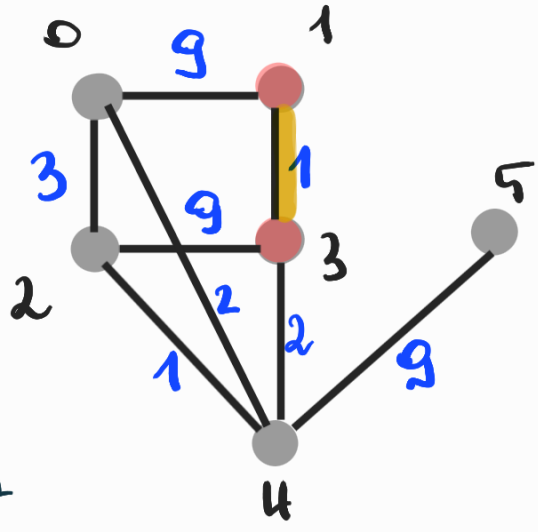
ALGORYTM KRUSKALA (ALGORYTM ZACHŁANNY)

- wyznacza minimalne drzewo rozpinające (MST)

IDEA:

Tworzymy pusty zbiór krawędzi T oraz listę L wszystkich krawędzi grafu uporządkowaną według rosnących wag.

Dopóki w T nie mamy krawędzi łączących wszystkie wierzchołki grafu, wybieramy z listy L krawędź, i jeśli nie tworzy ona cyklu z krawędziami już obecnymi w T, to dodajemy ją do zbioru T.



Suma wag = 1

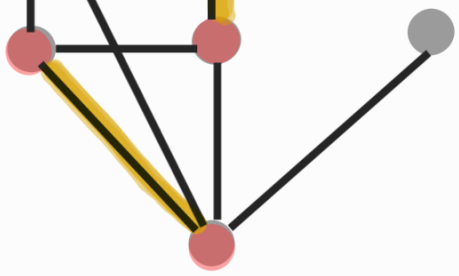
T L

[ ]

L:
1-3 : 1
2-4 : 1
0-4 : 2
3-4 : 2
0-2 : 3
0-1 : 9
4-5 : 9
2-3 : 9

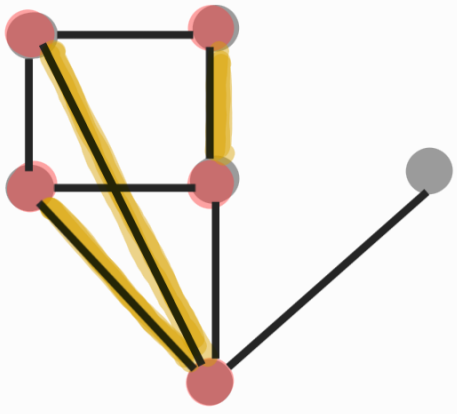
[1-3 : 1]

L:
2-4 : 1
0-4 : 2



Suma wag = 2

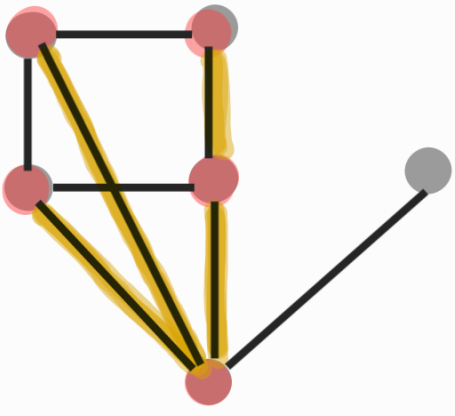
0-2 : 3
0-1 : 9
4-5 : 8
2-3 : 9



Suma wag = 4

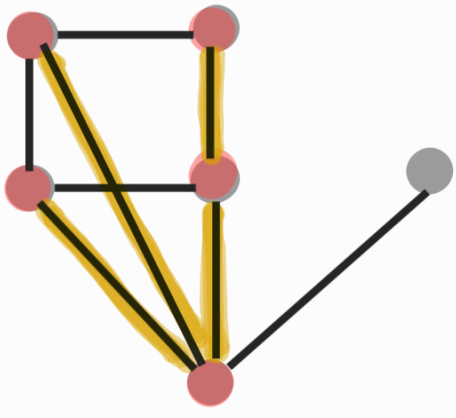
1-3 : 1
2-4 : 1

0-4 : 2
3-4 : 2
0-2 : 3
0-1 : 9
4-5 : 8
2-3 : 9



1-3 : 1
2-4 : 1
0-4 : 2

3-4 : 2
0-2 : 3
0-1 : 9
4-5 : 8
2-3 : 9

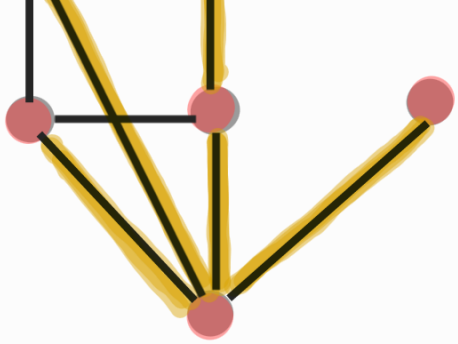


1-3 : 1
2-4 : 1
0-4 : 2
3-4 : 2

0-2 : 3
0-1 : 9
4-5 : 8
2-3 : 9

JEŚLI DODAMY STWORZY CYKL!

1-3 : 1



$$\begin{bmatrix} 2\text{-}4 : 1 \\ 0\text{-}4 : 2 \\ 3\text{-}4 : 2 \\ 4\text{-}5 : 9 \end{bmatrix}$$

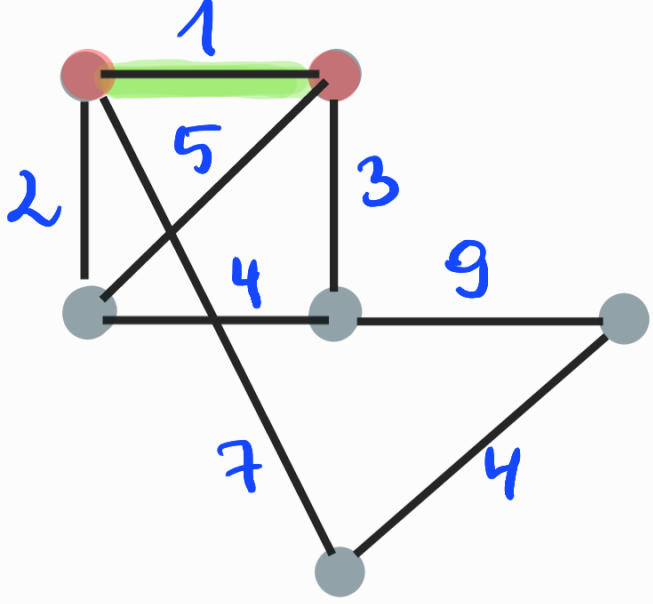
$$\begin{bmatrix} & \end{bmatrix}$$

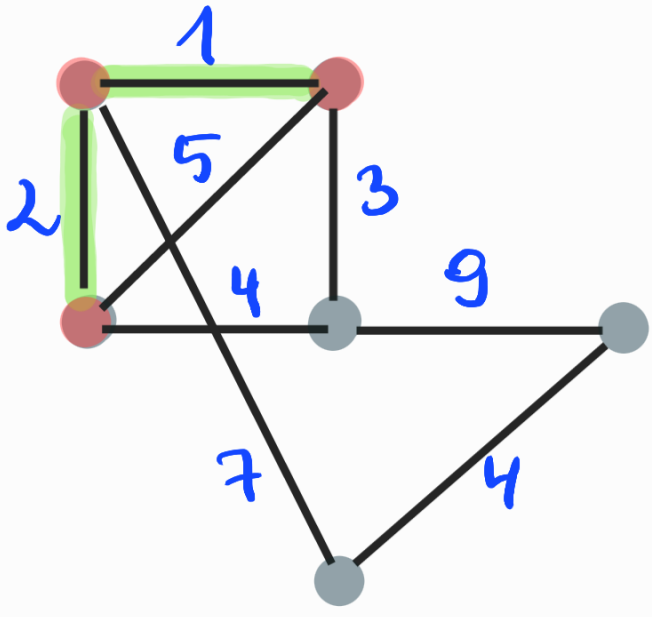
ALGORYTM PRIMA (ALGORYTM ZACHŁANNY)

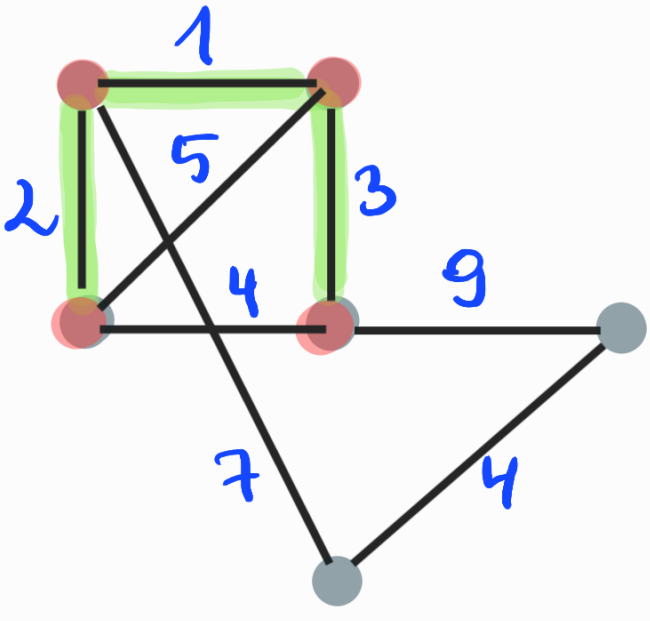
- algorytm wyznaczania minimalnego drzewa rozpinającego (MST)

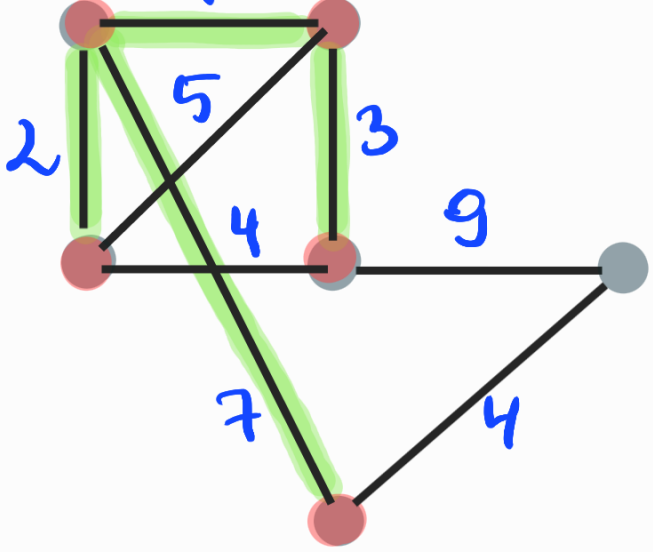
IDEA:

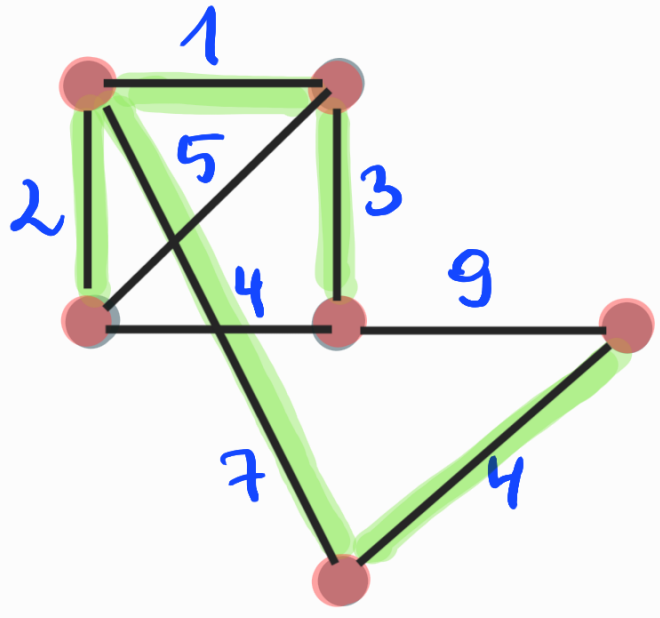
Wybierany w grafie dowolny wierzchołek startowy.
Dopóki drzewo nie pokrywa całego grafu znajdujemy krawędź o najmniejszym koszcie spośród wszystkich krawędzi od wybranych już wierzchołków do wierzchołków jeszcze niewybranych. Znalezioną krawędź dodajemy do drzewa rozpinającego.


1
5
2
3
4
9
7
4


1
5
2
3
4
9
7
4


1
5
2
3
4
9
7
4





→ nie może zawierać ujemnych wag

ALGORYTM DIJKSTRY

- rozwiązuje problem znalezienia ścieżki o najniższym koszcie z danego wierzchołka do każdego innego.

IDEA:

Tworzymy dwa zbiory wierzchołków Q i S. Początkowo zbiór Q zawiera wszystkie wierzchołki grafu, a S jest pusty

Dla wszystkich wierzchołków u grafu za wyjątkiem startowego v wstawiamy koszt dojście do u na nieskończoność, koszt dojścia d(v) zerujemy.
Dodatkowo wstawiamy poprzednik p(u) każdego wierzchołka u grafu na niezdefiniowany. Poprzedniki będą w kierunku odwrotnym wyznaczały najkrótsze ścieżki od wierzchołków u do wierzchołka startowego v. Teraz w pętli dopóki zbiór Q zawiera wierzchołki wykonujemy:

1) Wybieramy ze zboru Q_k wierzchołek u o najmniejszym koszcie dojścia d(u).

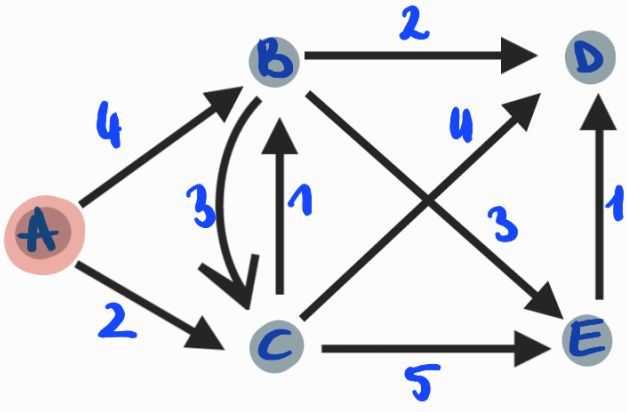
2) Wybrany wierzchołek u usuwamy ze zbioru Q i dodajemy do zbioru S.

3) Dla każdego sąsiada w wierzchołku u, który jest wciąż w zbiorze Q_k, sprawdzamy czy $d(w) > d(u) +$ + waga krawędzi u-w.
Jeśli tak to wyznaczamy nowy koszt dojścia do wierzchołka

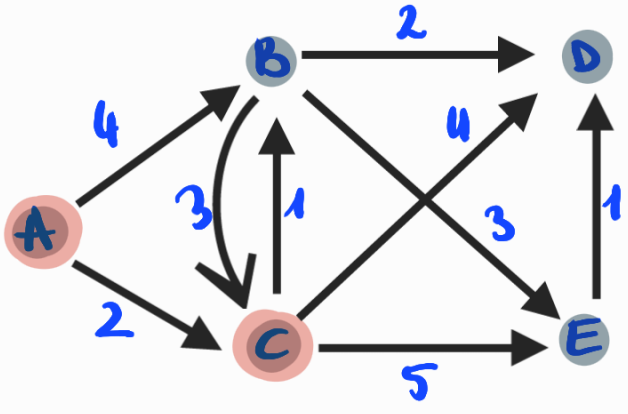
w jako:

$d(w) \leftarrow d(u) +$ waga krawędzi $u-w$

Następnie wierzchołek u czynimy poprzednikiem w $p(w) \leftarrow u$.

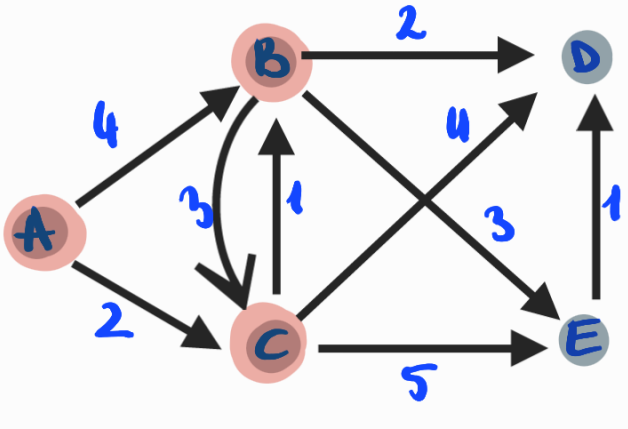


u	A	B	C	D	E
d[u]	0	∞	∞	∞	∞
p[u]	-1	-1	-1	-1	-1



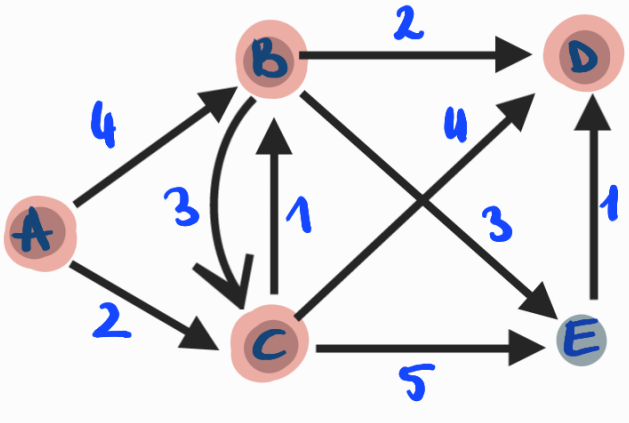
C JEST MNIEJSZE OD B

u	A	B	C	D	E
d[u]	0	4	2	∞	∞
p[u]	-1	A	A	-1	-1

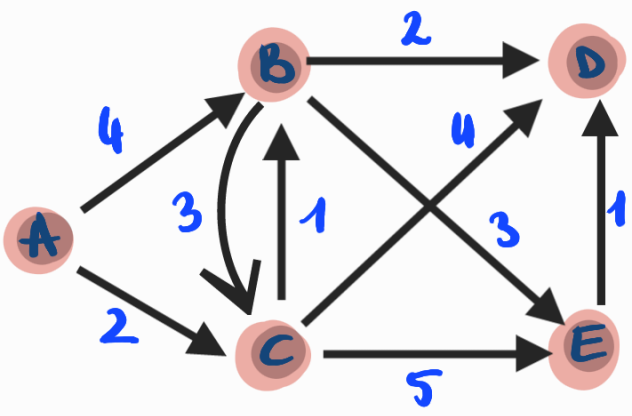


B JEST NAJMNIEJSZE

u	A	B	C	D	E
d[u]	0	3	2	6	5
p[u]	-1	C	A	C	C



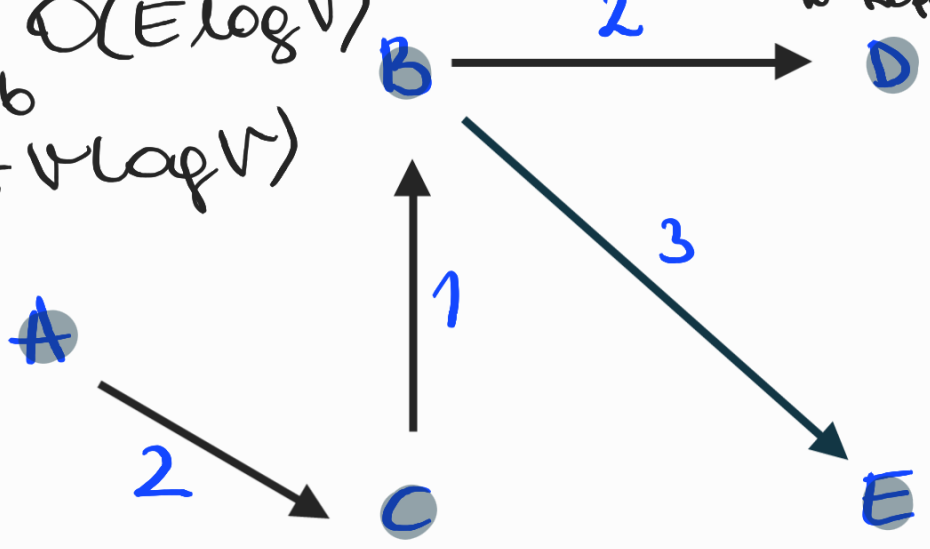
u	A	B	C	D	E
d[u]	0	3	2	5	6
p[u]	-1	C	A	B	B



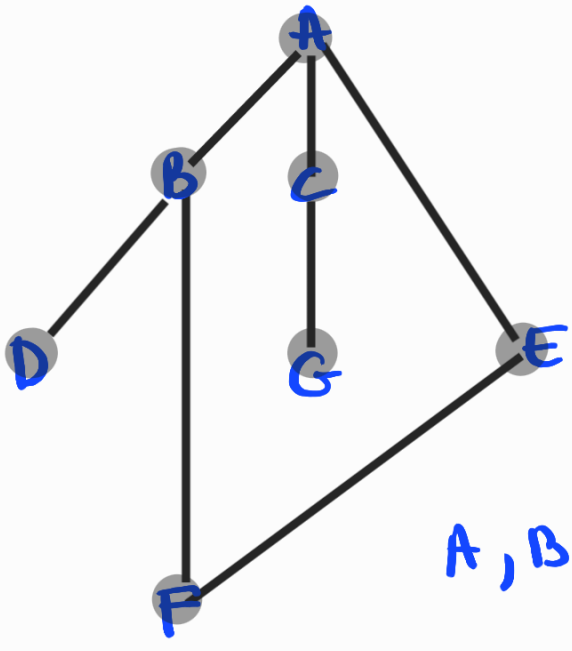
u	A	B	C	D	E
d[u]	0	3	2	5	6
p[u]	-1	C	A	B	B

- naiwnie $O(V^2)$
- kopiec $O(E \log V)$
- kopiec Fib $O(E + V \log V)$

nieodwiedzony wierzchołek o najmniejszej odległości możemy znaleźć w $O(\log |V|)$, bo nieodwiedzone wierzchołki trzymamy w kopcu.



DFS (przeszukiwanie w głąb)



Zaczynamy od wierzchołka A

A, B, D, F, E, C, G

BELLMAN-FORD

- wyszukiwanie najkrótszych ścieżek w grafie ważonym z wierzchołka źródłowego do wszystkich pozostałych

(działa dla ujemnych wag krawędzi, nie może jednak wystąpić cykl o ujemnej wadze osiągalny ze źródła)

BELLMAN-FORD(G, w, s):

dla każdego wierzchołka v w V[G] wykonaj:

d[v] = nieskończoność

poprzednik[v] = niezdefiniowany

d[s] = 0

dla i od 1 do |V[G]| - 1 wykonaj:

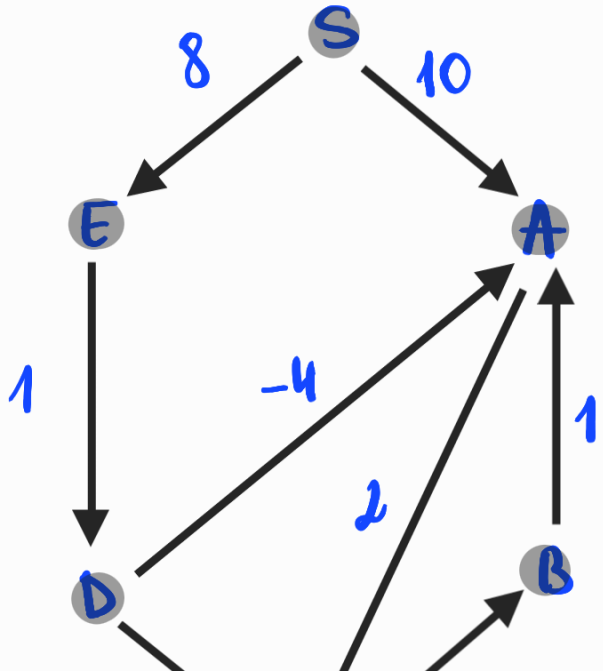
dla każdej krawędzi w E[G] wykonaj:

jeżeli d[v] > d[u] + w(u,v) to:

d[v] = d[u] + w(u,v)

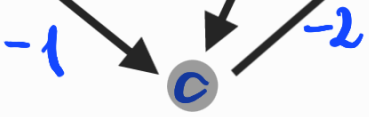
poprzednik[v] = u

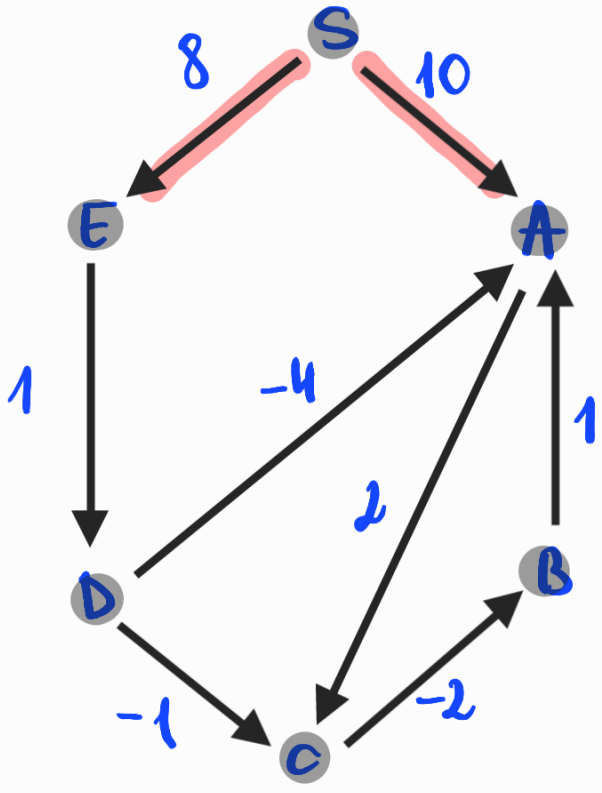
dla każdej jego krawędzi, która z niego wychodzi



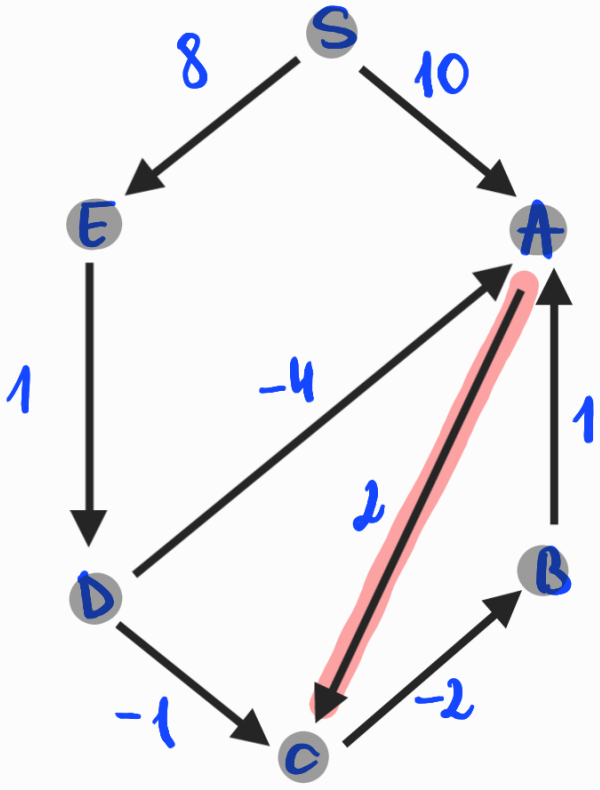
ITERACJA 1

0	∞	∞	∞	∞	∞
S	A	B	C	D	E

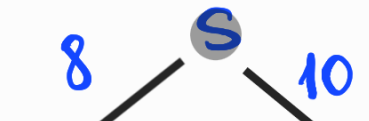


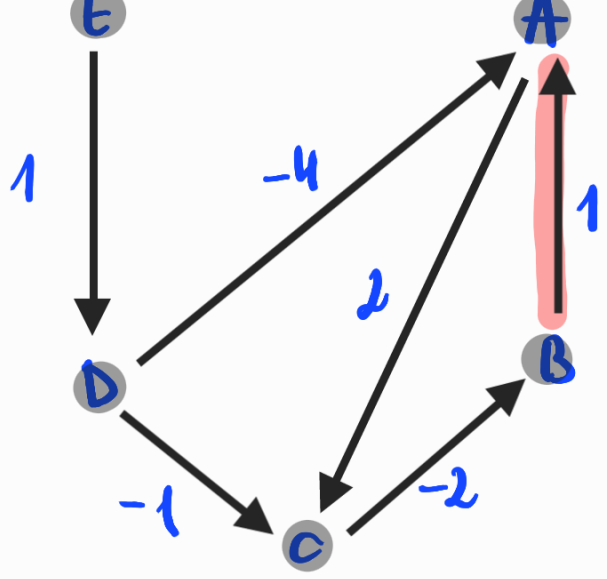


	10				8
0	~~∞~~	∞	∞	∞	~~∞~~
S	A	B	C	D	E



		12			
0	10	∞	~~∞~~	∞	8
S	A	B	C	D	E

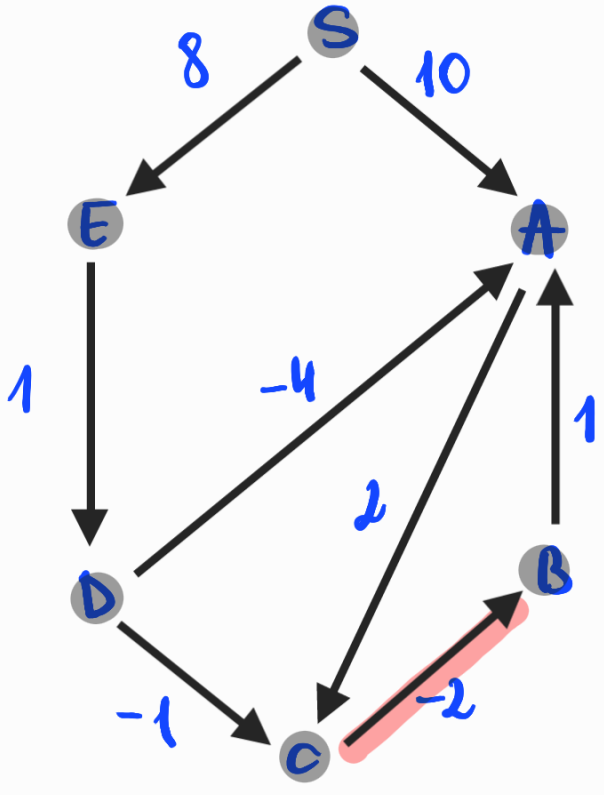




0 10 ∞ 12 ∞ 8

S A B C D E

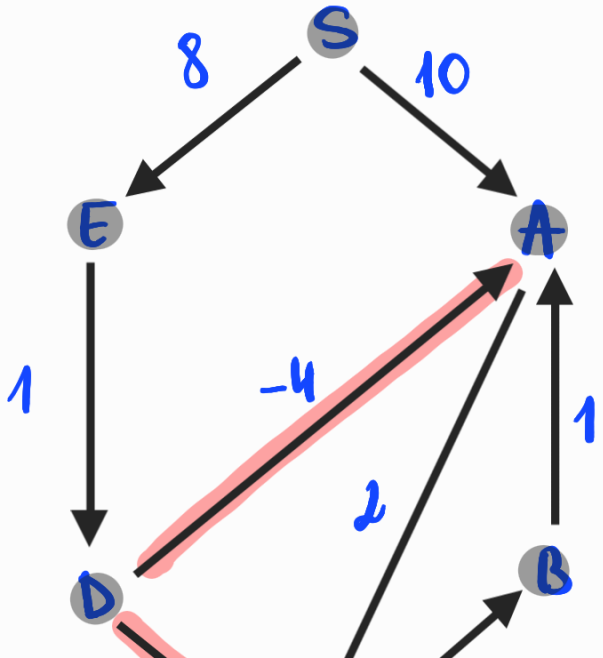
skip



10

0 10 ~~∞~~ 12 ∞ 8

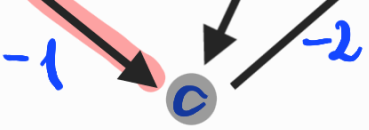
S A B C D E

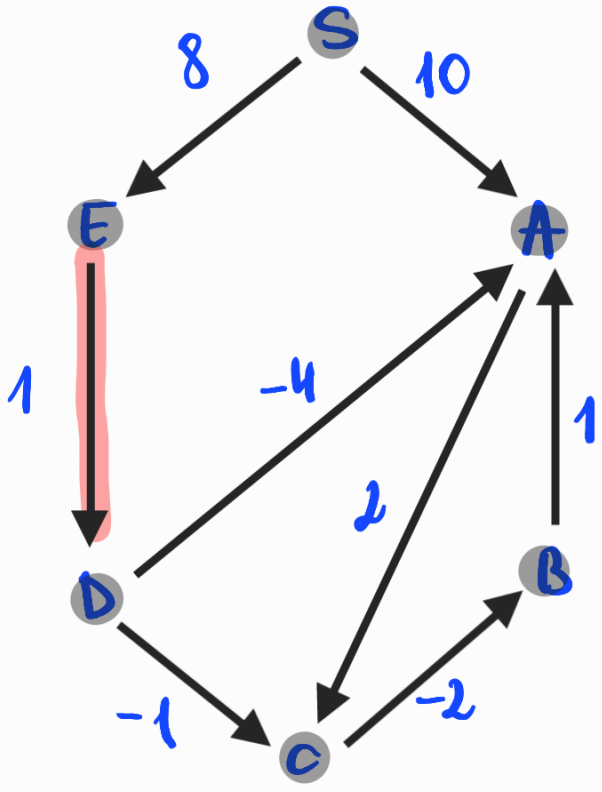


0 10 10 12 ∞ 8

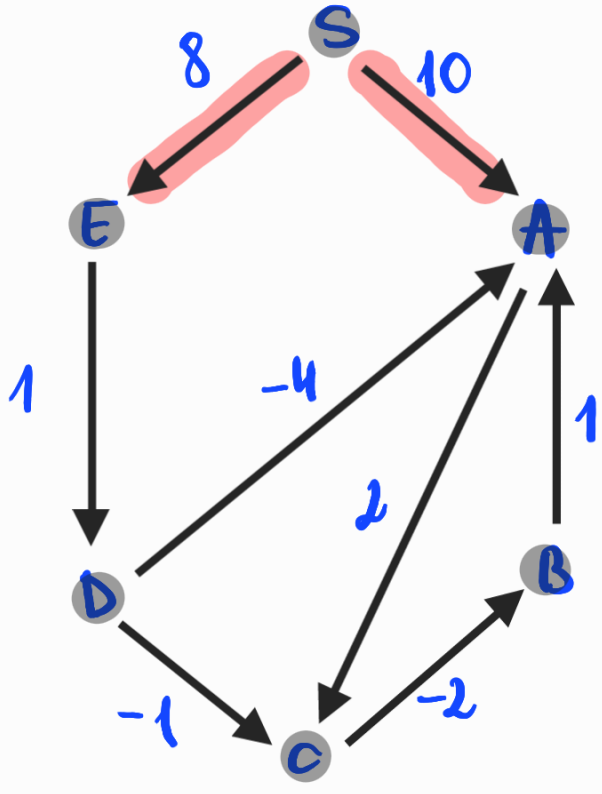
S A B C D E

skip



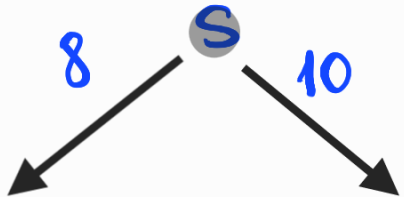


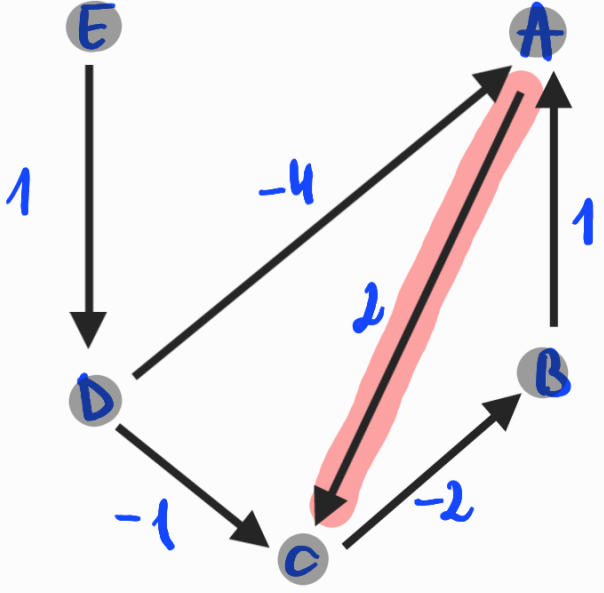
0	10	10	12	~~9~~	8
S	A	B	C	D	E



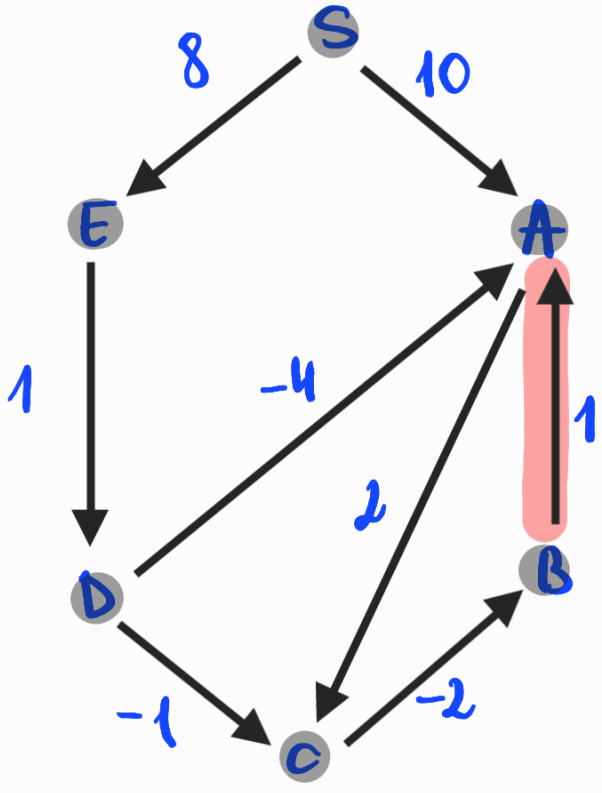
ITERACJA 2

0	10	10	12	9	8
S	A	B	C	D	E

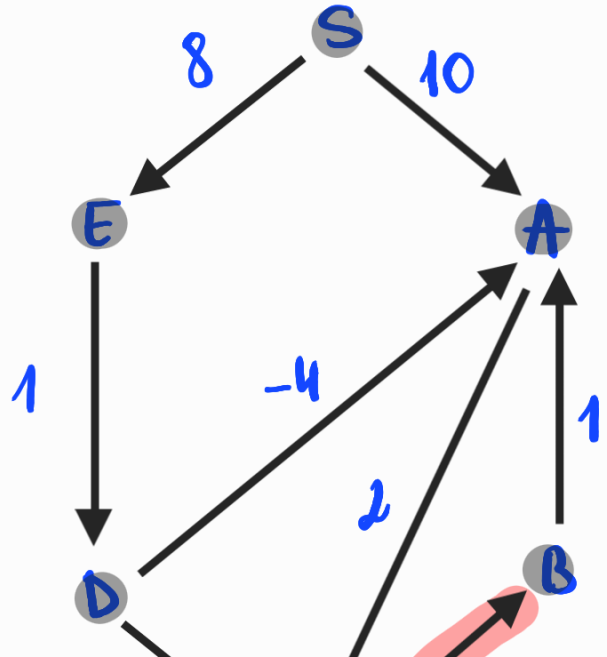




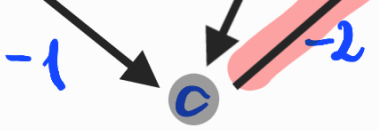
0	10	10	12	8	8
S	A	B	C	D	E

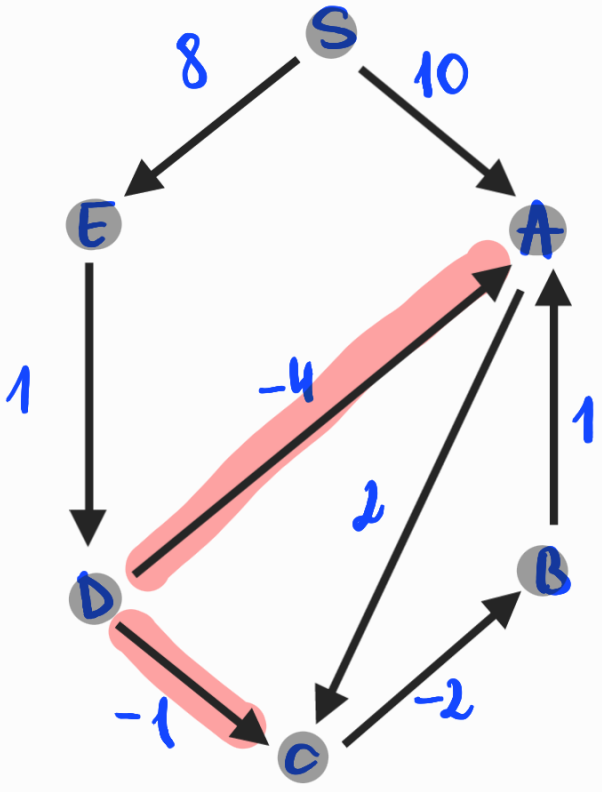


0	10	10	12	9	8
S	A	B	C	D	E

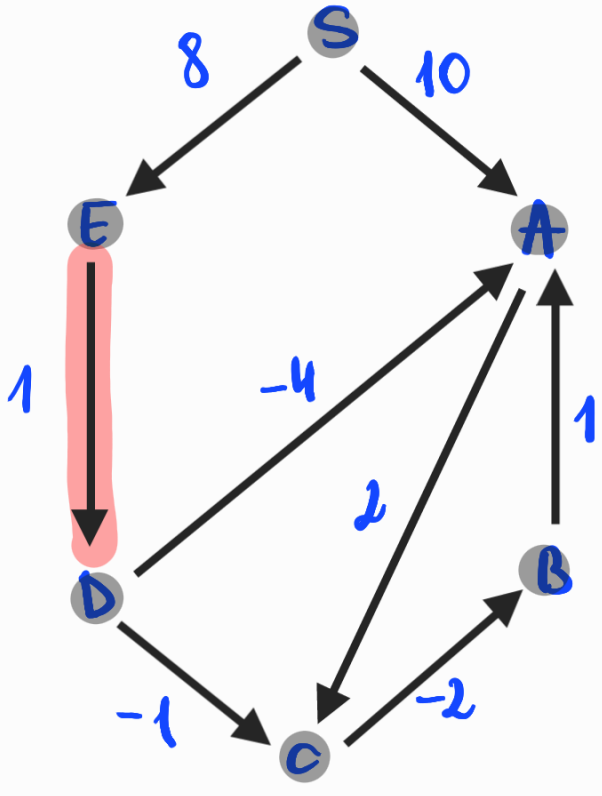


0	10	10	12	9	8
S	A	B	C	D	E



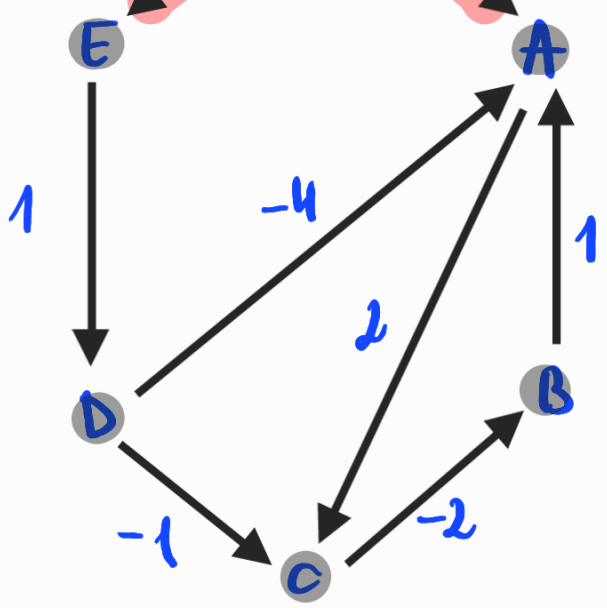


	5		8		
0	~~10~~	10	~~12~~	9	8
S	A	B	C	D	E

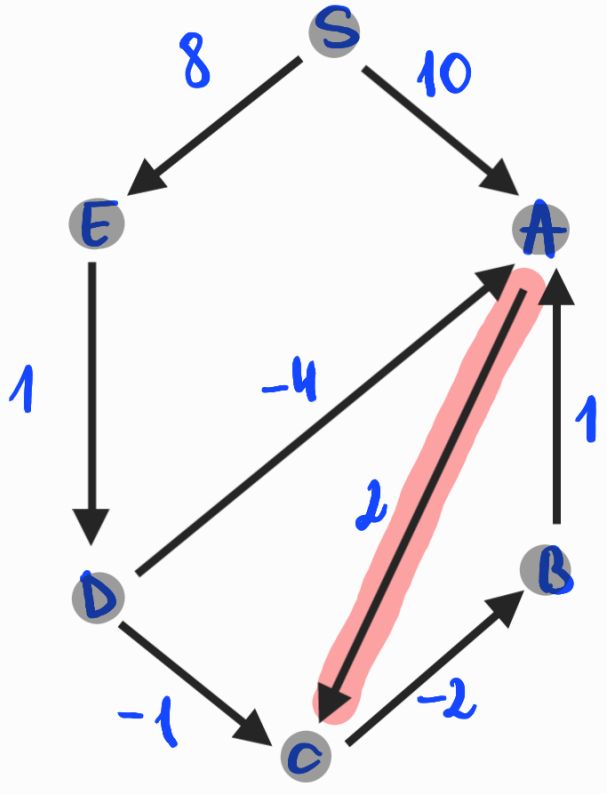


0	5	10	8	9	8
S	A	B	C	D	E

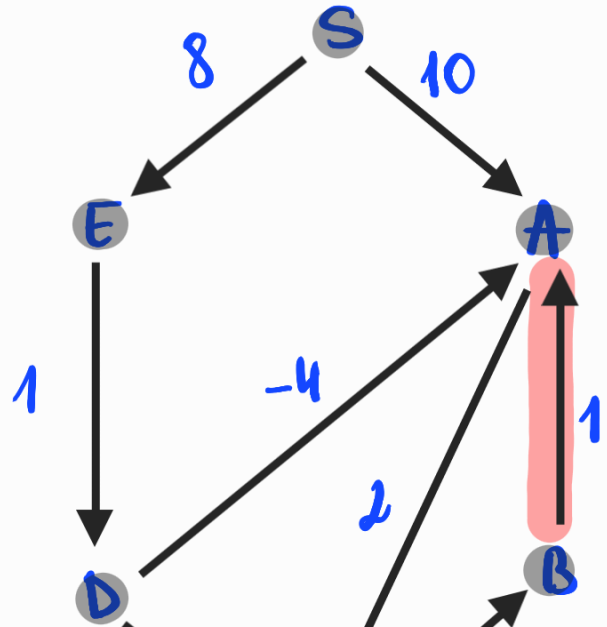
ITERACJA 3



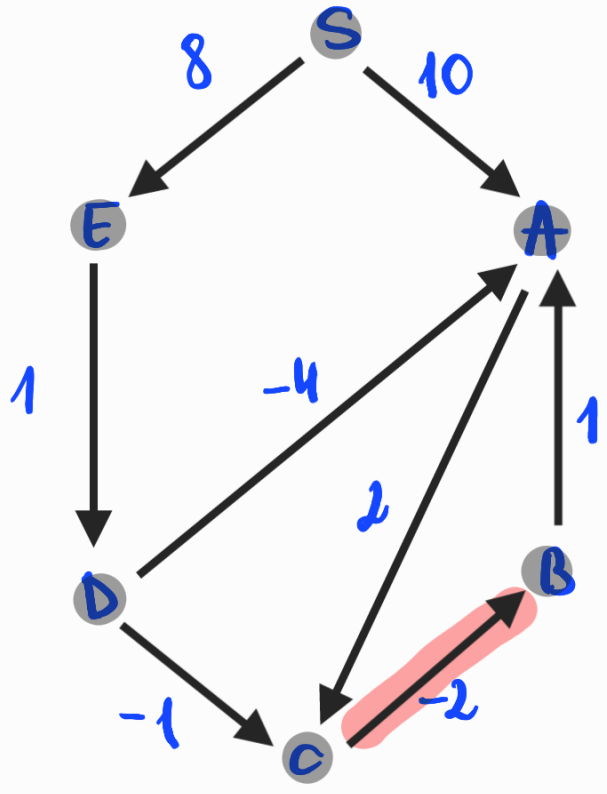
0	5	10	8	9	8
S	A	B	C	D	E



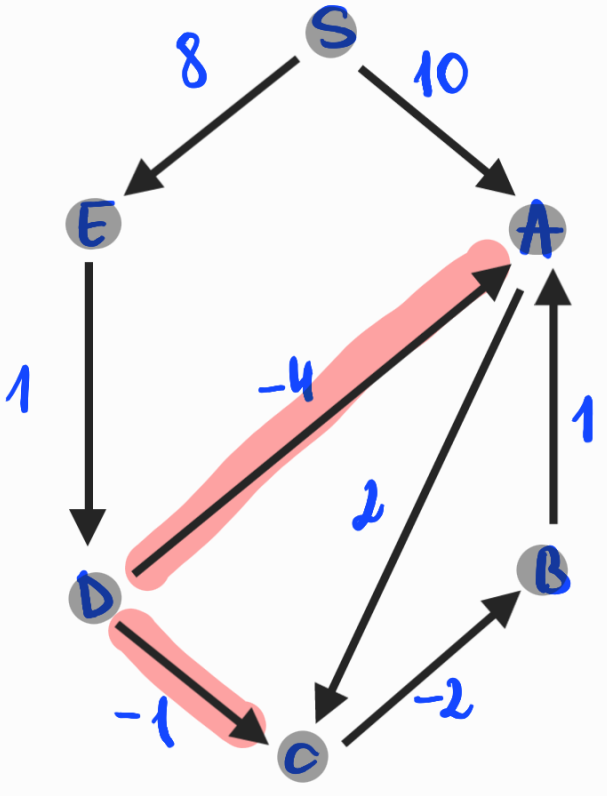
			7		
0	5	10	~~8~~	9	8
S	A	B	C	D	E



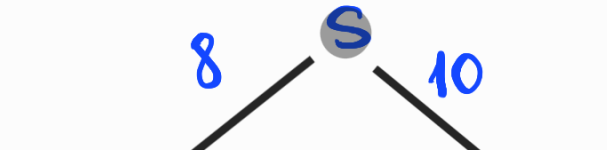
0	5	10	7	9	8
S	A	B	C	D	E

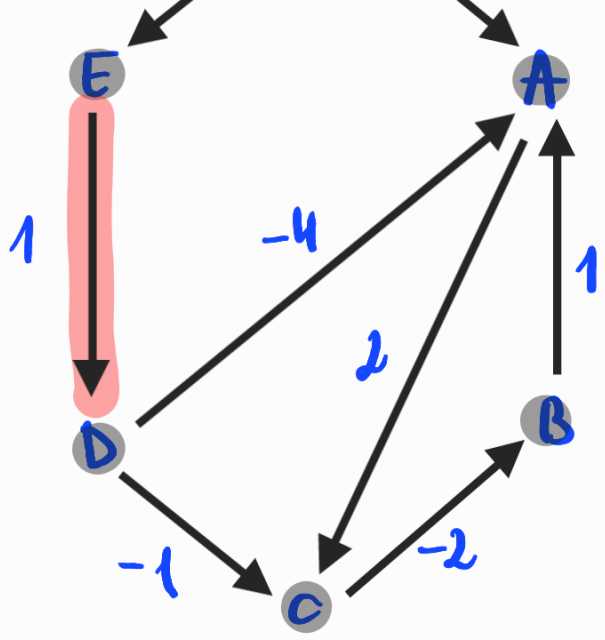


		5			
0	5	~~10~~	7	9	8
S	A	B	C	D	E



0	5	5	7	9	8
S	A	B	C	D	E





0	5	5	7	9	8
S	A	B	C	D	E

$O(VE)$

ITERACJA 4 - nic się nie zmieni

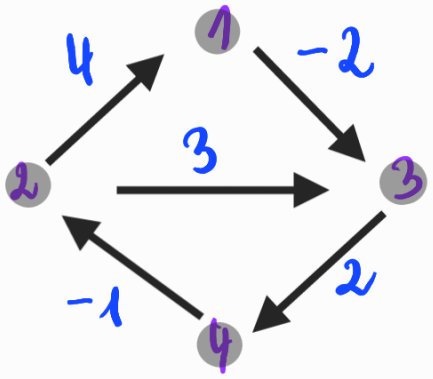
WARSHALL-FLOYD (wykorzystuje DP)

- znajduje najkrótsze ścieżki pomiędzy wszystkimi parami wierzchołków w grafie ważonym

(może zawierać ujemne wagi, ale nie może zawierać ujemnych cykli)

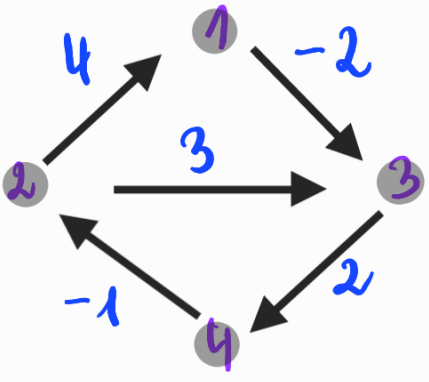
```
WARSHALL-FLOYD(  ):
    V = liczba wierzchołków w grafie
    dist = tablica min odległości (V x V)
    dla każdego wierzchołka v:
        dist[v][v] = 0
    dla każdej krawędzi (u,v):
```

```
dist[u][v] = wage(u,v)
for k from 1 to V:
    for i from 1 to V:
        for j from 1 to V:
            if dist[i][j] > dist[i][k] + dist[k][j]
                dist[i][j] = dist[i][k] + dist[k][j]
```



	1	2	3	4
1	0		-2	
2	4	0	3	
3			0	2
4		-1		0

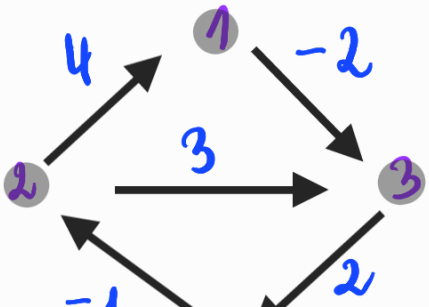
$k = 1 \quad i = 1 \quad j = 1$



	1	2	3	4
1	0		-2	
2	4	0	3	
3			0	2
4		-1		0

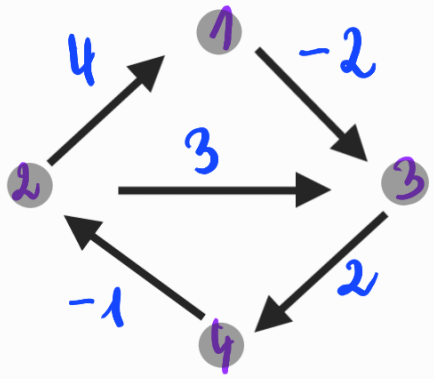
$\vdots$

$k = 1 \quad i = 1 \quad j = 4$



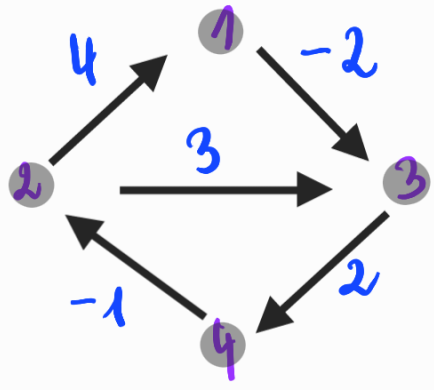
	1	2	3	4
1	0		-2	
2	4	0	3	
3			0	2
4		-1		0

$k=1 \quad i=2 \quad j=1$



	1	2	3	4
1	0		-2	
2	4	0	3	
3			0	2
4		-1		0

$k=1 \quad i=2 \quad j=3$



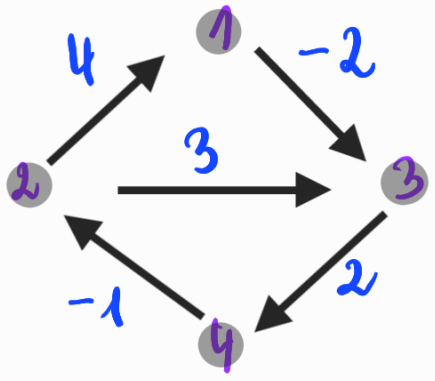
	1	2	3	4
1	0		-2	
2	4	0	2	
3			0	2
4		-1		0

$3 > 4 + (-2)$

$3 > 2$

$\vdots$

$k=2 \quad i=4 \quad j=1$

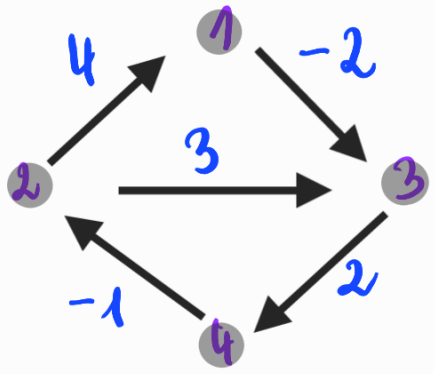


	1	2	3	4
1	0		-2	
2	4	0	2	
3			0	2
4	3	-1		0

$\infty > -1 + 4$

$\infty > 3$

$k=2 \quad i=4 \quad j=3$

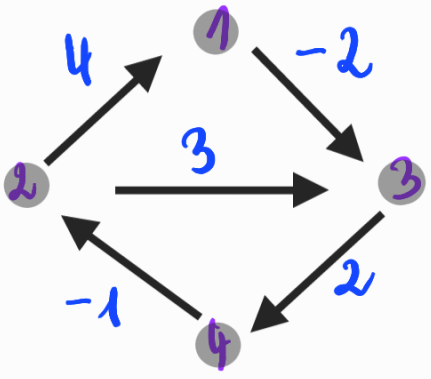


	1	2	3	4
1	0		-2	
2	4	0	2	
3			0	2
4	3	-1	1	0

$\infty > -1+2$

$\infty > 1$

$k=3 \quad i=1 \quad j=4$

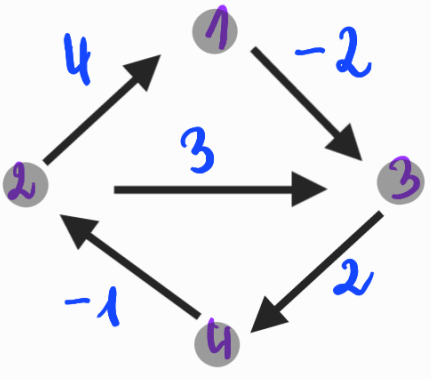


	1	2	3	4
1	0		-2	0
2	4	0	2	
3			0	2
4	3	-1	1	0

$\infty > -2+2$

$\infty > 0$

$k=3 \quad i=2 \quad j=4$

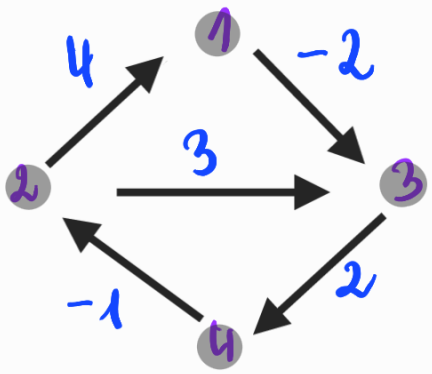


	1	2	3	4
1	0		-2	0
2	4	0	2	4
3			0	2
4	3	-1	1	0

$\infty > 2+2$

$\infty > 4$

$k=4 \quad i=1 \quad j=2$

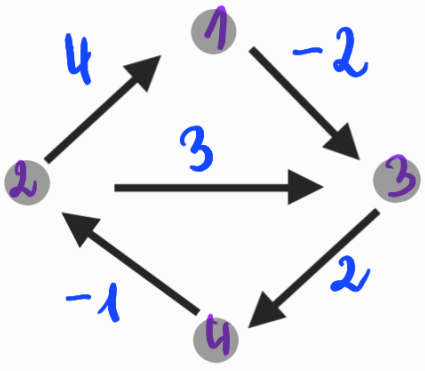


	1	2	3	4
1	0	-1	-2	0
2	4	0	2	4
3			0	2
4	3	-1	1	0

$\Delta > 0+(-1)$

$\Delta > -1$

$k=4 \quad i=3 \quad j=1$

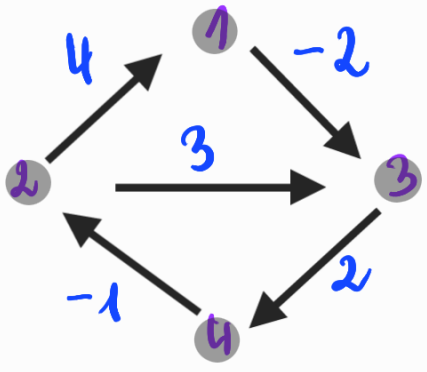


	1	2	3	4
1	0	-1	-2	0
2	4	0	2	4
3	5		0	2
4	3	-1	1	0

$\Delta > 2+3$

$\Delta > 5$

$k=4 \quad i=3 \quad j=2$



	1	2	3	4
1	0	-1	-2	0
2	4	0	2	4
3	5	1	0	2
4	3	-1	1	0

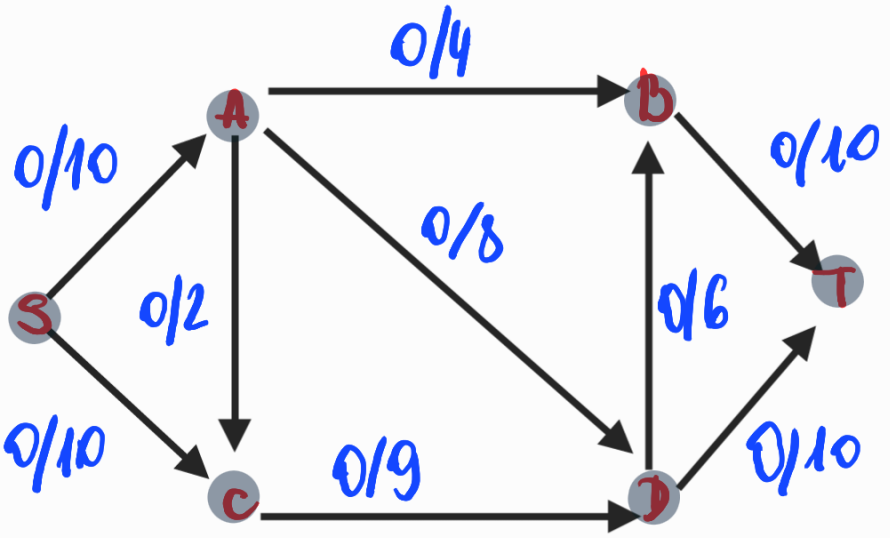
$\Delta > 2+(-1)$

$\Delta > 1$

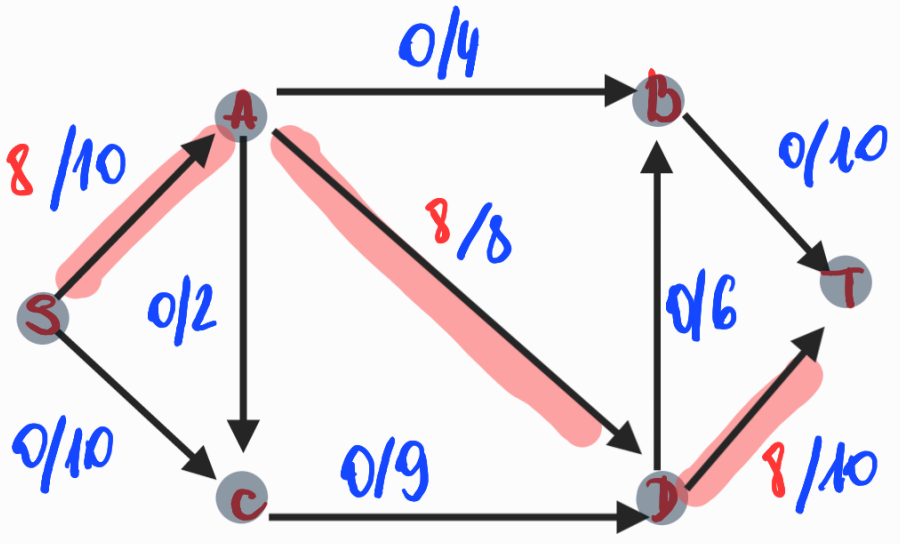
KONIEC $O(V^3)$

FORD-FULKERSON

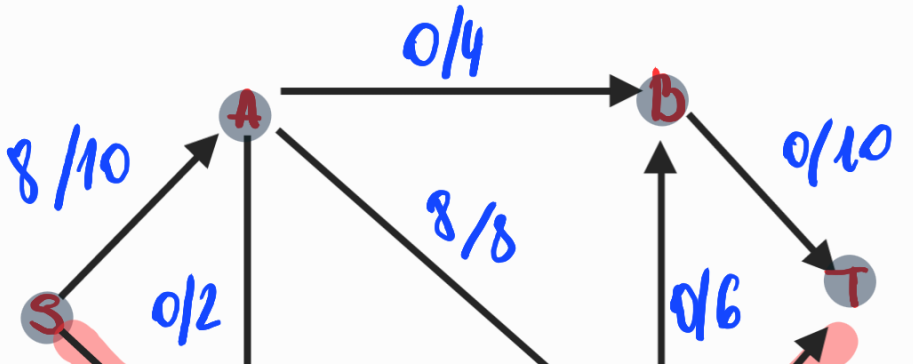
- maksymalny przepływ w sieci przepływowej



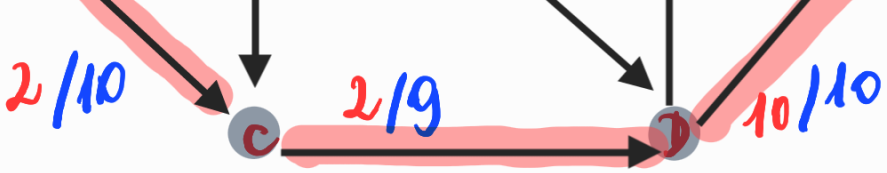
0

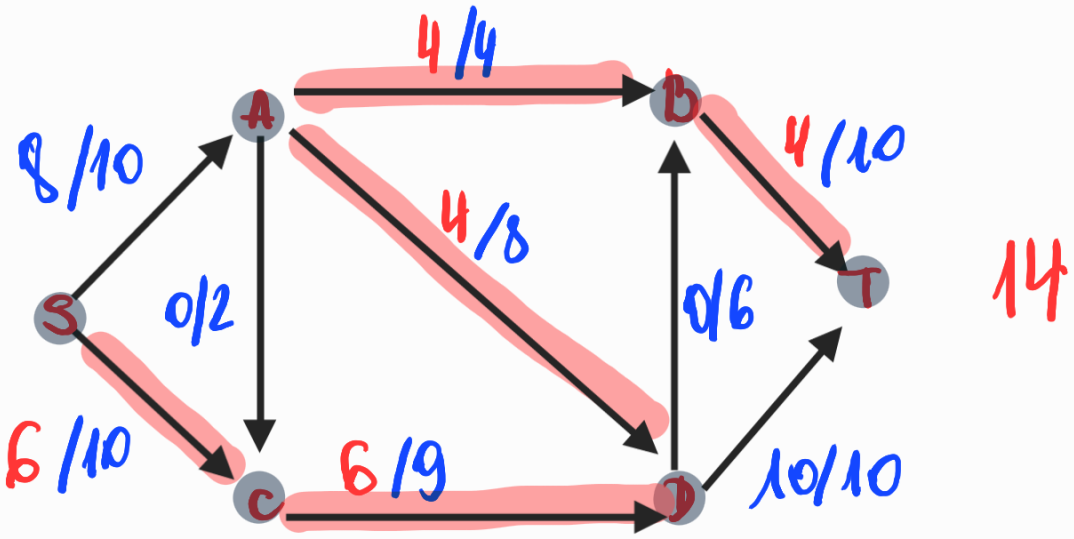


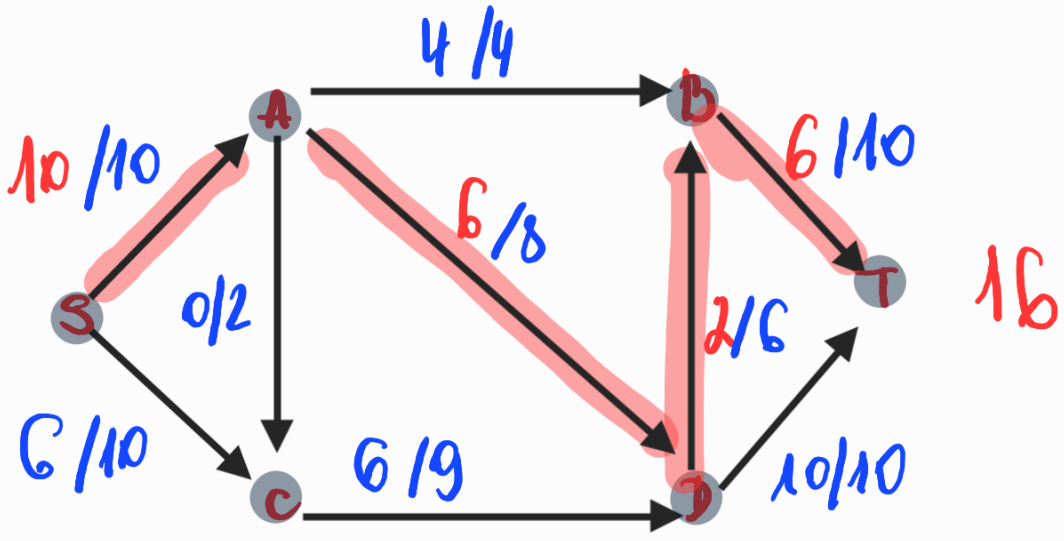
8



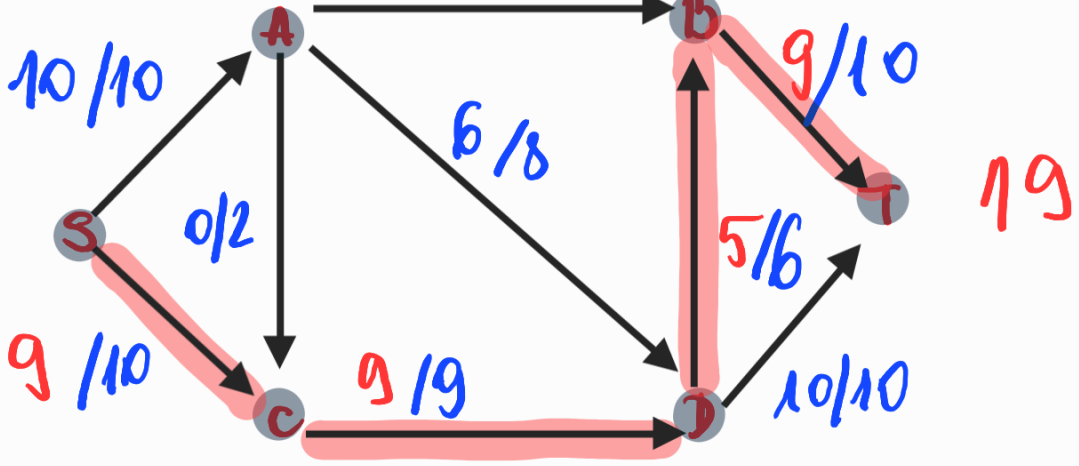
10

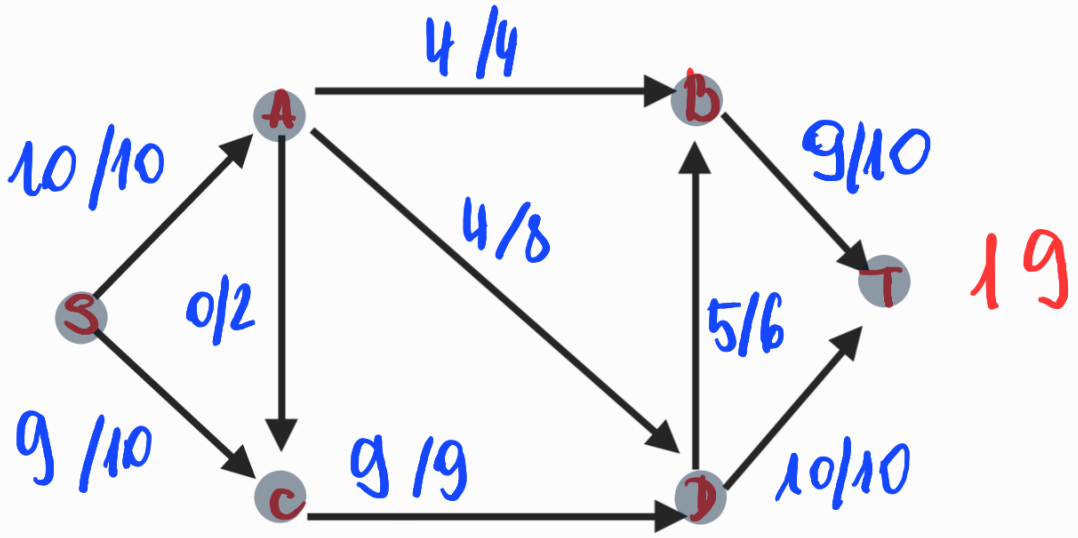

2/10
2/9
10/10


4/4
8/10
4/10
4/8
0/2
0/6
6/10
6/9
10/10
14


4/4
10/10
6/10
6/8
0/2
2/6
6/10
6/9
10/10
16


4/4





1. find an augmenting path
2. compute the bottleneck capacity
3. augment each edge and the total flow

while istnieje prosta ścieżka powiększająca $p \in G_f$ do

for each $(u,v) \in p$ do

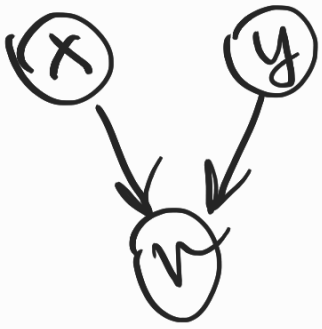
$f(u,v) = f(u,v) + cf(p)$

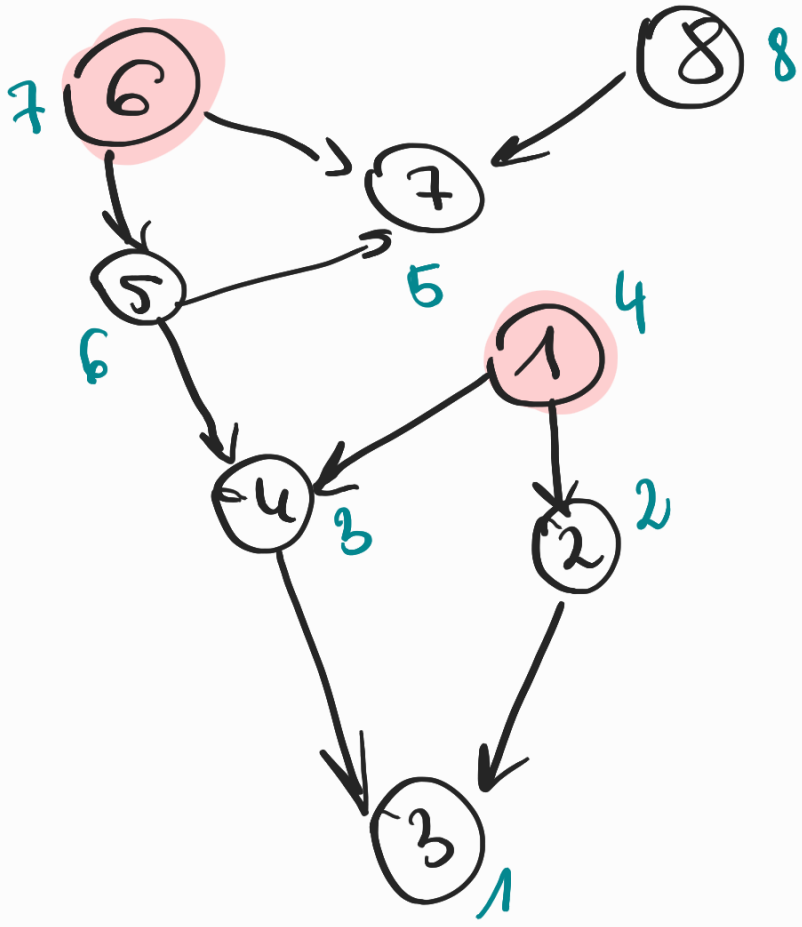
$f(v,u) = f(v,u) - cf(p)$

$O(VE^2)$

SORTOWANIE TOPOLOGICZNE

Chcemy żeby x, y wypisało przed wypisaniem v.



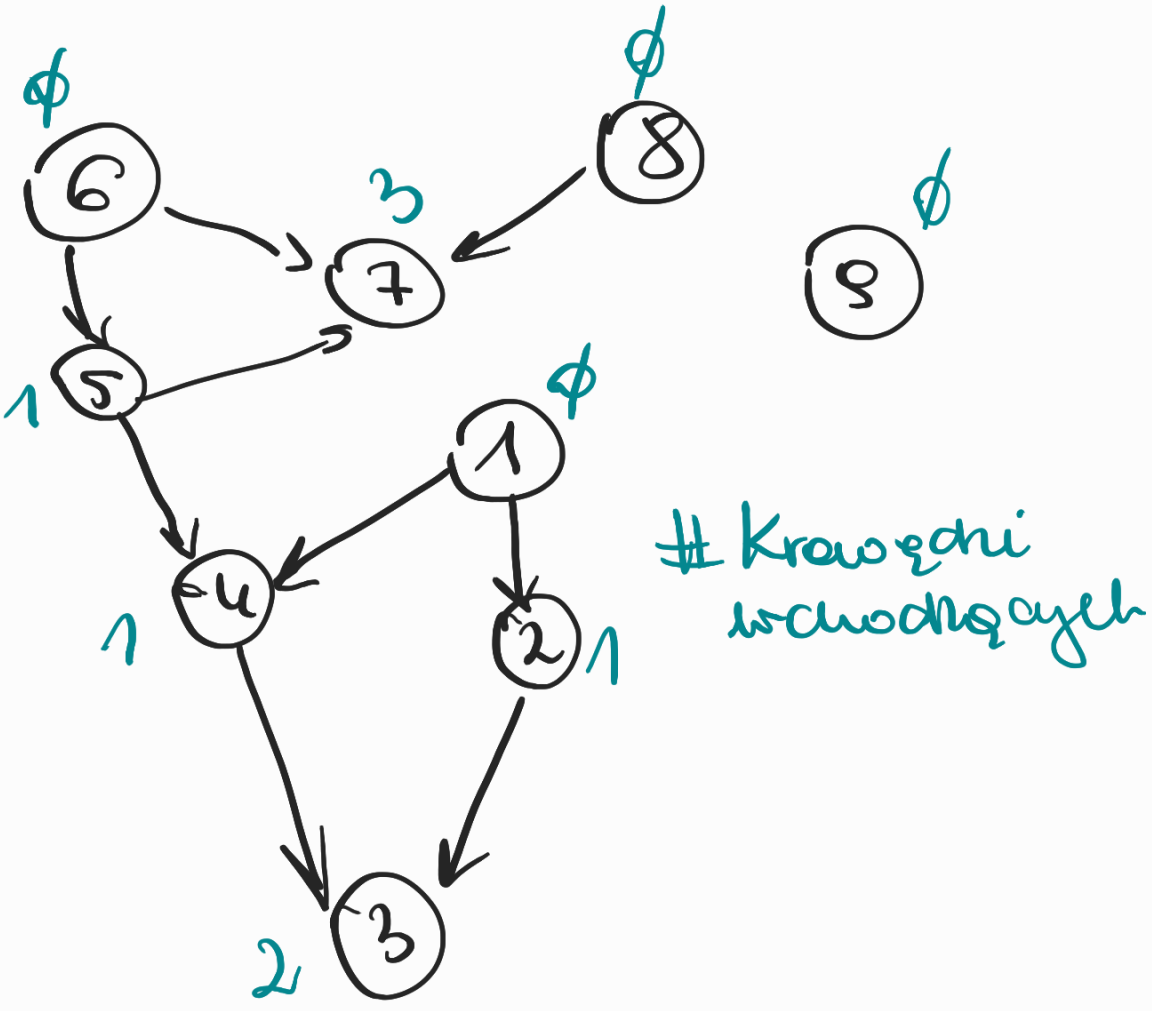


Już posortowane tylko na odwrót

ALGO 1: (DFS)

Puszczamy DFS zaczynając od wierzchołka do którego nie dochodzi żadna krawędź (takie -

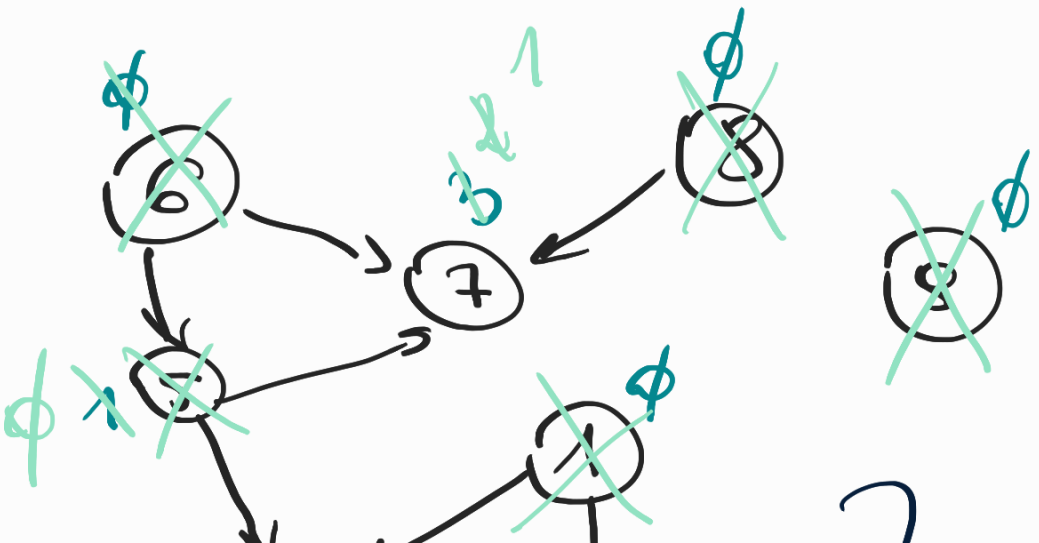
rowowej), np. 1.

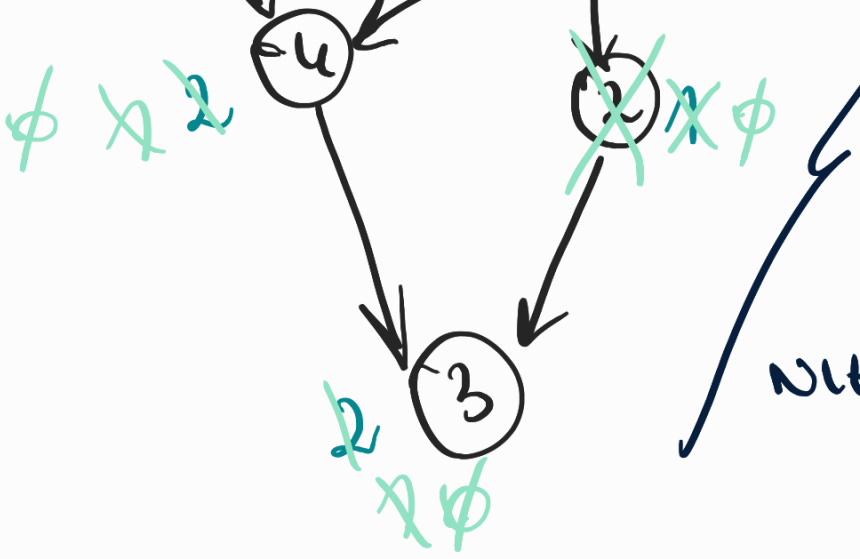


krawędzi wchodzących)

KOLEJKA: 1, 6, 8, 9

Patrzymy na wszystkich sąsiadów 1.





CZY JAKIEŚ PĘTLA DAŁBY ZMIENIŁO SIĘ NA ∞?

KOLEJKA : 1, 6, 8, 3, 2, 5, 7, 4

13

$O(n+m)$

PRIM
KRUSKAL
DIJKSTRA
} $O(|E| \lg |V|)$

B-FORD $O(E \cdot V)$

W-FLOYD $O(V^3)$

F-FULKERSON $O(EV^2)$