Wstęp do programowania

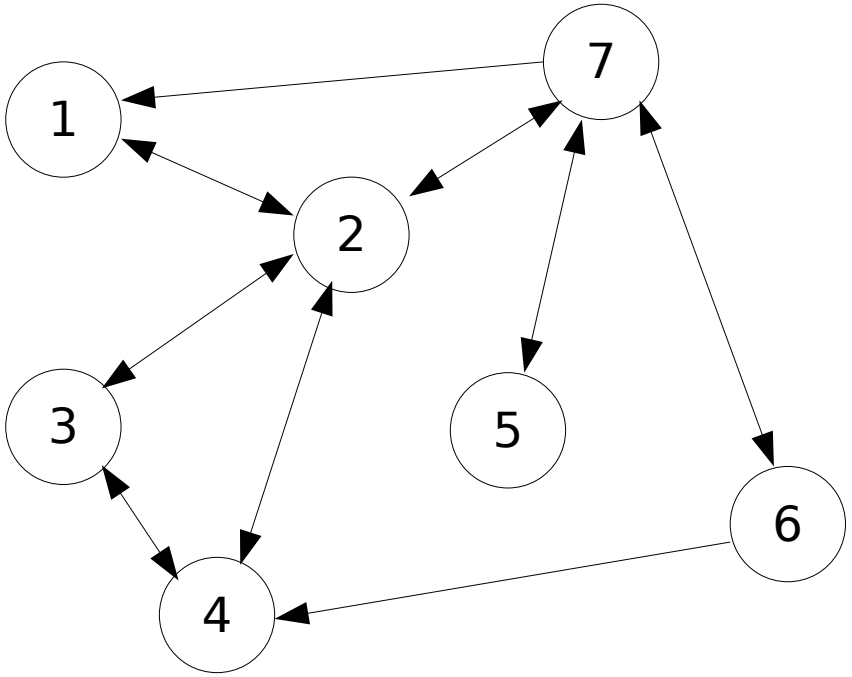
Wykład 7. Grafy

Plan wykładu

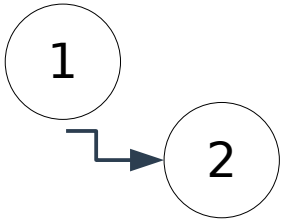
1. Grafy i ich reprezentacje.
2. Algorytmy grafowe
- przeszukiwanie grafu,
- kolorowanie grafu,
- minimalne drzewo rozpinające,
- najkrótsze ścieżki w grafie.

Grafy

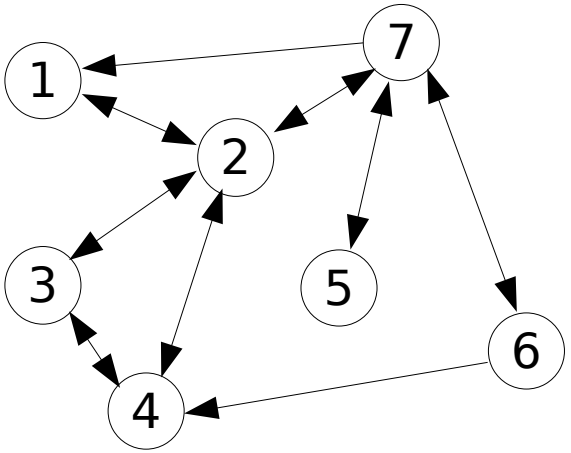
Graf składa się z wierzchołków i krawędzi. Zarówno wierzchołki jak i krawędzie mogą zawierać dodatkowe dane, np. krawędź może zawierać informację o odległości między wierzchołkami. Szczególnym przypadkiem grafu jest drzewo.

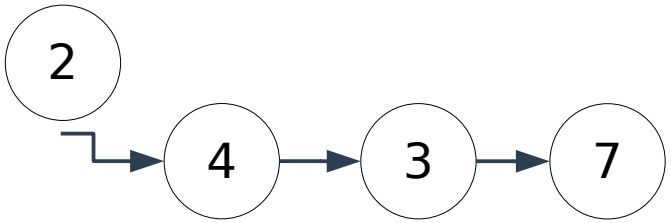


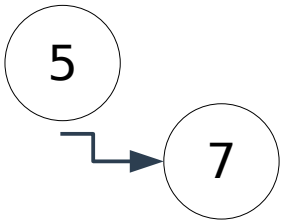
Grafy: reprezentacje

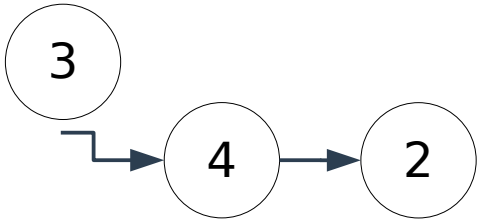


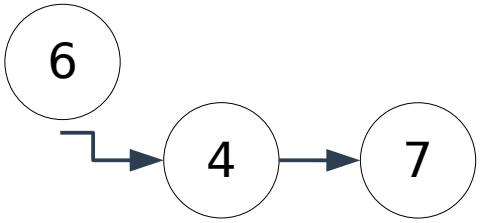
Listy sąsiedztwa – każdy wierzchołek zawiera listę (wskaźników) do swoich sąsiadów.

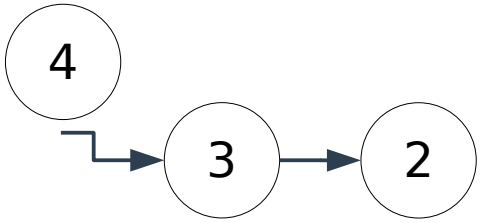


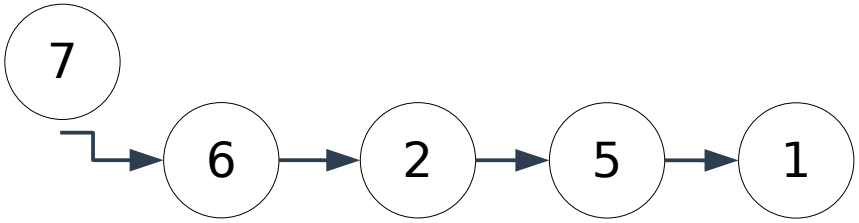






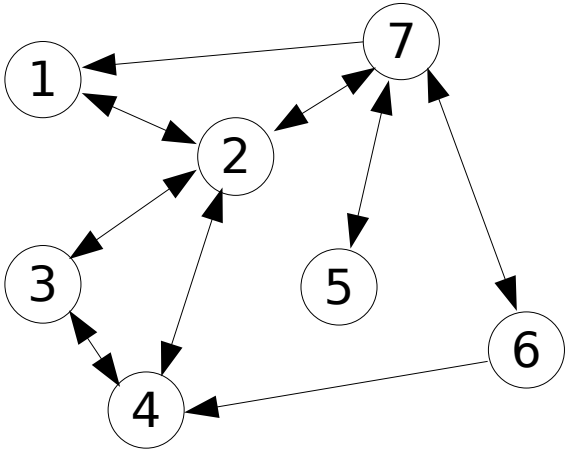






Grafy: reprezentacje

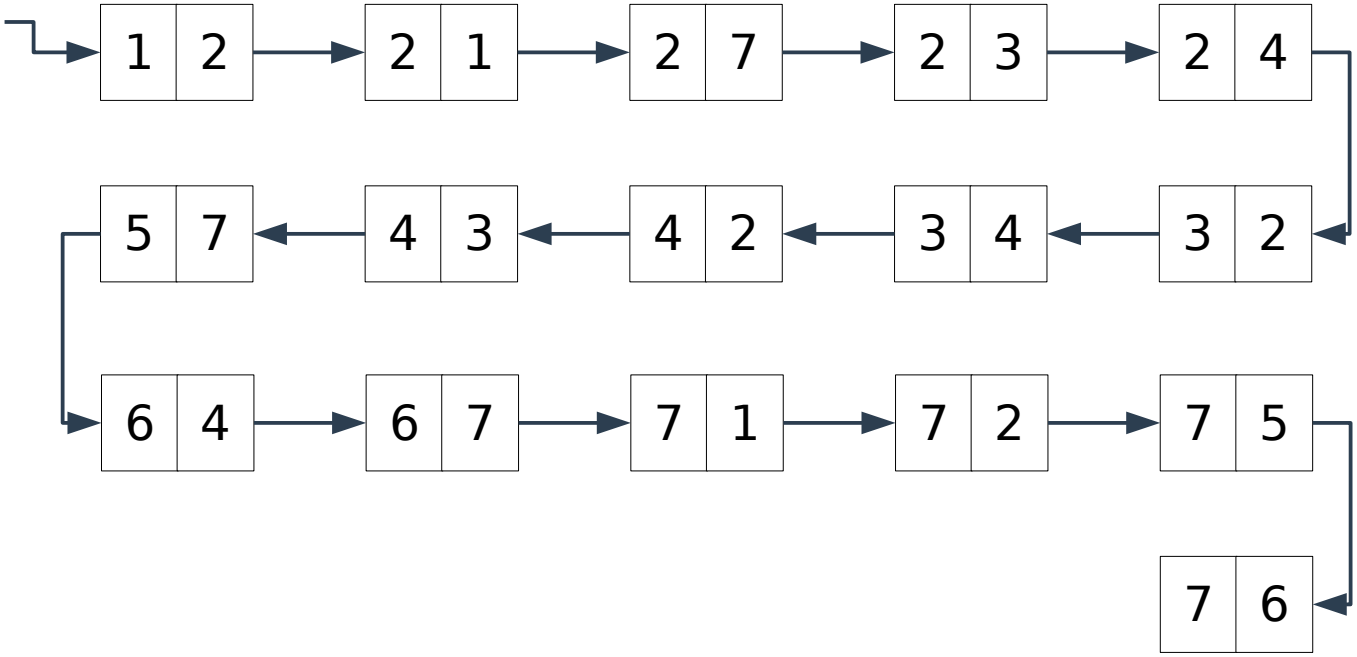
Macierz sąsiedztwa – elementy macierzy odpowiadają krawędziom grafu.

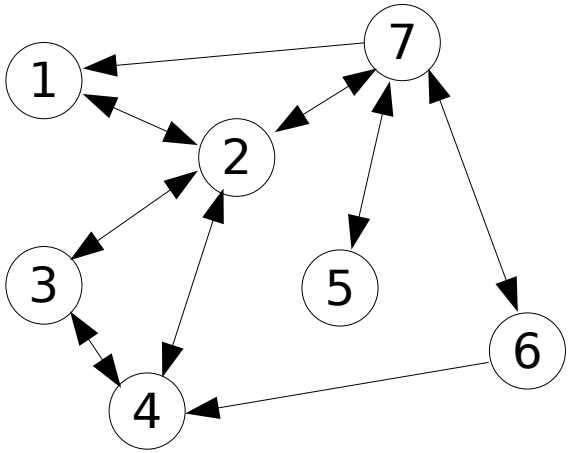


	1	2	3	4	5	6	7
1	x	1	0	0	0	0	0
2	1	x	1	1	0	0	1
3	0	1	x	1	0	0	0
4	0	1	1	x	0	0	0
5	0	0	0	0	x	0	1
6	0	0	0	1	0	x	1
7	1	1	0	0	1	1	x

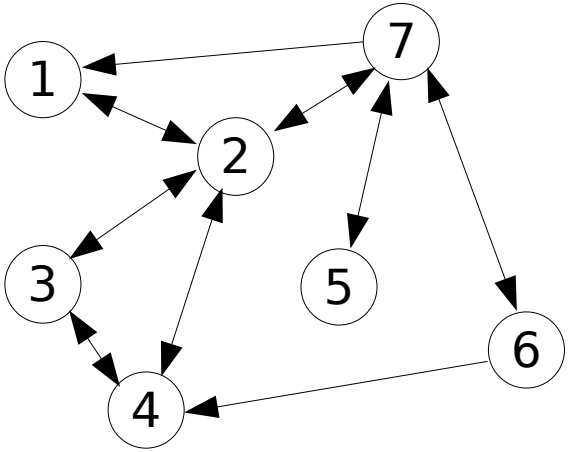
Grafy: reprezentacje

Lista krawędzi – każda krawędź to osobny element listy.





Grafy: reprezentacje



Nie ma ogólnie najlepszej reprezentacji.

Lista krawędzi jest efektywna, gdy krawędzi jest mało (tzw. grafy rzadkie)

Macierz sąsiedztwa jest zwykle najwygodniejsza i najwydajniejsza, ale zajmuje tyle samo (dużo) pamięci niezależnie od liczby krawędzi.

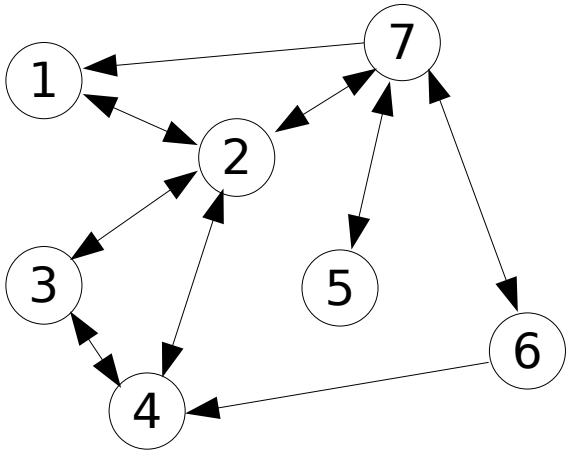
Listy sąsiedztwa są rozwiązaniem pośrednim.

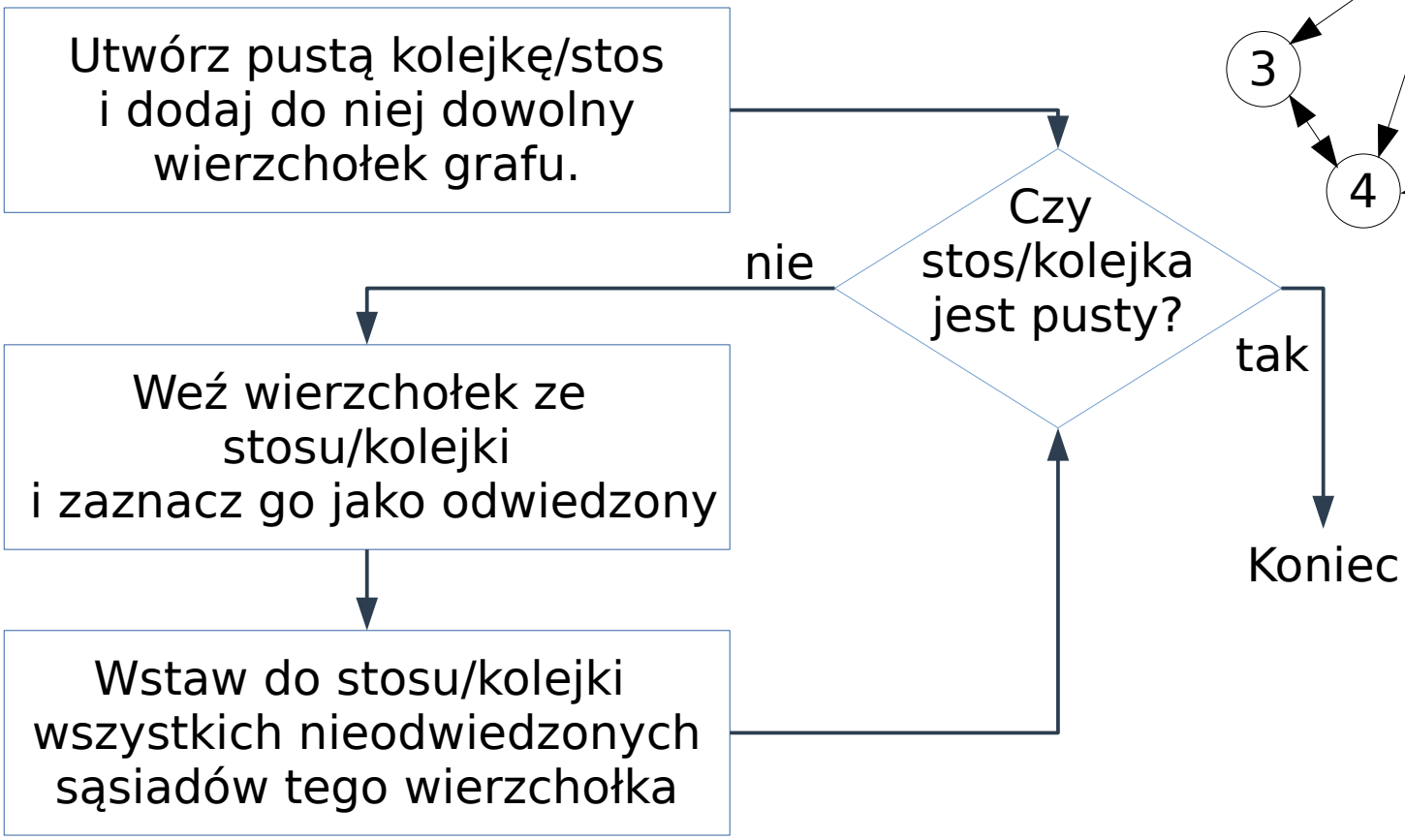
W przypadku listy krawędzi lub macierzy sąsiedztwa, dane zapisane w wierzchołkach grafu można przechowywać w osobnej tablicy, liście lub innej strukturze danych.

Grafy w których wszystkie krawędzie są dwukierunkowe (strzałki na obu końcach) nazywamy grafami nieskierowanymi.

Grafy: algorytmy

Podstawowym algorytmem jest przechodzenie grafu (odwiedzanie wszystkich wierzchołków).





Przechodzenie grafu: uwagi

1. Algorytm odwiedza tzw. spójną składową grafu, zawierającą wierzchołek początkowy. Jeśli pozostały jakieś nieodwiedzone wierzchołki należy uruchomić ten algorytm na kolejnym takim wierzchołku.

2. Stos to struktura z której wyciągamy elementy w kolejności odwrotnej do ich wstawienia (LIFO - Last In First Out). Kolejka to struktura, z której wyciągamy elementy w takiej kolejności w jakiej były wstawiane (FIFO - First In First Out).

3. Użycie kolejki to tzw. przechodzenie/przeszukiwanie grafu wszerz (BFS - Best First Search). Użycie stosu to tzw. przechodzenie/przeszukiwanie grafu wgłąb (DFS – Deep First Search).

4. DFS jest naturalną konsekwencją implementacji rekurencyjnej algorytmu przeszukiwania grafu:

DFS(wierzchołek){ odwiedź wierzchołek; dla każdego nieodwiedzonego sąsiada wywołaj DFS(sąsiad); }

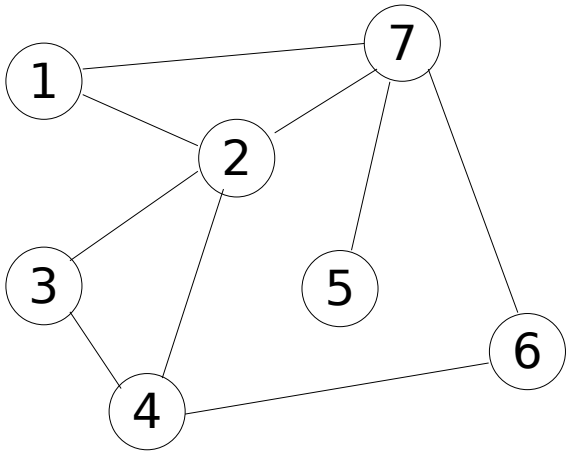
Grafy: kolorowanie grafu

Problem: pokolorować wierzchołki, tak aby sąsiednie wierzchołki miały różny kolor.

Stopień wierzchołka: liczba sąsiadów.

Stopień nasycenia wierzchołka: liczba już pokolorowanych sąsiadów.

Liczba chromatyczna grafu: najmniejsza liczba kolorów potrzebna do pokolorowania grafu.

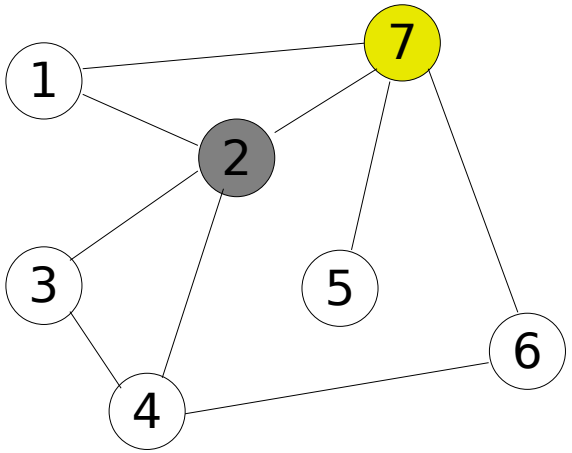


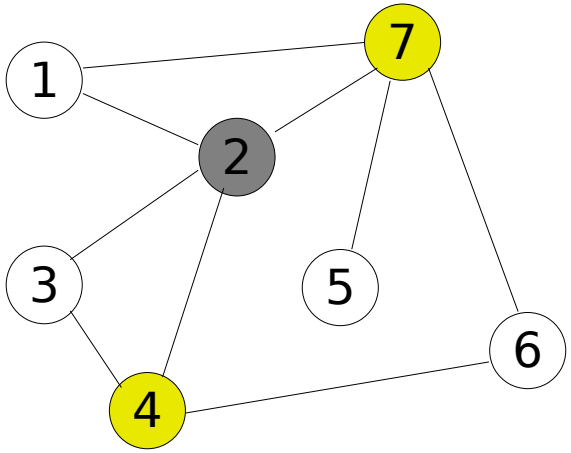
Grafy: kolorowanie

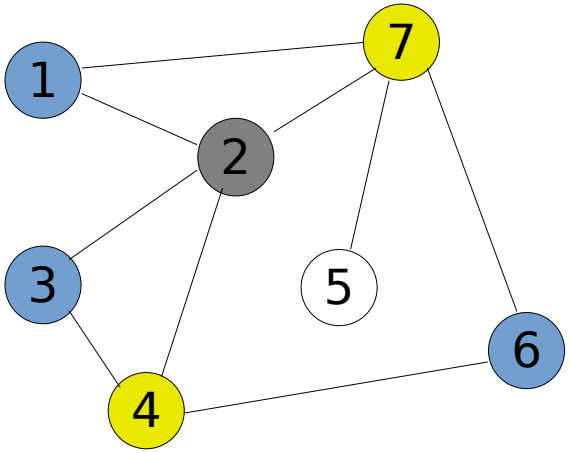
Algorytm LF (largest first).

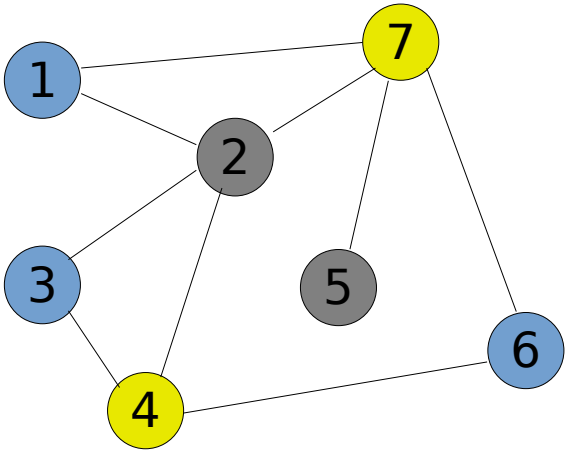
1. Uporządkuj wierzchołki malejąco względem ich stopni.

2. Koloruj po kolei wierzchołki.







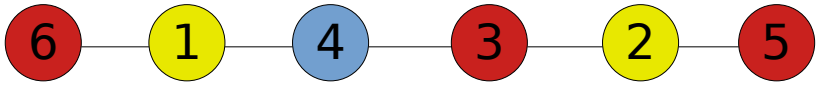


Grafy: kolorowanie

Uwagi:

- algorytm LF jest tzw. algorytmem zachłannym. Wybiera rozwiązanie, która w danym momencie jest najlepsze. Takie rozwiązanie może ale nie nie musi być globalnie najlepsze.

- algorytmy zachłanne nie kolorują grafu „optymalnie”. Problem pojawia się w związku z nieodpowiednią kolejnością kolorowanych wierzchołków.

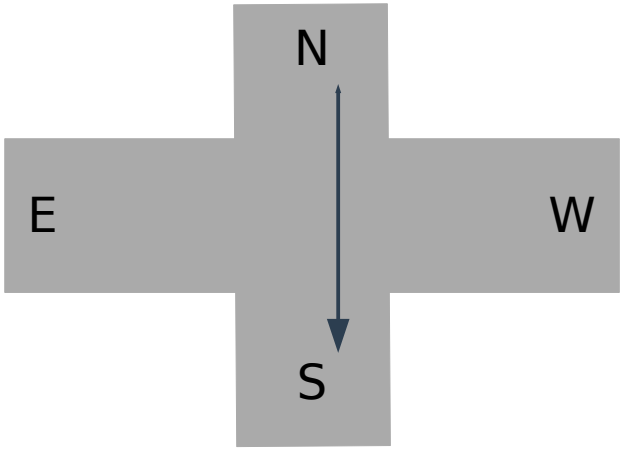


- istnieją grafy tzw. „well-colored graphs” dla których algorytmy zachłanne będą dobrze działać.

- w ogólności problem optymalnego kolorowania grafu jest NP-trudny. Sprowadza się do sprawdzenia wszystkich możliwości, które rosną wykładniczo (lub szybciej) z rozmiarem grafu.

Grafy: kolorowanie

Przykładowe zastosowanie – sterowanie światłami na skrzyżowaniu.



1. Z kierunku E można pojechać do N, W lub S. Każdej takiej trasie odpowiada jeden wierzchołek grafu.

2. Wierzchołki odpowiadające trasom, które się przecinają są połączone krawędziami. Np. trasa NS jest połączona z trasami WE, WS, WN, ES, EW, SW ale nie jest połączona z EN, NE, NW, SN, SE.

3. Optymalne pokolorowanie grafu mówi ile stanów świateł jest potrzebne do sterowania skrzyżowaniem – w ramach każdego stanu „zielone światło” mają trasy niekolidujące ze sobą.

Grafy: minimalne drzewa rozpinające

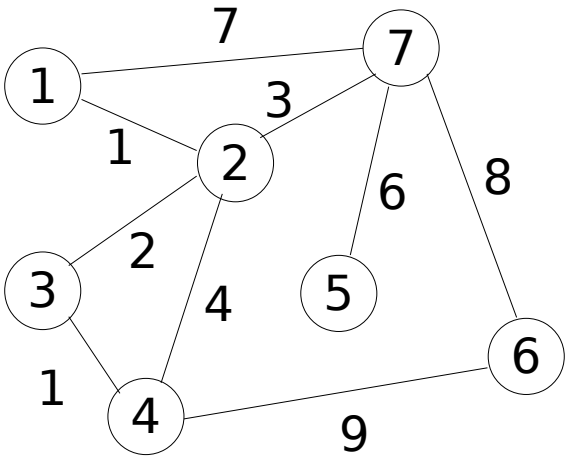
Problem: znaleźć drzewo (graf bez cykli), w którym suma wag krawędzi będzie najmniejsza.

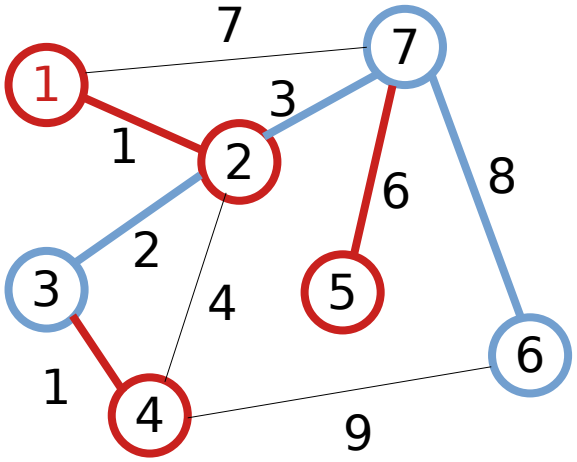
Rozwiązanie: algorytm zachłanny:

1. Weź najkrótszą krawędź i połączone nią wierzchołki – to będzie początek naszego drzewa

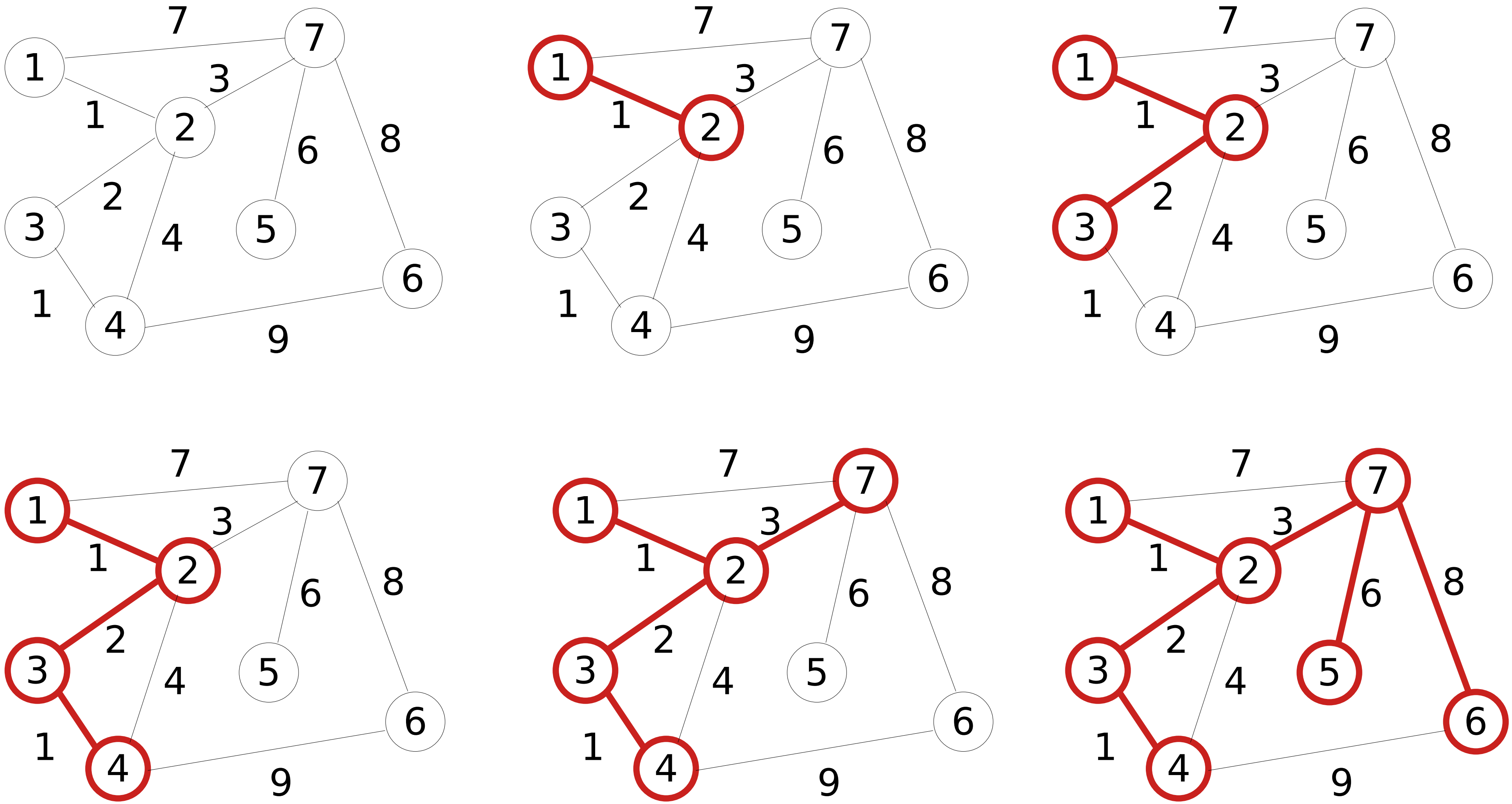
2. Dopóki istnieją wierzchołki nie należące do drzewa:
- dodawaj do drzewa wierzchołek (i krawędź) połączony najkrótszą krawędzią z drzewem.

To tzw. algorytm Kruskala. Istnieją także inne algorytmy zachłanne rozwiązujące ten problem.





Grafy: minimalne drzewa rozpinające



Grafy

Dziękuję za uwagę