Algorytmy i Struktury Danych

Wykład 9

Najkrótsze ścieżki między każdą parą wierzchołków

Konwencja

W specjalizowanych algorytmach stosuje się reprezentację macierzową

$D[u][v]$ – dł. najkrótszej ścieżki z u do v

$P[u][v]$ – "parent" – wierzchołek tuż przed v na najkrótszej ścieżce z u do v

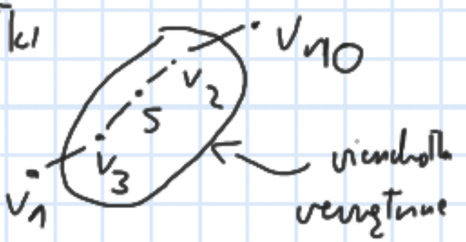
Algorytm Floyda - Warshalla

Idea: Jeśli znamy najkrótsze ścieżki między każdą parą wierzchołków, które jako wewnętrzne wierzchołki używają

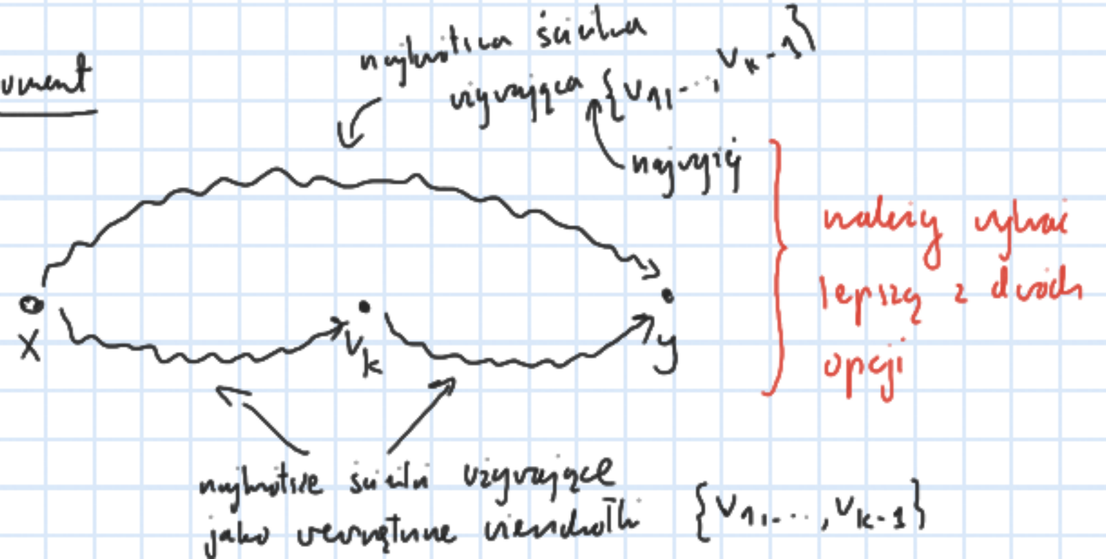
$\{v_1, \dots, v_{k-1}\}$

to możemy osiągnąć to samo dla

$\{v_1, \dots, v_k\}$



Argument



Konwencja

$V = \{v_1, \dots, v_n\}$

$S^{(t)}$ - macierz długości najkrótszych ścieżek spełniających o wierzchołkach ze zbioru $\{v_1, \dots, v_t\}$

$S^{(0)}$ - macierz wag krawędzi między wierzchołkami (∞ oznacza brak krawędzi)

złożoność

$\Theta(n^3)$

$P^{(0)}[x][x] = None$

$P^{(0)}[x][y] = \begin{cases} x & \text{- jeśli mamy krawędź z } x \text{ do } y \\ None & \text{- w p.p.} \end{cases}$

Algorytm

```
for t in range(1, n+1):
    for x ∈ V:
        for y ∈ V:
            S^(t)[x][y] = min(S^(t-1)[x][y], S^(t-1)[x][v_t] + S^(t-1)[v_t][y])
D = S^(n)
```

range(n)

w implementacjach używamy tylko jednej macierzy

uzupełnianie macierzy P

$P^{(t)}[x][y] = P^{(t-1)}[x][y]$

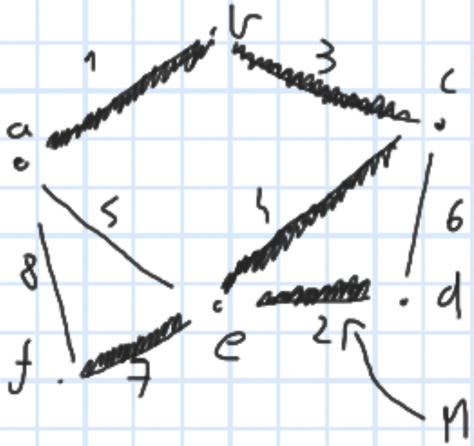
$P^{(t)}[x][y] = P^{(t-1)}[v_t][y]$

Minimalne drzewa rozpinające (minimal spanning trees, MST)

$G = (V, E)$ - graf nieskierowany ← spójny

$w: E \to \mathbb{R}_+$

Cel: znaleźć podzbiór krawędzi tworzący spójny podgraf (obejmujący wszystkie wierzchołki), o minimalnej sumie wag



Obserwacja

$G = (V, E)$, $w: E \to \mathbb{R}_+$

Jeśli $A \subseteq E$ jest podzbiorem krawędzi pewnego MST dla G oraz $e = \{u, v\}$ jest krawędzią, taką że:

a) $e \notin A$

b) $A \cup \{e\}$ nie zawiera cyklu

c) e ma minimalną wagę wśród krawędzi spełniających powyższe warunki

to $A \cup \{e\}$ jest podzbiorem krawędzi pewnego MST dla G

Dowód

Jeśli $e = \{u, v\}$ nie należy do żadnego MST zawierającego A, to istnieje inna krawędź, różna od e (spoza A) zapewniająca trasę z u do v w MST rozszerzającym A

Niech to będzie $e' = \{x, y\}$

Tworzymy $T = (T' - \{e'\}) \cup \{e\}$

$w(e') \geq w(e)$

$w(T) \leq w(T')$, więc T to MST

T' - rozważane A:

Algorytm Kruskala dla MST

1. Posortuj krawędzie po wagach
2. $A = \emptyset$
3. Przeglądaj krawędzie $e \in E$ w kolejności nierosnących wag
   - jeśli $A \cup \{e\}$ nie zawiera cyklu to
     $A := A \cup \{e\}$
4. Zwróć A

$O(E \log V)$ - złożoność czasowa
(o ile szybko wykrywamy krawędzie tworzące cykl)

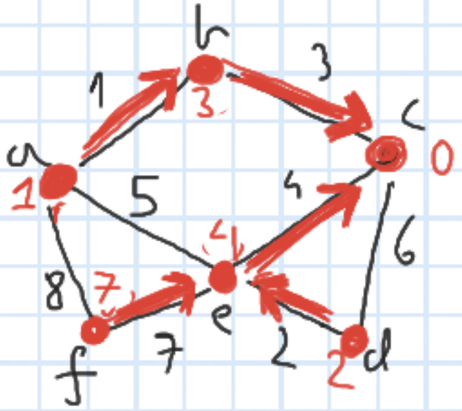
Algorytm Prima dla MST

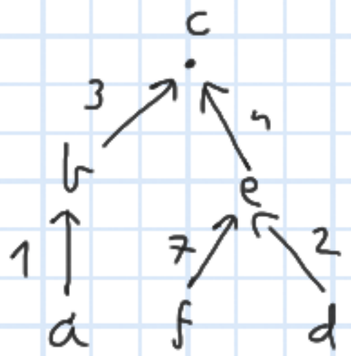
v - wierzchołek startowy

1. Umieść wszystkie wierzchołki w kolejce priorytetowej z wagą ∞
2. Zmień wagę v na 0
3. Póki wierzchołki są w kolejce:
   - wyjmij wierzchołek u o minimalnej wadze z kolejki
   - dla każdej krawędzi $\{u, x\}$, jeśli waga $w(\{u,x\}) <$ waga x w kolejce, to zmień wagę x na $w(\{u,x\})$ (zaktualnij parent)

$O(E \log V)$

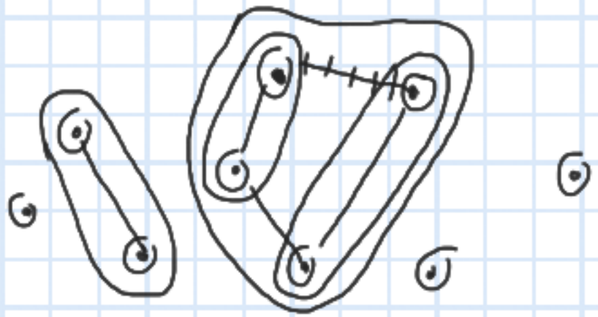
Przykład



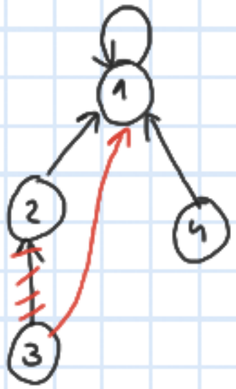


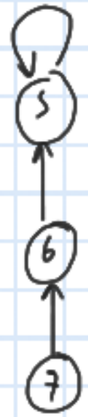
Rodzina zbiorów rozłącznych – struktura Find/Union

Idea



Struktura Find/Union oparta o las zbiorów rozłącznych





Operacja Find: Wędrujemy w górę drzewa do korzenia i jego zwracamy jako identyfikator zbioru (reprezentanta)
+ kompresja ścieżki

Operacja Union: dołączamy korzeń jednego drzewa do drugiego

z każdym zbiorem łączymy tzw. "rangę" i dołączamy drzewo o mniejszej randze do tego o większej (jeśli rangi równe, to dowolnie które dołączamy, ale ranga rośnie o 1)

```
class Node
    def __init__(self, value):
        self.parent = self
        self.rank = 0
        self.value = value
```

```
def findset(x):
    if x.parent ≠ x:
        x.parent = findset(x.parent)
    return x.parent
```

```
def union(x, y):
    x = findset(x)
    y = findset(y)
    if x.rank > y.rank:
        y.parent = x
    else:
        x.parent = y
        if x.rank = y.rank:
            y.rank += 1
```

Jeśli stosujemy łączenie wg. rangi oraz kompresję ścieżki to ciąg m operacji ma złożoność $O(m \log^* m)$

gdzie

$\log^*(m)$ to liczba zastosowań $\log$ do m potrzebna by wynik spadł do ≤ 1

$\log(\log(\log(m))) \leq 1$