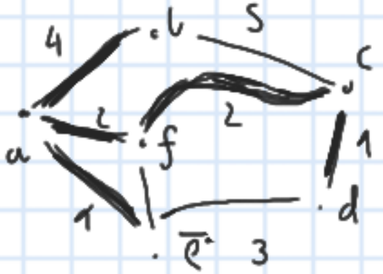
Algorytmy i Struktury Danych

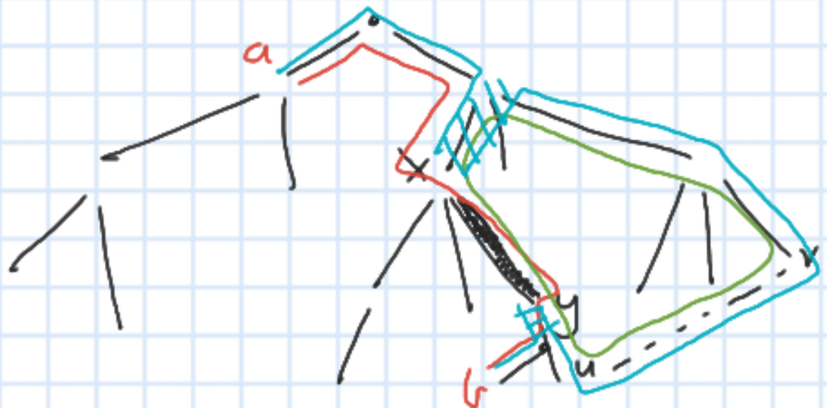
Wykład 9

Minimalne drzewo rozpinające – zbiór krawędzi, które łączą wszystkie wierzchołki danego grafu i których suma wag jest minimalna



(Problem dotyczy grafów spójnych, nieskierowanych)

MST – minimal spanning tree



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

Wówczas $A \cup \{e\}$ jest podzbiorem krawędzi pewnego MST

---

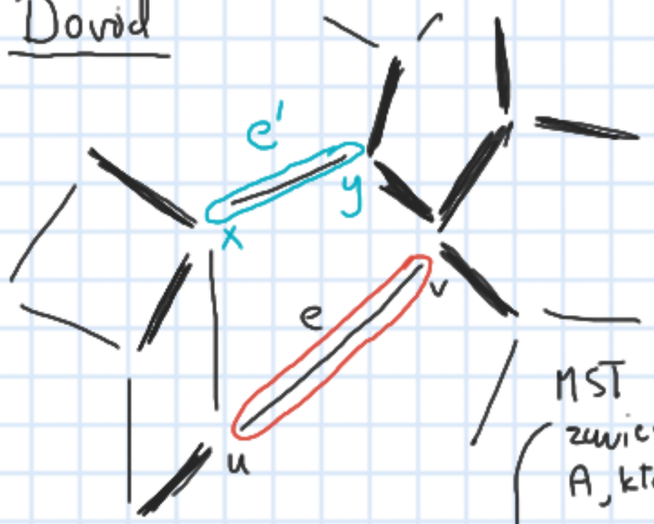
dla każdej pary wierzchołków istnieje łącząca je ścieżka

$a \rightsquigarrow x - y \rightsquigarrow b$

$\Downarrow$

$a \rightsquigarrow x \rightsquigarrow \{u, v\} \rightsquigarrow y \rightsquigarrow b$

Dowód



MST T zawierające A, które

Jeśli nie istnieje MST rozszerzające $A \cup \{e\}$, to istnieje zapewnia ścieżkę z u do v jakąś inną krawędzią $e' = \{x, y\}$ spoza A

Twierdzimy, że $T' = (T \setminus \{e'\}) \cup \{e\}$ również jest MST

Zauważmy, że:

① $w(T) \geq w(T')$ bo $w(e') \geq w(e)$

← ② T' ucięli jest drzewem

Algorytm Kruskala znajdowania MST

Na wejściu mamy $G = (V, E)$, $w : E \to \mathbb{R}$

① Posortuj krawędzie po wagach — $O(E \log E)$

② $A := \emptyset$ — $O(1)$

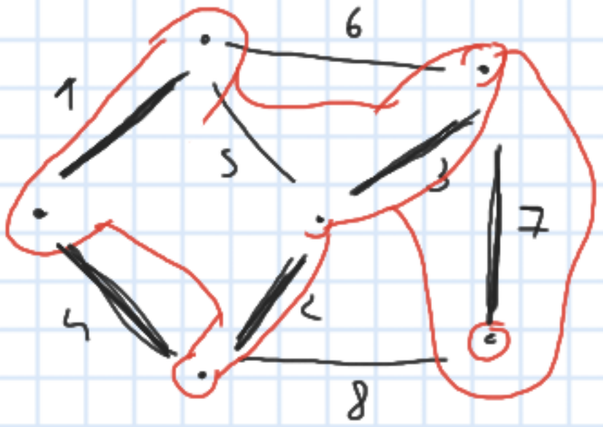
③ Przeglądaj krawędzie w kolejności niemalejących wag (rozważana krawędź to e)

Jeśli $A \cup \{e\}$ nie tworzy cyklu to

$A := A \cup \{e\}$

$O(E)$ op. na Find-Union

④ Zwróć A



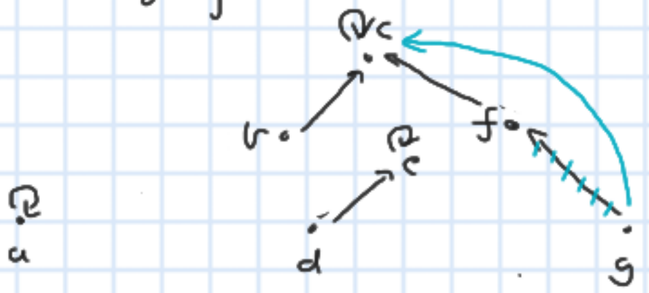
cały algorytm ma złożoność

$O(E \log E) =$

$O(E \log V)$

Struktura Find-Union dla zbiorów rozłącznych

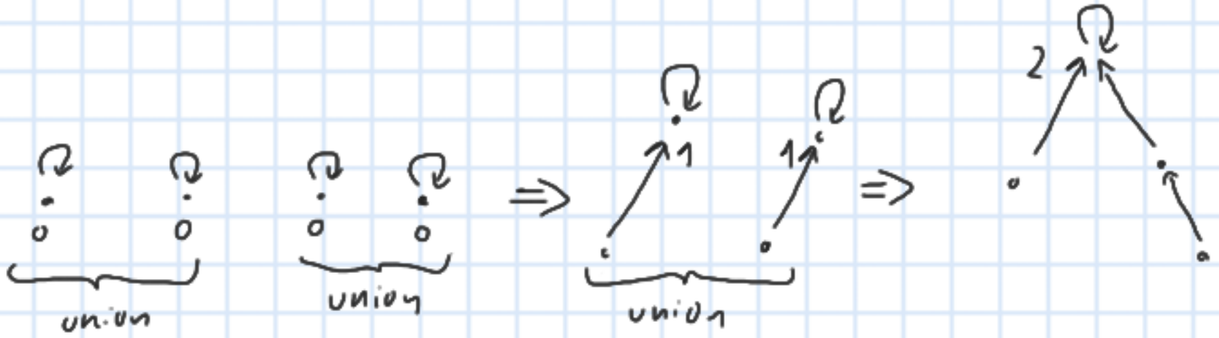
Strukturę Find-Union realizujemy jako las zbiorów rozłącznych




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

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

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



Obserwacja

Dowo o randze k ma co najmniej 2^k elementów oraz ścieżki o długości co najwyżej k

Obserwacja

Jeśli mamy n elementów to najwyższa ranga wynosi $\log n$, czyli każda operacja find ma złożoność $O(\log n)$

z Tarjanem według rangi i kompresją ścieżki

Fakt

Jeśli wykonujemy m operacji na strukturze Find-Union zawierającej n elementów, to ich łączny koszt wynosi $O(m \log^* n)$

gdzie $\log^* n$ to to ile razy należy zastosować log, żeby argument spadł do 1 lub mniej

Algorytm Prima znajdowania MST

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

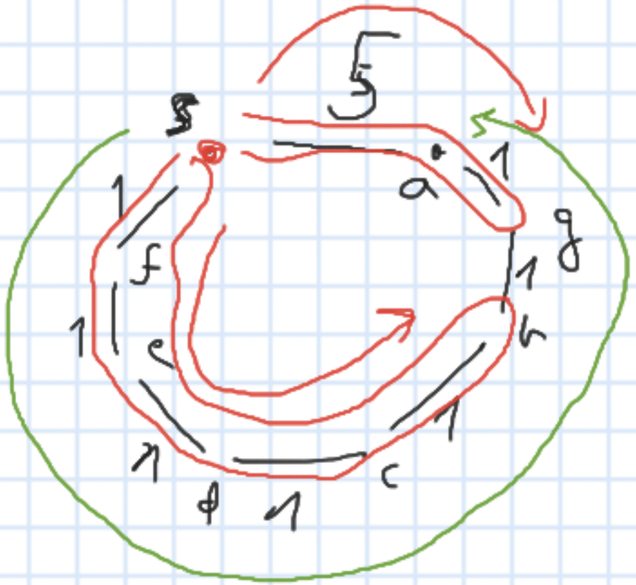
① Umieść wszystkie wierzchołki w kolejce, z wagą ∞

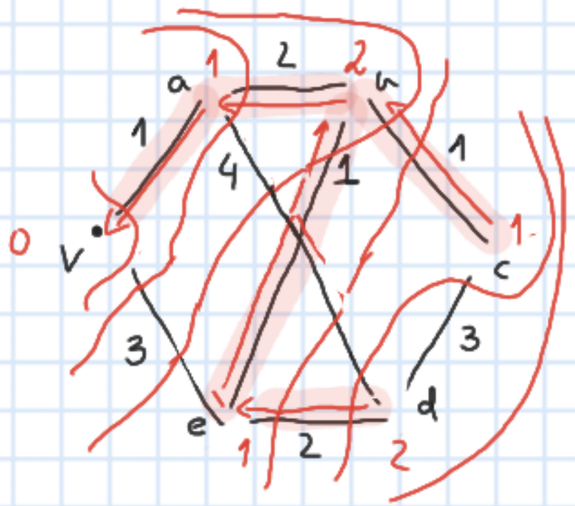
② Zmień wagę wierzchołka v na 0

↑ wierzchołek startowy, zadany na wejściu

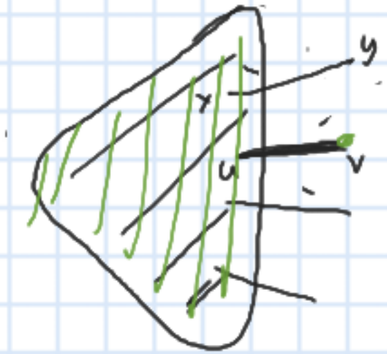
③ Jak długo są wierzchołki w kolejce:

- wyjmij wierzchołek t o minimalnej wadze z kolejki
- dla każdej krawędzi $e = \{t, u\}$, jeśli waga u jest $\geq w(t,u)$ to zmniejsz jej do $w(t,u)$ i ustaw $u.parent$ na t

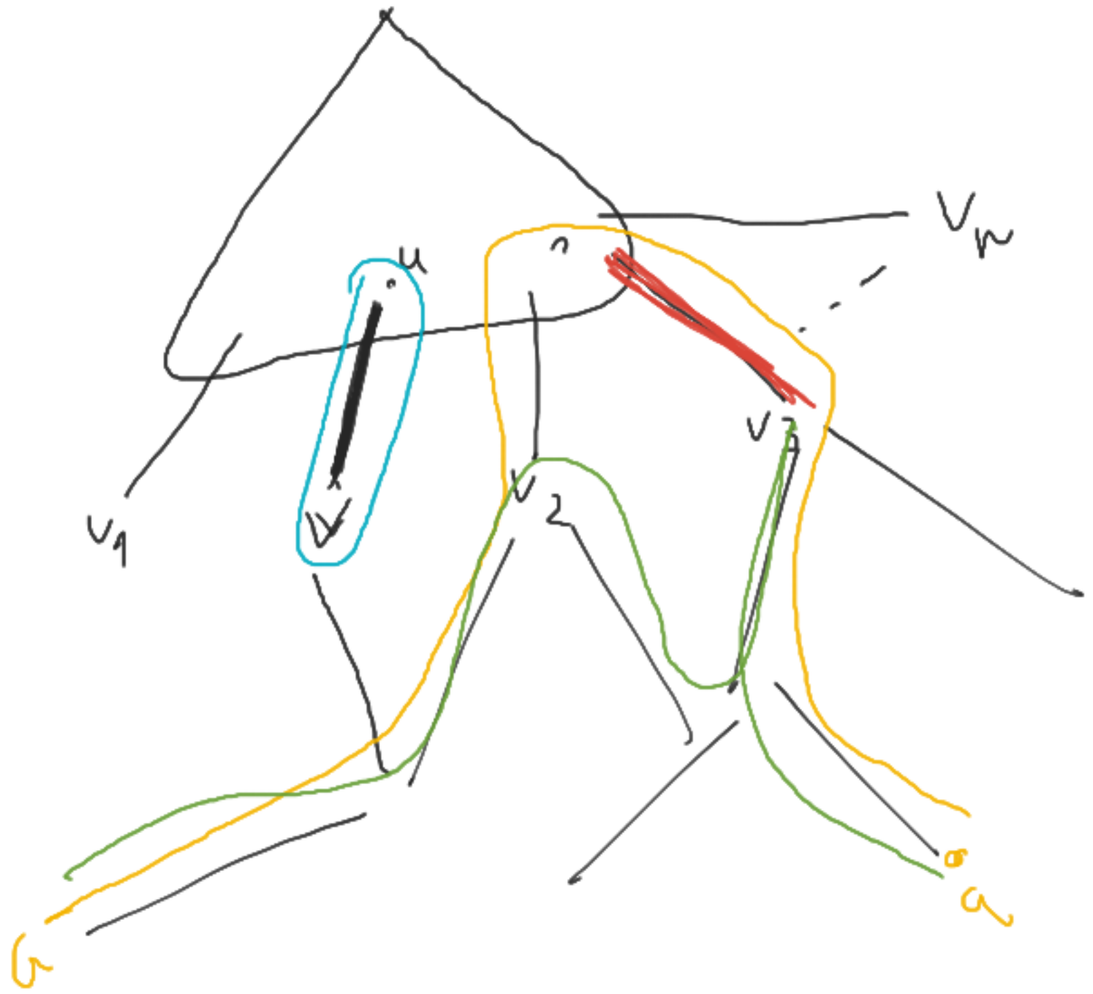




$O(E \log V)$



$e = \{u, v\}$ o minimalnej wadze, wychodząca z wierzchołka już policzonego przez alg. Prima


v_n
u
v_1
v
v_2
v_3
γ
σ