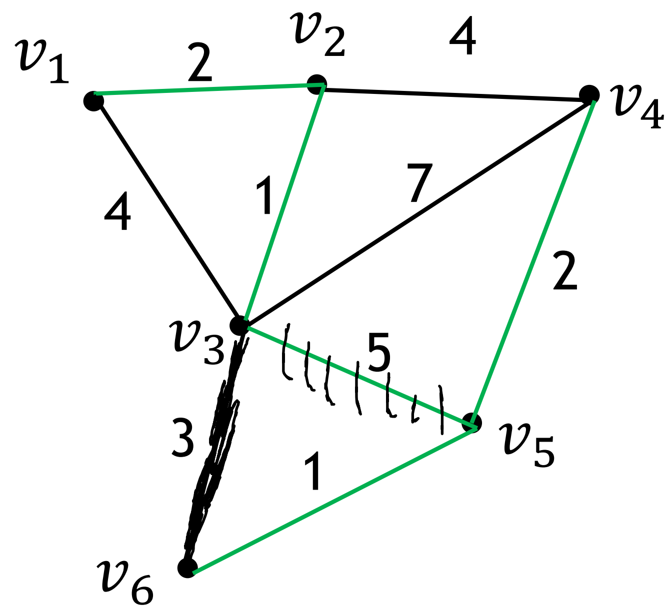


Waga drzewa rozpinającego 20

Algorytmy grafowe: podstawowe pojęcia

Przykład drzewa rozpinającego nieskierowanego grafu spójnego





Waga drzewa rozpinającego 11

Algorytmy grafowe: reprezentacja grafów

Kategorie grafów nieskierowanych wyróżnionych ze względu na liczbę krawędzi względem liczby wierzchołków grafu pełnego:

- Rzadki, gdy $|E| \ll |V|^2$,
- Gęsty, gdy $|E|$ bliskie $|V|^2$.

Reprezentacja grafów:

Rzadkich - raczej za pomocą list,

Gęstych - raczej macierzowa.

Reprezentacja macierzowa daje szybszy dostęp. Oszczędność pamięci można uzyskać przez pamiętanie ośmiu składowych macierzy w jednym bajcie.

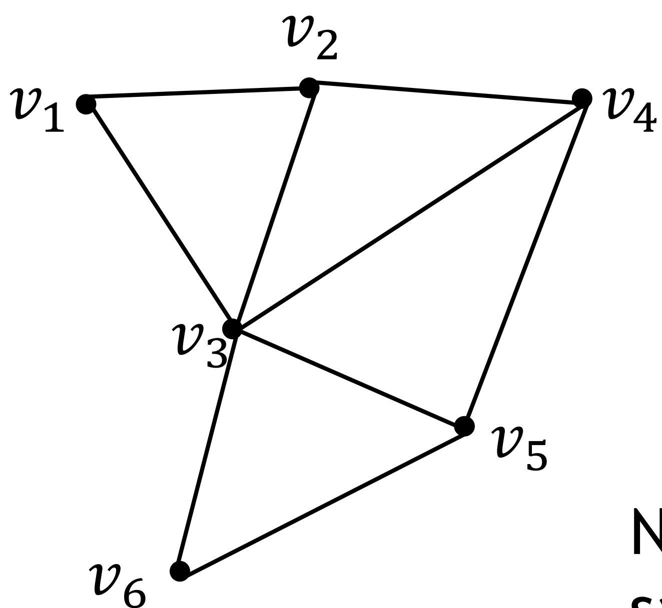
Algorytmy grafowe: podstawowe pojęcia

Rozmiary grafu:

- Liczba wierzchołków n,
- Liczba krawędzi (łuków) m.

Algorytmy grafowe: reprezentacja grafów

Graf nieskierowany i jego macierz sąsiedztwa.



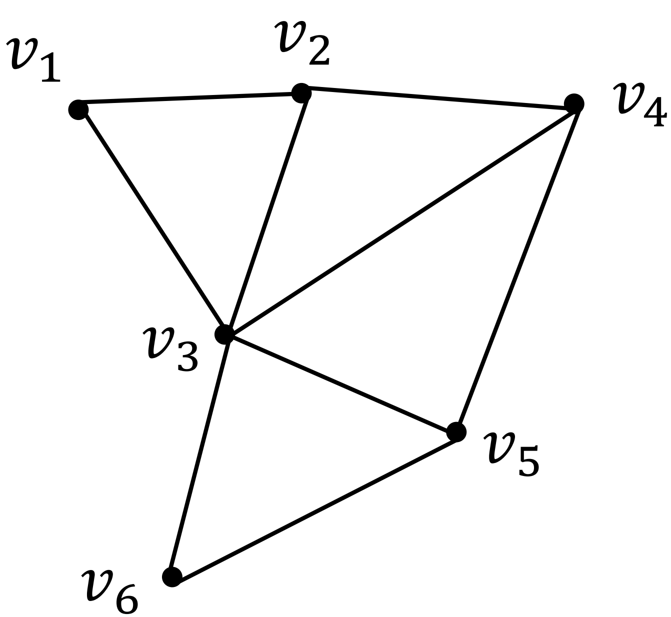
$$
\begin{array}{c c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} &
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0
\end{bmatrix}
\end{array}
$$

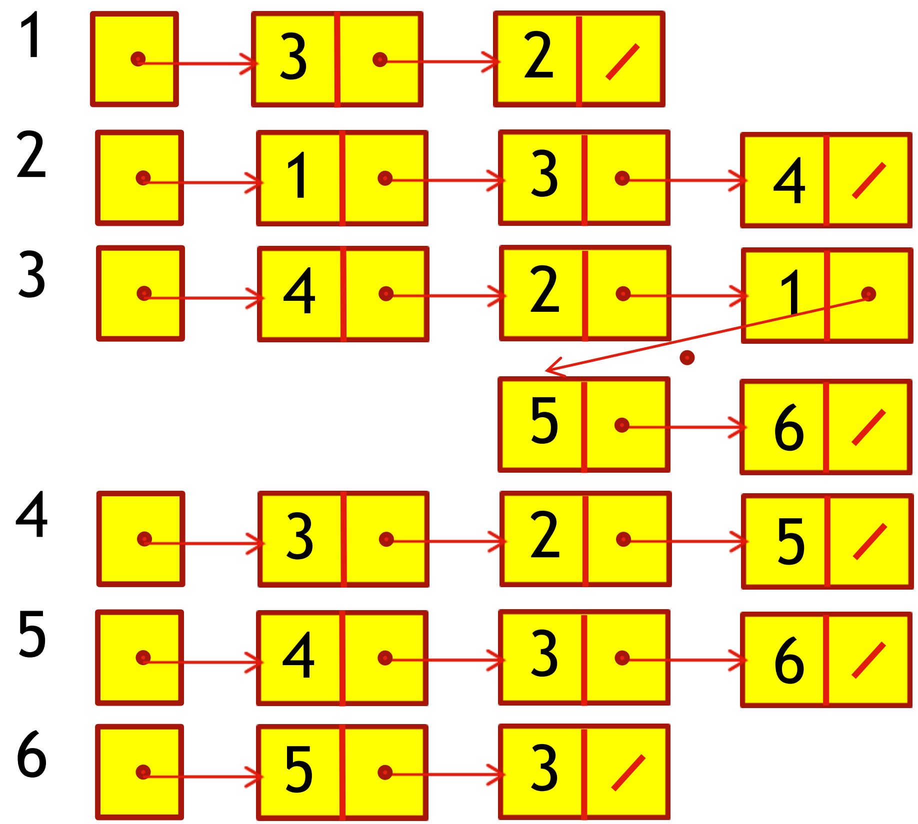
Niekiedy na przekątnej wpisywane są inne wartości wyróżnione.

Zajętość pamięci $O(n^2)$

Algorytmy grafowe: reprezentacja grafów

Graf nieskierowany i jego listy sąsiedztwa

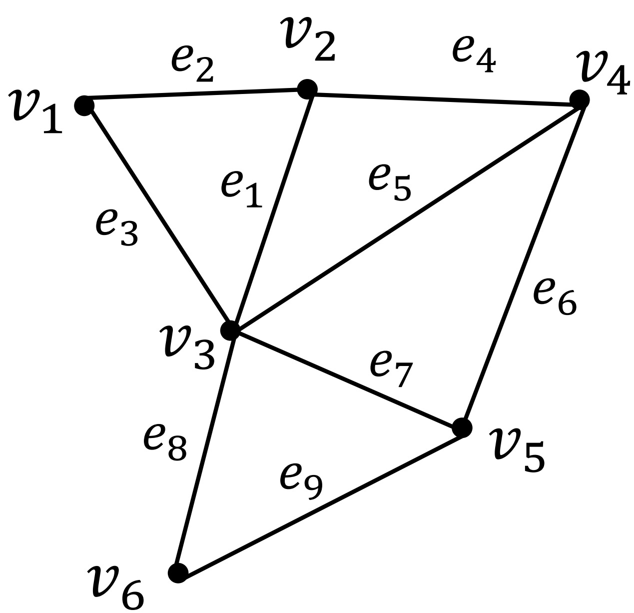




Zajętość pamięci $O(n+m)$

Algorytmy grafowe: reprezentacja grafów

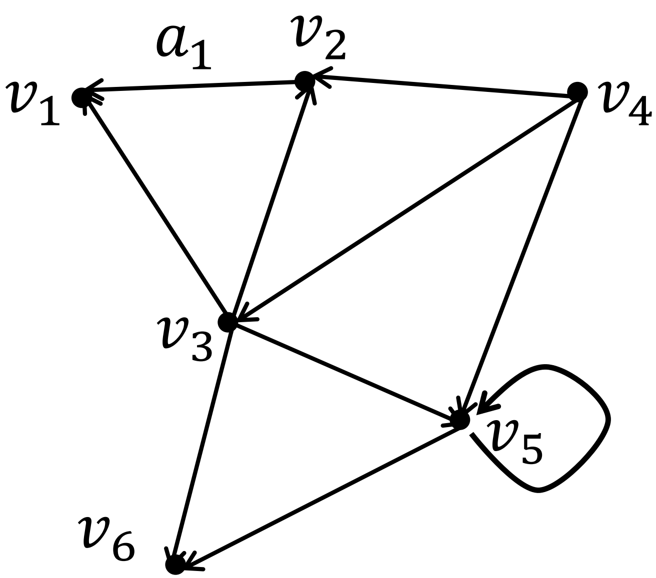
Graf nieskierowany i jego macierz incydencji.



$$\begin{array}{c c}
 & \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}
\end{array}$$

Algorytmy grafowe: reprezentacja grafów

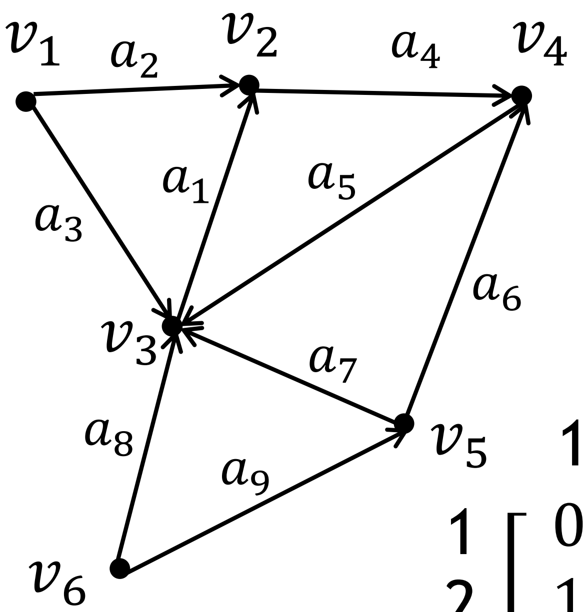
Graf skierowany i jego macierz sąsiedztwa



$$
\begin{array}{c c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} &
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Algorytmy grafowe: reprezentacja grafów

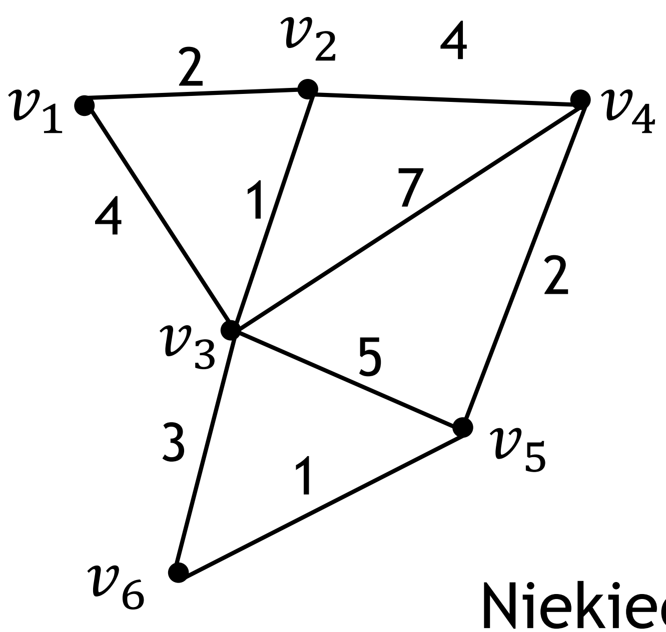
Graf skierowany i jego macierz incydencji



$$
\begin{array}{c|c}
 & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{matrix} \\
\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} &
\begin{bmatrix}
0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1
\end{bmatrix}
\end{array}
$$

Algorytmy grafowe: reprezentacja grafów

Przykład grafu nieskierowanego z wagami krawędzi i jego macierz wag.

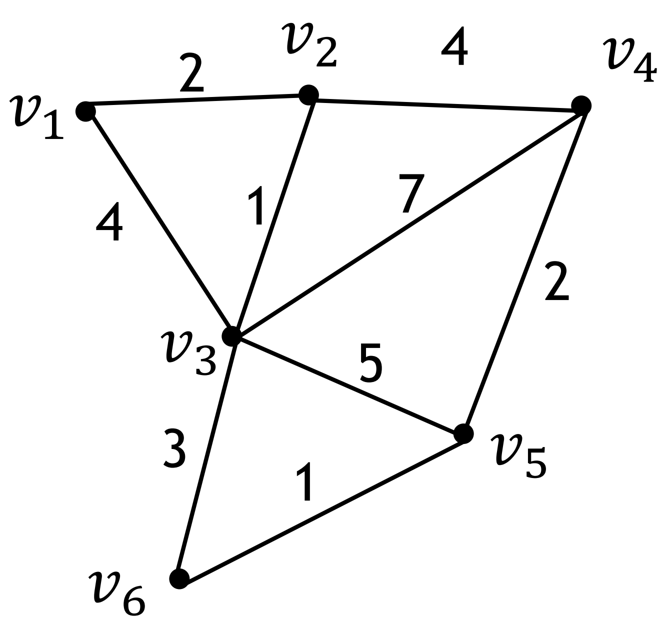


$$
\begin{array}{c c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} &
\begin{bmatrix}
\infty & 2 & 4 & \infty & \infty & \infty \\
2 & \infty & 1 & 4 & \infty & \infty \\
4 & 1 & \infty & 7 & 5 & 3 \\
\infty & 4 & 7 & \infty & 2 & \infty \\
\infty & \infty & 5 & 2 & \infty & 1 \\
\infty & \infty & 3 & \infty & 1 & \infty
\end{bmatrix}
\end{array}
$$

Niekiedy na przekątnej wpisywane są inne wartości wyróżnione.

Algorytmy grafowe: podstawowe pojęcia

Przykład grafu nieskierowanego z wagami krawędzi i jego lista krawędzi w postaci tablicy.



$ind(v_i)$	$ind(v_j)$	$W(\{v_i, v_j\})$
1	2	2
1	3	4
2	3	1
2	4	4
3	4	7
3	5	5
4	5	2
3	6	3
5	6	1

Zajętość pamięci $O(m)$

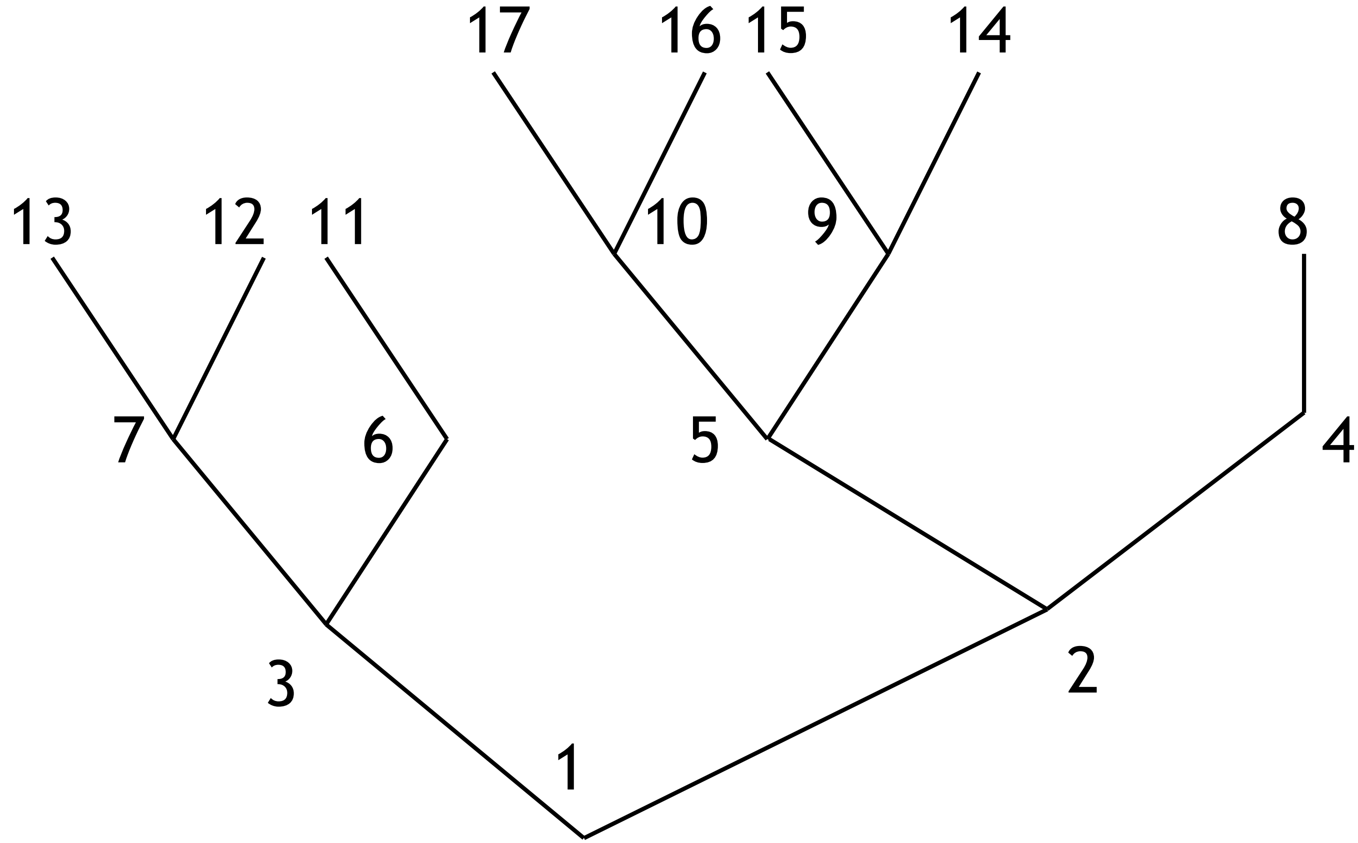
Algorytmy grafowe: metody przeszukiwania

Metody przeszukiwania:

- Wszerz (ang. Breadth-First Search),
- W głąb (ang. Depth-First Search).

Algorytmy grafowe: metody przeszukiwania

Kolejność odwiedzania wierzchołków w BFS działającej na drzewie



Algorytmy grafowe: metody przeszukiwania

Przeszukiwanie wszerz

Kolory (ang. *color*) wierzchołków:

- Biały (nieodwiedzony),
- Szary (odwiedzony, nie wszyscy sąsiedzi tegoż odwiedzeni),
- Czarny (odwiedzeni: ten i jego sąsiedzi).

Metoda wykonywana z:

- **jednego źródła**,
- wielu źródeł.

```
BFS(G, s)
for każdy wierzchołek u ∈ V[G] − {s}
  do color[u] ← BIAŁY
     d[u] ← ∞
     π[u] ← NIL
color[s] ← SZARY
d[s] ← 0
π[s] ← NIL
Q ← ∅
ENQUEUE(Q, s)
while Q ≠ ∅
  do u ← DEQUEUE(Q)
     for każdego v ∈ Adj[u]
        do if color[v] = BIAŁY
              then color[v] ← SZARY
                   d[v] ← d[u] + 1
                   π[v] ← u
                   ENQUEUE(Q, v)
     color[u] ← CZARNY
```

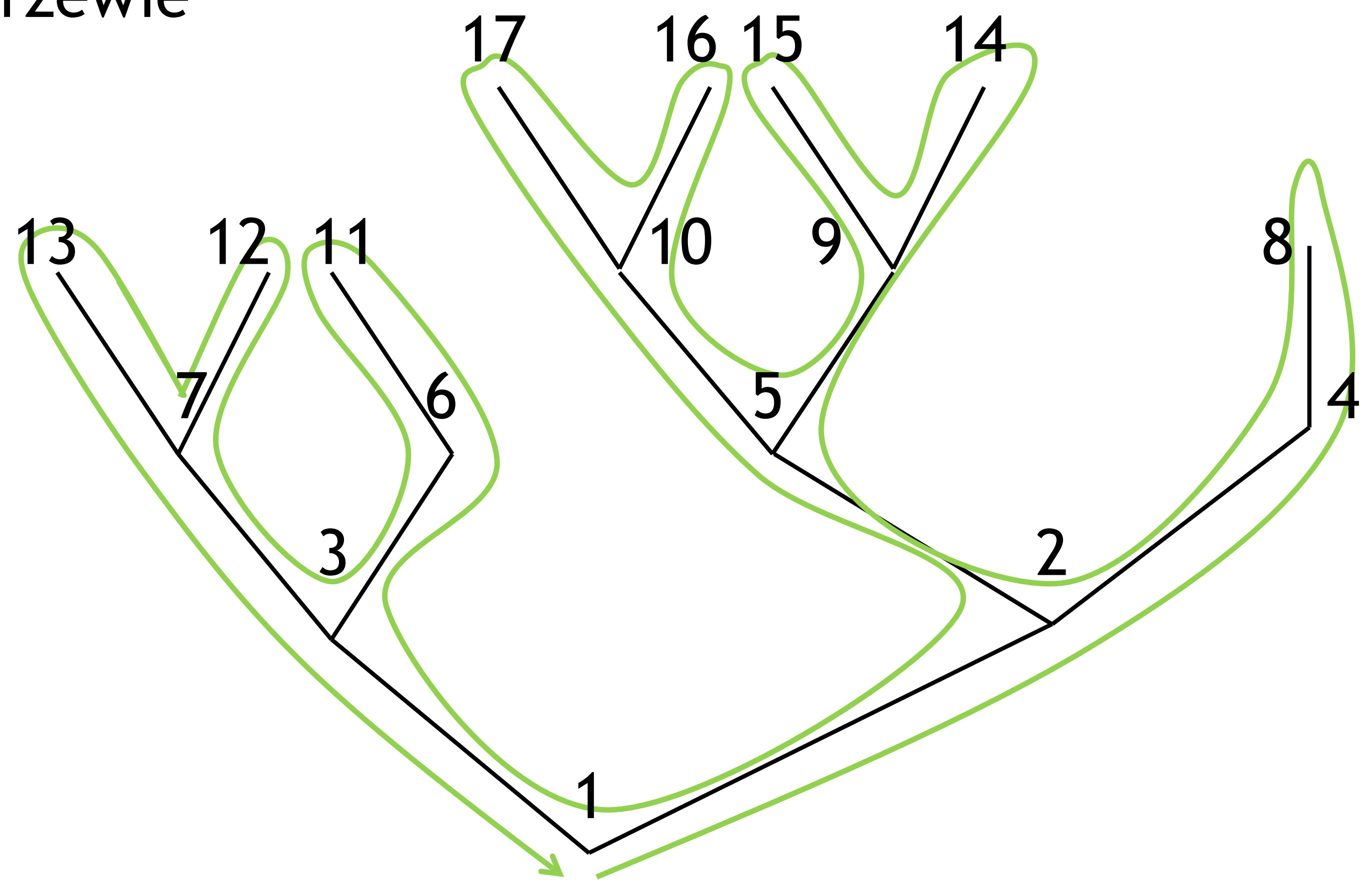
$d[u]$ – zmienna przechowująca odległość od źródła s do wierzchołka u – najkrótszą odległość daną liczbą krawędzi, $d[s] = 0$, $\pi[u]$ – zmienna wskazująca wierzchołek, z którego wierzchołek u został odwiedzony, u=NIL, jeśli $u = s$ lub u nie został jeszcze odwiedzony.

$$O(|V| + |E|)$$

[CLRS, Wprowadzenie do algorytmów]

Algorytmy grafowe: metody przeszukiwania

Kolejność odwiedzania wierzchołków w DFS działającej na drzewie



Algorytmy grafowe: metody przeszukiwania

Przeszukiwanie w głąb

Kolory (ang. *color*) wierzchołków:

- Biały (nieodwiedzony),
- Szary (odwiedzony, nie wszyscy sąsiedzi tegoż odwiedzeni),
- Czarny (odwiedzeni: ten i jego sąsiedzi).

Metoda wykonywana z:

- jednego źródła,
- **wielu źródeł**.

```
DFS(G)
for każdy wierzchołek u ∈ V[G]
  do color[u] ← BIAŁY
     π[u] ← NIL
for każdy wierzchołek u ∈ V[G]
  do if color[u] = BIAŁY
        then DFS-VISIT(u)
```

```
DFS-VISIT(u)
color[u] ← SZARY          ▷ Biały wierzchołek u został właśnie odwiedzony.
for każdy v ∈ Adj[u]      ▷ Zbadaj krawędź (u, v).
  do if color[v] = BIAŁY
        then π[v] ← u
             DFS-VISIT(v)
color[u] ← CZARNY         ▷ Pokoloruj u na czarno; został przetworzony.
```

$O(|V| + |E|)$

[CLRS, Wprowadzenie do algorytmów]

Algorytmy grafowe: minimalne drzewa rozpinające

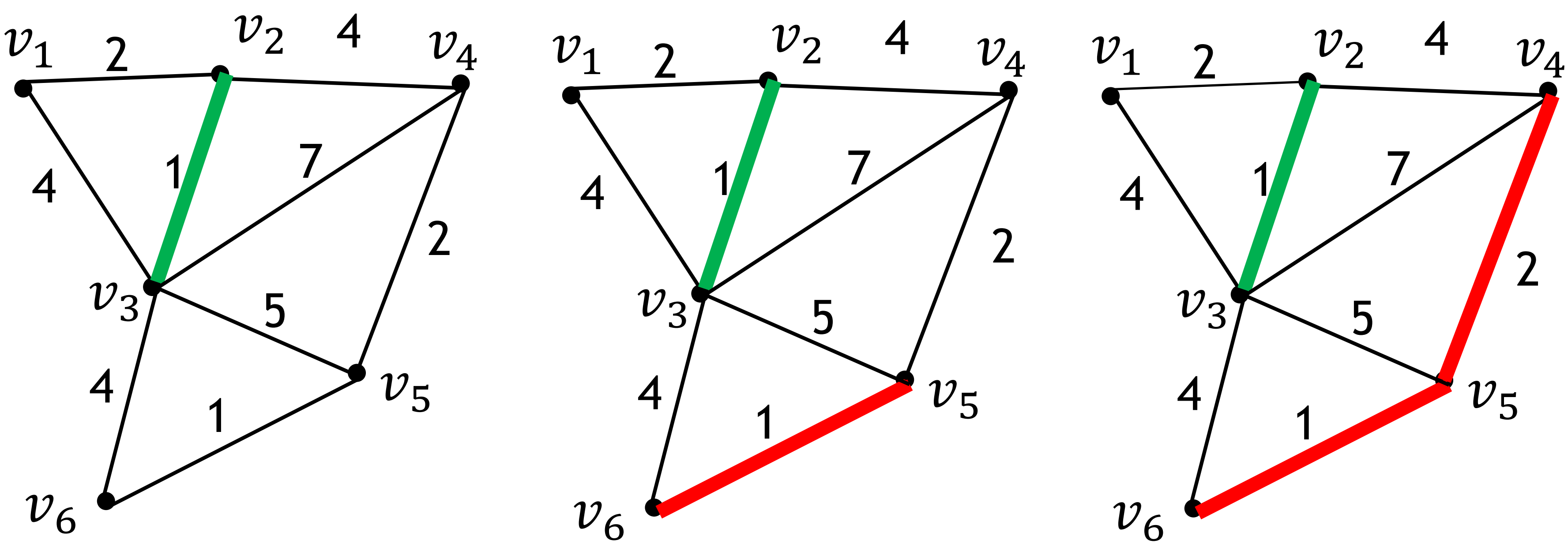
Drzewo rozpinające nieskierowanego grafu spójnego $G = (V, E)$ jest podgrafem $S = (V^S, E^S)$ spójnym będącym drzewem takim, że $V^S = V$.

Minimalne drzewo rozpinające grafu nieskierowanego z wagami krawędzi jest drzewem rozpinającym o minimalnej sumie wag krawędzi.

Algorytmy grafowe: minimalne drzewa rozpinające

Idea algorytmu Kruskala: z posortowanej niemalejąco wg wag listy krawędzi – dołączane są kolejne, jeśli nie tworzą cyklu (algorytm zachłanny).

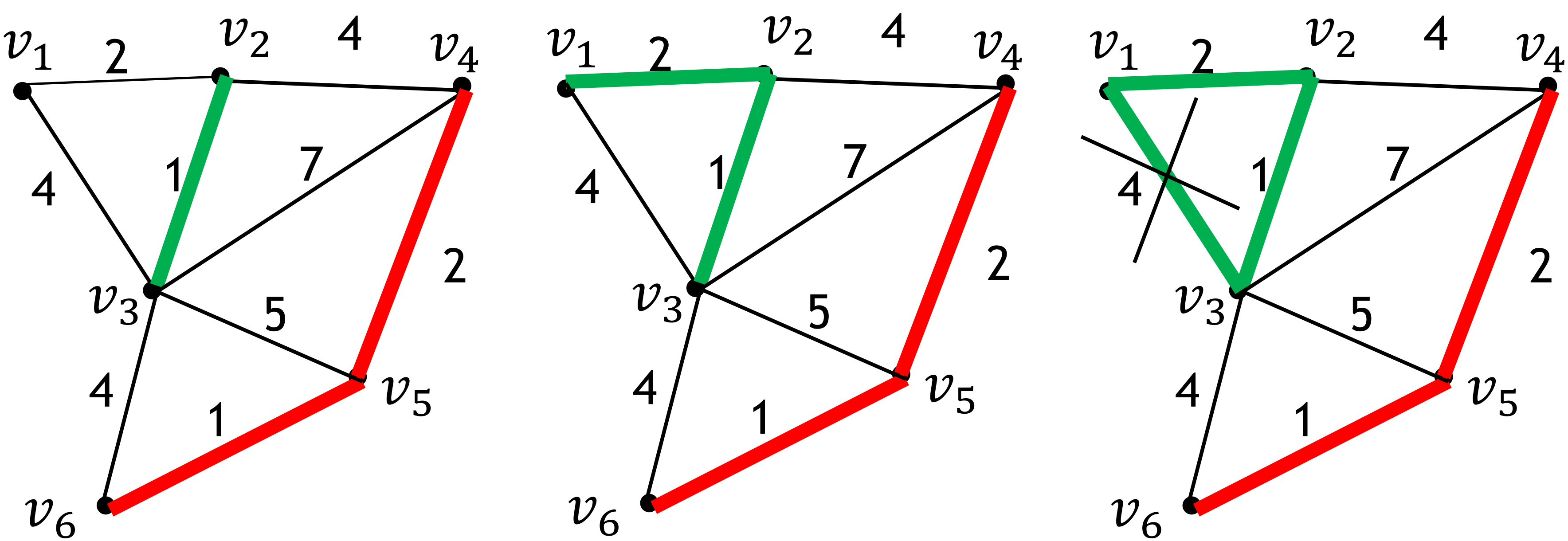
Lista uporządkowana: $\{v_2,v_3\}$ $\{v_5,v_6\}$ $\{v_5,v_4\}$ $\{v_1,v_2\}$ $\{v_1,v_3\}$ $\{v_2,v_4\}$ $\{v_6,v_3\}$ $\{v_5,v_3\}$ $\{v_4,v_3\}$



Algorytmy grafowe: minimalne drzewa rozpinające

Idea algorytmu Kruskala: z posortowanej niemalejąco wg wag listy krawędzi – dołączane są kolejne, jeśli nie tworzą cyklu (algorytm zachłanny).

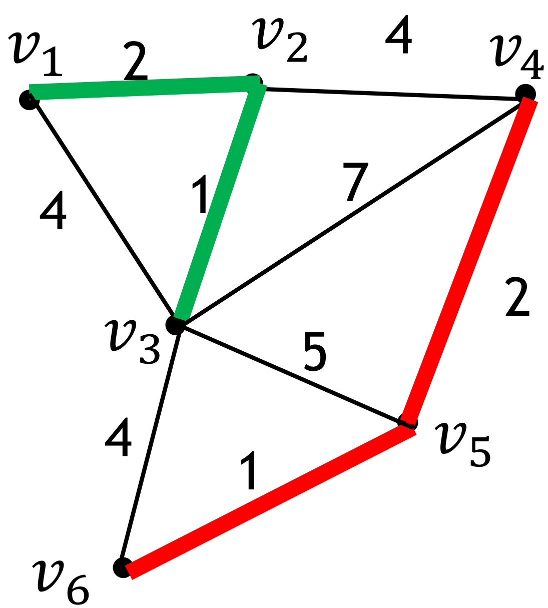
Lista uporządkowana: $\{v_2,v_3\}$ $\{v_5,v_6\}$ $\{v_5,v_4\}$ $\{v_1,v_2\}$ $\{v_1,v_3\}$ $\{v_2,v_4\}$ $\{v_6,v_3\}$ $\{v_5,v_3\}$ $\{v_4,v_3\}$

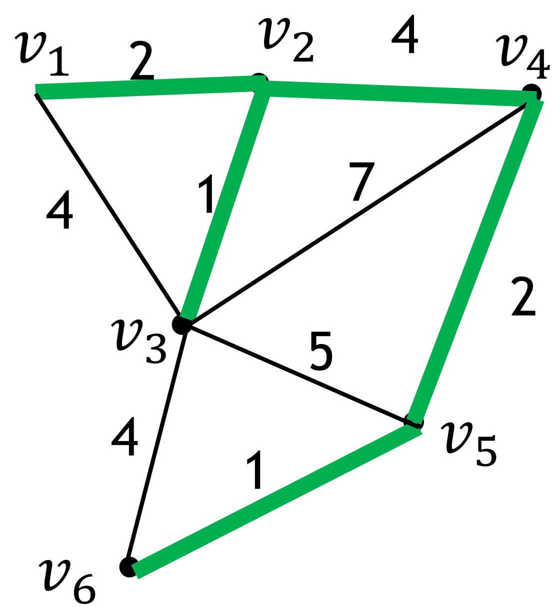


Algorytmy grafowe: minimalne drzewa rozpinające

Idea algorytmu Kruskala: z posortowanej niemalejąco wg wag listy krawędzi – dołączane są kolejne, jeśli nie tworzą cyklu (algorytm zachłanny).

Lista uporządkowana: $\{v_2, v_3\}$ $\{v_5, v_6\}$ $\{v_5, v_4\}$ $\{v_1, v_2\}$ $\{v_1, v_3\}$ $\{v_2, v_4\}$ $\{v_6, v_3\}$ $\{v_5, v_3\}$ $\{v_4, v_3\}$





Algorytmy grafowe: minimalne drzewa rozpinające

Idea algorytmu Kruskala:

1. Wszystkie wierzchołki zostają parami różnie pokolorowane tworząc jednowierzchołkowe drzewa,
2. Z posortowanej niemalejąco wg wag listy krawędzi – dołączane są kolejne, jeśli nie tworzą cyklu (algorytm zachłanny),
3. Dla dwu drzew łączonych w jedno ujednolicane zostają kolory.

Algorytmy grafowe: minimalne drzewa rozpinające

Idea algorytmu Kruskala: z posortowanej niemalejąco wg wag listy krawędzi – dołączane są kolejne, jeśli nie tworzą cyklu (algorytm zachłanny).

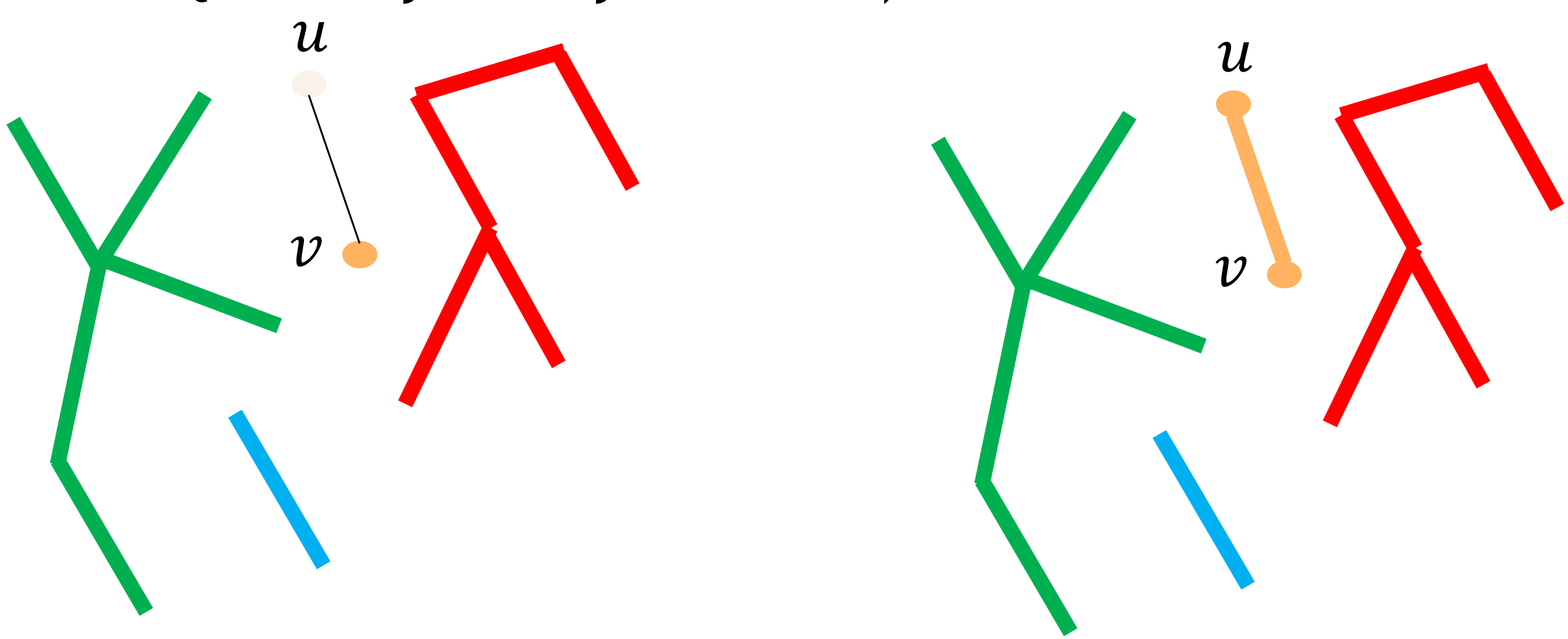
Przy próbie dołączenia krawędzi (u, v) występują następujące przypadki:

1. Wierzchołki u, v nie zostały wybrane (nie dołączono krawędzi incydentnych z nimi), są różnie pokolorowane,
2. Dokładnie jeden z wierzchołków, przyjmijmy u, został dotychczas wybrany, są różnie pokolorowane,
3. Oba wybrane ale różnie pokolorowane,
4. Oba wybrane ale jednolicie pokolorowane.

Algorytmy grafowe: minimalne drzewa rozpinające

Zał: Na początku wierzchołki o parami różnych kolorach.

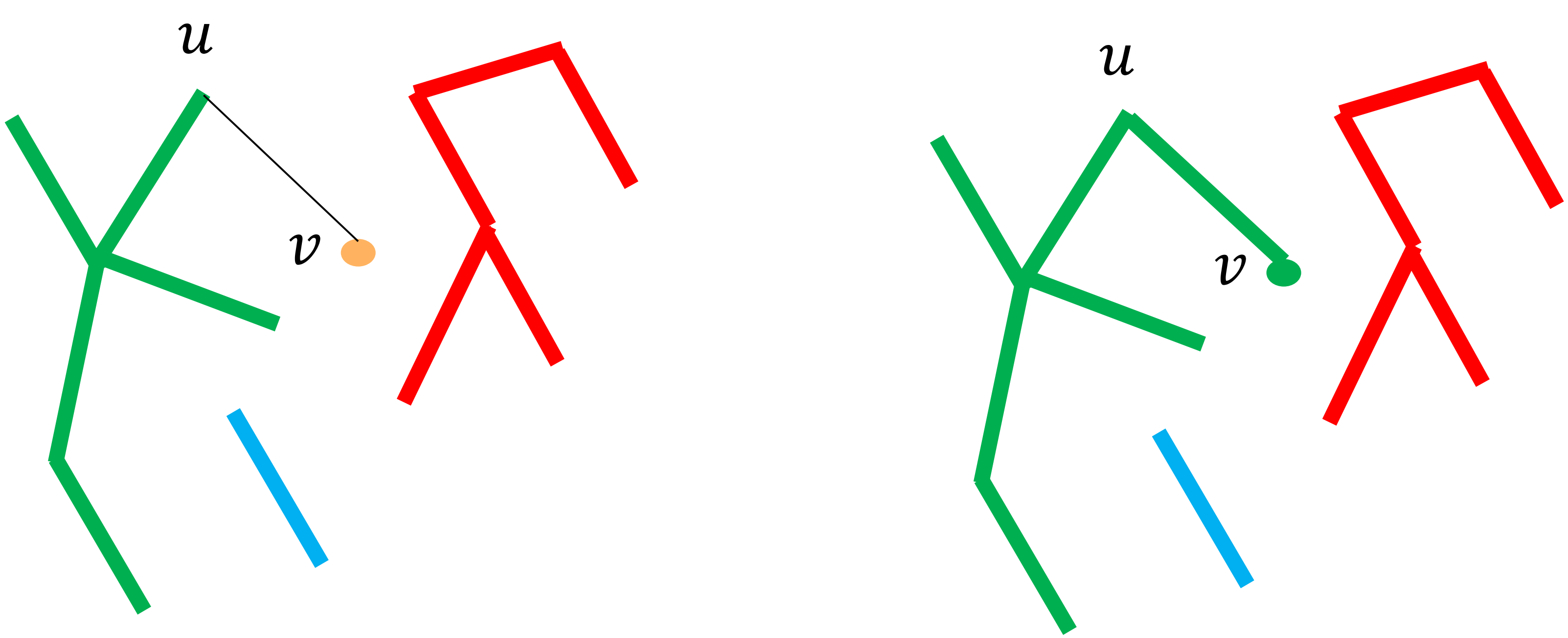
1. Wierzchołki u, v nie zostały dotąd wybrane (nie dołączono krawędzi incydentnych z nimi)



Dołączenie nowej krawędzi nie powoduje cyklu, a ujednolicenie kolorów wierzchołków u, v daje nowe poddrzewo

Algorytmy grafowe: minimalne drzewa rozpinające

2. Dokładnie jeden z wierzchołków, przyjmijmy u, został dotychczas wybrany

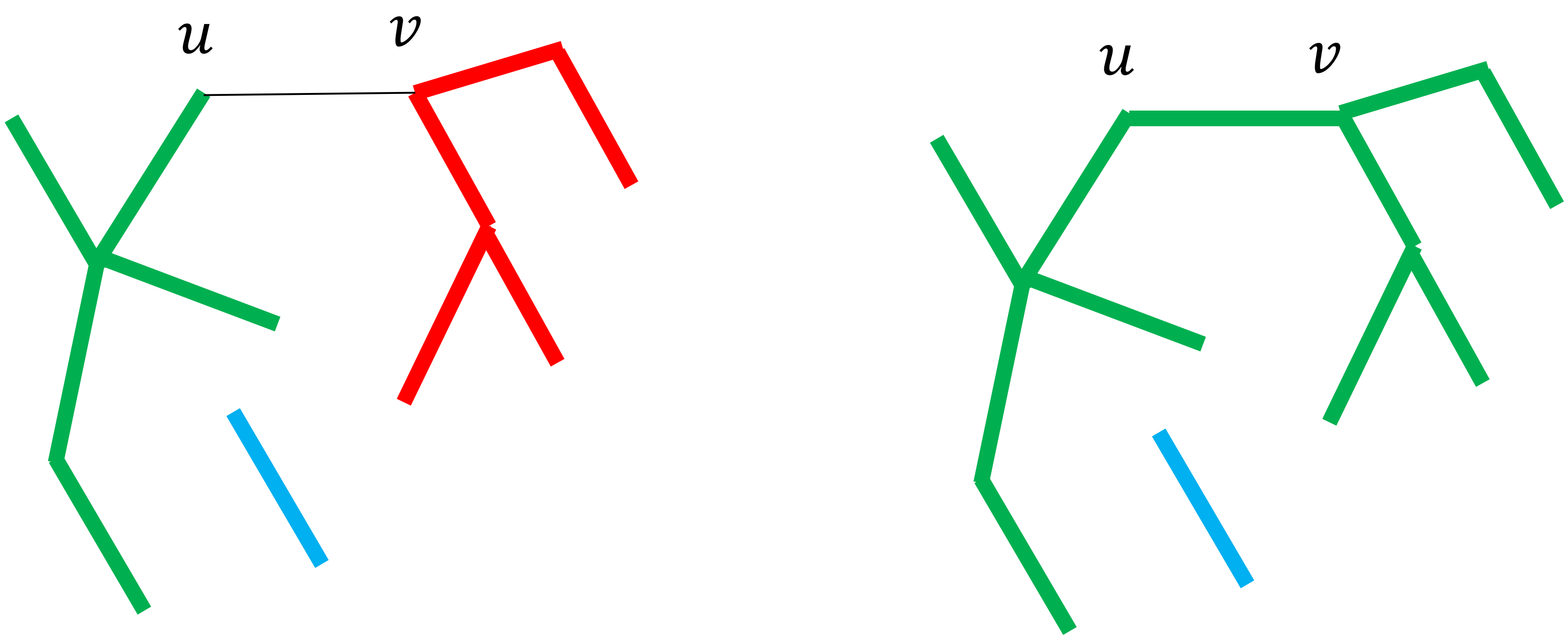


Dołączenie nowej krawędzi nie powoduje cyklu, a ujednolicenie kolorów daje większe poddrzewo

Politechnika Wrocławska

Algorytmy grafowe: minimalne drzewa rozpinające

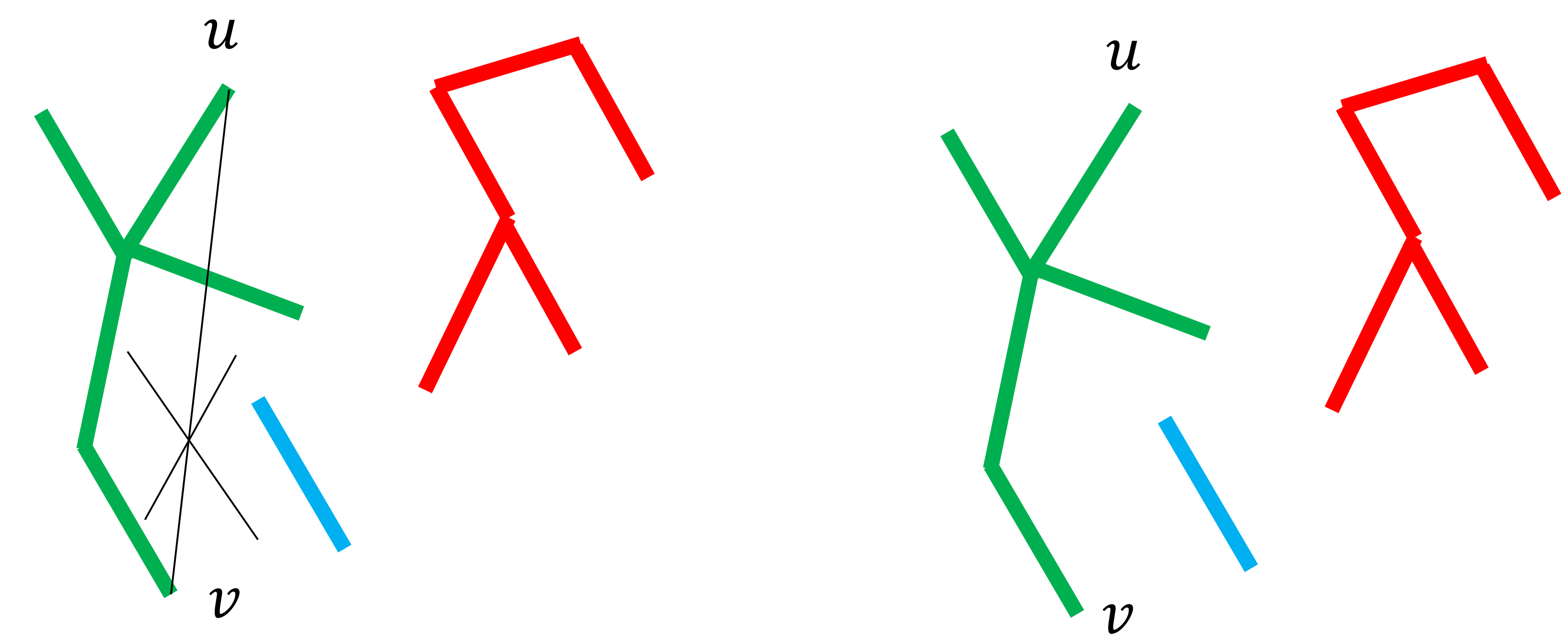
3. Oba wybrane ale różnie pokolorowane



Dołączenie nowej krawędzi nie powoduje cyklu, a ujednolicenie kolorów daje większe poddrzewo

Algorytmy grafowe: minimalne drzewa rozpinające

4. Oba wybrane ale jednolicie pokolorowane



Dołączenie nowej krawędzi spowodowałoby cykl

Algorytmy grafowe: minimalne drzewa rozpinające

Algorytm Kruskala

$A \leftarrow \emptyset$

for każdy wierzchołek $v_i \epsilon V$

do $C(v_i) \leftarrow i$

posortuj krawędzie z E niemalejąco wg wag $W(e_j)$

for każda krawędź $(v_l, v_k) \in E$ w kolejności niemalejących wag

do if $C(v_l) \neq C(v_k)$

then $A \leftarrow A \cup \{(v_l, v_k)\}$

Ujednolicenie kolorów wierzchołków poddrzewa zawierającego v_l, v_k $\leftarrow O(|V|)$

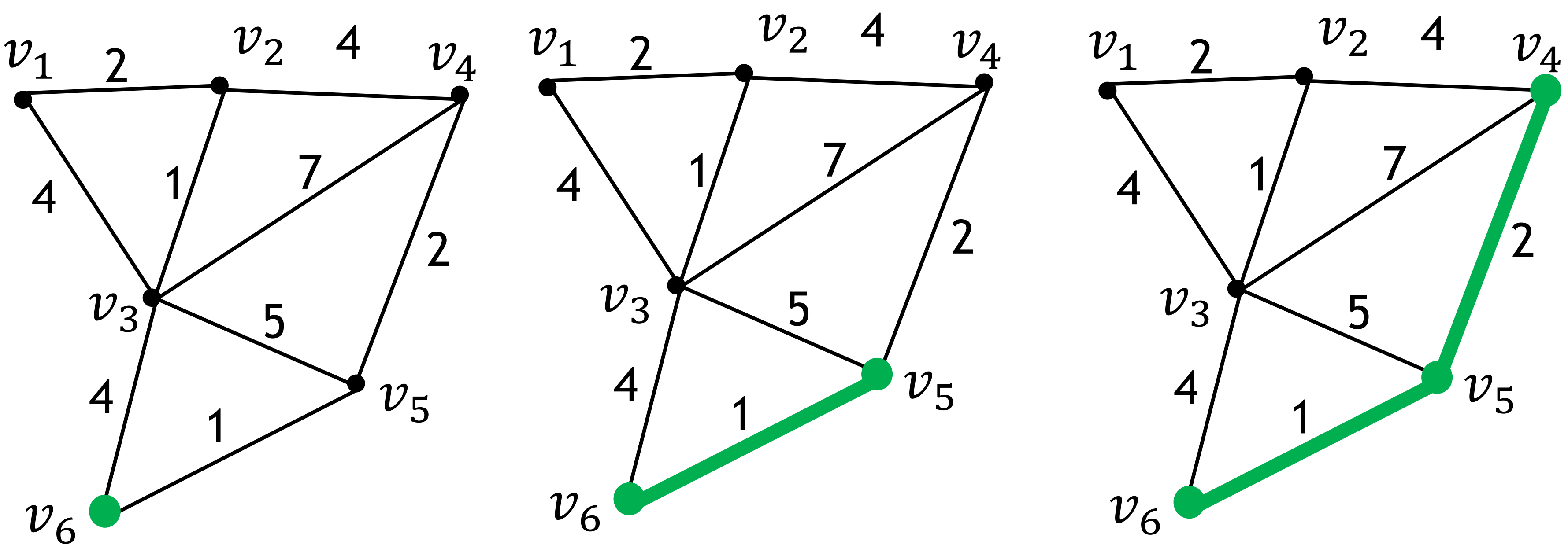
return A

$O(|E| \log|E| + |V| \cdot |E|)$

Algorytmy grafowe: minimalne drzewa rozpinające

Idea algorytmu Prima:

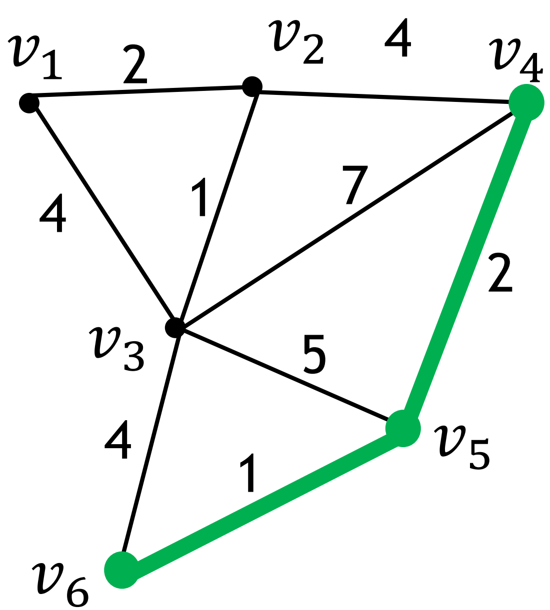
Po wybraniu dowolnego wierzchołka, kolejno dołączany jest najbliższy sąsiad czyli wierzchołek połączony krawędzią o najmniejszej wadze z wcześniej dołączonymi wierzchołkami.

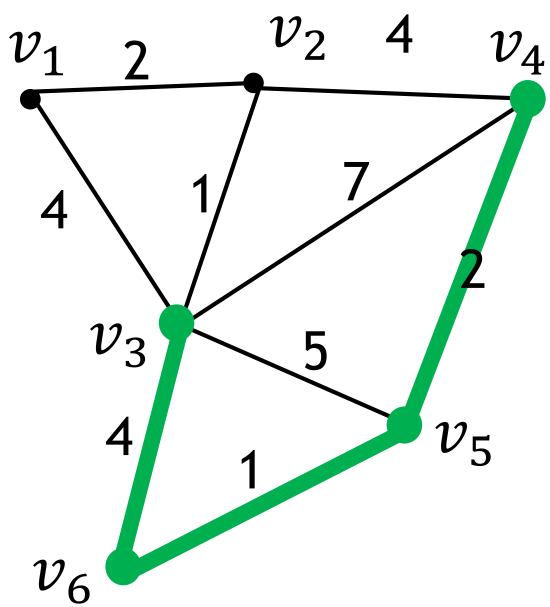


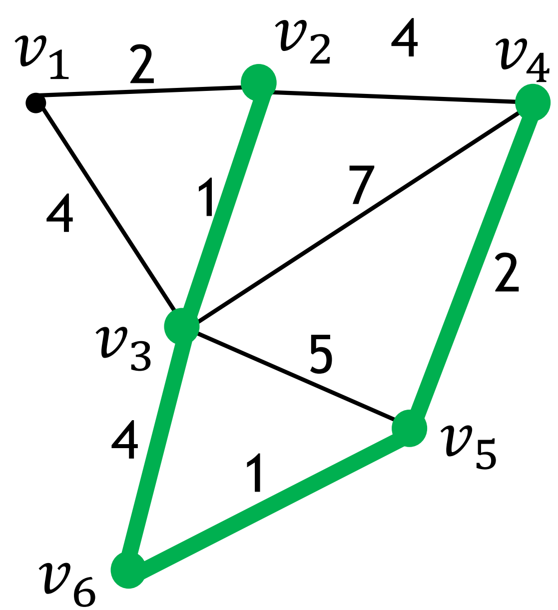
Algorytmy grafowe: minimalne drzewa rozpinające

Idea algorytmu Prima:

Po wybraniu dowolnego wierzchołka, kolejno dołączany jest najbliższy sąsiad czyli wierzchołek połączony krawędzią o najmniejszej wadze z wcześniej dołączonymi wierzchołkami.



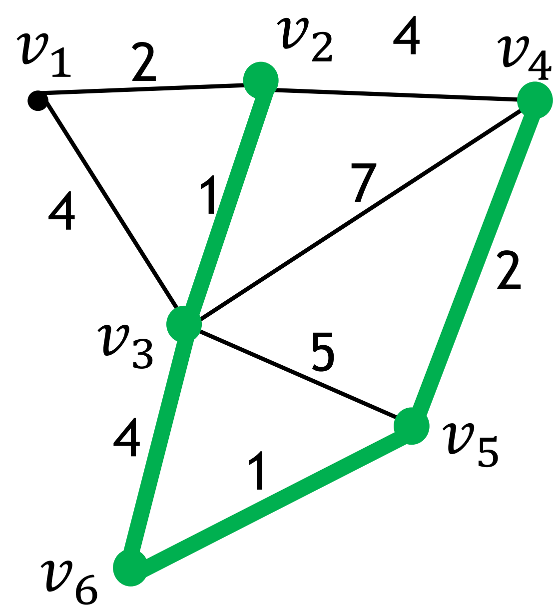


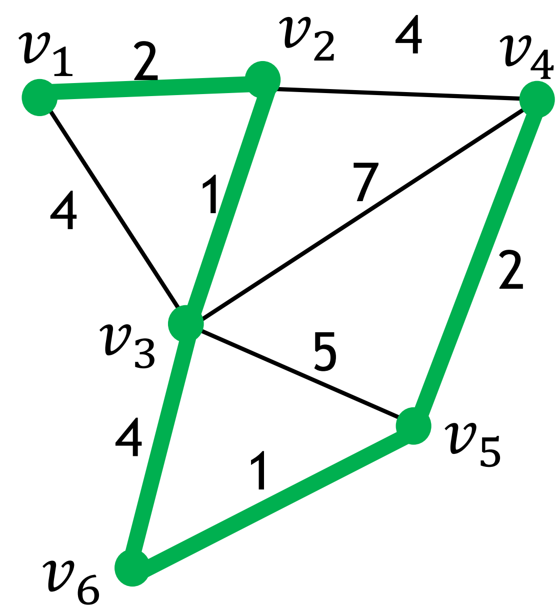


Algorytmy grafowe: minimalne drzewa rozpinające

Idea algorytmu Prima:

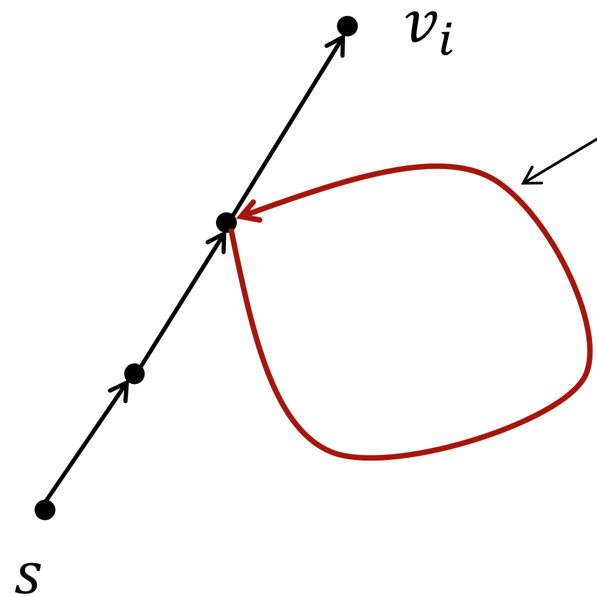
Po wybraniu dowolnego wierzchołka, kolejno dołączany jest najbliższy sąsiad czyli wierzchołek połączony krawędzią o najmniejszej wadze z wcześniej dołączonymi wierzchołkami.





Algorytmy grafowe: problemy ścieżkowe

Cykle o ujemnej wadze (długości)



Cykl o ujemnej wadze

Długość najkrótszej ścieżki z s do v_i dąży do $-\infty$.

Algorytmy grafowe: problemy ścieżkowe

Wyznaczanie najkrótszych ścieżek z jednego źródła do wszystkich wierzchołków dla grafów skierowanych:

Wagi łuków mogą być ujemne

Algorytm Bellmana-Forda $O(|V| \cdot |E|)$

Wagi łuków są nieujemne

Algorytm Dijkstry $O((|V| + |E|)\log|V|)$

(Kolejka priorytetowa jako kopiec binarny typu min)

Acykliczny graf skierowany, Wagi łuków mogą być ujemne

Algorytm oparty na sortowaniu topologicznym $O(|V| + |E|)$

DIJKSTRA(G, s)

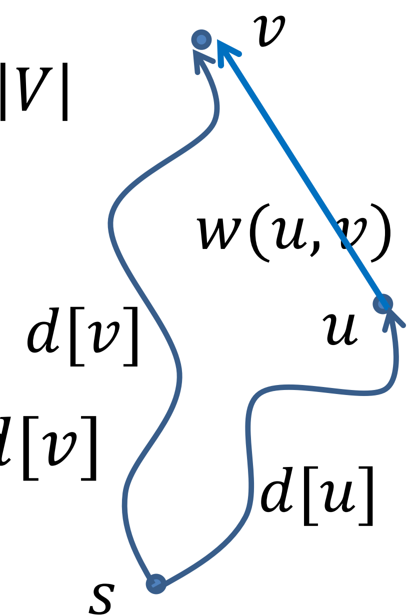
for każdy wierzchołek $v \in V[G]$
  do $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
  do $Q \leftarrow Q \setminus \{$wierzchołek u o minimalnej wartości $d[u]\}$;
    $S \leftarrow S \cup \{u\}$
    for każdy wierzchołek $v \in Adj[u]$
      do if $d[v] > d[u] + w(u, v)$
        then $d[v] \leftarrow d[u] + w(u, v)$
          $\pi[v] \leftarrow u$
          umieść wierzchołek v w Q wg $d[v]$

$d[u]$ – zmienna przechowująca górne ograniczenie odległości od źródła s do wierzchołka u, $d[s] = 0$,
$\pi[u]$ – zmienna wskazująca wierzchołek, z którego wierzchołek u został odwiedzony przy wyznaczaniu tego ograniczenia,
Q – kolejka priorytetowa z kryterium min $d[u]$,
$i(u)$ – pozycja u w tablicowej implementacji kopca,
$u = (name(u), d[u], \pi(u), i(u))$,
S – rozwiązanie

$\log|V|$

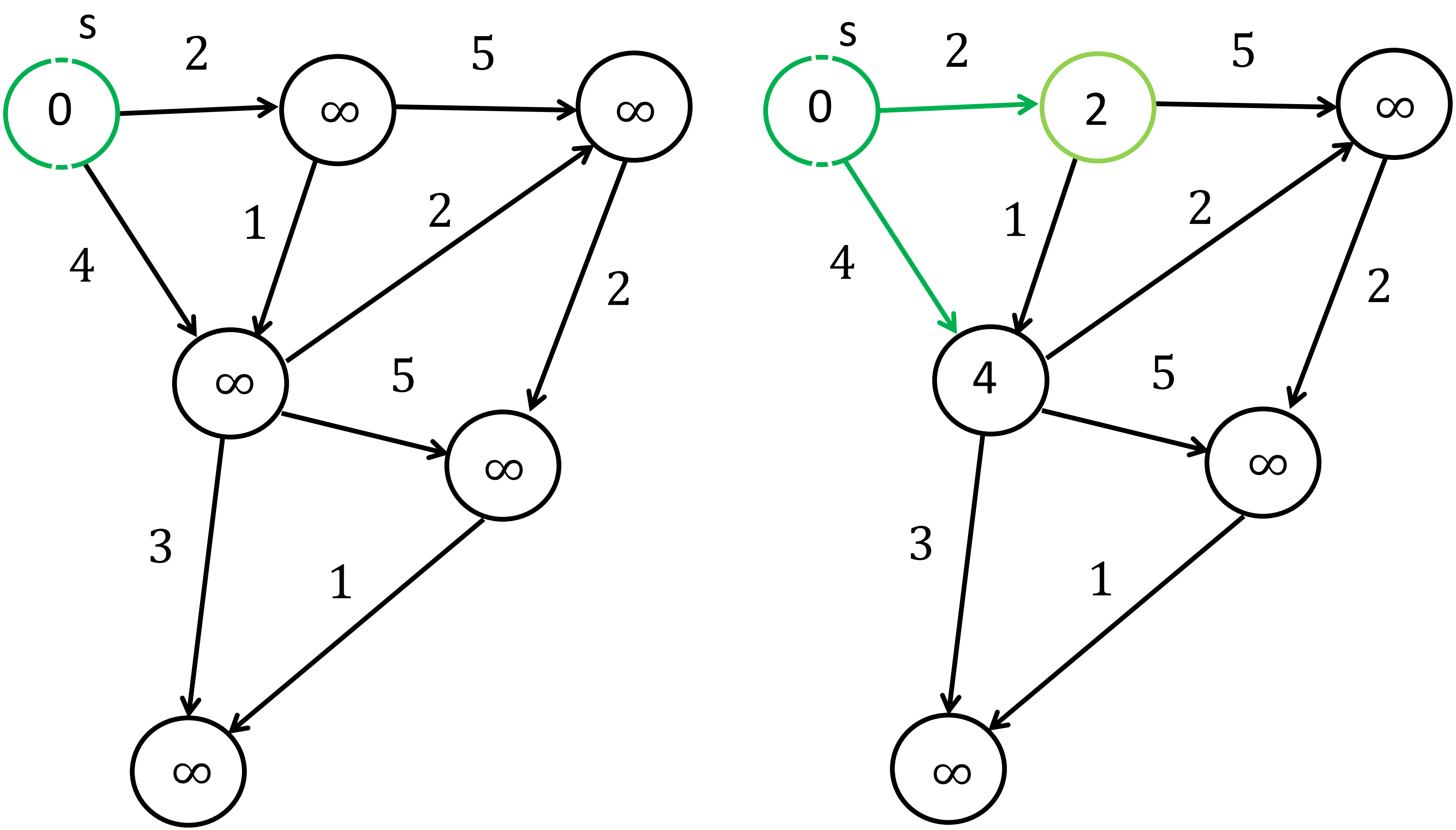
Operacje na kolejce priorytetowej (jako kopiec binarny typu min)

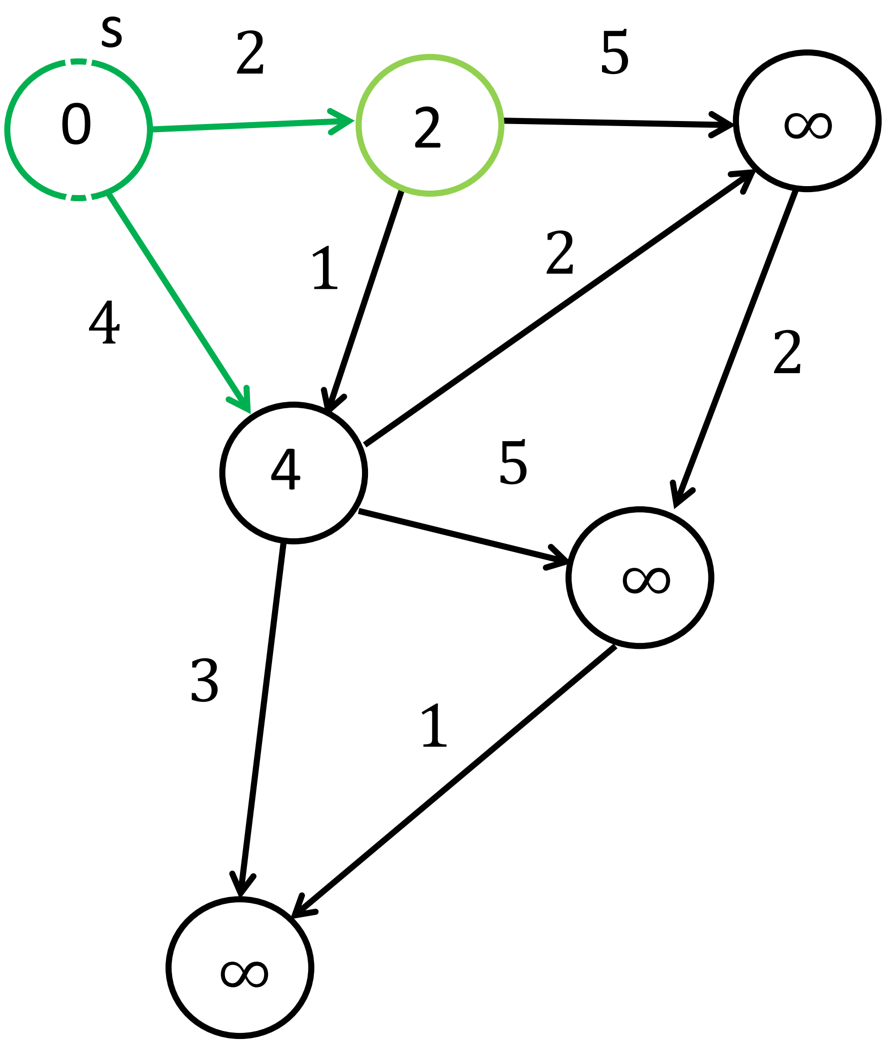
$O((|V| + |E|) \log|V|)$

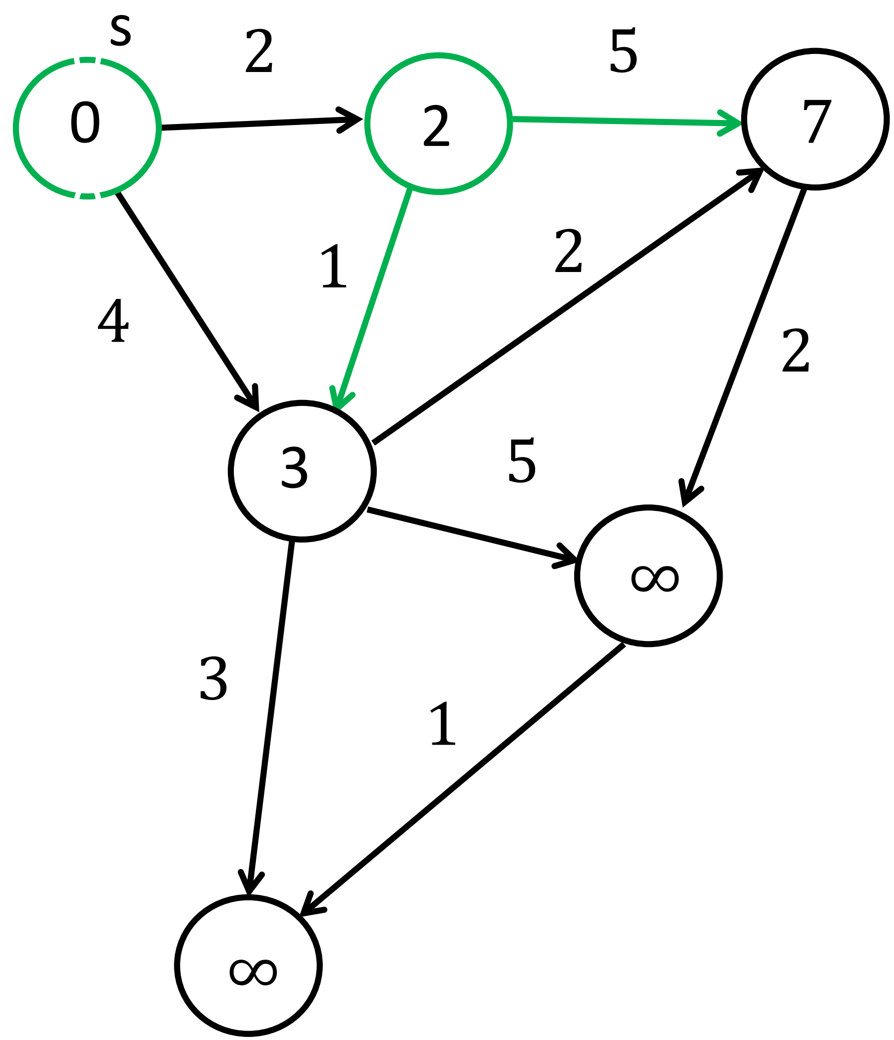


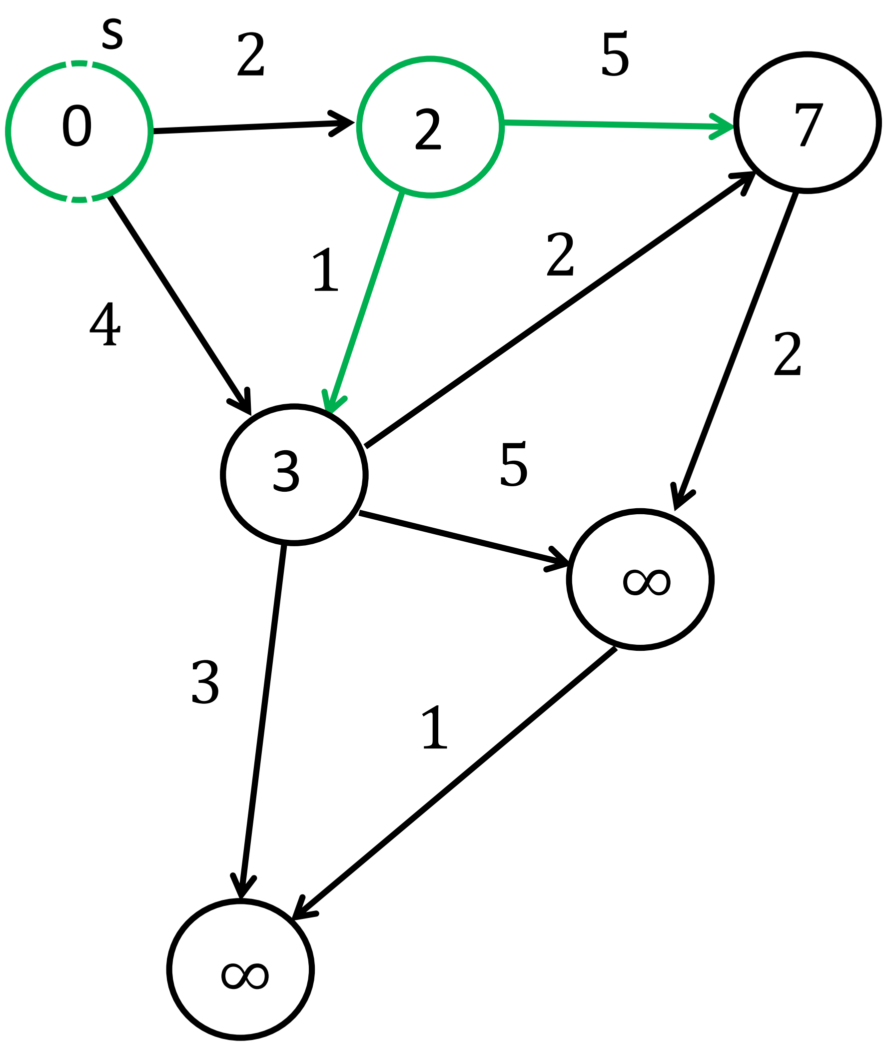
7 Elementy zbioru S z oszacowaniem odległości od wierzch. s

Łuki badane w aktualnym kroku

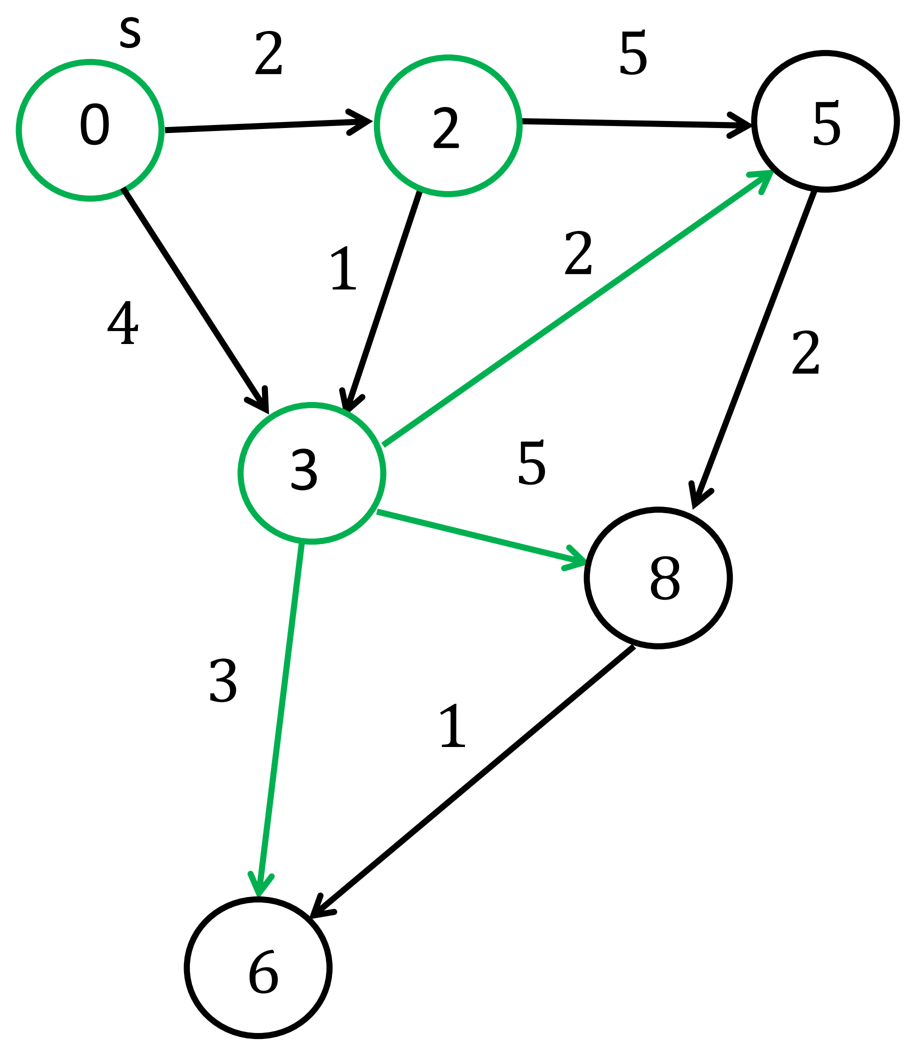



s
0
2
2
5
∞
4
1
2
2
4
5
∞
3
1
∞

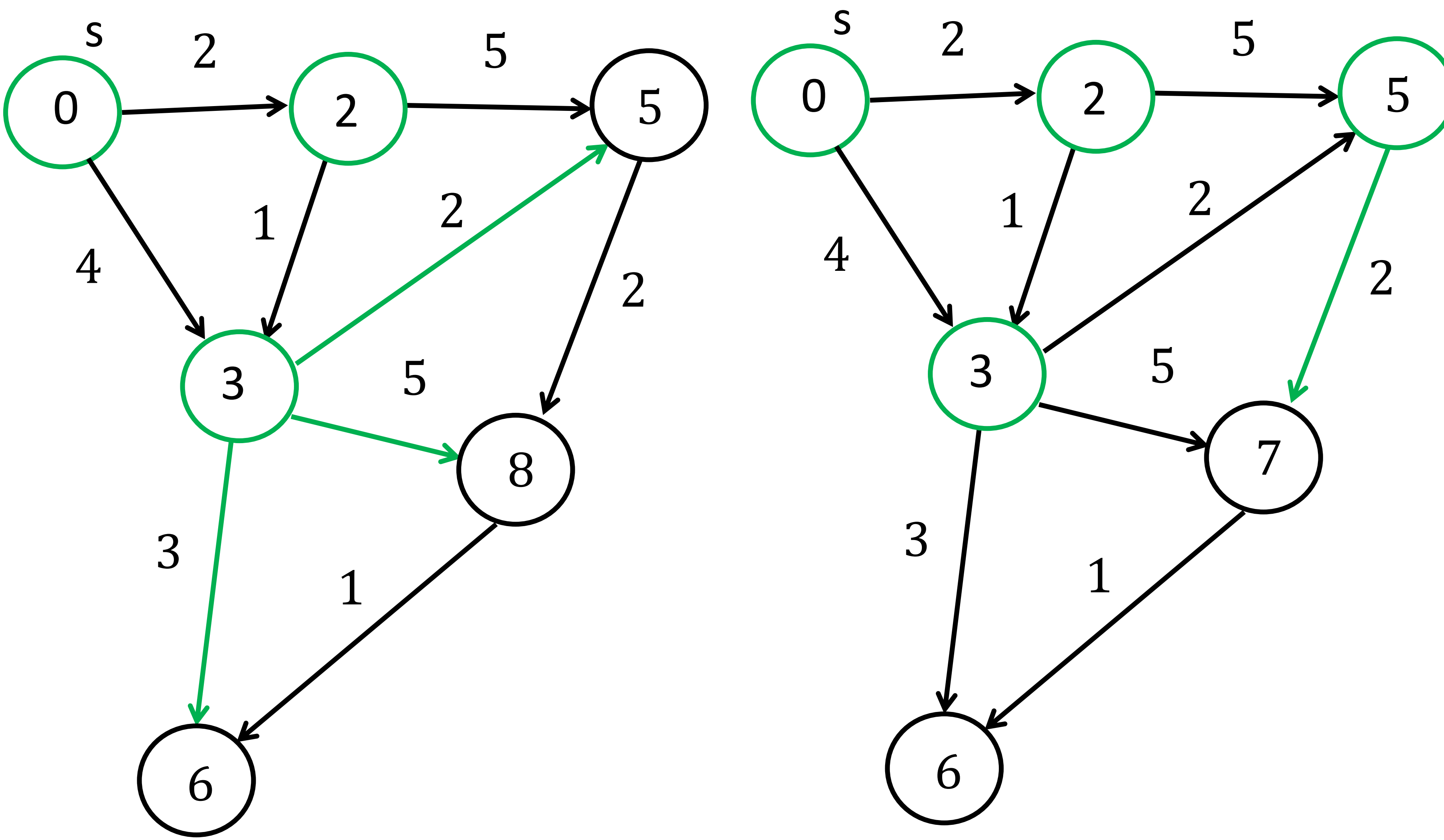

s
0
2
2
5
7
4
1
2
2
3
5
∞
3
1
∞

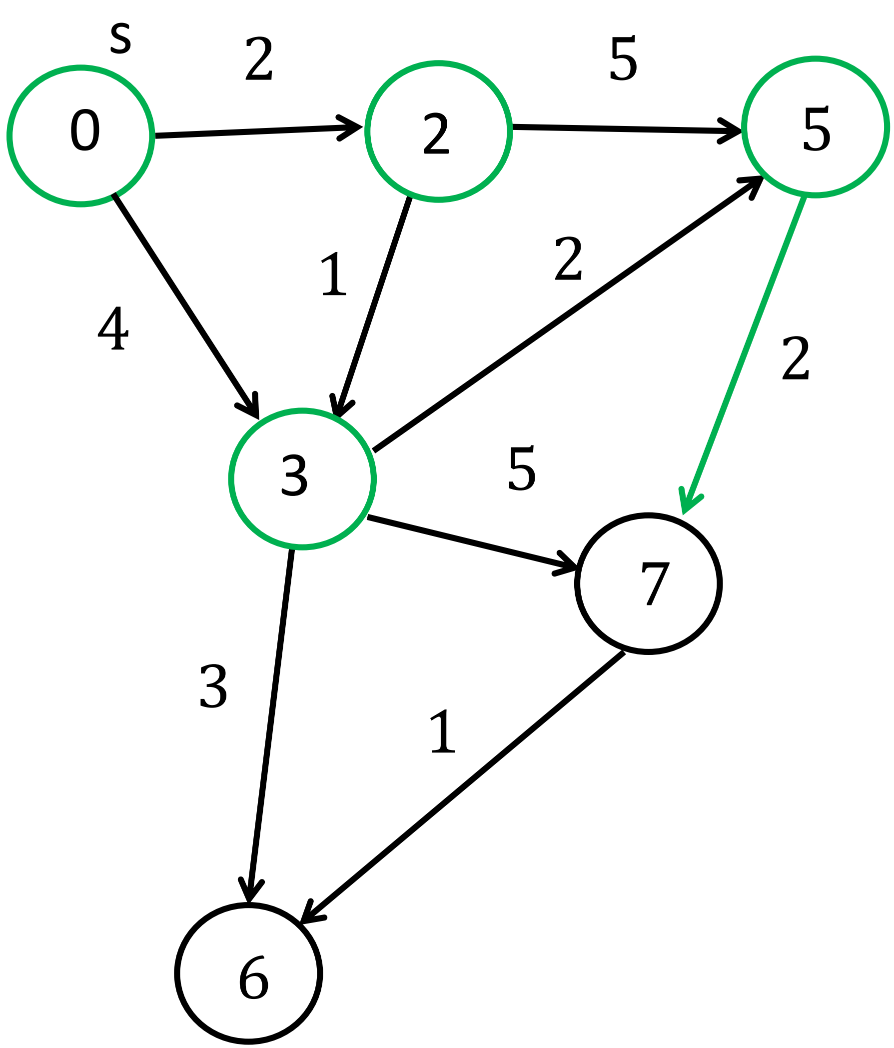


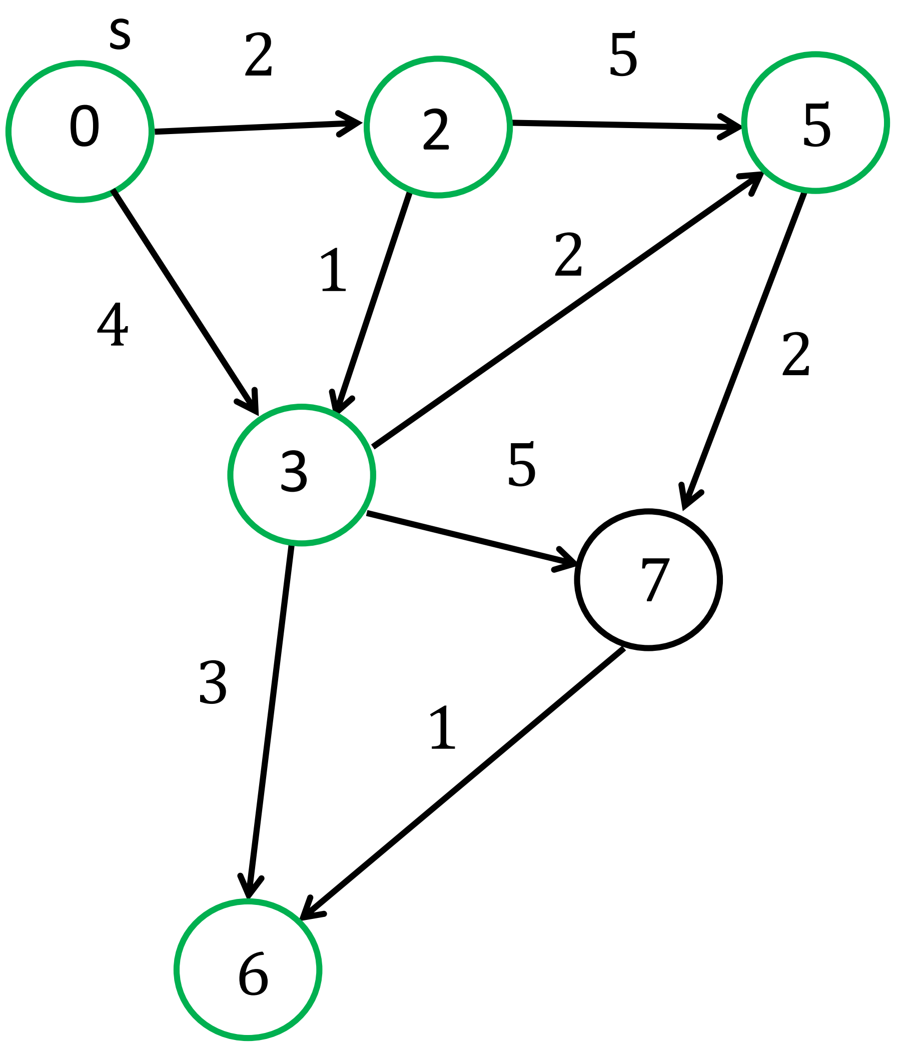
s
0
2
2
5
7
1
2
4
3
2
5
∞
3
1
∞

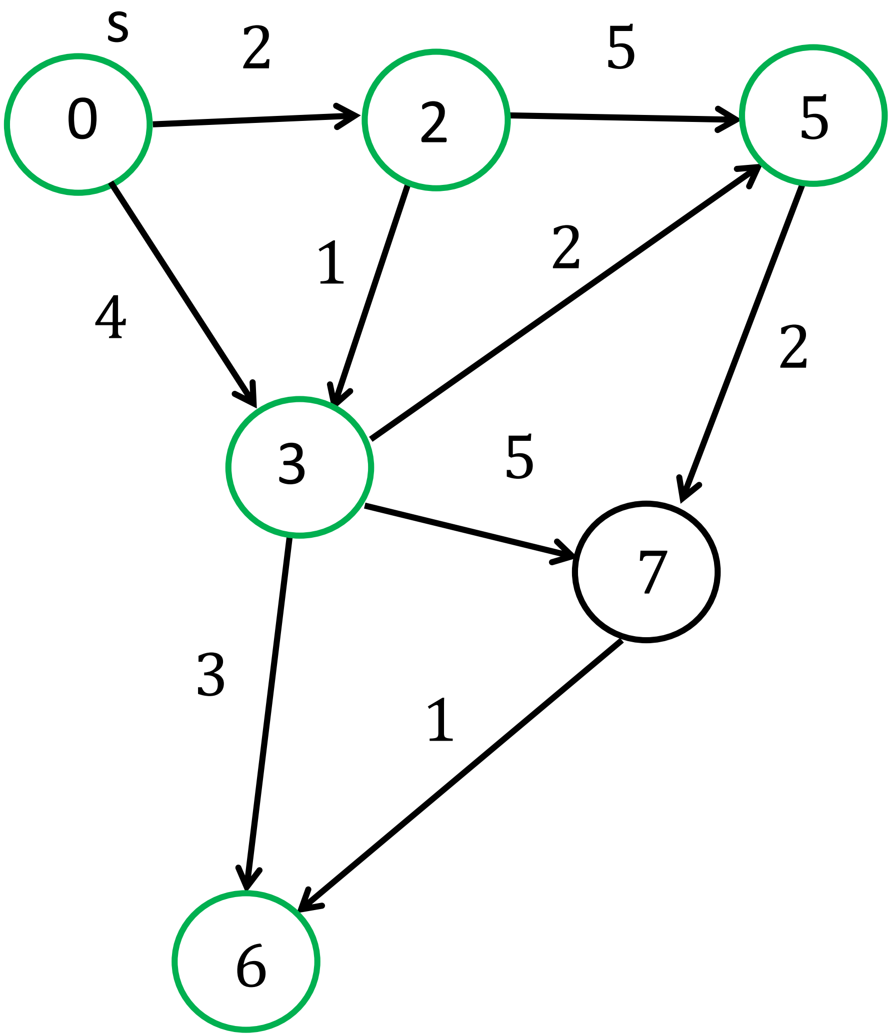


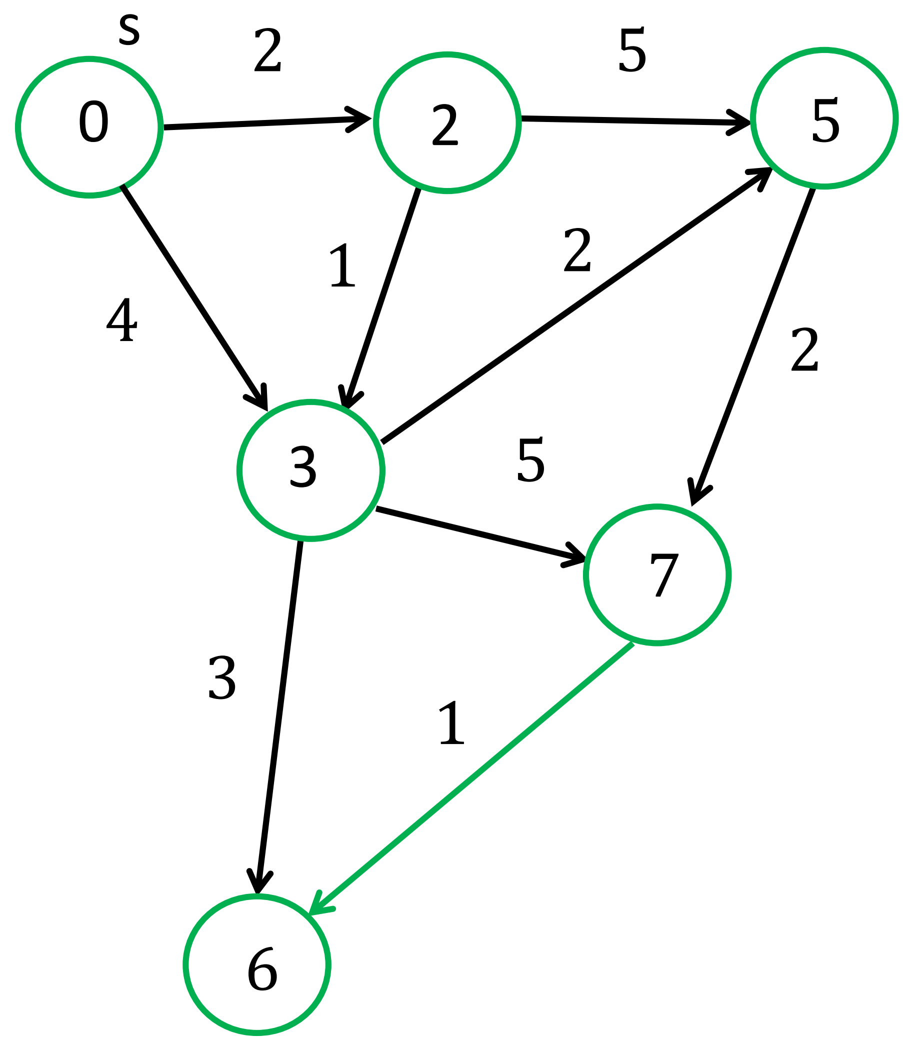
s
0
2
2
5
5
1
2
4
2
3
5
8
3
1
6


s
0
2
5
3
8
6
2
5
4
1
2
2
5
3
1
s
0
2
5
3
7
6
2
5
4
1
2
2
5
3
1


s
0
2
5
3
7
6
2
5
4
1
2
2
5
3
1


s
0
2
5
3
7
6
2
5
4
1
2
2
5
3
1


s
0
2
5
3
7
6
2
5
1
2
4
2
5
3
1


s
0
2
5
3
7
6
2
5
1
2
4
2
5
3
1