Matematyka dyskretna, Wykład 11, 19/12/2019

Najkrótsze ścieżki w grafach:

3 sformułowania problemu:

- znajdź najkrótszą ścieżkę z u do v
- znajdź najkrótszą ścieżkę z v_0 do wszystkich innych wierzchołków
- znajdź najkrótsze ścieżki między wszystkimi parami wierzchołków

Algorytm Dijkstry (dla nieujemnych długości)

Algorytm Warshalla-Floyda (znajduje długości najkrótszych ścieżek)

$a[i,j] \leftarrow$ długość krawędzi (i,j) lub ∞ gdy między i,j nie ma krawędzi

$a[i,i] \leftarrow 0$

```
FOR k ← 1 TO n:
    FOR i ← 1 TO n:
        FOR j ← 1 TO n:
            a[i,j] ← min {a[i,j]; a[i,k] + a[k,j]}
```

Na końcu otrzymamy tablicę/macierz $a[i,j]$, w której będą znajdować się długości najkrótszych dróg z i do j. Algorytm ten działa także dla grafów z ujemnymi wagami krawędzi pod warunkiem braku cykli o długości ujemnej, ponieważ w takiej sytuacji:

przechodząc z u do v moglibyśmy wchodzić w cykl cały czas, dzięki czemu najkrótsza ścieżka nie istniałaby (zawsze możemy przejść cykl kolejny raz, uzyskując krótszą ścieżkę).

Można powiedzieć, że niezmiennikiem algorytmu Warshalla-Floyda jest:

Po k-tej iteracji najbardziej zewnętrznej pętli, $a[i,j]$ zawiera długość najkrótszej drogi z i do j, w której wierzchołkami pośrednimi mogą być $\{1, 2, \ldots, k\}$.

Przechodnie domknięcie grafu skierowanego

G - graf skierowany
G^* - graf przechodniego domknięcia G

1° $V(G) = V(G^*)$
2° W G^* jest krawędź $(u,v) \iff$ w G istnieje droga z u do v
3° $uRv \iff (u,v) \in E(G)$

Znajdowanie przechodniego domknięcia:

- Algorytm Warshalla - Floyda

```
FOR k ← 1 TO n:
    FOR i ← 1 TO n:
        FOR j ← 1 TO n:
            a_ij ← a_ij ∨ (a_ik ∧ a_kj)
```

macierz sąsiedztwa G $\longrightarrow a_{ij}$

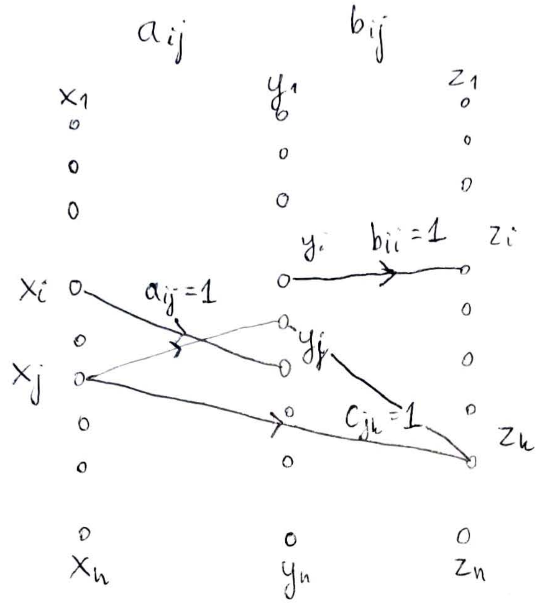
Na końcu a_{ij} jest macierzą sąsiedztwa grafu G^*, czyli macierzą przechodniego domknięcia.

Niech A - macierz sąsiedztwa, a A^* - macierz przechodniego domknięcia, wtedy $A^* = I + \underbrace{A + A^2 + A^3 + \dots}_{\text{potęgi oznaczają długość drogi z } i \text{ do } j} = I + A + A^2 + \dots + A^{n-1}$ = ↑ długość najdłuższej drogi

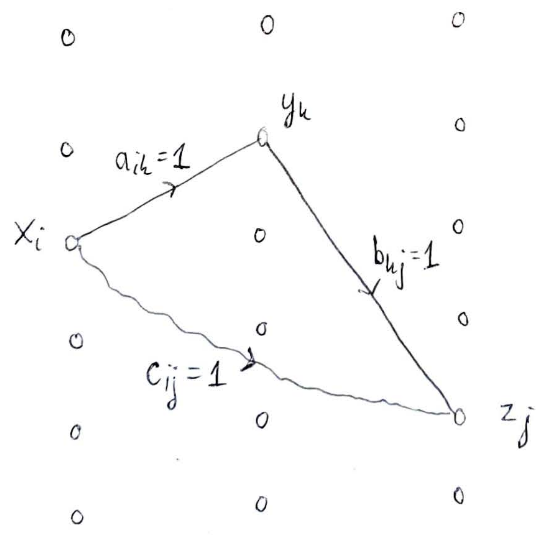
$$= (I+A)^{n-1} = \underbrace{((((I+A)^2)^2 \dots)^2)^2}_{\log n}$$

FAKT: A^* można policzyć w czasie $M(n) \cdot \log n$, gdzie $M(n)$ to czas mnożenia macierzy $n \times n$.

Twierdzenie: Czas obliczania przechodniego domknięcia nie jest asymptotycznie szybszy niż czas mnożenia macierzy $n \times n$.



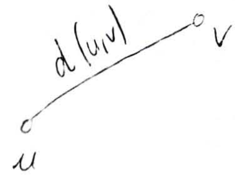
$$\{a_{ij}\}\{b_{ij}\} = \{c_{ij}\}$$



$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Najlżejsze drzewo spinające (rozpinające)

G – graf nieskierowany z wagami na krawędziach

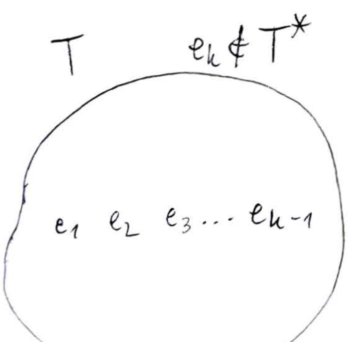


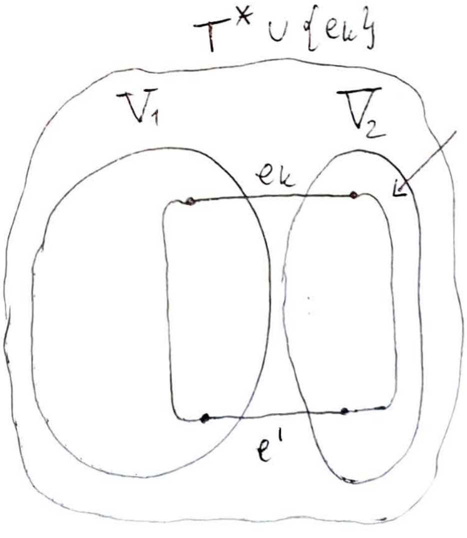
Ogólny schemat:

- Na początku T jest grafem pustym.
- Dopóki T nie jest drzewem:
    - do T dodajemy najlżejszą krawędź między **zbiorami** wierzchołków V_1 i V_2 takimi, że $V_1 \cup V_2 = V(G)$, $V_1 \cap V_2 = \emptyset$, V_1 i V_2 nie są połączone w T krawędzią.

Dowód:

Załóżmy nie wprost, że ten schemat nie produkuje najlżejszego drzewa spinającego. Oznaczmy drzewo zwracane przez ten schemat jako T. Niech T^* będzie najlżejszym drzewem spinającym zawierającym najdłuższy początkowy ciąg krawędzi dodawanych przez schemat.





cykl $C \in T^* \cup \{e_k\}$

$d(e') \geq d(e_k)$

$\underbrace{T^* \cup \{e_k\} \setminus \{e'\}}_{\text{graf spójny, ma } n-1 \text{ krawędzi}}$

Niech $T^{**} = T^* \cup \{e_k\} \setminus \{e'\}$, wtedy $d(T^{**}) \leq d(T^*)$.

T^{**} zawiera $e_1 e_2 \dots e_{k-1} e_k$.

[...]

Dochodzimy do sprzeczności. ∎

Algorytm Kruskala

1° Uporządkuj krawędzie w porządku $d(e_1) \le d(e_2) \le \ldots \le d(e_m)$.

2° $T \leftarrow \emptyset$

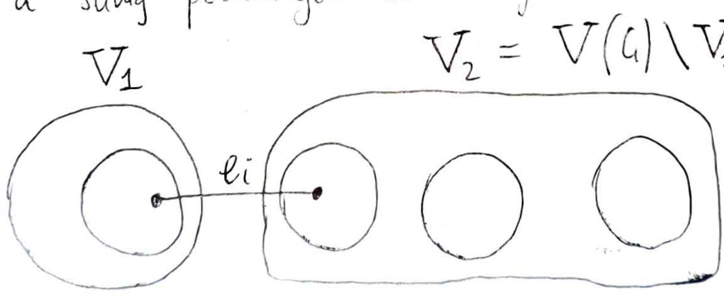
3° FOR $i \leftarrow$ TO m:

IF (dodanie e_i do T nie tworzy cyklu):

$T \leftarrow T \cup \{e_i\}$

Dowód poprawności:

- T nie posiada cykli
- T jest spójny (bo G jest spójny, więc dla dowolnych V_1, V_2 takich że $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V(G)$, istnieje e_i, który łączy te zbiory - algorytm Kruskala doda do T najwcześniejszą taką krawędź)
- algorytm Kruskala dodaje krawędź między składowymi, w której jest jeden z końców dodawanej krawędzi a sumą pozostałych składowych



Algorytm Prima - Dijkstry (v_0)

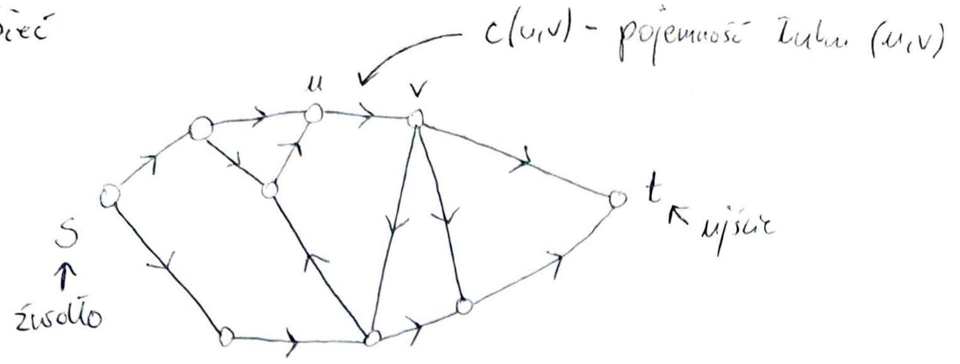
INICJALIZACJA:

$V' \leftarrow \{v_0\}$
FOR $v \in N(v_0)$
  $d[v] \leftarrow d(v, v_0)$ - graf nieskierowany, więc $d(v,v_0) \equiv d(v_0, v)$
  $p[v] \leftarrow v_0$
FOR $v \notin N(v_0)$
  $d[v] \leftarrow \infty$

WHILE $V' \neq V$
  wybierz z $V \setminus V'$ takie v, że $d[v]$ jest minimalny
  $V' \leftarrow V' \cup \{v\}$
  FOR $u \in N(v)$
    IF $(d[u] > d(v,u))$
      $d[u] \leftarrow d(v,u)$
      $p[u] \leftarrow v$

- Algorytm dodaje do V' $n-1$ wierzchołków (jednocześnie dodając $n-1$ krawędzi)
- Algorytm działa wg ogólnego schematu dodając w każdym kroku najkrótszy krawędź między V' i $V \setminus V'$ (po każdym kroku w $d[u]$ jest długość najkrótszej krawędzi łączącej u z V')

Sieć



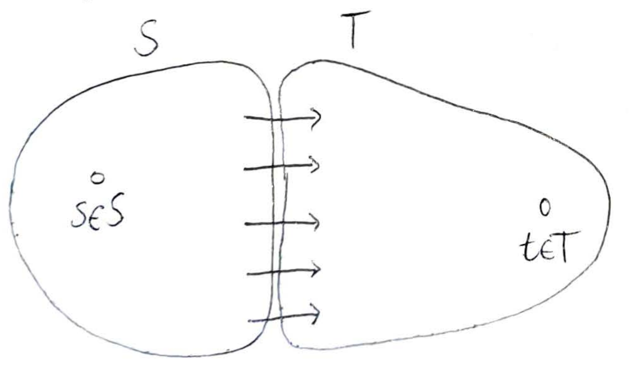
$f(u,v)$ - przepływ z u do v, spełnia on warunki:

1° $0 \leq f(u,v) \leq c(u,v)$

2° prawo Kirchoffa, tzn. $\sum_u f(u,v) = \sum_u f(v,u)$, czyli tyle co wchodzi na wejściu, musi wyjść na wyjściu $(v \neq s,t)$

$$\hat{f} = \sum_u f(s,u)$$

Przekrój (sieci)



Pojemność przekroju S,T to

$$c(S,T) = \sum_{\substack{u \in S, \\ v \in T}} c(u,v).$$

$f(S,T)$ - przepływ netto z S do T

$$f(S,T) = \sum_{\substack{u \in S, \\ v \in T}} f(u,v) - \sum_{\substack{u \in S, \\ v \in T}} f(v,u)$$

Lemat: $f(S,T) = \hat{f}$ dla każdego S,T

$$f(S,T) = \sum_{\substack{u\in S,\\ v\in T}} f(u,v) - \sum_{\substack{u\in S,\\ v\in T}} f(v,u) \quad \sum_{u,v\in S} f(u,v) - \sum_{u,v\in S} f(u,v) =$$

$$= \underbrace{\sum_{u\in S\setminus\{s\}} f(s,u)}_{\hat{f}} + \underbrace{\sum_{v\in S} \underbrace{\left(\sum_{u\in V} f(v,u) - \sum_{u\in V} f(u,v) \right)}_{0 \text{ - z prawa Kirchoffa}}}_{\sum 0 = 0} = \hat{f} \ \blacksquare$$