

Algorytmy i Struktury Danych

Wykład 10

Rodzina zbiorów rozłącznych

- makeSet
- find
- union

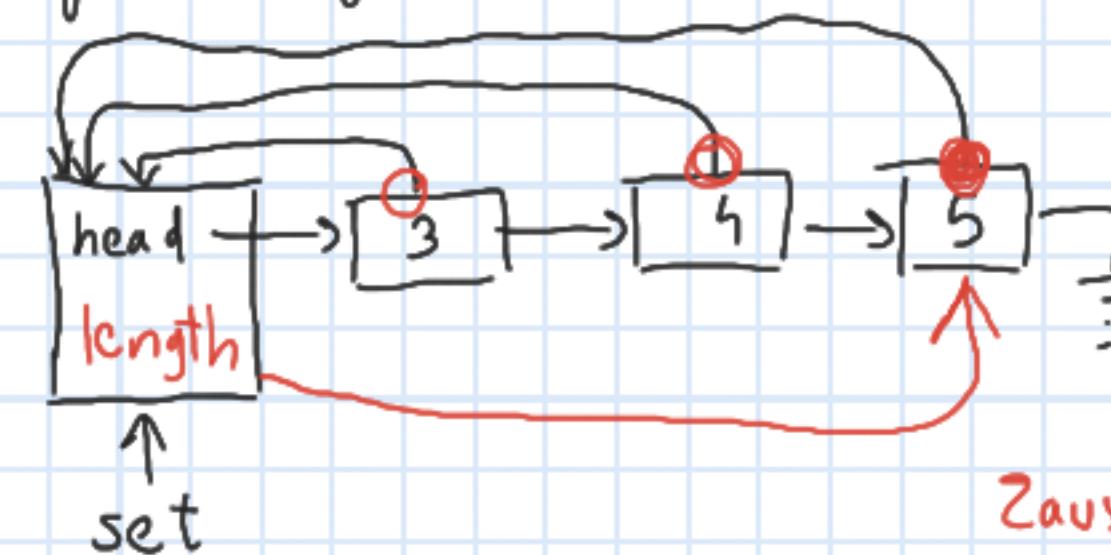
Punkłod



$$\text{find}(3) = \text{find}(4) = \text{find}(5)$$

$$\text{find}(3) \neq \text{find}(6)$$

Implementacja listowa



makeSet $\rightarrow O(1)$

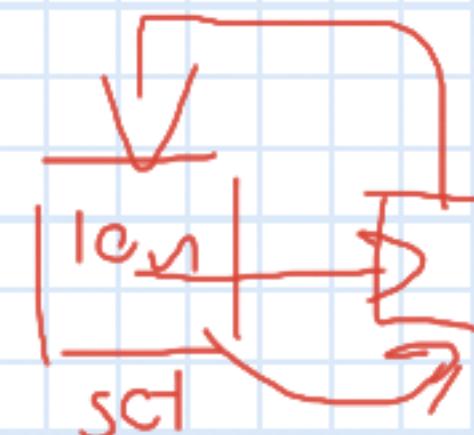
find $\rightarrow O(1)$

union $\rightarrow O(n)$

drugią listę

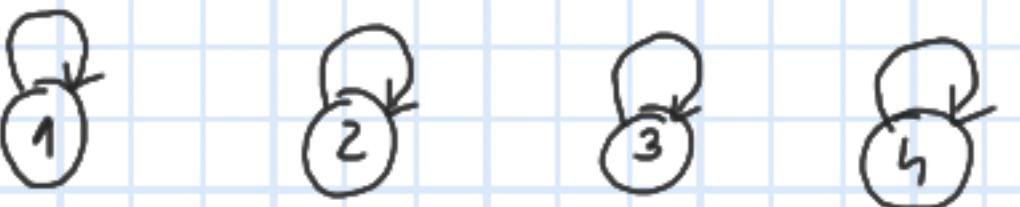
Zawsze doklejamy krotkę
listę do dłuższej

Wykonanie m operacji, z liczbą
n to makeSet ma złożoność
 $O(m + n \log n)$



Las zbiorów rozłącznych

a) seria operacji makeSet



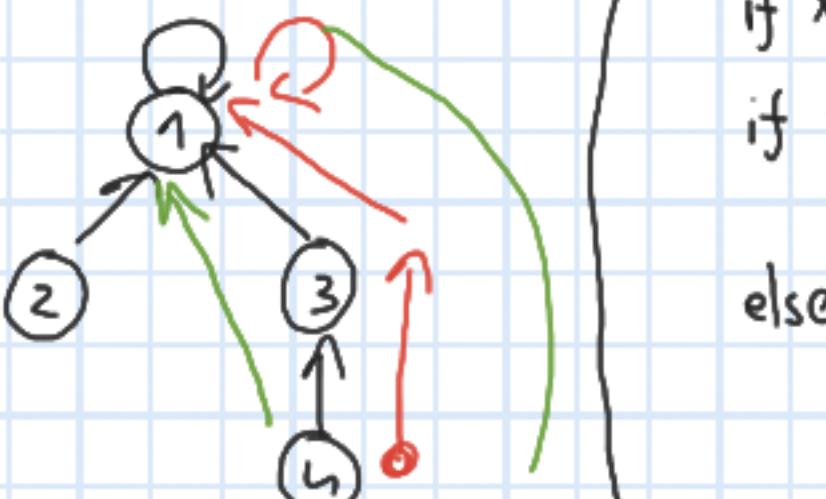
b) union (1, 2)



c) union (3, 1)



d) union (2, 3)



Implementacja

class Node:

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

def find(x):

```
if x != x.parent:
    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
```

Jesli wykonamy m operacji, z tego n to malest, to złożoność wynosi $O(m \log n)$

zakłada się, że operacje wg. ranku, bez kompresji ścieżki



Dniero o ranku

rank ma w najmniej

rank

ugiętow

Jesli stosujemy obie heurystyki to ciąg
m operacji, z których n to male set

ma złożoność

$$O(m \log^* n)$$

$$O(m\alpha(n))$$

↑
odwrotność funkcji Ackermann

$$x = 2^{2^2} = 2^{2^4} = 2^{16} = 65536$$

find - Union

Algorytmy na grafach wazonych

- minimalne dwoe rozpinajce
- najkrótsze ścieżki

Reprezentacja grafów wazonych:

$$G = (V, E)$$

$$w: E \rightarrow \mathbb{N} (\mathbb{Z} ? \mathbb{R} ?)$$

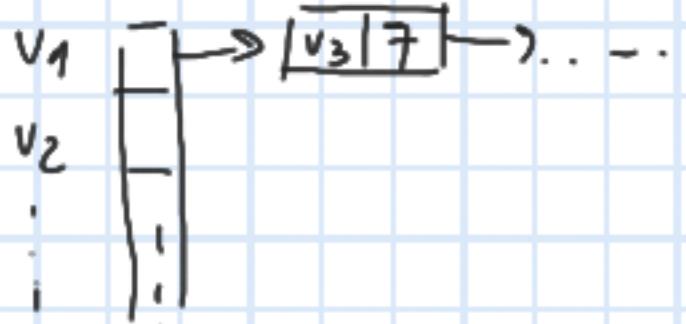
Reprezentacja macierza:

W - macierz wag

$w_{i,j}$ - waga krawędzi $\{v_i, v_j\}$

0 lub ∞ jeśli krawędzi nie ma

Reprezentacja listowa:

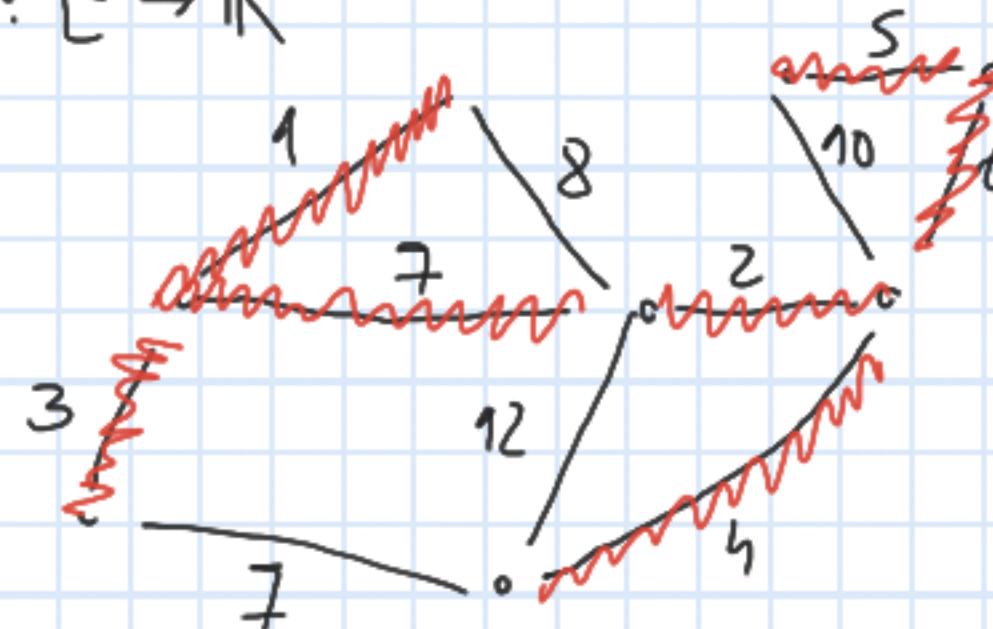


Minimalne dwoe rozpinajce

MST - min. spanning tree

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

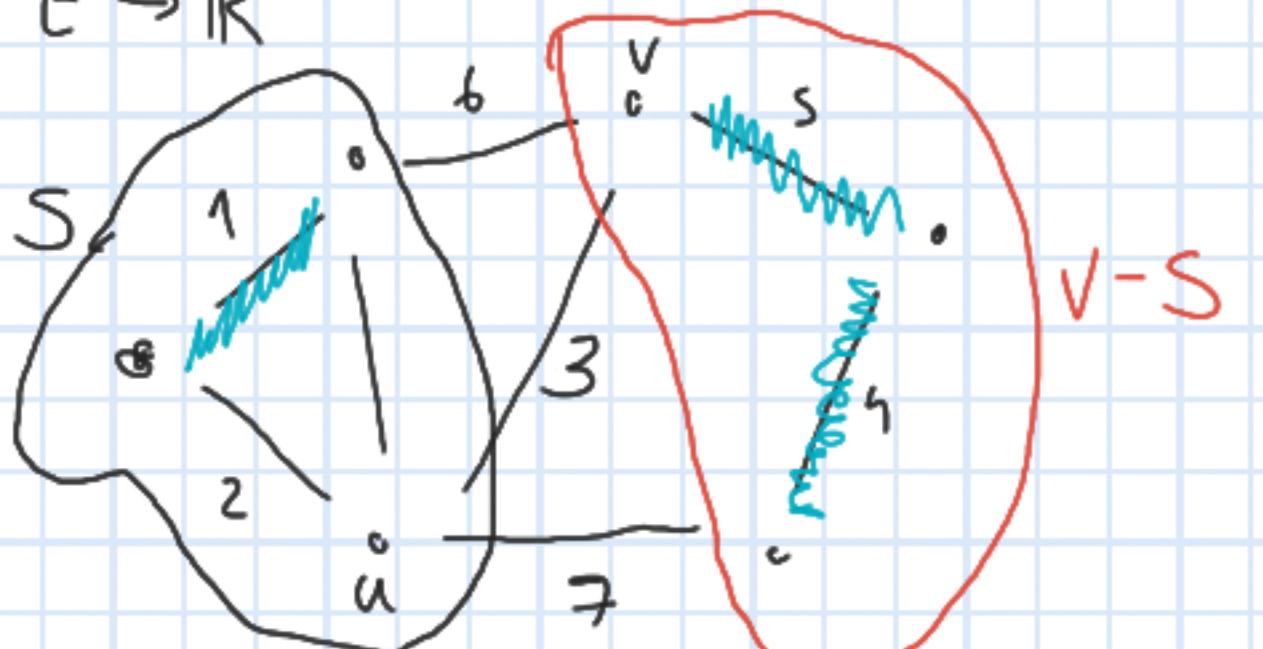
$$w: E \rightarrow \mathbb{R}$$



Należy znaleźć zbiór krawędzi, które łączą każdą parę wierzchołków i których suma wag jest minimalna

$G = (V, E)$ - spojny, nieskierowany graf

$$w: E \rightarrow \mathbb{R}$$



$A \subseteq E$ - podzbiór krawędzi pierwotego MST

$(S, V - S)$ - podzbiór V względem najmniejszej krawędzi z S

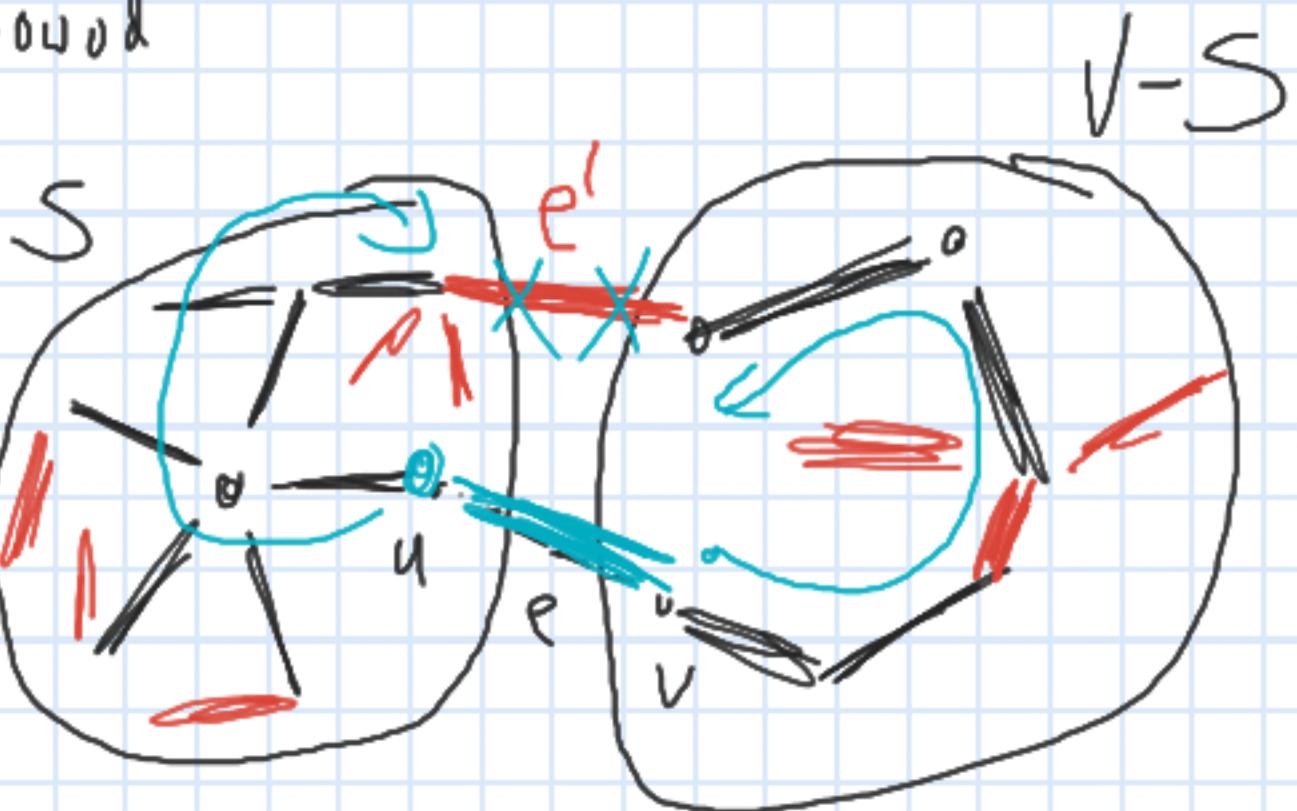
$$S \subseteq V$$

kiedy krawędź e nie ma winców z S
lub z $V - S$

$e = \{u, v\}$ - krawędź o minimalnej wadze
gdzie $u \in S$, $v \in V - S$

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

"dowód"



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



W MST możemy zastąpić e' przez e

Algorytmy i Struktury Danych

Układ 11

Algorytmy obliczające minimalne drene
rozpiąwanie

Algorytm Kruskala

$$G = (V, E)$$

$$w: E \rightarrow \mathbb{R}$$

1. Posortuj krawędzie po wagach

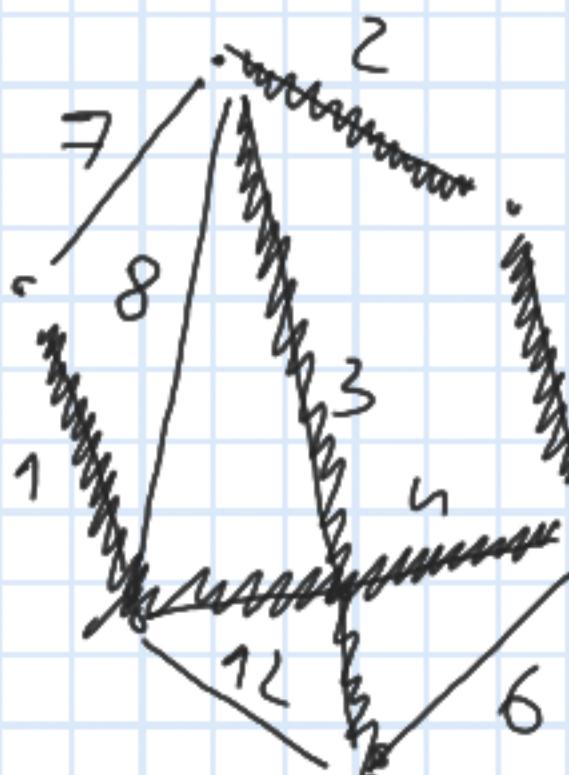
2. $A = \emptyset$

3. Przeglądaj krawędzie $e \in E$ w kolejności
niesmalejących wag:

jeśli $A \cup \{e\}$ nie zauważa cyklu
to $A := A \cup \{e\}$

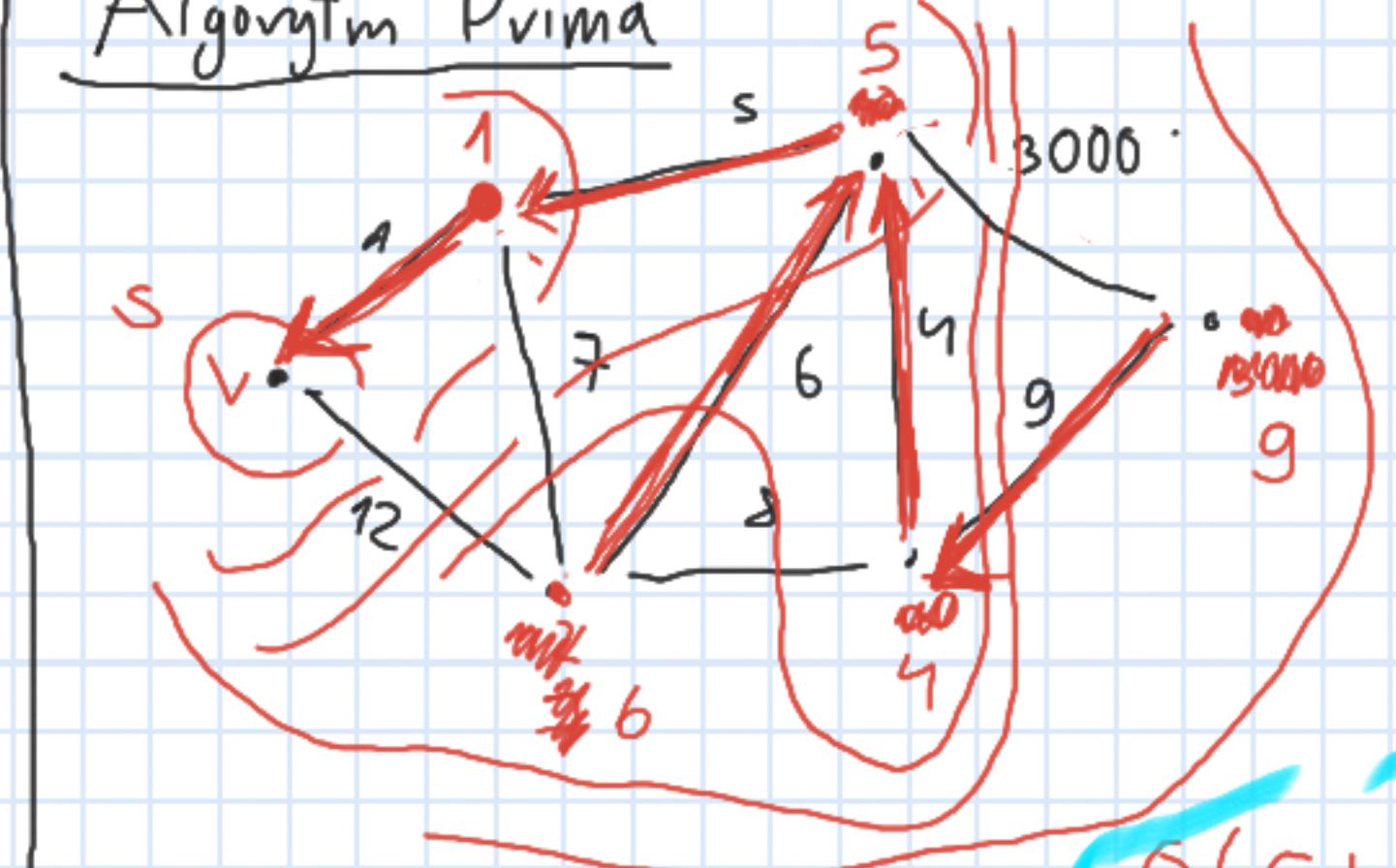
4. Zwróć A

$$O(E \log E)$$



stosujemy strukturę
find/union

Algorytm Prima



$$O(E \log V)$$

1. Umieść wszystkie wierzchołki w kolejce
priorytetowej z wagą ∞

2. zmień wagę v na 0

3. Później sq wierzchołki w kolejce priorytetowej
- ujmij wierzchołek t o minimalnej wadze
- dla każdej krawędzi $\{t, u\}$, jeśli waga
 $w(\{t, u\})$ jest mniejsza niż waga u w kolejce
to zmień wagę u na $w(\{t, u\})$ i ustaw

v.parent = t

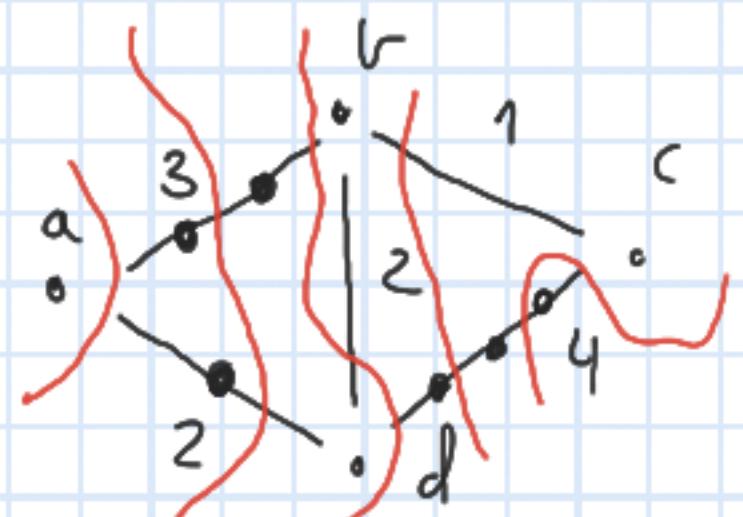
Znajdowanie najkrótszych ścieżek w grafach ważonych

Uwianity

- 1 - 1
- 1 - wszyscy
- wszyscy - wszyscy

} trudno wykonać
} standardowe wersje

Podając elementarne



Algorytm Dijkstry

$$G = (V, E), w: E \rightarrow \mathbb{R}_+ \cup \{0\}$$

Startujemy z $s \in V$

1. Umieść wszystkie wierzchołki w kolejce priorytetowej z oszacowanym odł. od s równym ∞

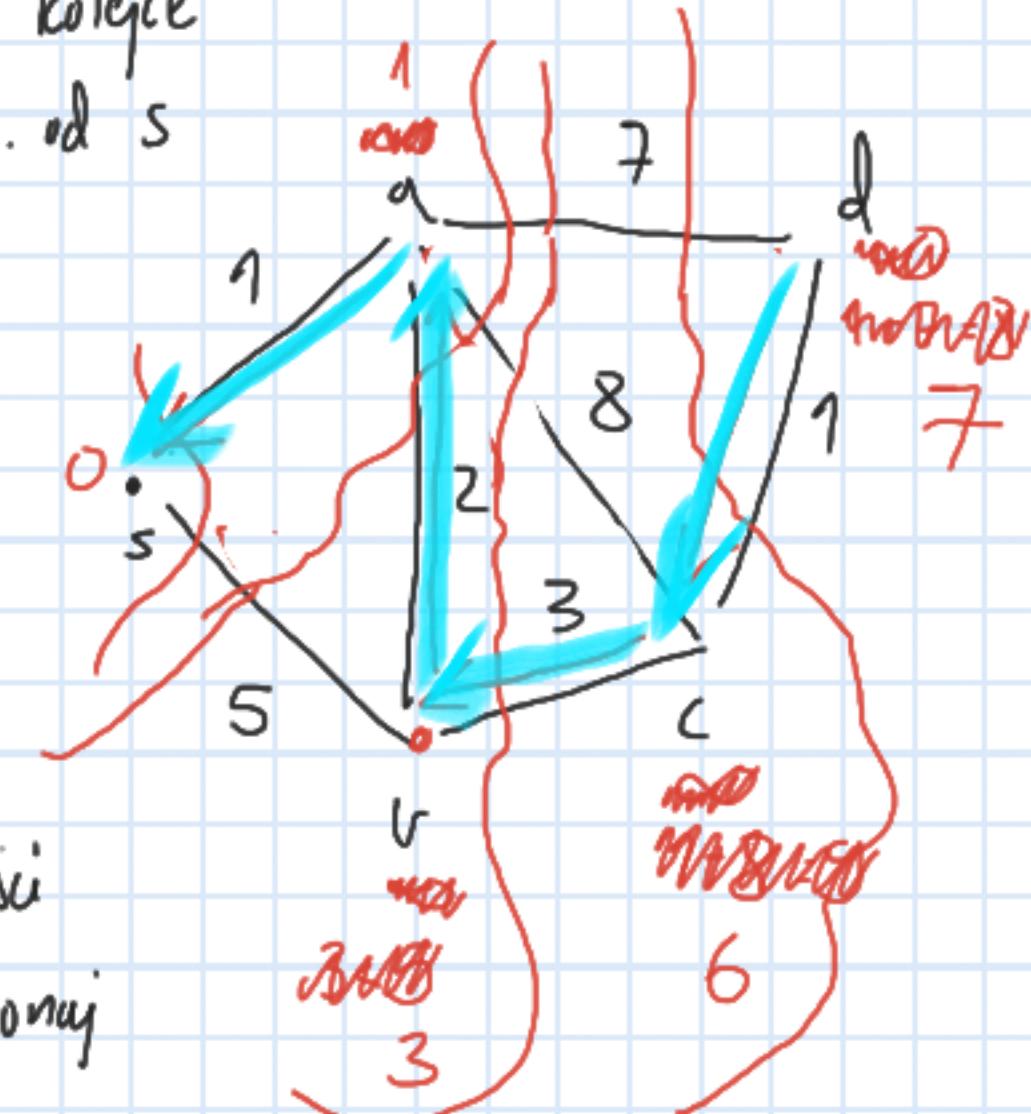
2. Zmień odległość s na 0

3. Polci się wierzchołki w kolejce:

- ujmij z kolejki wierzchołek $u \in V$ o minimalnym oszacowaniu odległości
- dla każdej krawędzi $\{u, v\}$ wykonaj relaksację: $\text{def relax}(u, v):$

```

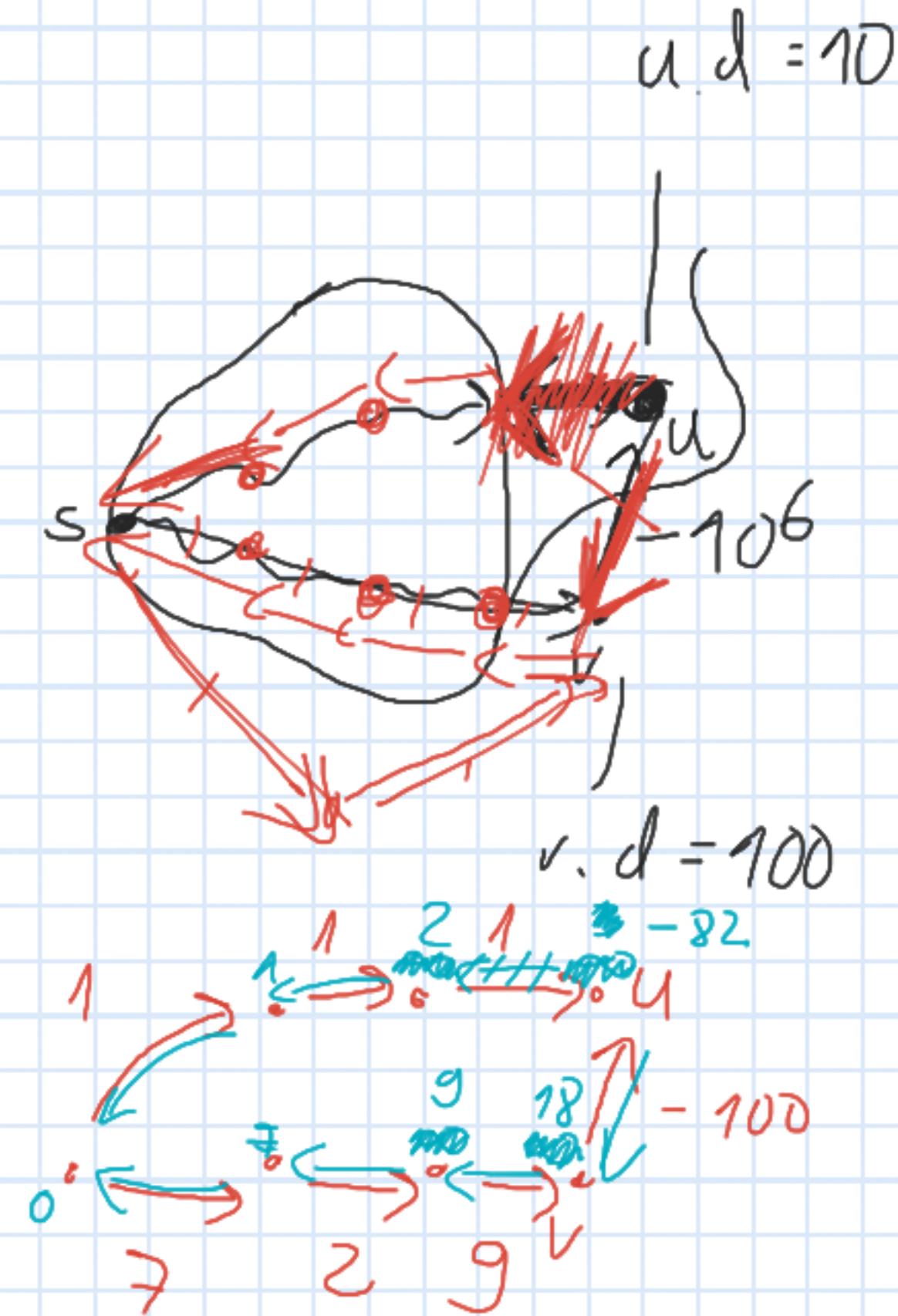
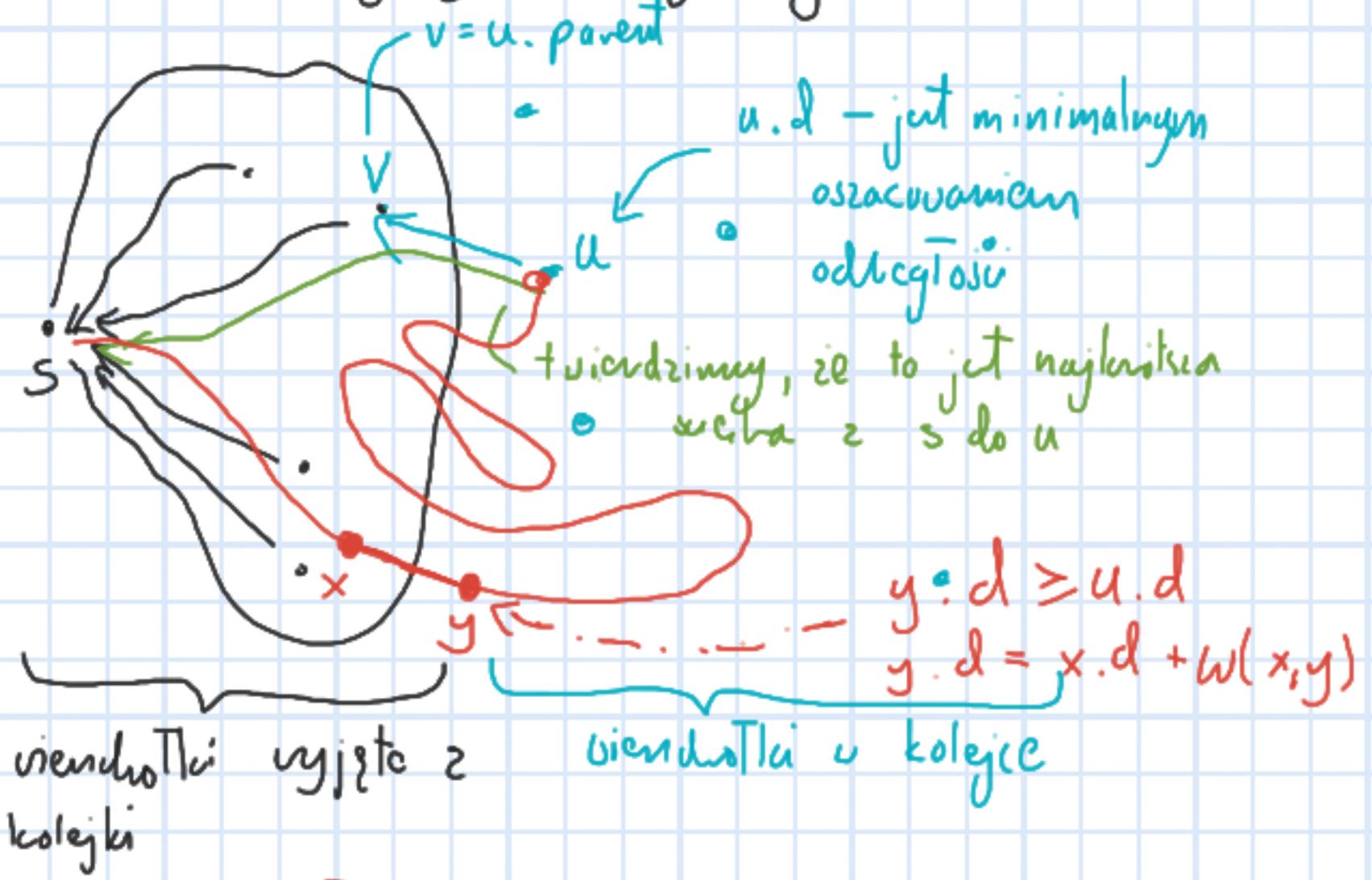
if  $v.d > u.d + w(\{u, v\})$ :
     $v.d = u.d + w(\{u, v\})$ 
     $v.parent = u$ 
  
```



Algorytmy i Struktury Danych

Wykład 12

Poprawność algorytmu Dijkstry



Algorytm Bellmana - Forda

Oblinanie najkrótszych ścieżek w sytuacji gdy
wag. mogą być ujemne

$$G = (V, E), \quad v: E \rightarrow \mathbb{R}$$

① Inicjalizacja (s - punkt startowy)

for $v \in V$:

$$v.d = \infty$$

$v.parent = \text{None}$

$$s.d = 0$$

② Relaksacja

for i in range ($|V| - 1$):

[for $(u, v) \in E$:

 Relax (u, v)

③ Weryfikacja

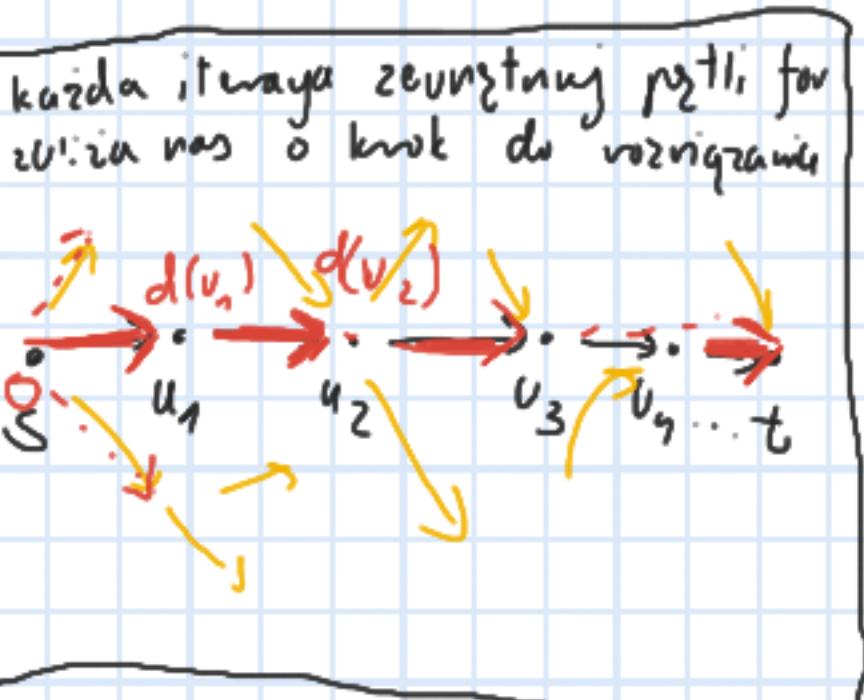
wy dla każdej $(u, v) \in E$:

$$v.d \leq u.d + w(u, v) \leftarrow \text{mamy ujemny cykl}$$

uwaga na cykle o
ujemnej wadze



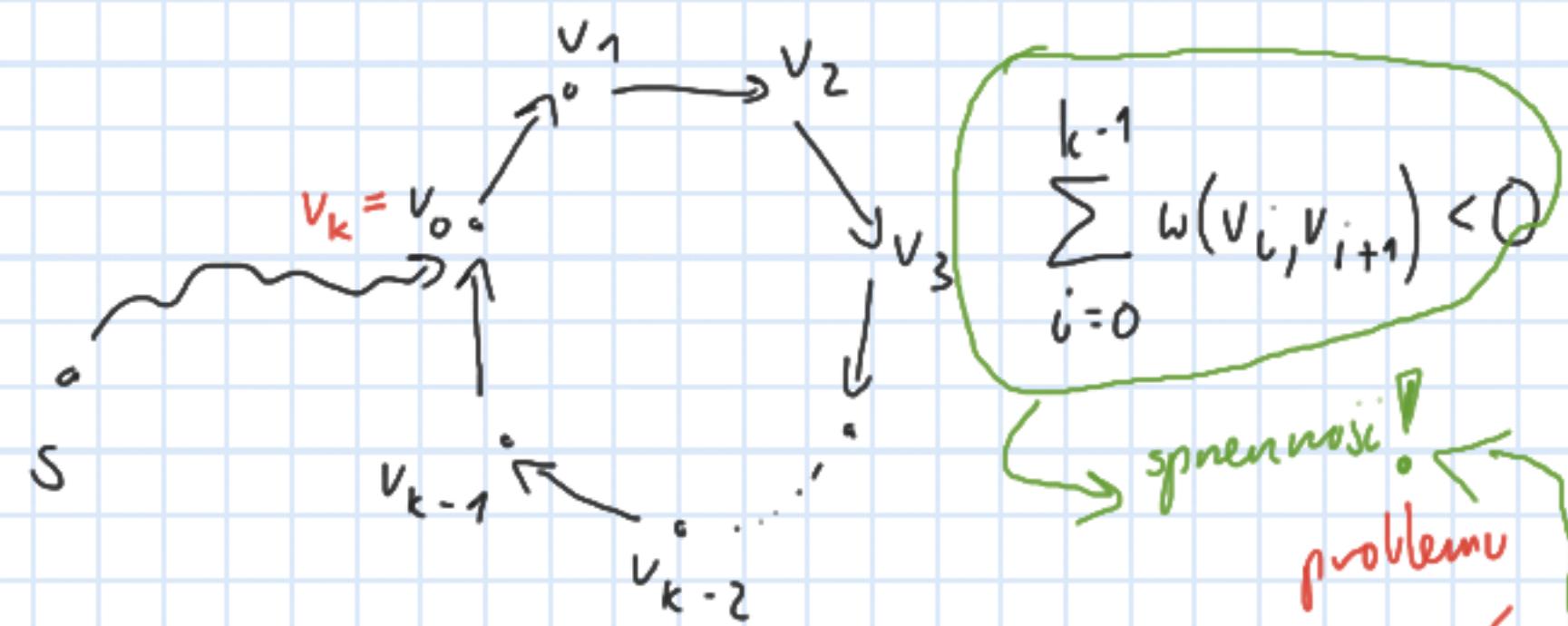
$\mathcal{O}(V \cdot E)$



jako nie, to

mamy ujemny cykl

Czywo weryfikacja wykrywa ujemne cykle?



Zauważmy, że weryfikacja nie wykryła

wątpliwości dla każdego v_i :

$$v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$$

Najczęściej zsumowali te nierówności:

$$\sum_{i=1}^k v_i.d \leq \sum_{i=1}^k (v_{i-1}.d + w(v_{i-1}, v_i))$$

$$0 \leq \sum_{i=0}^{k-1} w(v_i, v_{i+1})$$

Najkrótsze ścieżki między każdą parą wierzchołków

- $|V|$ wywołani algorytmu Dijkstry $O(V \cdot \log V)$, $O(V^3)$
- $|V|$ wywołani alg. Bellmana - Forda $O(V^2 E)$

Konwencja

Stosujemy reprezentację macierzyową

macierza najkrótszych odległości:

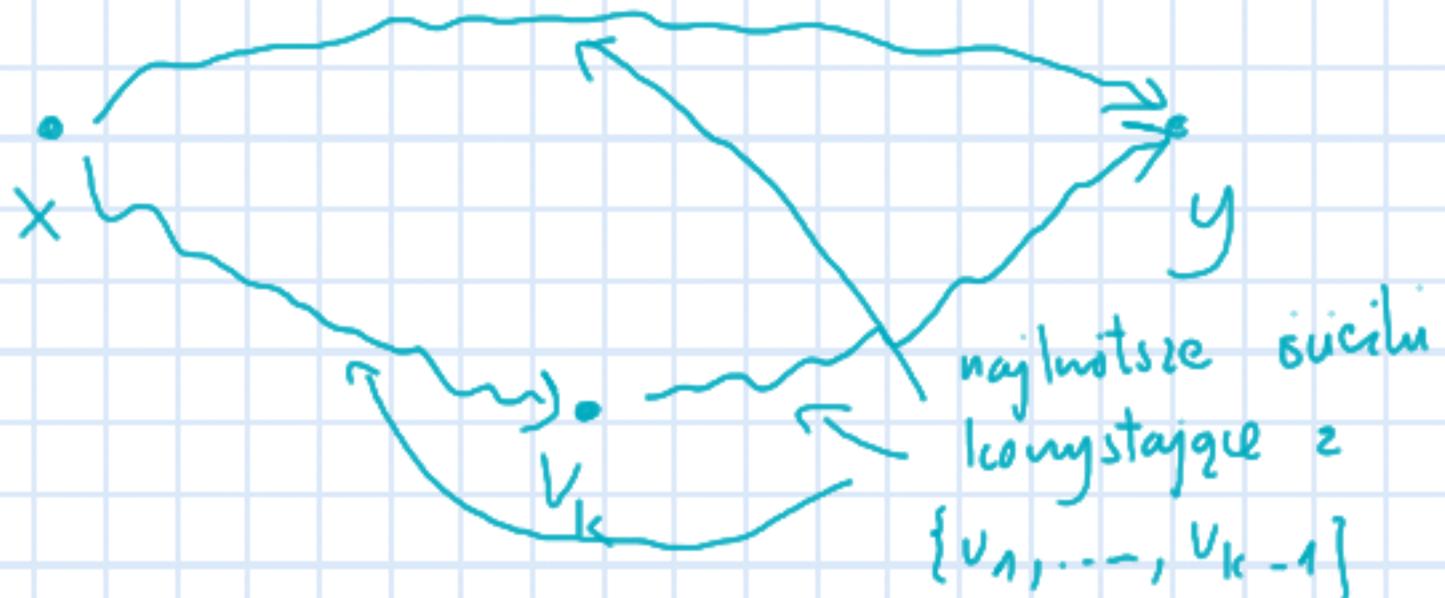
$D[u][v]$ - dt. najkrótszej ścieżki z u do v

// $P[u][v]$ - poprzednik v na najkrótszej ścieżce z u do v

Algorytm Floyda - Warshalla

Idea: znamy najkrótsze ścieżki między każdą parą wierzchołków, tzn. że te ścieżki kompletują się w wierzchołkach $\{v_1, \dots, v_{k-1}\}$

Na tej podstawie możemy rozszerzyć zbiór wierzchołków wew. do $\{v_1, \dots, v_k\}$



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

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

W - mauerz uag

$S^{(t)}$ - mauerz d \bar{t} . najkrótszych ścieżek
wierzchołkowych $\{v_1, \dots, v_t\}$ jako uel.

$S^{(0)} = W$

$O(V^3)$

Algorytm

for t in range $(1, n + 1)$:

 for $u \in V$:

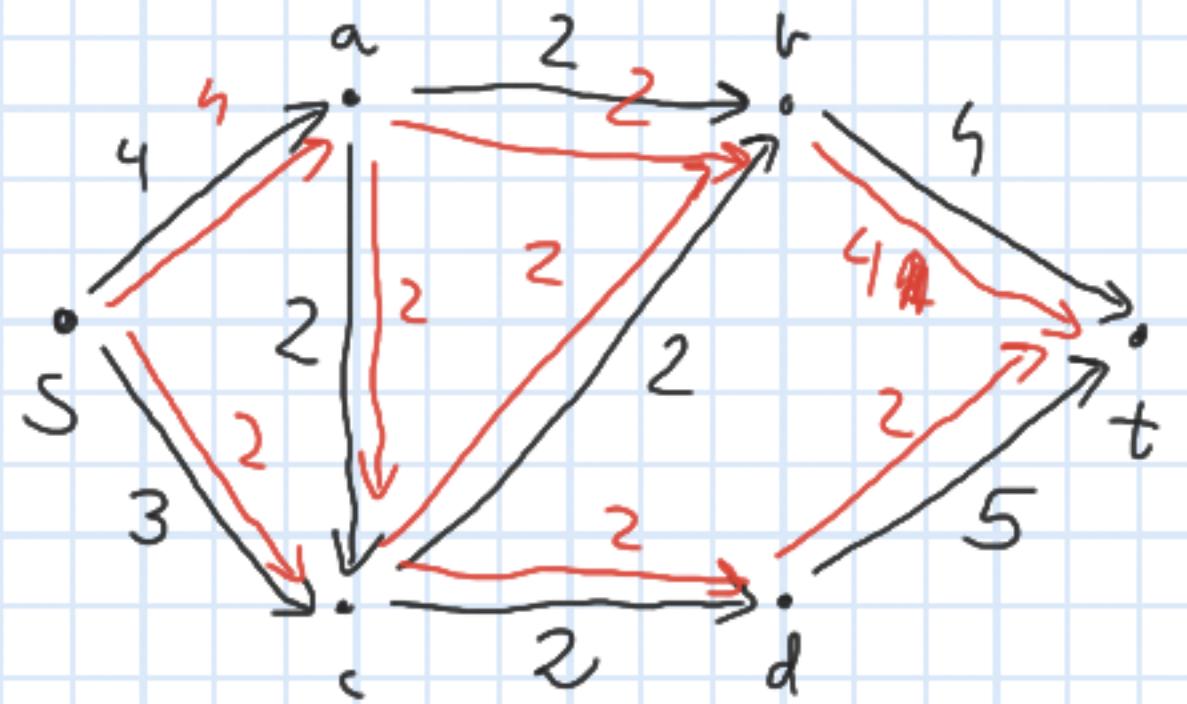
 for $w \in V$:

$$S^{(t)}[u][w] = \min \left(S^{(t-1)}[u][w], S^{(t-1)}[u][v_t] + S^{(t-1)}[v_t][w] \right)$$

return $S^{(n)}$

można stosować
te same mauerz

Problem maksymalnego przepływu



Uczenie

$$G = (V, E) \text{ - graf skierowany}$$

$$(\forall u, v \in V) [\neg ((u, v) \in E \wedge (v, u) \in E)]$$

↳ nic nie ma krawędzi w obie strony

$c: V \times V \rightarrow \mathbb{N}$ ← funkcja pojemności krawędzi

↳ jeśli $(u, v) \notin E$ to $c(u, v) = 0$

$$c(u, u) = 0$$

Zadanie

Znaleźć "przepływ" o maksymalnej wartości

Przepływ f to funkcja:

$$f: V \times V \rightarrow \mathbb{N}$$

taka, i.e.:

$$(\forall u, v \in V) [f(u, v) \leq c(u, v)]$$

$$(\forall v \in V - \{s, t\}) [\sum_{u \in V} f(u, v) = \sum_{w \in V} f(v, w)]$$

Wartość przepływu:

$$|f| = \sum_{v \in V} f(s, v)$$

zakładaemy, i.e. talcoż krawędzi nie ma, więc to jest 0

$$- \sum_{v \in V} f(v, s)$$

Siec residualna

$G = (V, E)$ }
 $s, t \in V$ sicc pmeptyvoua

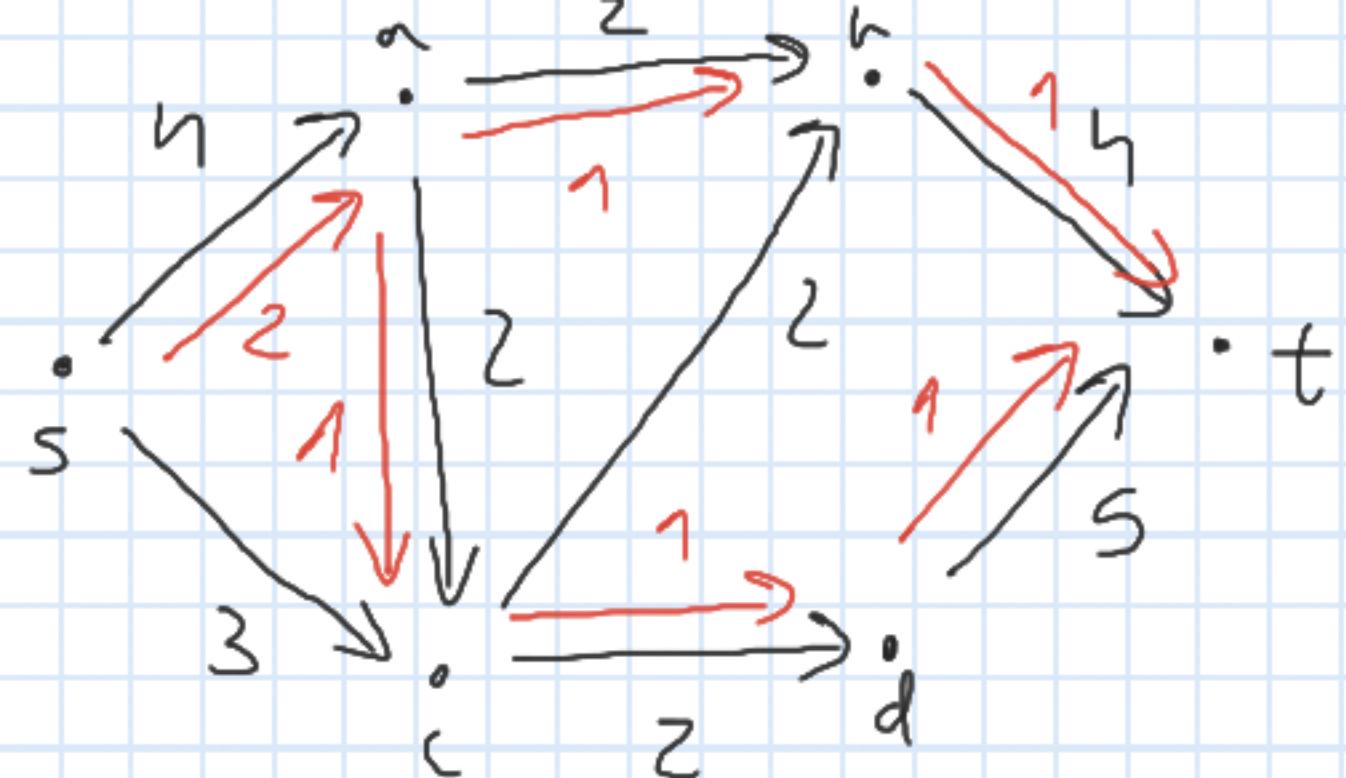
$$c: V \times V \rightarrow \mathbb{N}$$

$$f: V \times V \rightarrow \mathbb{N} - \text{pmeplyw}$$

Definiujemy siec residualna G_f, c_f

pmez funkjs c_f :

$$c_f(u, v) = \begin{cases} c(u, v) - f(u, v), & (u, v) \in E \\ f(v, u), & (v, u) \in E \\ 0, & \text{up.p.} \end{cases}$$



*sieka
mowikscayga*



Algorytmy i Struktury Danych

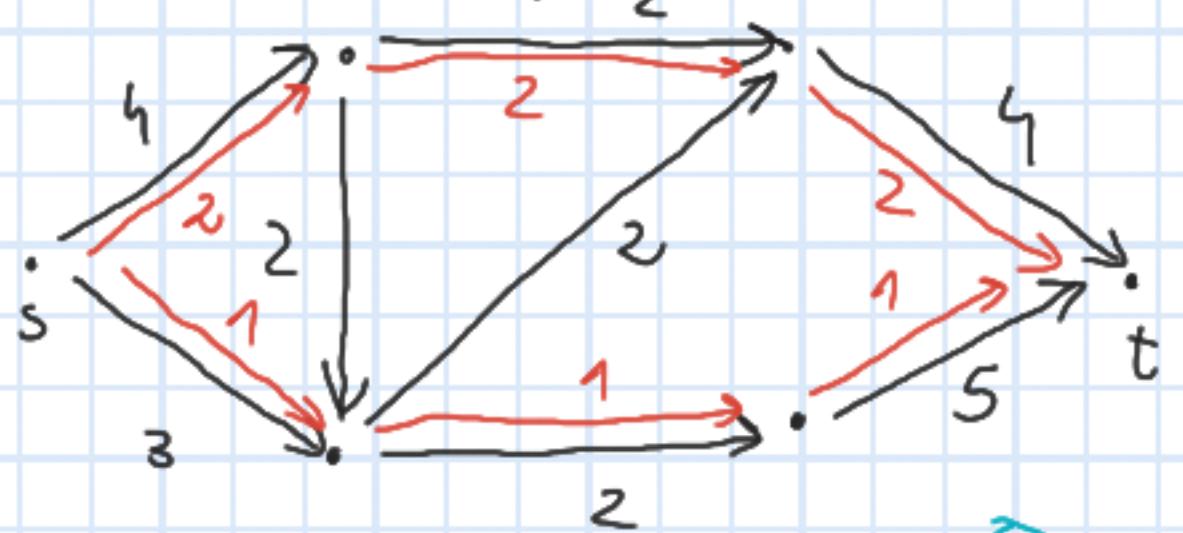
Wykład 13

Maksymalny przepływ (dolcanicenie)

Graf $G = (V, E)$ - skierowany

$$c: E \rightarrow \mathbb{N}$$

Szukamy przepływu $f: E \rightarrow \mathbb{N}$ z s
zródła s do ujścia t



Sieć residualna:



Ścieżka powiększająca - ścieżka z s do t u sieci residualnej

Metoda Forda-Fulkersona

1. Znajdź od pustego przepływu f

2. Poki istnieje jakis ścieżka powiększająca
powiększ f wzdłuż tej ścieżki

Pnieważ w sieci

$$G = (V, E) \quad s, t \in V$$

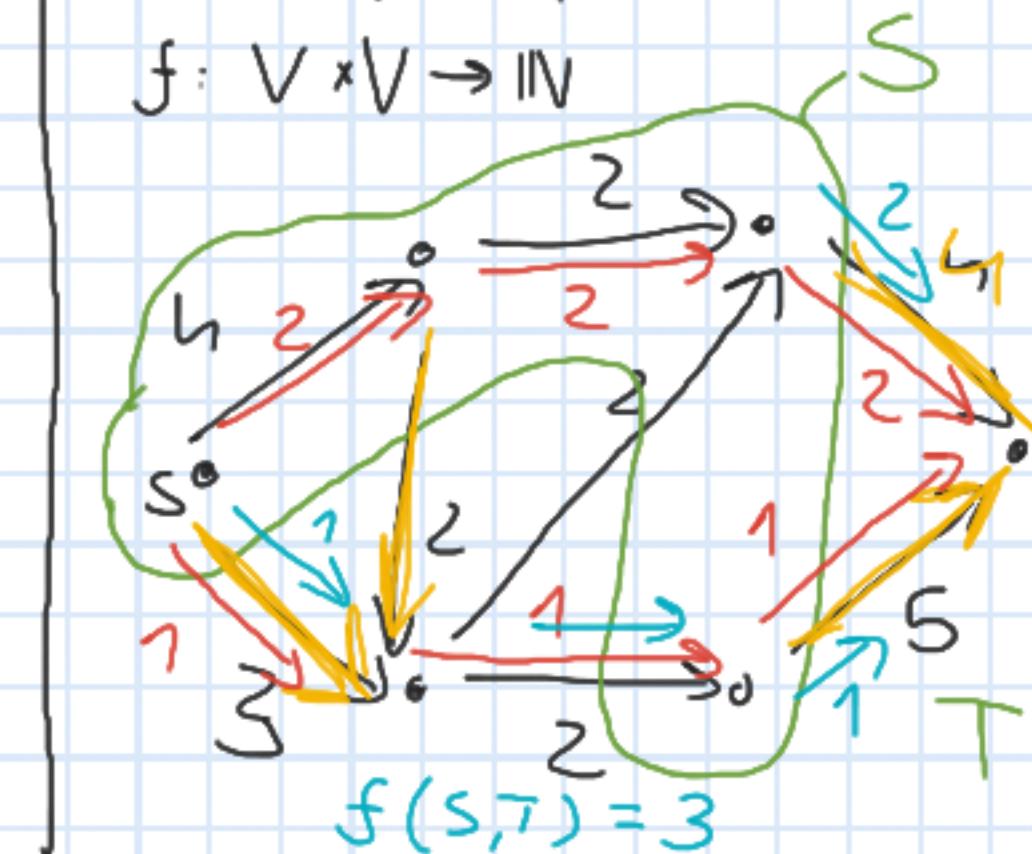
$$c: V \times V \rightarrow \mathbb{N}$$

$$f: V \times V \rightarrow \mathbb{N}$$

Pnieważ sieci to podzbiór V na

zbiory wzajemne zbiory, S, T tzn.

$$S \subseteq V, T \subseteq V - S$$



Pnepustowosc przepływu:

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

Przepływ netto:

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

Tw

Niech f będzie przepływem w sieci G
 ze źródłem s , ujściem t , a (S, T) niech
 będzie paryskiem. Wówczas $f(S, T) = |f|$

Dowód

$$|f| = \left[\sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) \right] + \sum_{u \in S - \{s\}} \sum_{v \in V} f(u, v) - \sum_{u \in S - \{s\}} \sum_{v \in V} f(v, u)$$



$$= \sum_{u \in S} \left(\sum_{v \in V} f(u, v) \right) - \sum_{u \in S} \left(\sum_{v \in V} f(v, u) \right)$$

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

$$+ \sum_{u \in S} \sum_{v \in S} f(u, v) - \sum_{u \in S} \sum_{v \in S} f(v, u) = 0$$

$$= f(S, T)$$

Iw Wartosci przepływu jest
 niegdyś niż przepustowość
 dowolnego parysku

Dowód

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

$$\leq \sum_{u \in S} \sum_{v \in T} f(u, v)$$

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

tw (max-flow/min-cut theorem)

Niech $G = (V, E)$, $s, t \in V$, $c: V \times V \rightarrow \mathbb{N}$

opisują się przepływy i f będzie przepływem

w tej sieci. Następujące warunki są równoważne:

1) f jest maksymalnym przepływem w G

2) G_f, c_f nie zawiera ścieżek powiększających

3) dla pewnego podziału S, T zachodzi
 $|f| = c(S, T)$

Dowód

3) \Rightarrow 1

1) \Rightarrow 2) – oczywiste!

2) \Rightarrow 3)

$S = \{v \in V \mid \begin{array}{l} \text{istnieje ścieżka z } s \text{ do } v \\ \text{w sieci residualnej } G_f, c_f \end{array}\}$

$T = V - S$

$s \in S, t \in T$

$$f(S, T) = \sum_{u \in S} \sum_{v \in T} \left(f(u, v) - \frac{f(v, u)}{c(u, v)} \right) = c(S, T)$$

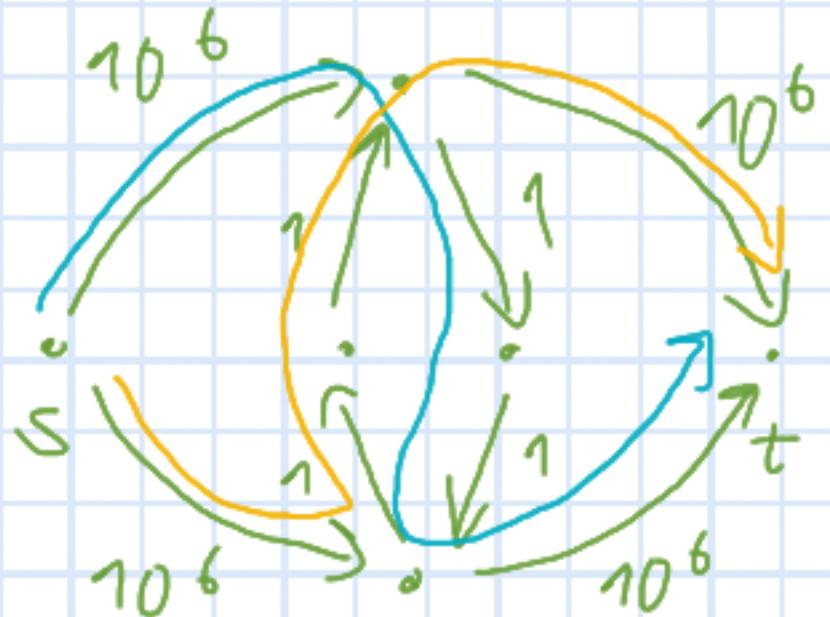
$$c_f(u, v) = 0 \quad \begin{cases} \text{jedźli } (u, v) \in E \text{ to } c_f(u, v) = c(u, v) - f(u, v) \\ \text{w przeciwnym wypadku } f(u, v) = c(u, v) \end{cases}$$

$$\begin{cases} \text{jedźli } (v, u) \in E \text{ to } c_f(u, v) = f(v, u) \\ \text{w przeciwnym wypadku } f(v, u) = 0 \end{cases}$$

Złożoność metody Forda - Fulkersona:

$$O((V+E)|f^+|)$$

vartości pmeptyu



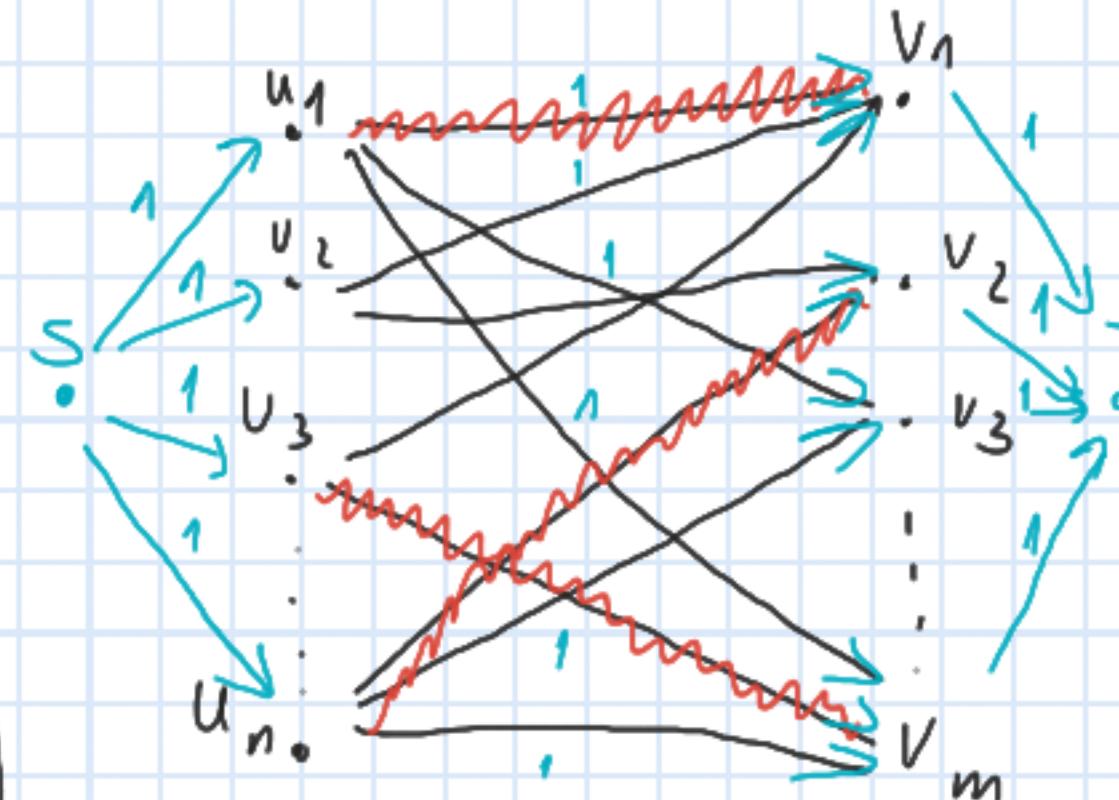
Metoda Edmonda - Karpia

→ ulepszaj scialu prorasczajce BFSem

$$O(VE^2)$$

Problem maksymalnego skojarzenia w grafie dwudzielnym

$G = (U, V; E)$ - graf dwudzielnug



Skojarzenie to zbiór krawędzi takie, że żadne dwie nie mają wspólnego wierzchołka

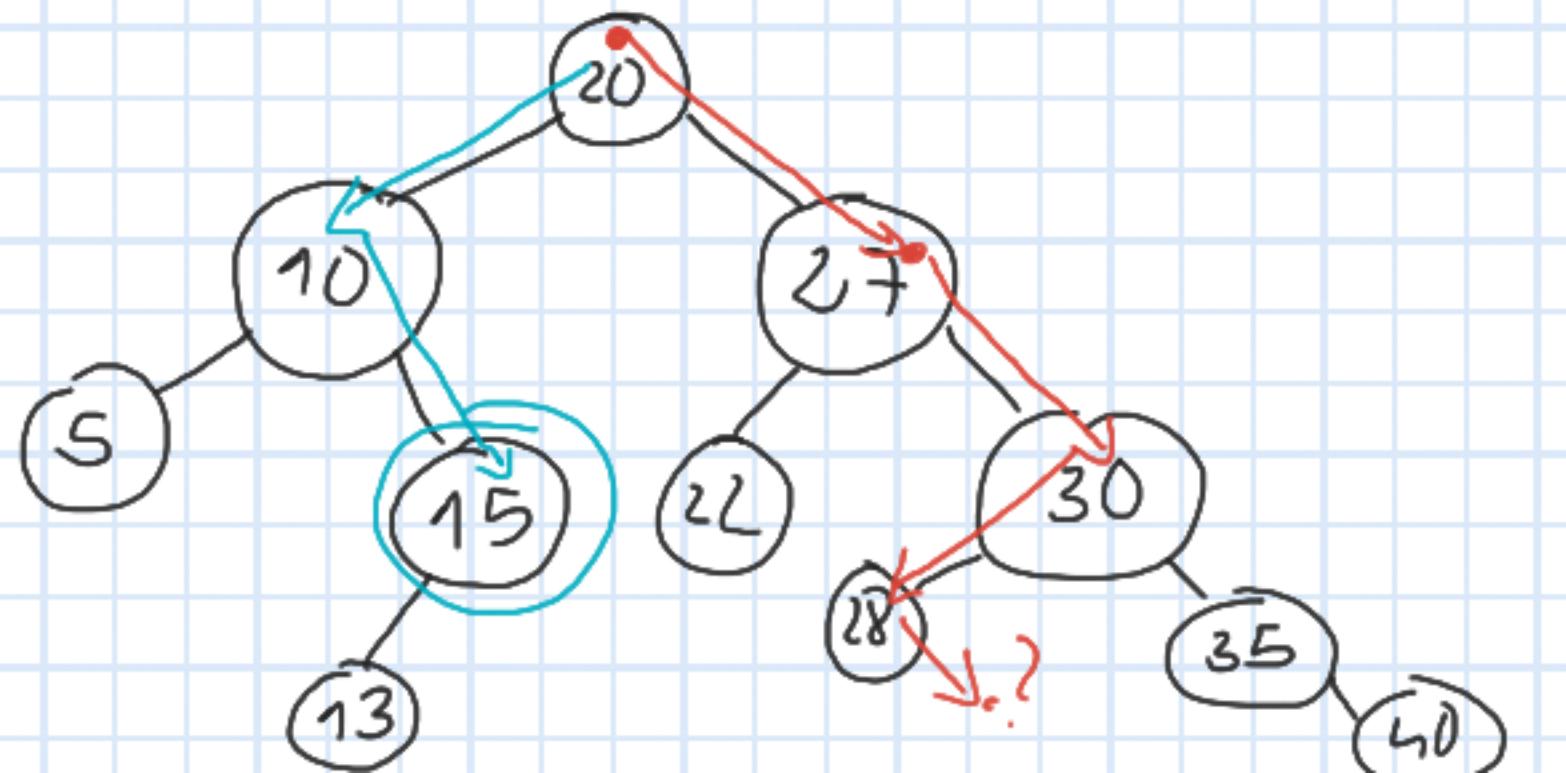
Tablica asocjacyjne (słowniki, mapy)

A["słów"] = 17 wartość
key

Operacje

- insert, remove, find
- min, max
- prev, next

Dzewo BST (Binary search tree)



```
def find(root, key):  
    while root != None:  
        if root.key == key:  
            return root  
        elif key < root.key:  
            root = root.left  
        else:  
            root = root.right  
  
    return None
```

class BST Node:

```
def __init__(self, key)  
    self.key = key  
    self.left = None  
    self.right = None  
    self.parent = None
```

