



Politechnika Wroclawska

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.

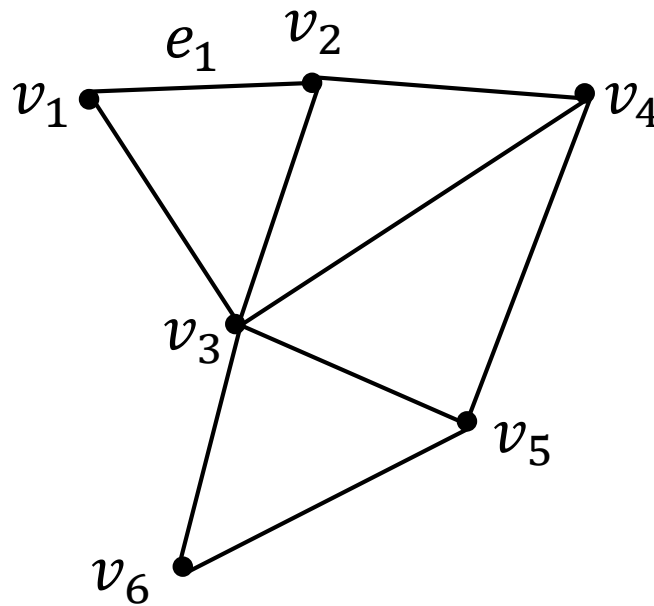


Algorytmy grafowe: podstawowe pojęcia

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.



$$e_1 = \{v_1, v_2\}$$

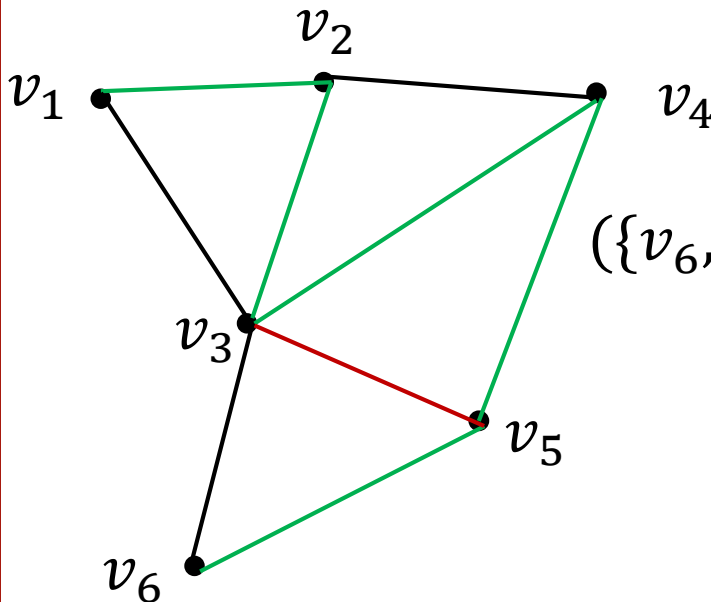
Nie ma pętli $\{v_i, v_i\}$.



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**

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



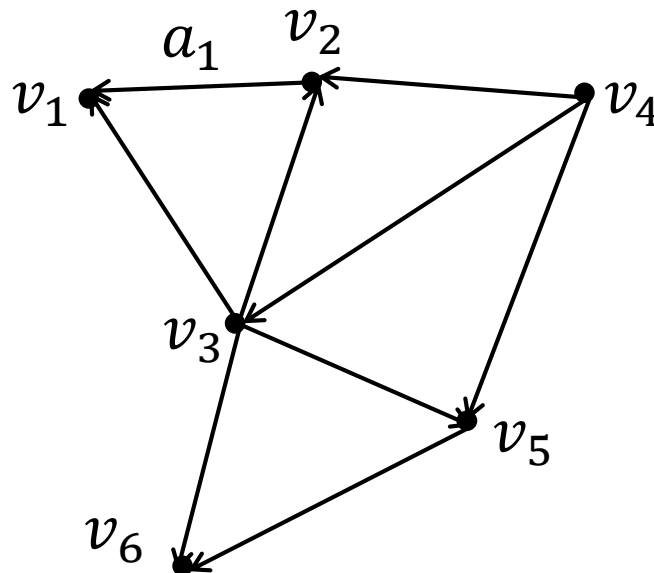
$(\{v_6, v_5\}, \{v_5, v_4\}, \{v_4, v_3\}, \{v_3, v_2\}, \{v_2, v_1\})$

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



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 $a_i = \langle v_j, v_k \rangle$ jest łukiem będącym parą uporządkowaną wierzchołków.

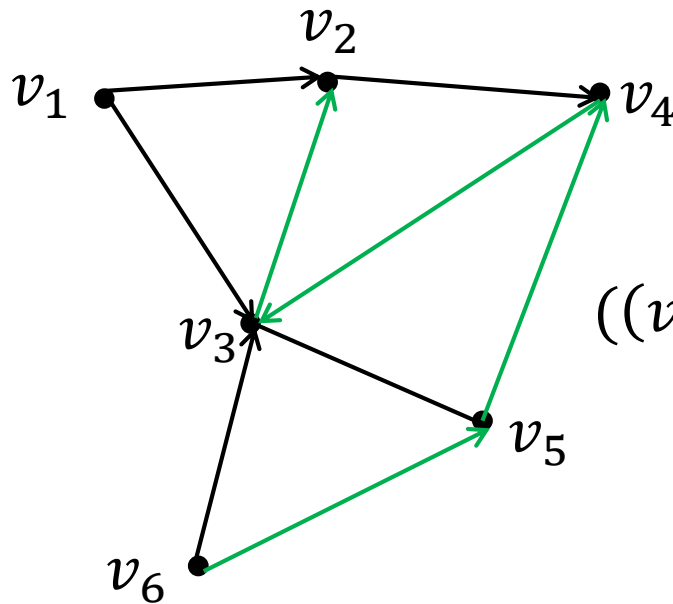


$$a_1 = (v_2, v_1)$$



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

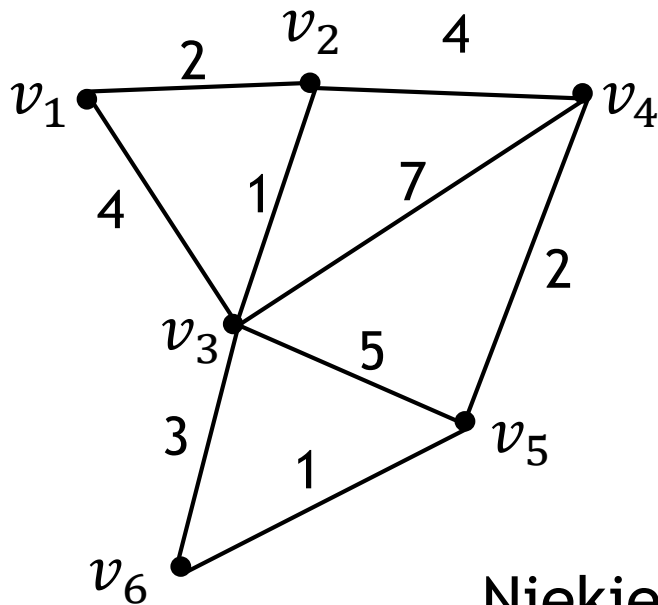
$W: V \rightarrow R$ jest funkcją wagi wierzchołków.

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.



	1	2	3	4	5	6
1	∞	2	4	∞	∞	∞
2	2	∞	1	4	∞	∞
3	4	1	∞	7	5	3
4	∞	4	7	∞	2	∞
5	∞	∞	5	2	∞	1
6	∞	∞	3	∞	1	∞

Niekiedy na przekątnej wpisywane są inne wartości wyróżnione.



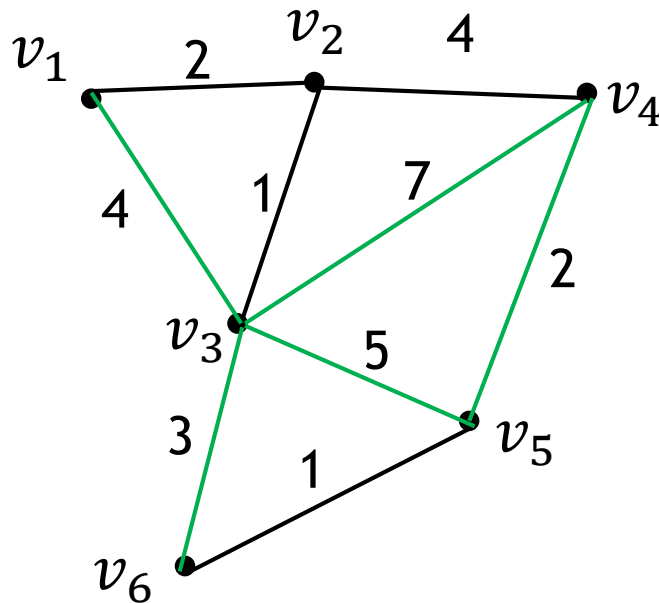
Algorytmy grafowe: podstawowe pojęcia

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

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



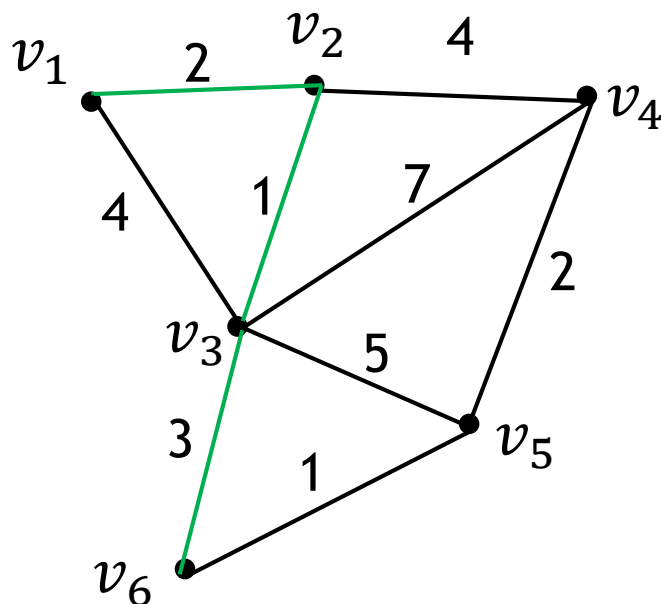
Algorytmy grafowe: podstawowe pojęcia



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

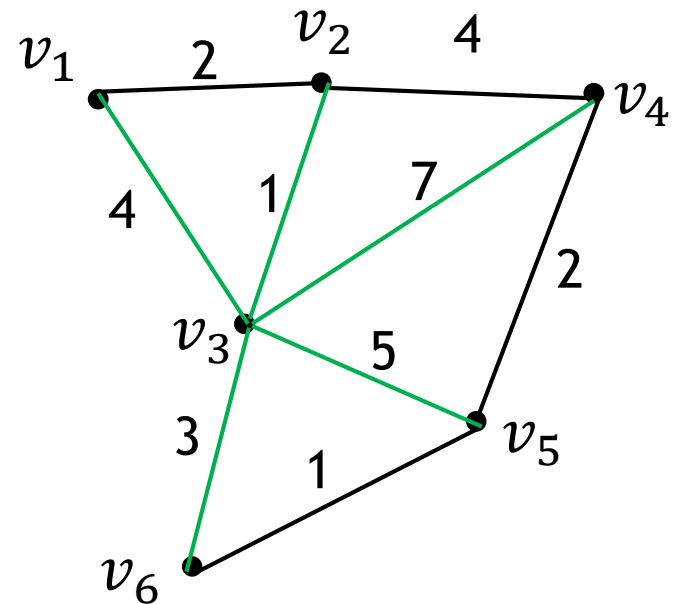
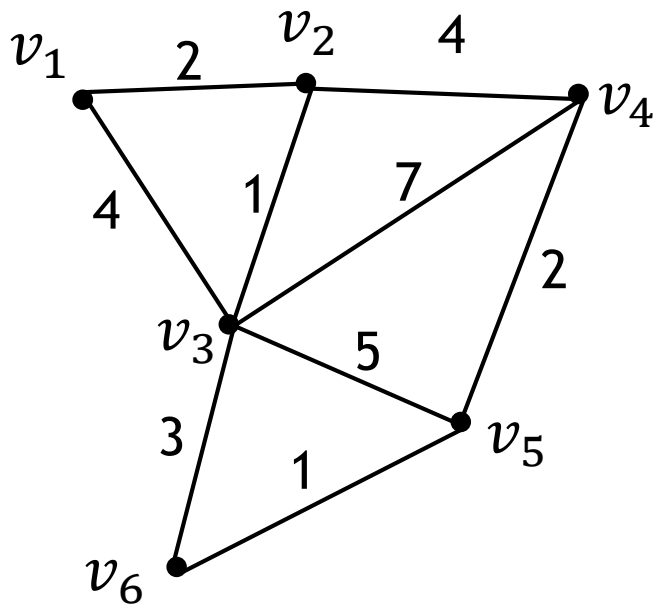
Podgrafem grafu nieskierowanego $G = (V, E)$ jest taki graf, że $V' \subset V$ i $E' \subset E$.

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

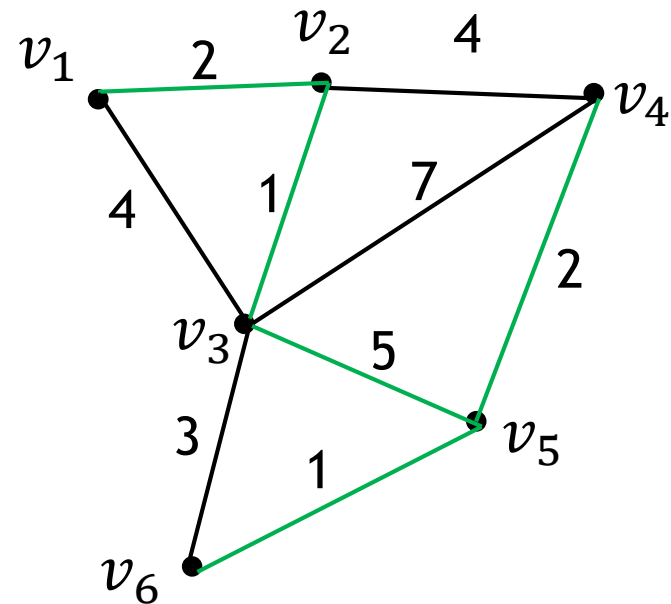
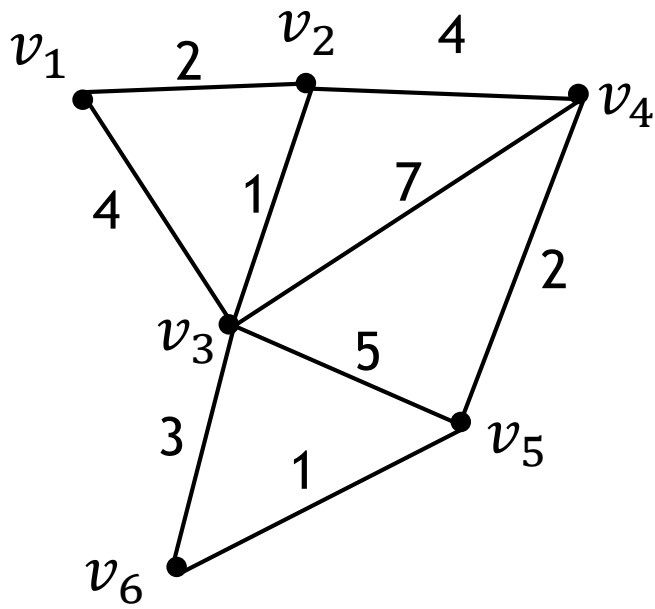
Przykład drzewa rozpinającego nieskierowanego grafu spójnego



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

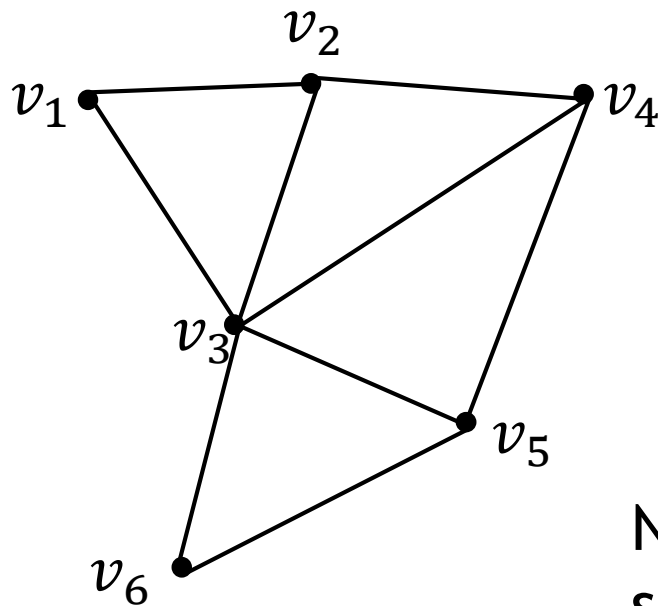
Rozmiary grafu:

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



Algorytmy grafowe: reprezentacja grafów

Graf nieskierowany i jego macierz sąsiedztwa.



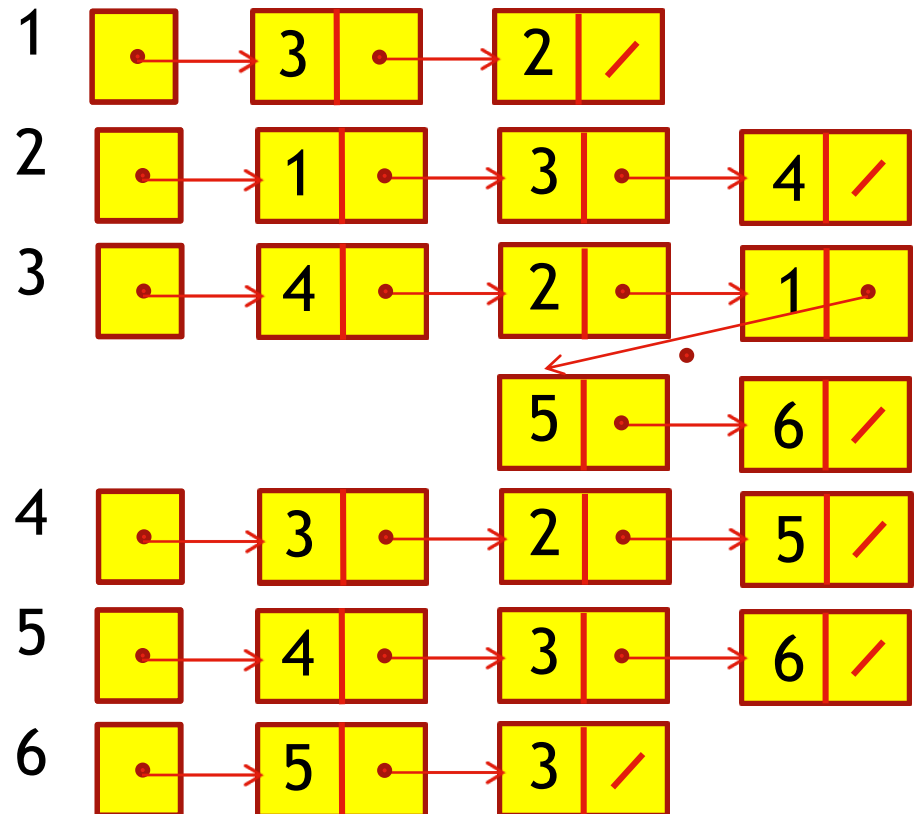
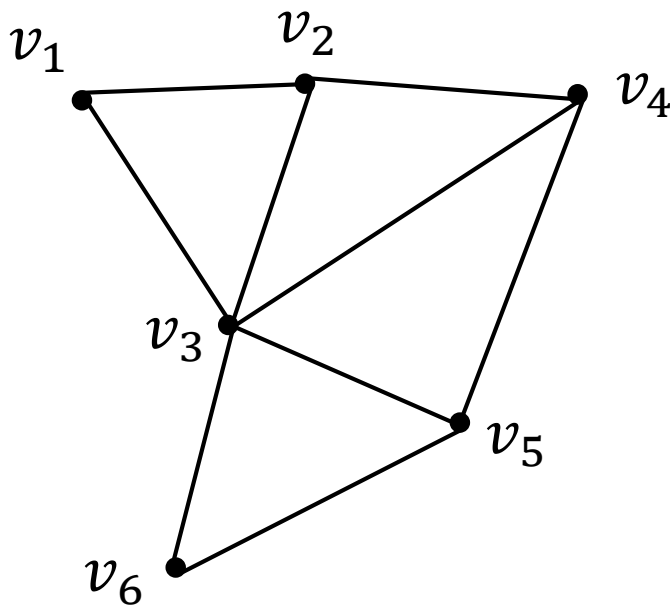
	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	1	1	0	0
3	1	1	0	1	1	1
4	0	1	1	0	1	0
5	0	0	1	1	0	1
6	0	0	1	0	1	0

Niekiedy na przekątnej wpisywane są inne wartości wyróżnione.

Zajętość pamięci $O(n^2)$

Algorytmy grafowe: reprezentacja grafów

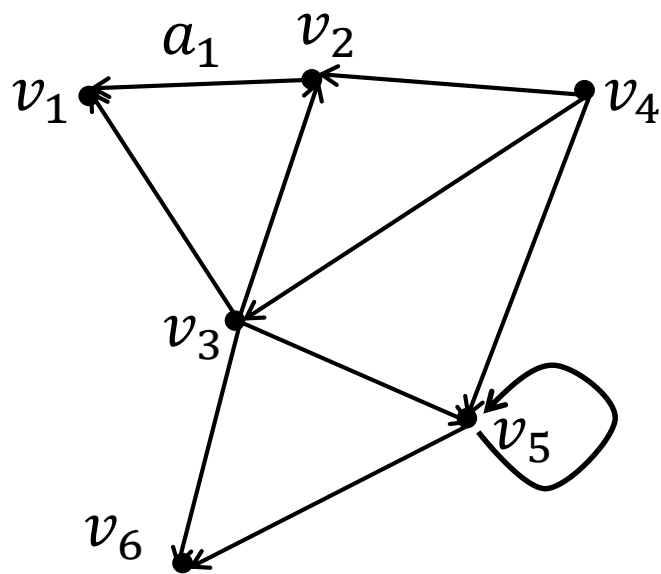
Graf nieskierowany i jego listy sąsiedztwa



Zajętość pamięci $O(n + m)$

Algorytmy grafowe: reprezentacja grafów

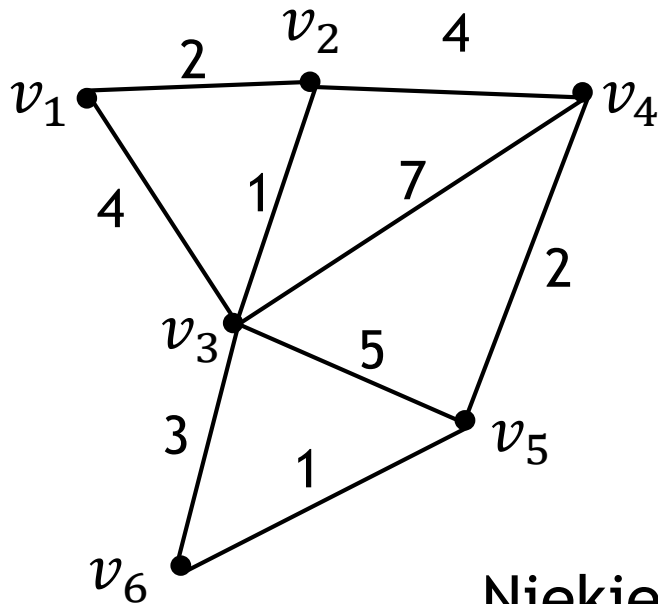
Graf skierowany i jego macierz sąsiedztwa



	1	2	3	4	5	6
1	0	0	0	0	0	0
2	1	0	0	0	0	0
3	1	1	0	0	1	1
4	0	1	1	0	1	0
5	0	0	0	0	1	1
6	0	0	0	0	0	0

Algorytmy grafowe: reprezentacja grafów

Przykład grafu nieskierowanego z wagami krawędzi i jego macierz wag.



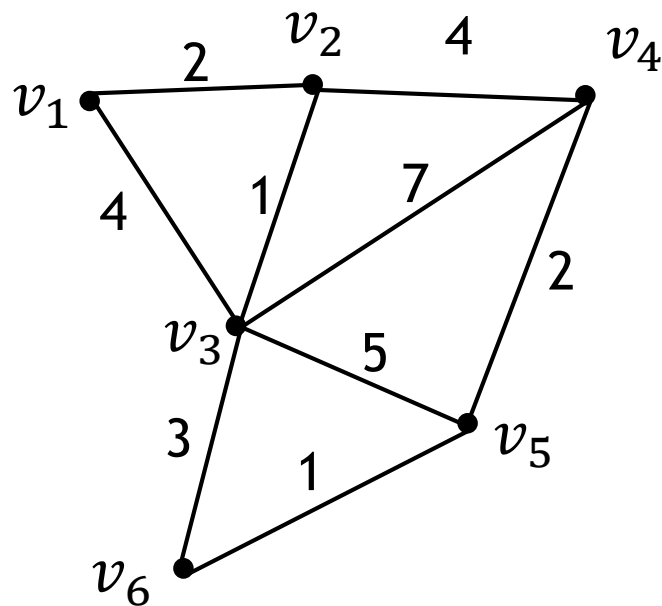
	1	2	3	4	5	6
1	∞	2	4	∞	∞	∞
2	2	∞	1	4	∞	∞
3	4	1	∞	7	5	3
4	∞	4	7	∞	2	∞
5	∞	∞	5	2	∞	1
6	∞	∞	3	∞	1	∞

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

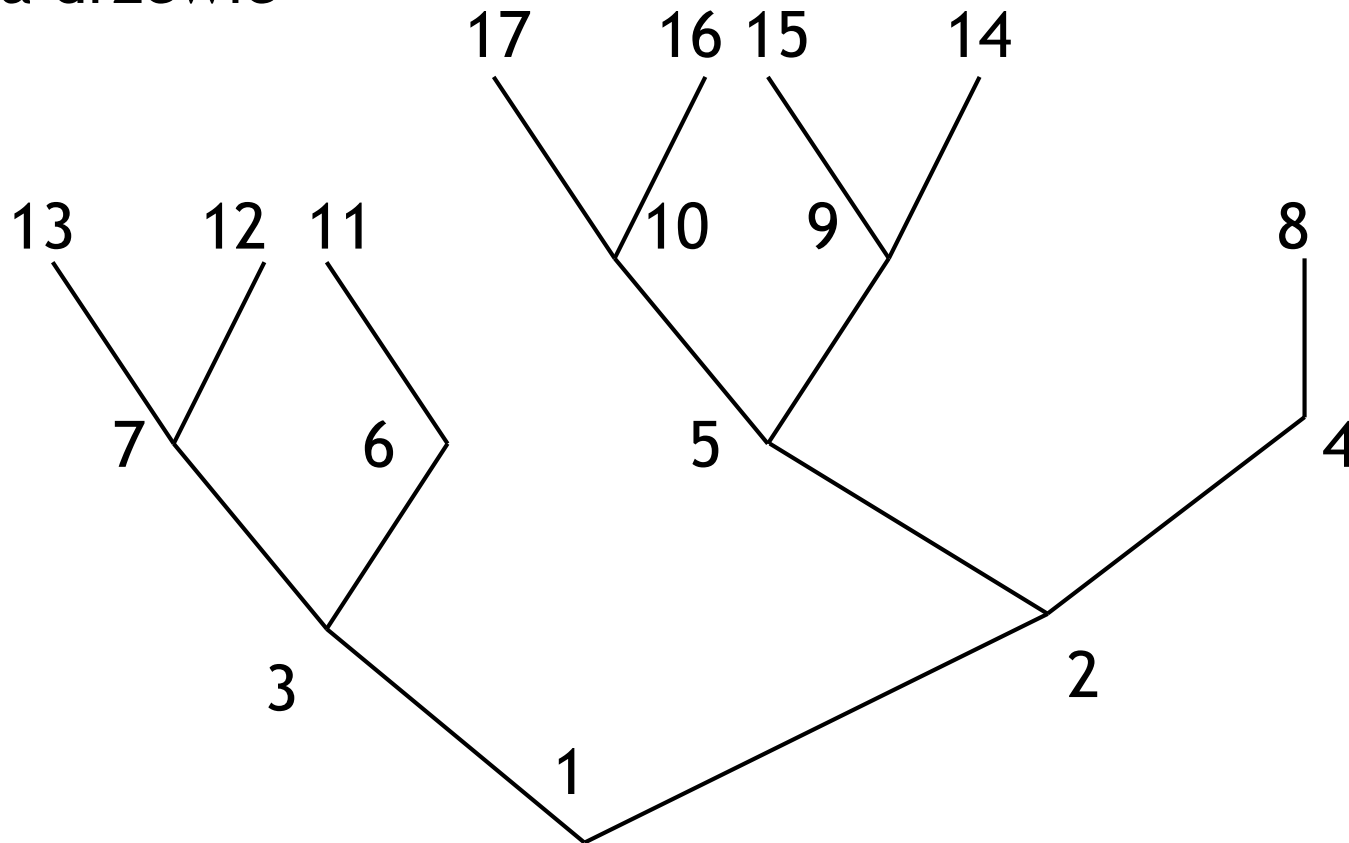
Metody przeszukiwania:

- 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)

```
1  for każdy wierzchołek  $u \in V[G] - \{s\}$ 
2    do  $color[u] \leftarrow$  BIAŁY
3       $d[u] \leftarrow \infty$ 
4       $\pi[u] \leftarrow$  NIL
5   $color[s] \leftarrow$  SZARY
6   $d[s] \leftarrow 0$ 
7   $\pi[s] \leftarrow$  NIL
8   $Q \leftarrow \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11   do  $u \leftarrow$  DEQUEUE( $Q$ )
12     for każdego  $v \in Adj[u]$ 
13       do if  $color[v] =$  BIAŁY
14         then  $color[v] \leftarrow$  SZARY
15            $d[v] \leftarrow d[u] + 1$ 
16            $\pi[v] \leftarrow u$ 
17           ENQUEUE( $Q, v$ )
18      $color[u] \leftarrow$  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, $u = \text{NIL}$, jeśli $u = s$ lub u nie został jeszcze odwiedzony.

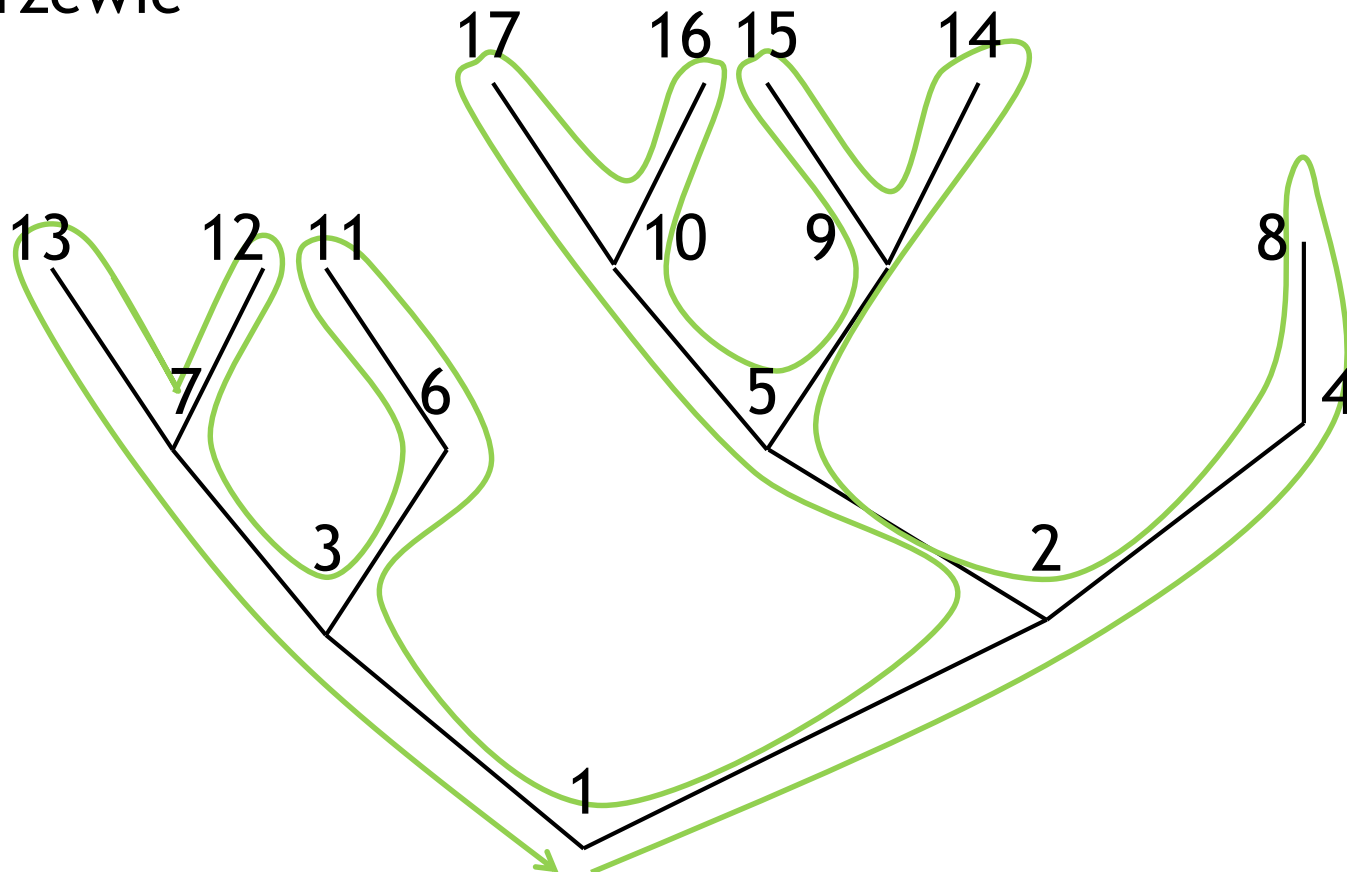
$O(|V| + |E|)$

[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)

```
1  for każdy wierzchołek  $u \in V[G]$ 
2    do  $color[u] \leftarrow$  BIAŁY
3        $\pi[u] \leftarrow$  NIL
4  for każdy wierzchołek  $u \in V[G]$ 
5    do if  $color[u] =$  BIAŁY
6       then DFS-VISIT( $u$ )
```

DFS-VISIT(u)

```
1   $color[u] \leftarrow$  SZARY    ▷ Biały wierzchołek  $u$  został właśnie odwiedzony.
2  for każdy  $v \in Adj[u]$       ▷ Zbadaj krawędź  $(u, v)$ .
3    do if  $color[v] =$  BIAŁY
4       then  $\pi[v] \leftarrow u$ 
5           DFS-VISIT( $v$ )
6   $color[u] \leftarrow$  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 $V^S = V$.

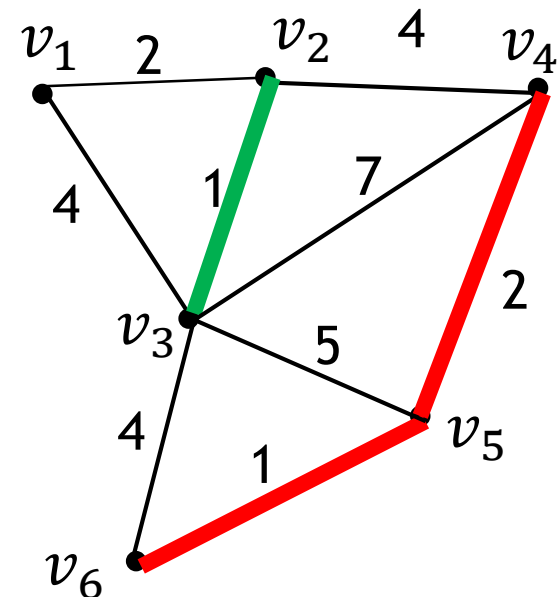
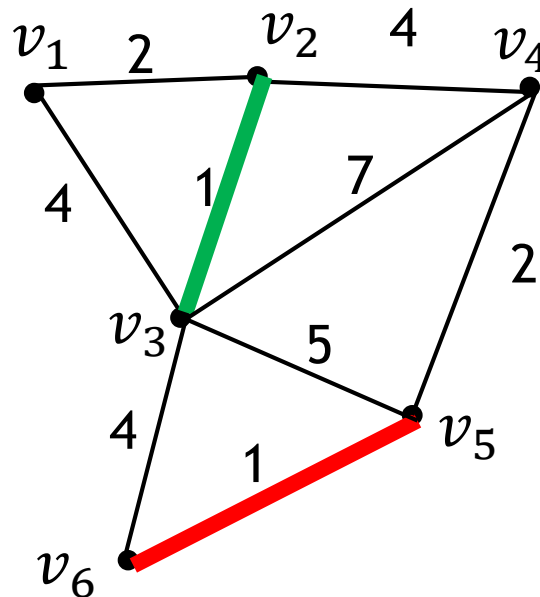
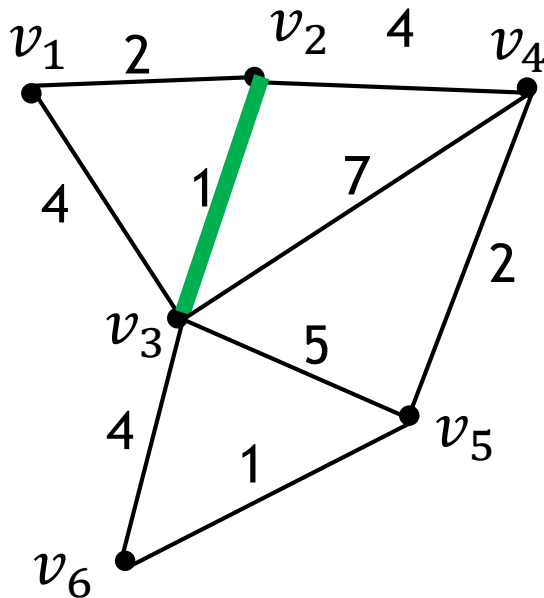
Minimalne drzewo rozpinające grafu nieskierowanego z wagami krawędzi jest drzewem rozpinającym o minimalnej sumie wag krawędzi.



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

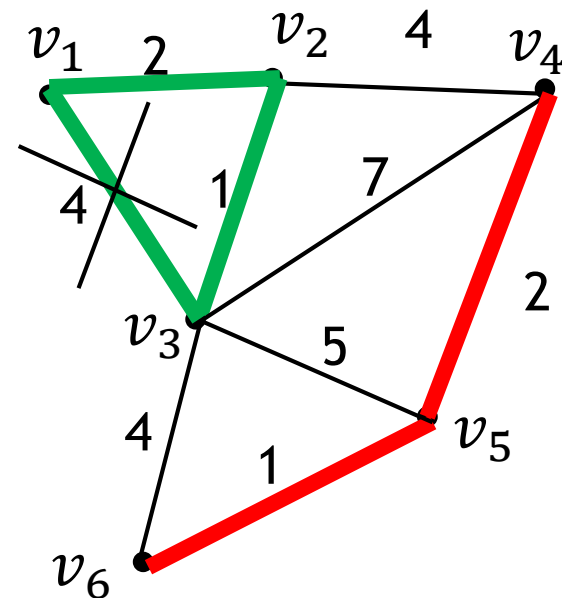
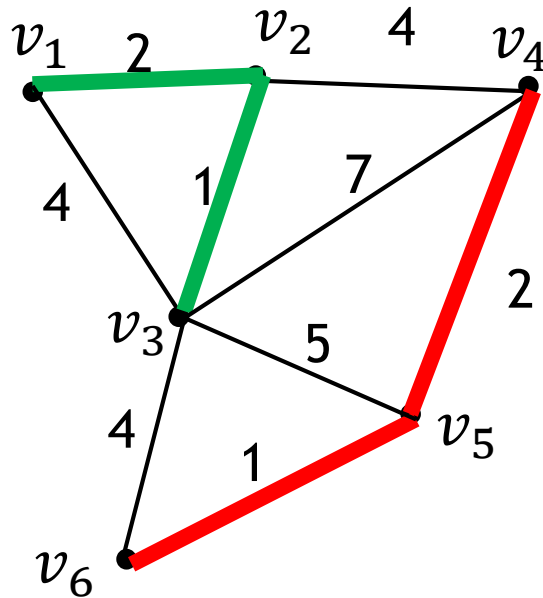
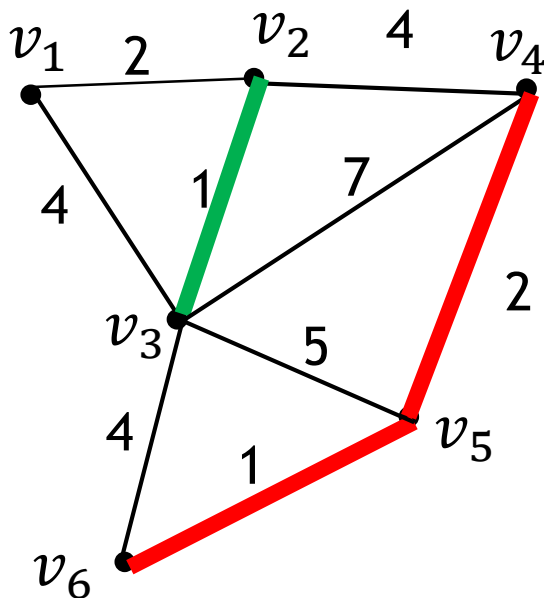




Algorytmy grafowe: minimalne drzewa rozpinające

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

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

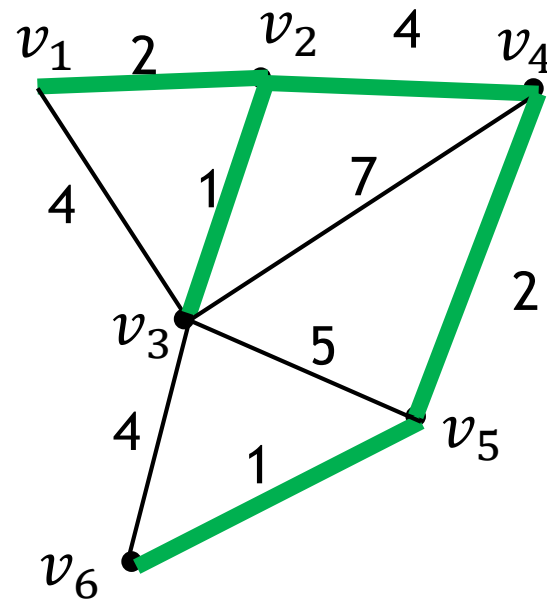
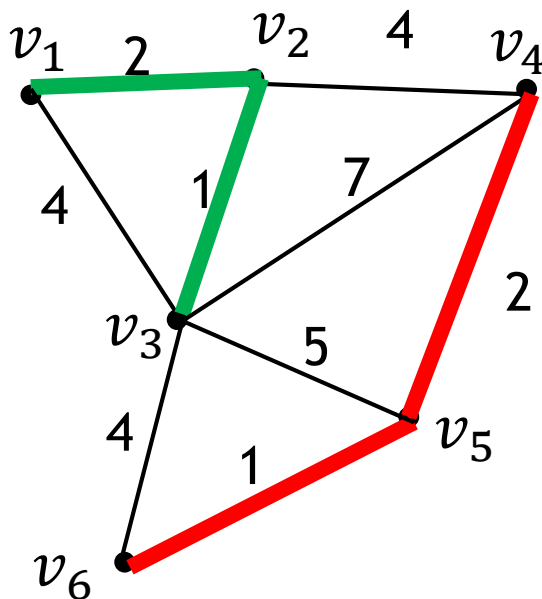




Algorytmy grafowe: minimalne drzewa rozpinające

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 ujednocicane 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).

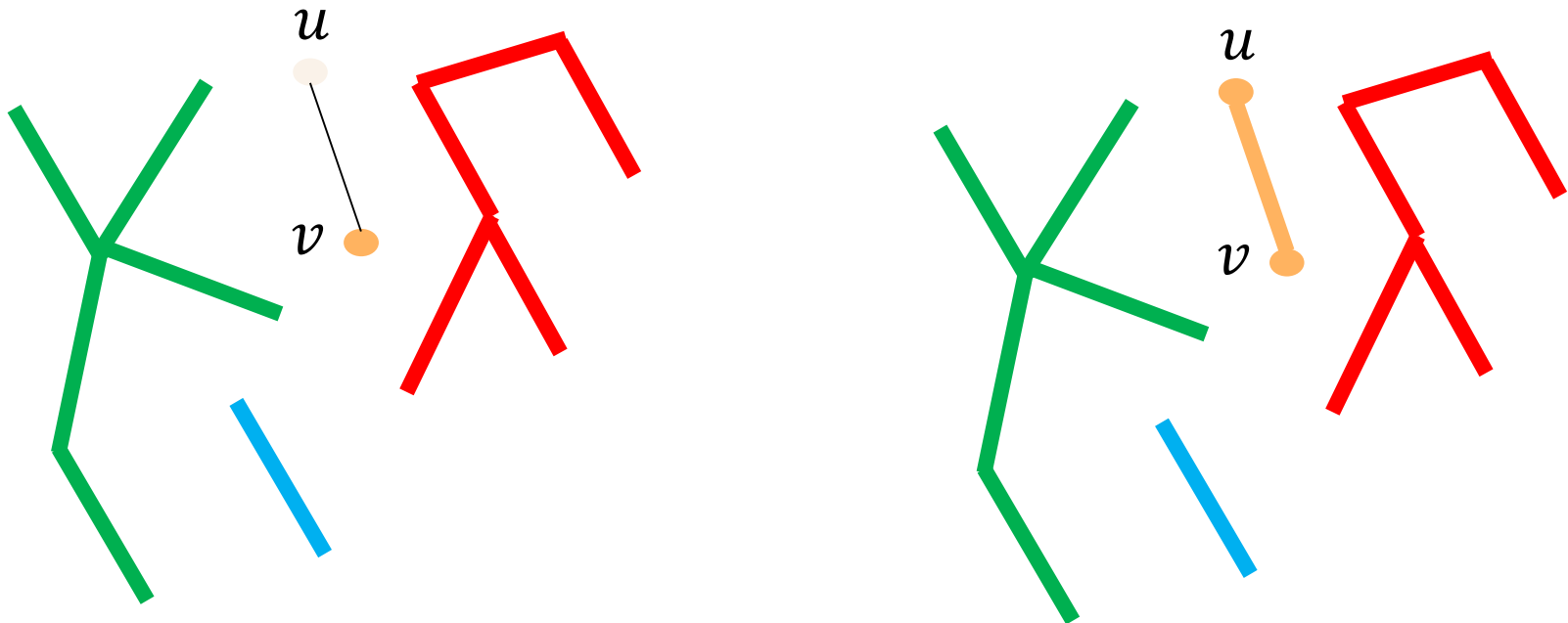
Przy próbie dołączenia krawędzi (u, v) występują następujące przypadki:

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.

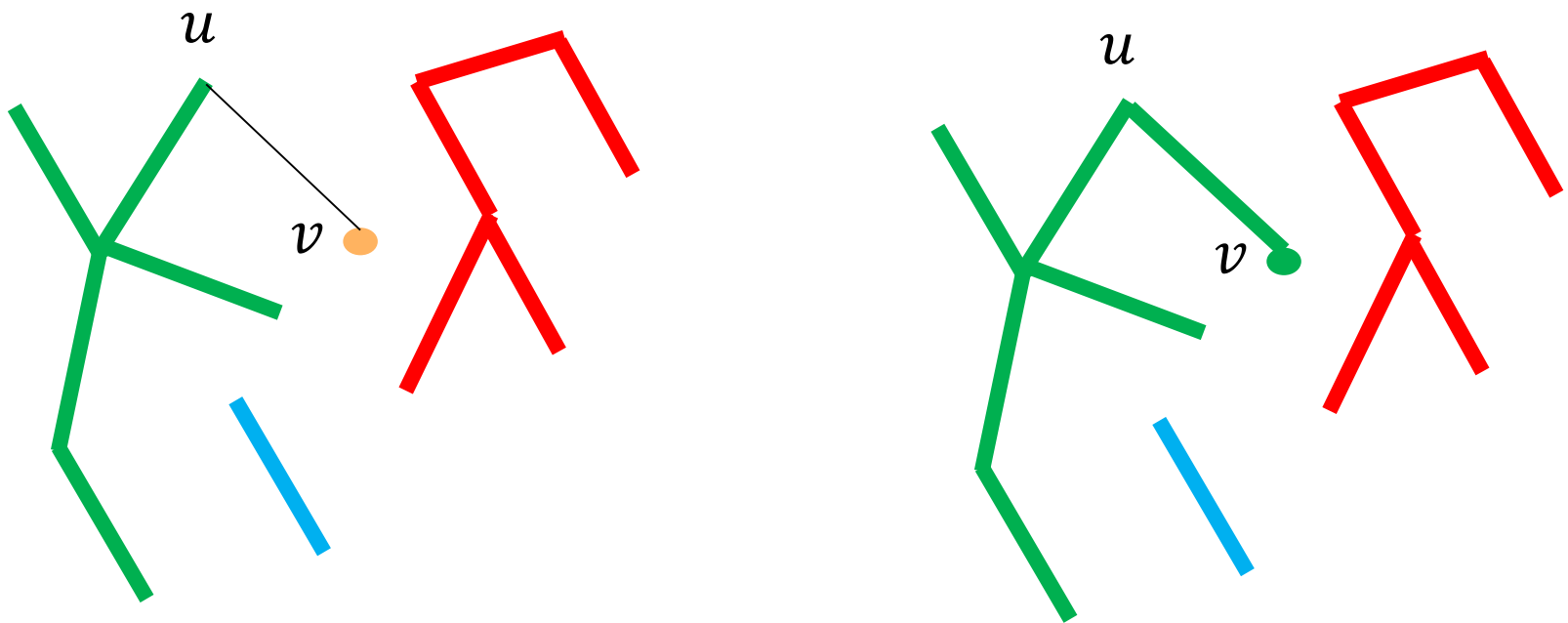
1. Wierzchołki u, v nie zostały dotąd wybrane (nie dołączono krawędzi incydentnych z nimi)



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

Algorytmy grafowe: minimalne drzewa rozpinające

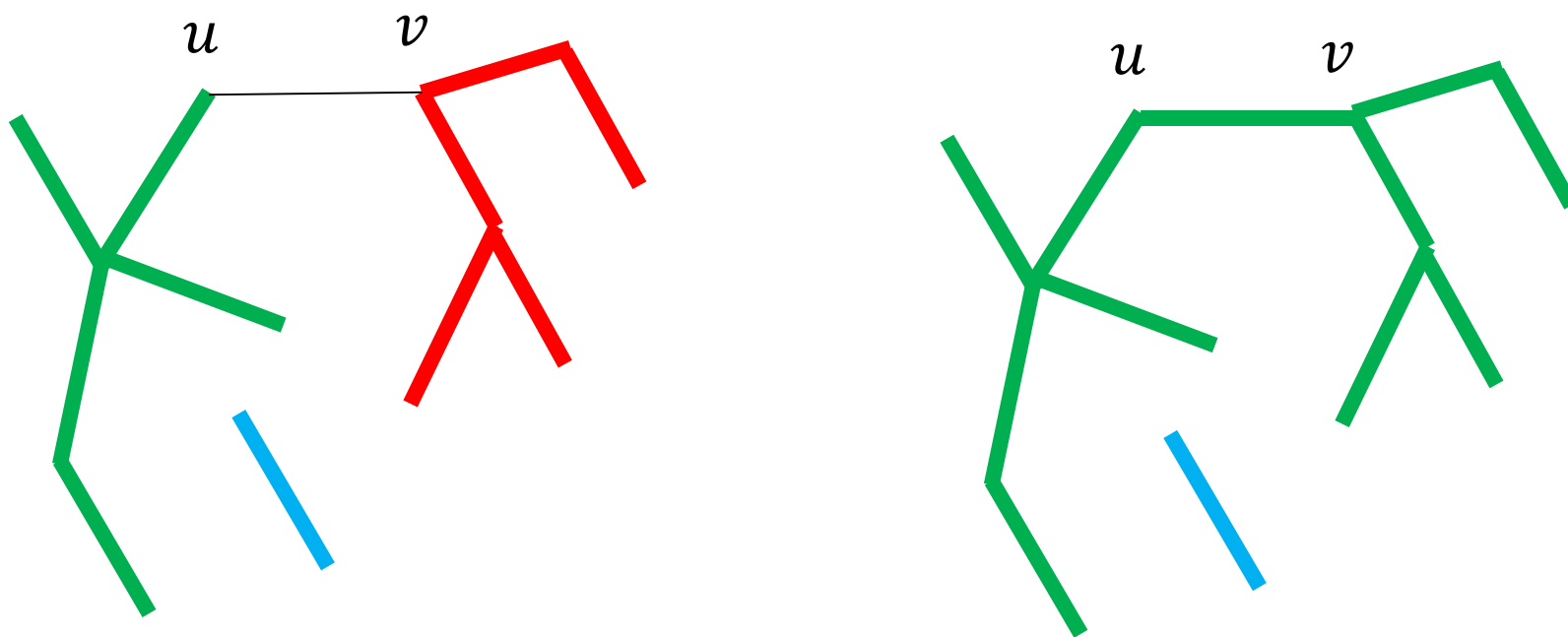
2. Dokładnie jeden z wierzchołków, przyjmijmy u , został dotychczas wybrany



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

Algorytmy grafowe: minimalne drzewa rozpinające

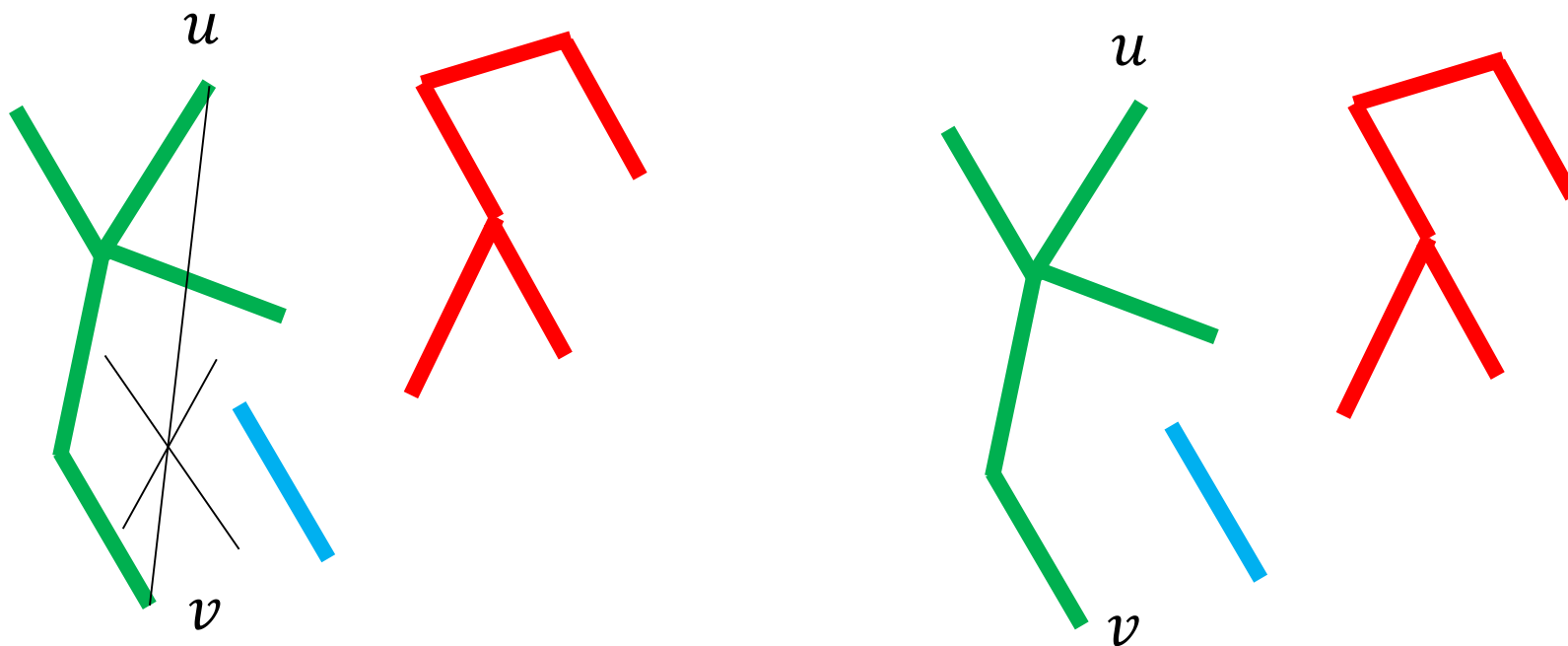
3. Oba wybrane ale różnie pokolorowane



Dołączenie nowej krawędzi nie powoduje cyklu, a ujednoczenie 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 $v_i \in V$

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

Ujednoczenie kolorów wierzchołków

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

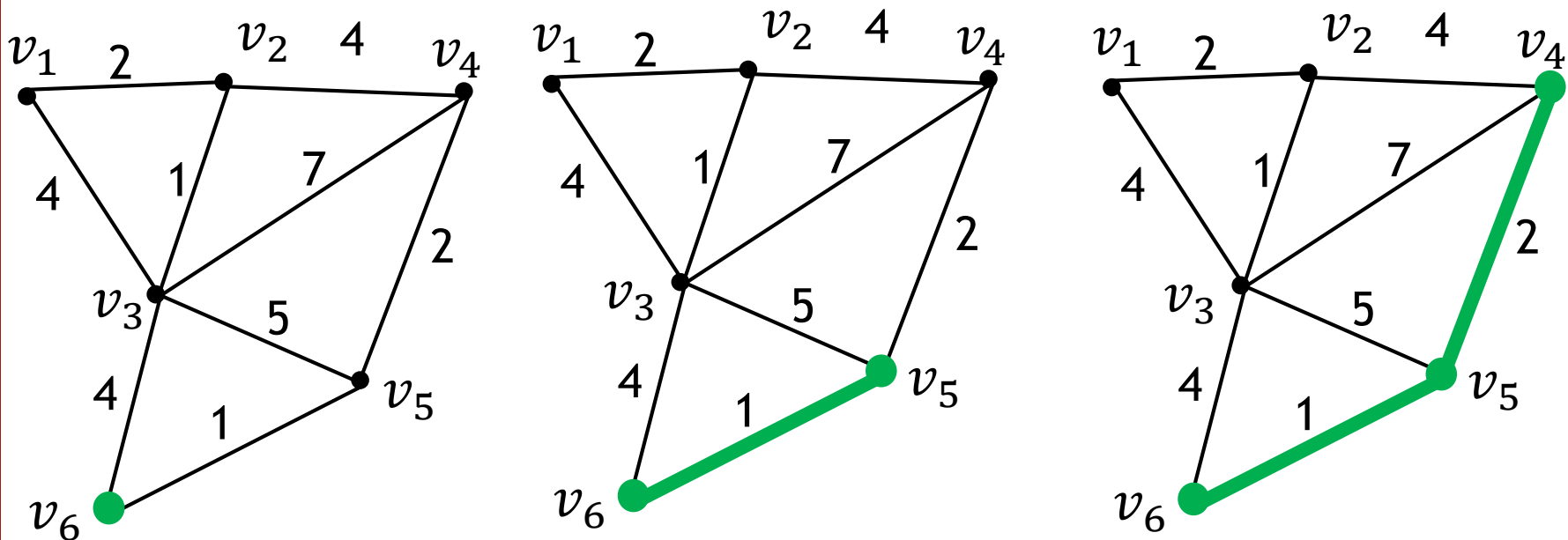
return A

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

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.

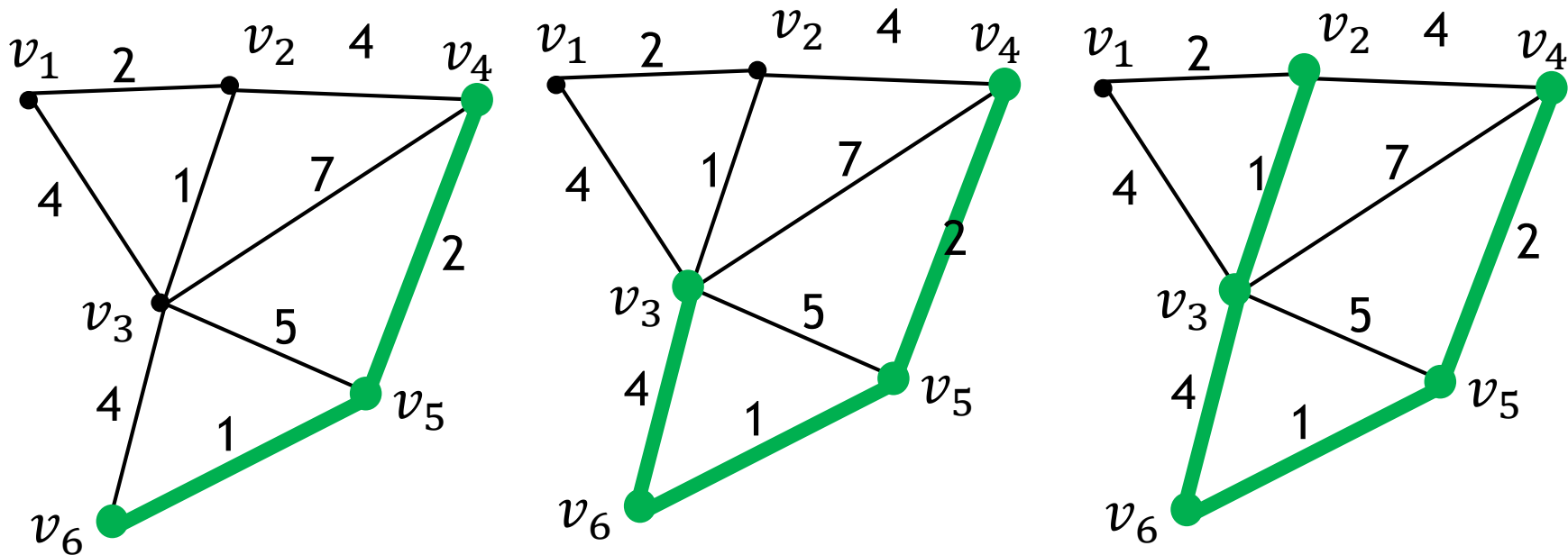




Algorytmy grafowe: minimalne drzewa rozpinające

Idea algorytmu Prima:

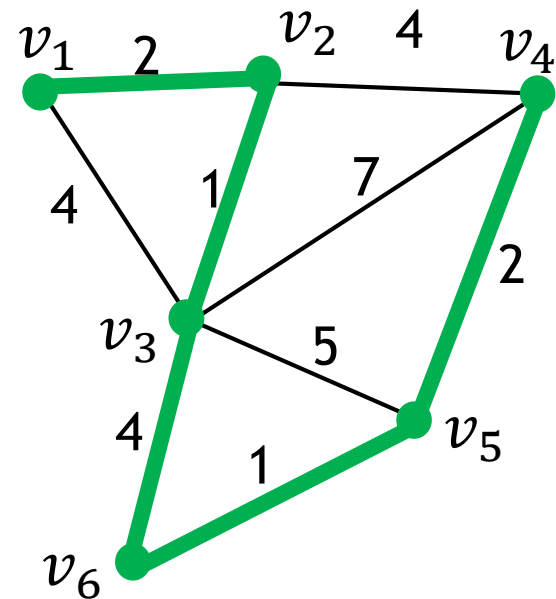
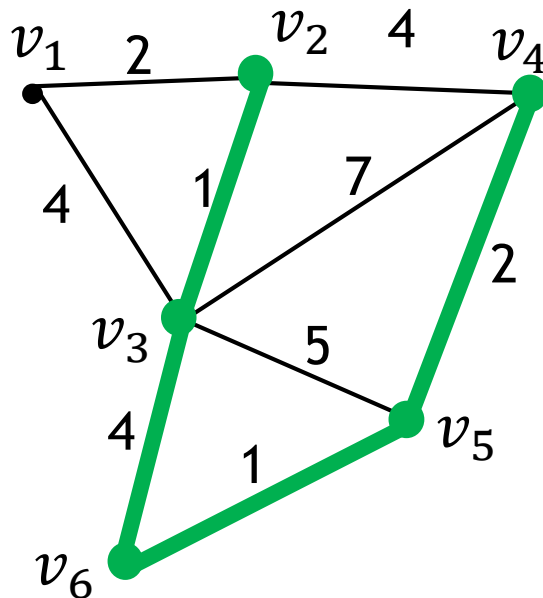
Po wybraniu dowolnego wierzchołka, kolejno dołączany jest najbliższy sąsiad czyli wierzchołek połączony krawędzią o najmniejszej wadze z wcześniej dołączonymi wierzchołkami.



Algorytmy grafowe: minimalne drzewa rozpinające

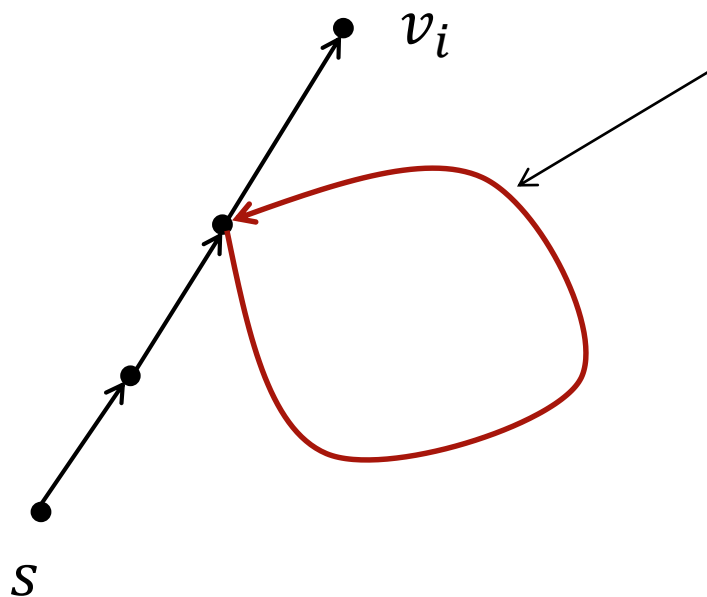
Idea algorytmu Prima:

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

7

Elementy zbioru S z oszacowaniem odległości od wierzch. s



Łuki badane w aktualnym kroku

