Wstęp do programowania

Wykład 8. Grafy i złożoność obliczeniowa

Plan wykładu

1. Algorytmy grafowe c.d.

- najkrótsze ścieżki w grafie,

- maksymalne przepływy.

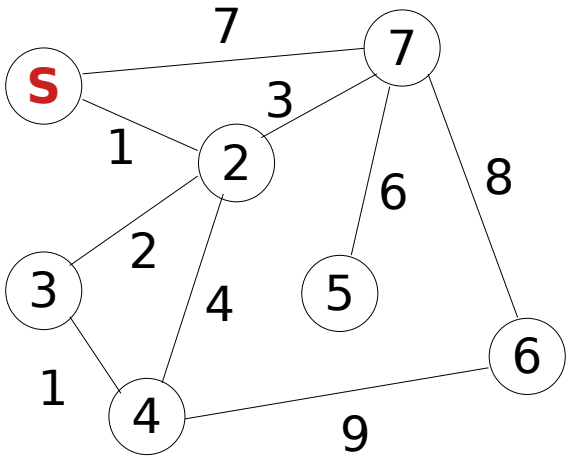
2. Złożoność obliczeniowa

- problemy NP,

- klasyfikacja złożoności algorytmów, problemy NP zupełne,

- problem stopu.

Grafy: najkrótsze ścieżki



Problem: znaleźć najkrótszą drogę między dwoma wierzchołkami.

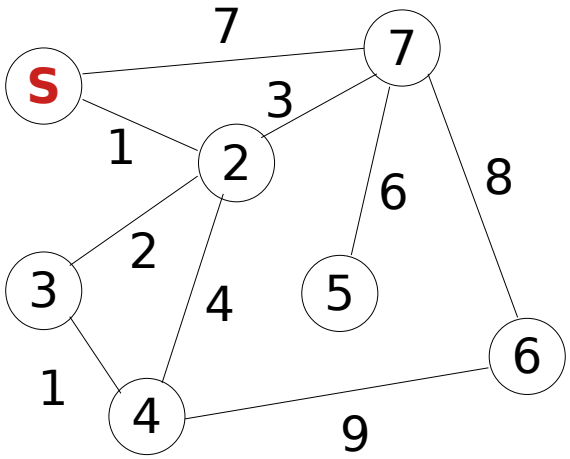
Rozwiązanie: algorytm zachłanny znajdujący drogi z ustalonego wierzchołka S do wszystkich innych:

1. Utwórz tablicę, w której przechowywane będą odległości S do innych wierzchołków. Na początek zainicjuj wartości tablicy nieskończonościami.

2. Weź wierzchołek S i zapisz w tablicy odległości do jego sąsiadów. Zaznacz S jako przeanalizowany.

3. Dopóki istnieją wierzchołki nieprzeanalizowane:
- weź wierzchołek X o najmniejszej odległości. Zaktualizuj odległości jego sąsiadów Y: d(Y) = min{d(Y), d(X)+d(X,Y)}.

Grafy: najkrótsze ścieżki



S	2	3	4	5	6	7
0	∞	∞	∞	∞	∞	∞
-	-	-	-	-	-	-

S	2	3	4	5	6	7
0	1	∞	∞	∞	∞	7
-	S	-	-	-	-	S

S	**2**	3	4	5	6	7
0	1	3	5	∞	∞	4
-	S	2	2	-	-	2

S	**2**	**3**	4	5	6	7
0	1	3	4	∞	∞	4
-	S	2	3	-	-	2

S	**2**	**3**	4	5	6	**7**
0	1	3	4	10	12	4
-	S	2	3	7	7	2

itd.
Kolejne iteracje nie wprowadzą zmian.

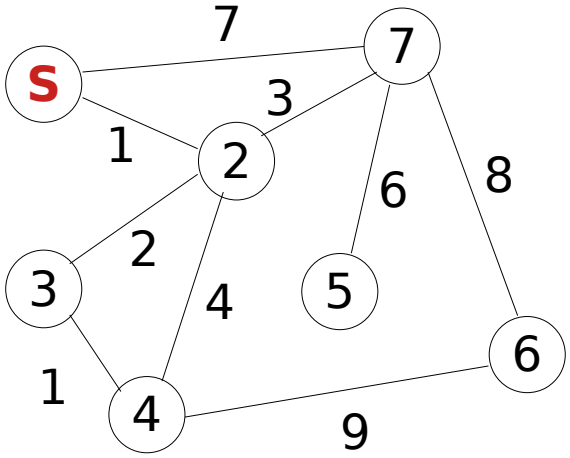
Grafy: najkrótsze ścieżki

Algorytm Dijkstry:

- nie wymaga spełnienia przez odległość nierówności trójkąta.

- wymaga nieistnienia ścieśek o ujemnej długości

- jest kolejnym przykładem algorytmu zachłannego



Grafy: najkrótsze ścieżki

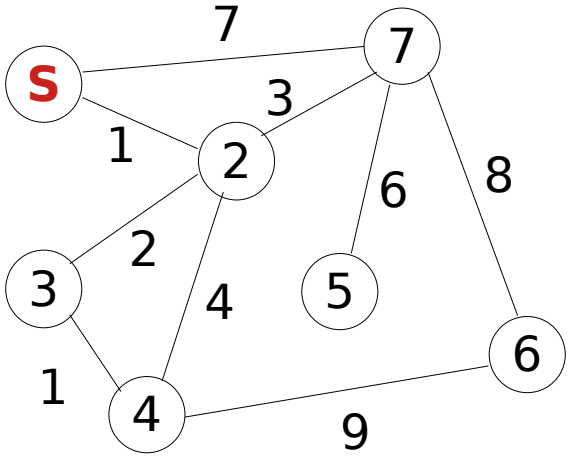
Algorytm Bellmana-Forda:

- ścieżki mogą mieć ujemne wagi

- wymaga nieistnienia cykli o ujemnej długości

1. Wykonuj poniższe operacje tyle razy ile jest wierzchołków w grafie - 1.

2. Dla każdej z krawędzi ((X,Y) zaktualizuj:
   d(Y) = min{d(Y), d(X)+d(X,Y)}



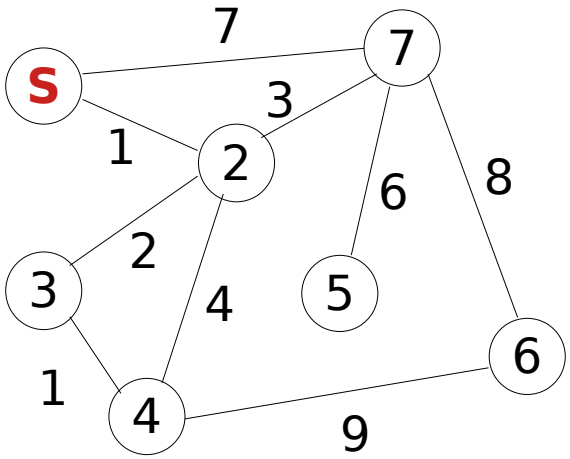
Grafy: najkrótsze ścieżki

S	2	3	4	5	6	7
0	∞	∞	∞	∞	∞	∞
-	-	-	-	-	-	-

S	2	3	4	5	6	7
0	1	∞	∞	∞	∞	7
-	S	-	-	-	-	S

S	2	3	4	5	6	7
0	1	3	5	13	15	4
-	S	2	2	7	7	2

S	2	3	4	5	6	7
0	1	3	4	10	12	4
-	S	2	3	7	7	2



itd.
Kolejne iteracje nie wprowadzą zmian.

Grafy: przepływy

Sieć przepływu: dla każdej krawędzi określamy ile jednostek przez nią aktualnie przepływa f(X,Y), na początku f(X,Y) = 0.

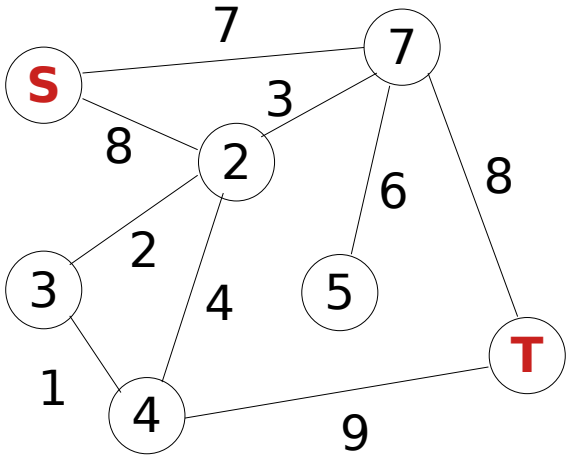
Sieć rezydualna: dla każdej krawędzi określamy ile jeszcze może przez nią przepłynąć c(X,Y), na początku C(X,Y) =w(X,Y).

Ścieżka powiększająca: ścieżka w sieci rezydualnej z S do T.

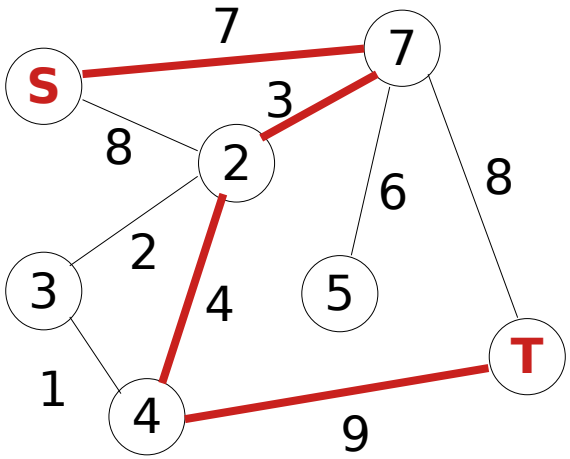
Algorytm Forda-Fulkersona:

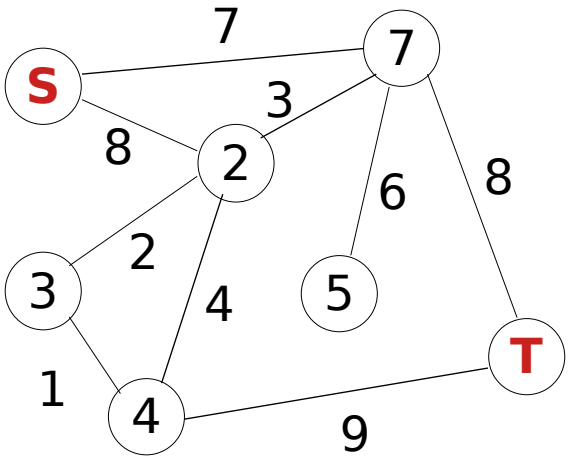
1. Jeśli istnieje ścieżka powiększająca to używając jej zaktualizuj sieć przepływu i sieć rezydualną,

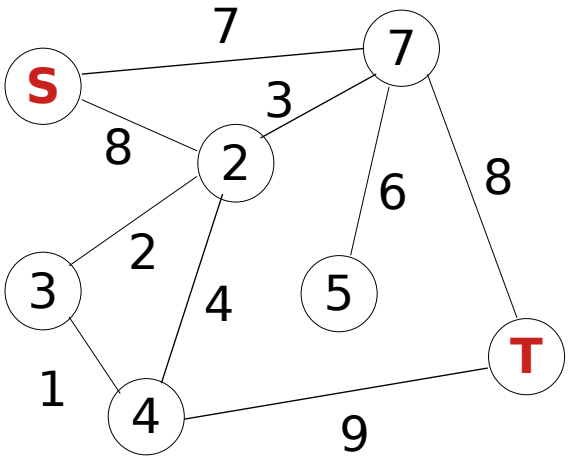
2. Wróć do punktu 1.

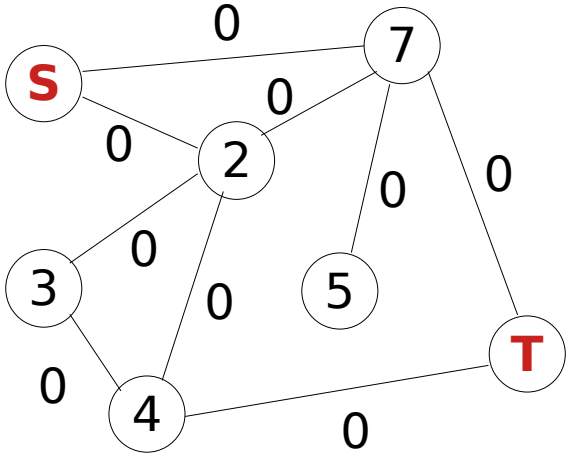


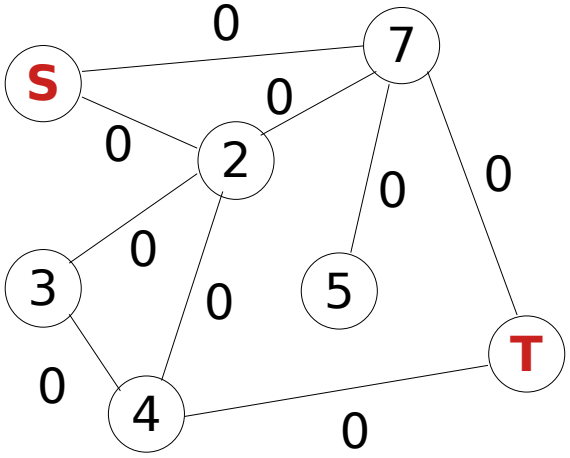
Grafy: przepływy

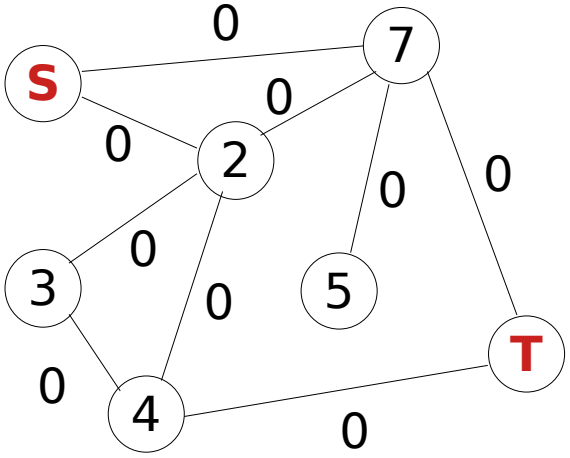












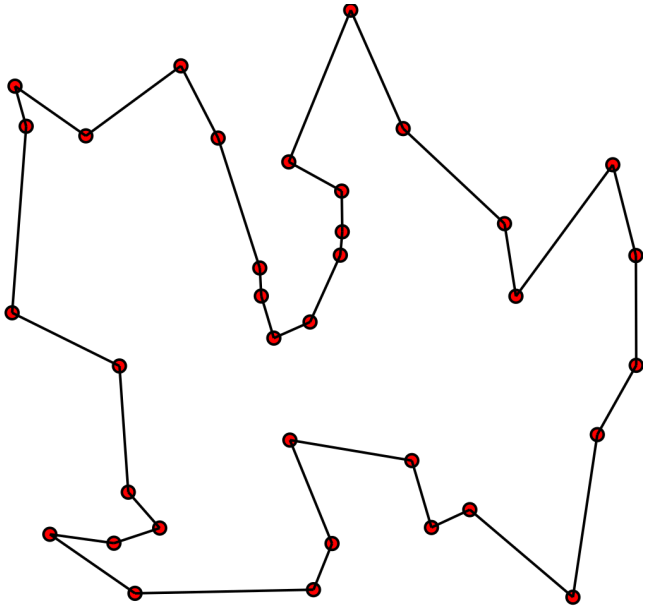
Problem komiwojażera

Wersja decyzyjna:

dany jest zbiór miast i odległości między nimi. Czy istnieje droga zaczynająca się i kończąca w tym samym mieście, odwiedzająca wszystkie inne miasta o długości mniejszej niż zadana wartość?

Wersja optymalizacyjna:

Wskazać najkrótszą taką drogę.



https://en.wikipedia.org/wiki/File:GLPK_solution_of_a_travelling_salesman_problem.svg

Problem komiwojażera

Własności:

Jeśli dostaniemy rozwiązanie (wersji decyzyjnej), to łatwo (w czasie wielomianowym – tutaj nawet liniowym - O(n)) sprawdzić, czy jest ono poprawne.

Poszukiwanie rozwiązania:

Startujemy z wybranego wierzchołka i przechodzimy do jednego z (n-1) pozostałych wierzchołków grafu. Potem z każdego kolejnego do (n-2) wierzchołków itd.

Problem komiwojażera

Komputer deterministyczny:

Analiza (n-1)! tras.

Komputer niedeterministyczny:

W pierwszym kroku uruchomia się (n-1) niezależnych „komputerów deterministycznych”, każdy dla innego odwiedzonego wierzchołka. W drugim kroku dla każdego z węzłów uruchamia się kolejnych (n-2) „komputerów deterministycznych” itd.

Komputer niedeterministyczny znajdzie odpowiedź po (n-1) krokach. To oznacza, że problem komiwojażera jest tzw. problemem NP.

P versus NP

Problemy P to takie, które można rozwiązać w czasie co najwyżej wielomianowym na komputerze deterministycznym.

P: wyszukiwanie elementu w posortowanej tablicy O(log n), wyszukiwanie elementu w nieposortowanej tablicy O(n), sortowanie O(n log n), mnożenie macierzy $O(n^3)$, ...

Problemy NP to takie, które można rozwiązać w czasie co najwyżej wielomianowym na komputerze niedeterministycznym.

NP: wszystkie P oraz problem komiwojażera, kolorowanie grafu, ...

Nie wiadomo, czy istnieje problem który jest NP a nie jest P.

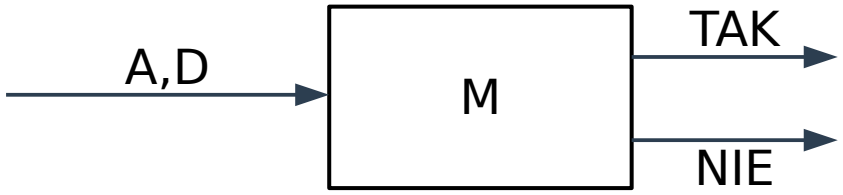
Istnieje klasa problemów, do których w czasie wielomianowym można sprowadzić wszystkie problemy NP. Ta klasa to problemy NP zupełne.. Wskazanie algorytmu o złożoności wielomianowej dla dowolnego problemu z tej klasy wykazałoby, że NP=P.

Problem stopu

Dany jest program (komputerowy) A oraz zestaw danych D, na których on operuje.

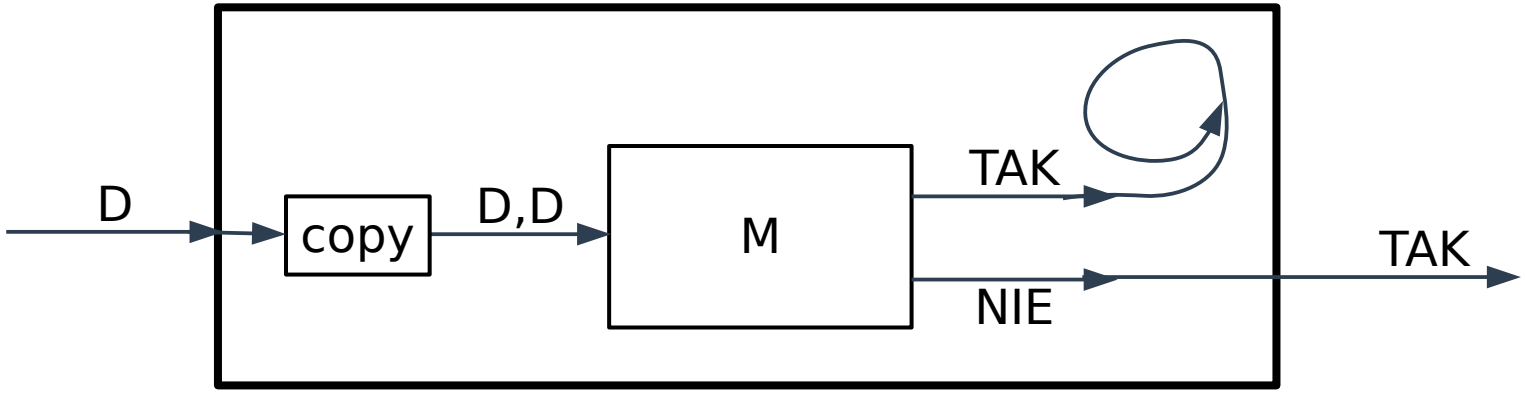
Pytanie: czy ten program zakończy działanie, czy będzie działał w „nieskończoność”.

Hipoteza: istnieje program M, który odpowiada na to pytanie w skończonym czasie.

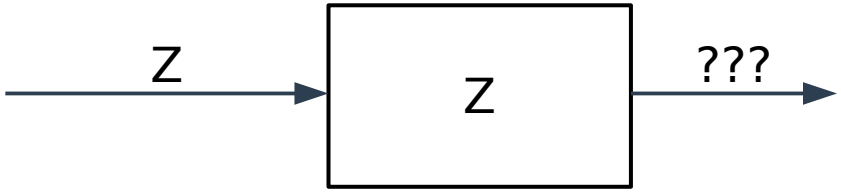


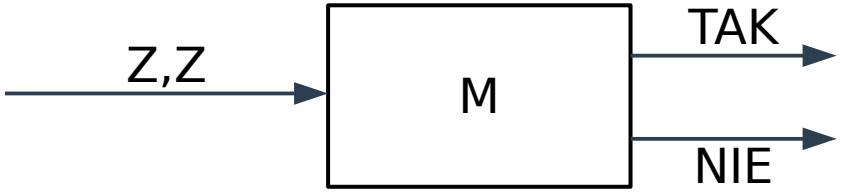
Problem stopu

Konstruujemy program (algorytm Z) w następujący sposób:



Jak zadziała algorytm Z jeśli dostanie na wejściu swój własny kod?





Grafy

Dziękuję za uwagę