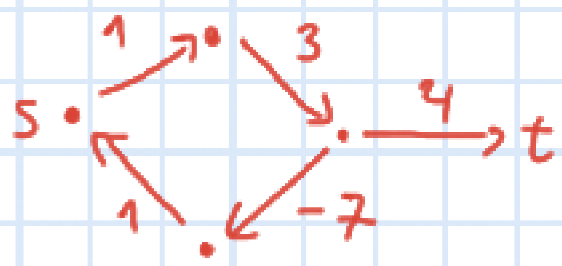
Algorytmy i Struktury Danych
Wykład 8

Szukamy najkrótszych ścieżek w grafie gdy wagi krawędzi mogą być ujemne

rozwiązanie może nie istnieć (nie być dobrze zdefiniowane) gdy istnieje cykl o ujemnej wadze



Algorytm Bellmana-Forda (ścieżki wychodzące z s)
$G=(V,E)$

① Inicjalizacja

for $v \in V$:
  $v.d = \infty$
  $v.parent = None$
$s.d = 0$

Złożoność

$\Theta(VE)$

② Relaksacja

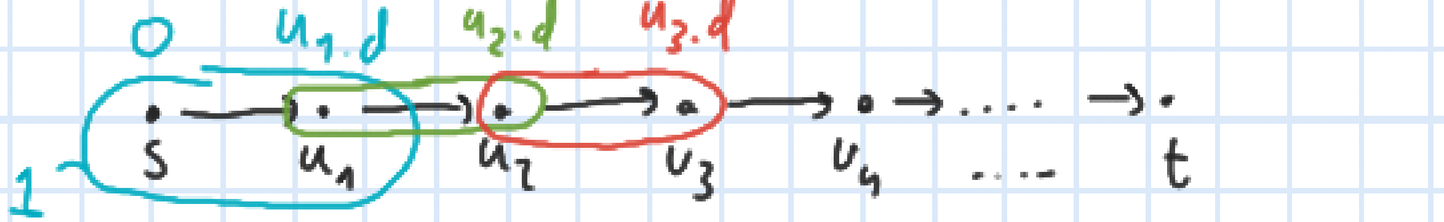
for i in range$(|V|-1)$:
  for $(u,v) \in E$
    Relax(u,v)

③ Weryfikacja
dla każdego $(u,v) \in E$ sprawdzamy czy $v.d \le u.d + w(u,v)$

jeśli dla jakiejś krawędzi nie zachodzi, to mamy cykl o ujemnej wadze

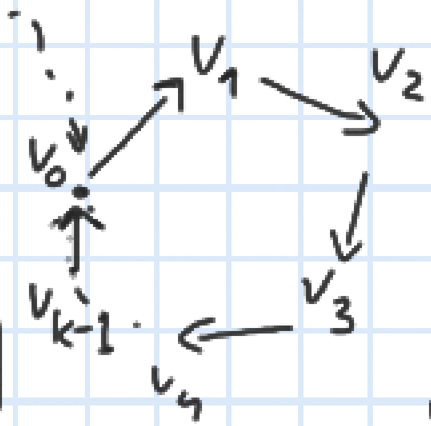
Czemu algorytm jest poprawny?

Wybierzmy sobie pewną najkrótszą ścieżkę z s do t



Po każdej iteracji głównej pętli w części ② algorytmu coraz dłuższy prefiks tej ścieżki ma poprawne wartości d

Jak działa wykrywanie ujemnych cykli?



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

równe

Mamy powyższy cykl o ujemnej wadze, ale wykrywanie ujemnego cyklu zawiodło.

Ale wówczas zachodzi:

te wartości są równe

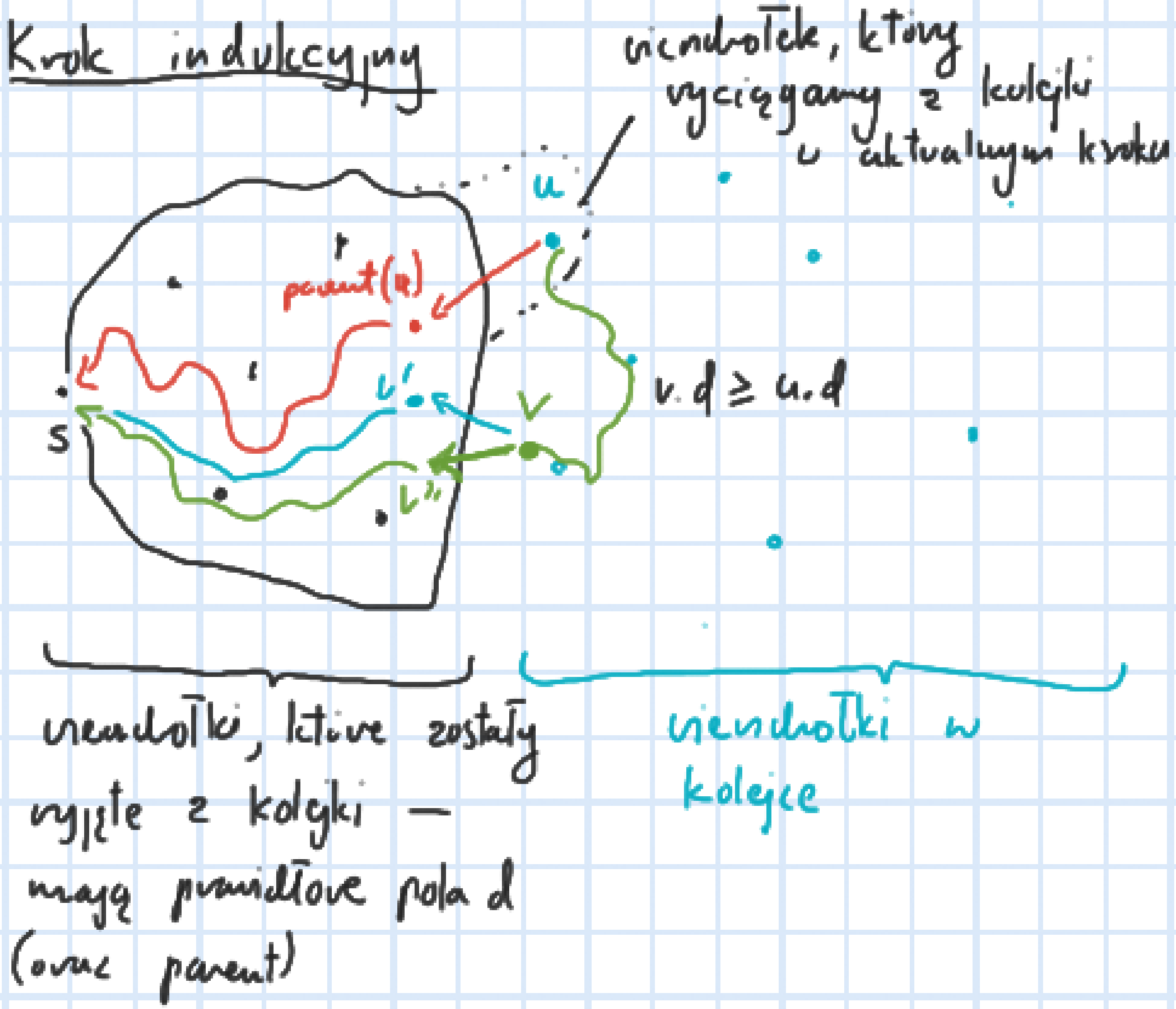
$$\sum_{i=1}^{k} v_i.d \le \sum_{i=1}^{k} \left(v_{i-1}.d + w(v_{i-1}, v_i)\right)$$

$$0 \le \sum_{i=1}^{k} w(v_{i-1}, v_i) \rightarrow \text{Sprzeczność}$$

Poprawność algorytmu Dijkstry

Dowód przez indukcję — dla pierwszego kroku sytuacja jest jasna, bo jedyny wyciągnięty z kolejki wierzchołek to s i jego wartość s.d = 0

Krok indukcyjny



Najkrótsze ścieżki między każdą parą wierzchołków

① |V| wywołań algorytmu Dijkstry $O(VE\log V)$

② |V| wywołań algorytmu Bellmana-Forda $O(V^2E)$

Konwencja

W specjalizowanych algorytmach dla najkrótszych ścieżek między każdą parą wierzchołków stosujemy reprezentację macierzową

— i tak szukamy macierzy najkrótszych odległości

D[u][v] to dł. najkrótszej ścieżki z u do v

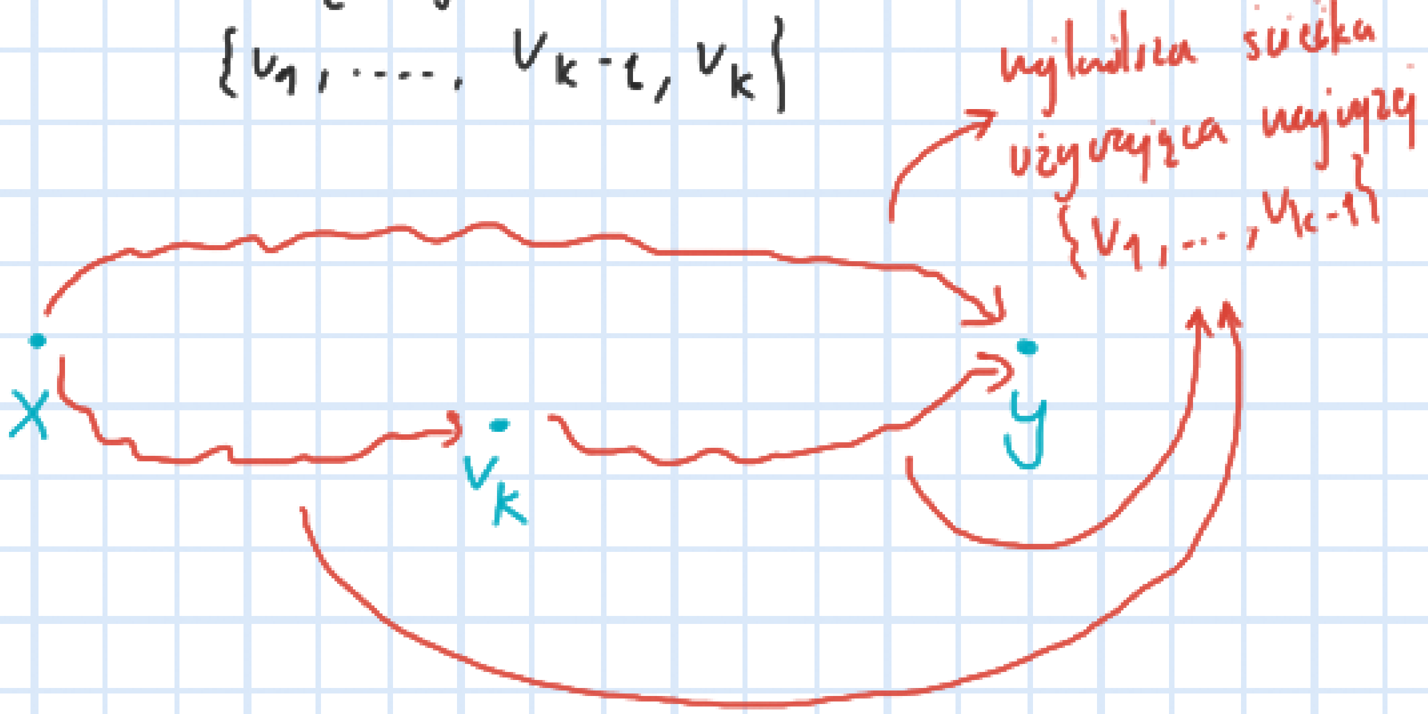
Algorytm Floyda-Warshalla

Idea: Jeśli znamy najkrótsze ścieżki między każdą parą wierzchołków, które wierzchołki używają tylko tych ze zbioru

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

to możemy osiągnąć podobne najkrótsze ścieżki dla wewnętrznych wierzchołków

$\{v_1, \ldots, v_{k-1}, v_k\}$



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

$S^{(k)}$ – macierz długości najkrótszych ścieżek korzystających z $\{v_1, \ldots, v_k\}$ jako wewnętrznych

$S^{(0)}$ – macierz długości krawędzi w grafie

Algorytm

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

można pominąć!

P[x][y] się nie zmienia

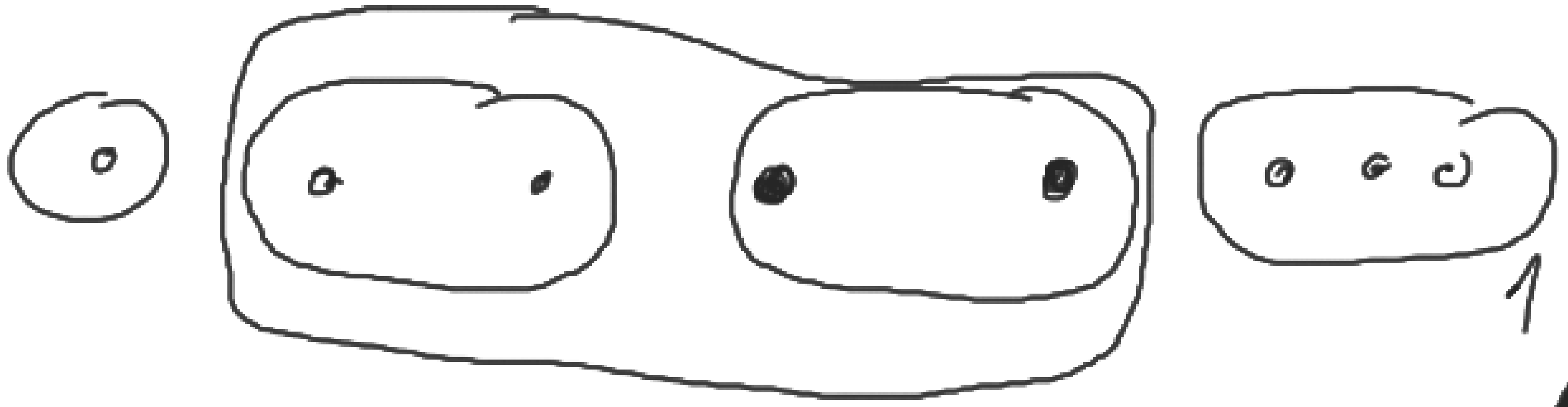
P[x][y] = P[v_k][y]

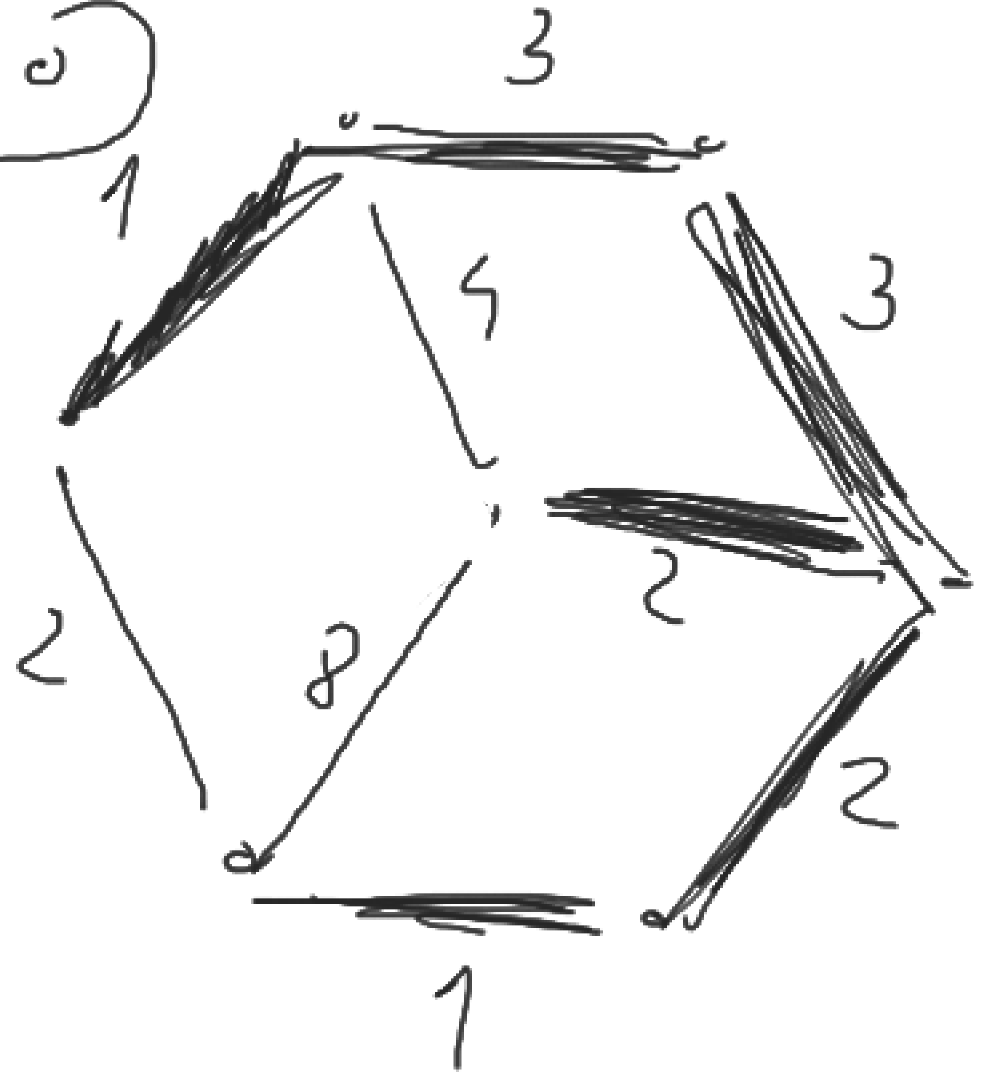
$D = S^{(n)}$

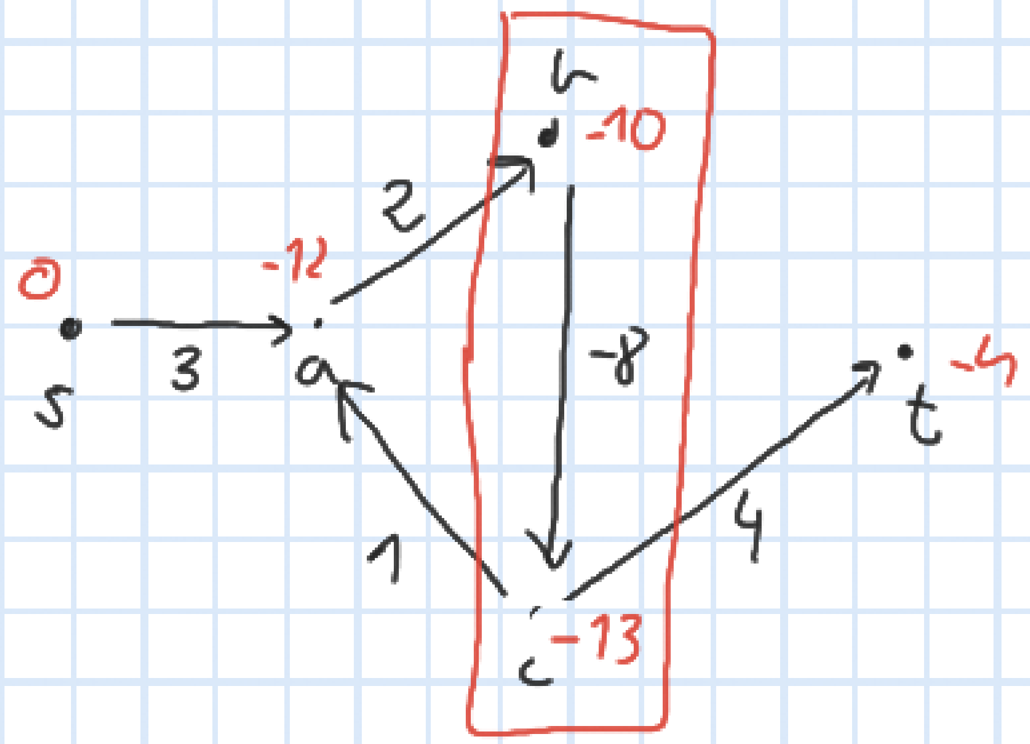
$\boxed{\text{Złożoność } \Theta(V^3)}$

możemy na tym etapie obliczać macierz P "rodziców"

P[x][y] – ostatni wierzchołek przed y na ścieżce z x do y




3
1
5
3
2
2
8
2
1



0
s
3
-12
a
2
b
-10
-8
1
c
-13
4
t
-4