

Obliczanie najkrótszych ścieżek pomiędzy wszystkimi parami wierzchołków

Obliczanie najkrótszych ścieżek metodą mnożenia macierzy

- problem: szukamy długości najkrótszych ścieżek pomiędzy wszystkimi parami wierzchołków
- wykorzystanie metody programowania dynamicznego
- związek z problemem mnożenia macierzy
- pewne optymalizacje

Rekurencyjna definicja rozwiązania

Lemat *Jeśli v_0, \dots, v_l jest najkrótszą ścieżką łączącą v_0 z v_l , to podścieżka v_s, \dots, v_t gdzie $0 \leq s \leq t \leq l$ jest najkrótszym połączeniem pomiędzy wierzchołkami v_s i v_t .*

Oznaczenie: d_{ij}^m - długość najkrótszej ścieżki z i do j zawierającej co najwyżej m krawędzi. Wówczas:

$$d_{ij}^m = \begin{cases} \min(d_{ij}^{m-1}, \min\{d_{ik}^{m-1} + w_{kj} : k = 1, \dots, n\}) & i \neq j \\ 0 & i = j \end{cases}$$

dla $m < n$ oraz

$$d_{ij}^0 = \begin{cases} +\infty & i \neq j \\ 0 & i = j \end{cases}$$

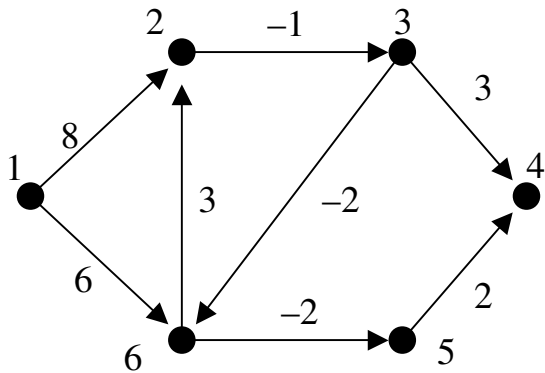
Algorytm

Wejście: macierz D^m współczynników d_{ij}^m oraz macierz W opisująca wagi w_{ij}

Wyjście: macierz D^{m+1} współczynników d_{ij}^{m+1}

```
procedure SearchShortestPaths(  $D^m$ ,  $W$  )  
begin  
  for  $i := 1$  to  $n$  do  
    for  $j := 1$  to  $n$  do begin  
       $d_{ij}^{m+1} = d_{ij}^m$ ;  
      for  $k := 1$  to  $n$  do  
         $d_{ij}^{m+1} = \min(d_{ij}^{m+1}, d_{ik}^m + w_{kj})$ ;  
      end  
    return  $D^{m+1}$ ;  
end
```

Przykład



$$W = \begin{bmatrix} 0 & 8 & \infty & \infty & \infty & 6 \\ \infty & 0 & -1 & \infty & \infty & \infty \\ \infty & \infty & 0 & 3 & \infty & -2 \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 2 & 0 & \infty \\ \infty & 3 & \infty & \infty & -2 & 0 \end{bmatrix}$$

$$D^2 = \begin{bmatrix} 0 & 8 & 7 & \infty & 4 & 6 \\ \infty & 0 & -1 & 2 & \infty & -3 \\ \infty & 1 & 0 & 3 & -4 & -2 \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 2 & 0 & \infty \\ \infty & 3 & 2 & 0 & -2 & 0 \end{bmatrix}$$

$$D^3 = \begin{bmatrix} 0 & 8 & 7 & 6 & 4 & 5 \\ \infty & 0 & -1 & 2 & -5 & -3 \\ \infty & 1 & 0 & -2 & -4 & -2 \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 2 & 0 & \infty \\ \infty & 3 & 2 & 0 & -2 & 0 \end{bmatrix}$$

$$D^4 = \begin{bmatrix} 0 & 8 & 7 & 6 & 3 & 5 \\ \infty & 0 & -1 & -3 & -5 & -3 \\ \infty & 1 & 0 & -2 & -4 & -2 \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 2 & 0 & \infty \\ \infty & 3 & 2 & 0 & -2 & 0 \end{bmatrix}$$

$$D^5 = \begin{bmatrix} 0 & 8 & 7 & 5 & 3 & 5 \\ \infty & 0 & -1 & -3 & -5 & -3 \\ \infty & 1 & 0 & -2 & -4 & -2 \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 2 & 0 & \infty \\ \infty & 3 & 2 & 0 & -2 & 0 \end{bmatrix}$$

Związek z mnożeniem macierzy

```

procedure SearchShortestPaths(  $D^m, W$  )
begin
  for  $i := 1$  to  $n$  do
    for  $j := 1$  to  $n$  do begin
       $d_{ij}^{m+1} = +\infty$ ;
      for  $k := 1$  to  $n$  do
         $d_{ij}^{m+1} = \min(d_{ij}^{m+1}, d_{ik}^m + w_{kj})$ ;
      end
    return  $D^{m+1}$ ;
  end

```

Stąd notacja: $D^{m+1} = D^m \cdot W$

```

procedure MatrixMultiplication(  $A, B$  )
begin
  for  $i := 1$  to  $n$  do
    for  $j := 1$  to  $n$  do begin
       $c_{ij} := 0$ ;
      for  $k := 1$  to  $n$  do
         $c_{ij} := c_{ij} + a_{ik} \cdot b_{kj}$ ;
      end
    return  $C$ ;
  end

```

Optymalizacje

Uwagi:

- $(n - 1)$ -krotne wykonanie mnożenia wg. schematu

$$D^{n-1} = (\dots (((D^0 \cdot W) \cdot W) \cdot W) \dots) \cdot W$$
 powoduje, że czas działania algorytmu wynosi $O(n^4)$
- Z punktu widzenia końcowego rozwiązania istotna jest tylko macierz D^{n-1}
- Zwiększenie wydajności algorytmu uzyskuje się poprzez rezygnację z obliczania większości macierzy pośrednich D^k

Sposób postępowania:

- zamierzamy obliczyć macierz o numerze $k = 2^{\lceil \log(n-1) \rceil}$, która stanowi poprawne rozwiązanie, gdyż $D^k = D^{n-1}$ dla każdego $k > n - 1$
- wykonujemy k razy podnoszenie macierzy do kwadratu obliczając tylko macierze o indeksach będących potęgami 2:

$$D^1 = W, D^2 = D^1 \cdot D^1, D^4 = D^2 \cdot D^2, \dots, D^{2^k} = D^{2^{k-1}} \cdot D^{2^{k-1}}$$

czas działania takiego algorytmu to $O(n^3 \log n)$

Algorytm Johnsona

- algorytm znajduje najkrótsze ścieżki pomiędzy wszystkimi parami wierzchołków
- algorytmy Dijkstry oraz Bellmana-Forda są wykorzystywane jako podprogramy
- asymptotyczny czas działania to $O(n^2 \log n + nm)$, gdzie n i m to odpowiednio liczba wierzchołków oraz krawędzi grafu

Zmiana funkcji wagowej

Uwaga Algorytm wywołuje algorytm Dijkstry dla każdego wierzchołka. Graf wejściowy może zawierać krawędzie o ujemnych wagach. Aby algorytm działał poprawnie modyfikowana jest funkcja wagowa w .

Nowa funkcja wagowa w' musi spełniać poniższe własności:

1. ścieżka P jest najkrótszym połączeniem pomiędzy wierzchołkami u, v przy funkcji wagowej w wtedy i tylko wtedy, gdy P jest najkrótszą ścieżką pomiędzy u i v z funkcją wagową w'
2. $w'(e) \geq 0$ dla każdej krawędzi e

Def. $w'((u, v)) = w((u, v)) + h(u) - h(v)$

Lemat Jeśli P jest ścieżką złożoną z v_0, \dots, v_k , to $w'(P) = w(P) + h(v_0) - h(v_k)$

Dowód:

$$\begin{aligned} w'(P) &= \sum_{i=1}^k w'(v_{i-1}, v_i) = \sum_{i=1}^k (w(v_{i-1}, v_i) + h(v_{i-1}) - h(v_i)) = \\ &= \sum_{i=1}^k w(v_{i-1}, v_i) + h(v_0) - h(v_k) = w(P) + h(v_0) - h(v_k) \end{aligned}$$

Zmiana funkcji wagowej

Lemat *Ścieżka P zawierająca wierzchołki v_0, \dots, v_k jest najkrótszą drogą z v_0 do v_k przy funkcji wagowej w wtedy i tylko wtedy, gdy P jest najkrótszą ścieżką z v_0 do v_k przy f-cji wagowej w' .*

Dowód: (\Rightarrow) (implikację przeciwną dowodzi się analogicznie)

Dowodzimy implikację metodą nie wprost. Załóżmy, że istnieje inna ścieżka P' taka, że $w'(P') < w'(P)$. Korzystając z poprzedniego lematu

$$w(P') + h(v_0) - h(v_k) = w'(P') < w'(P) = w(P) + h(v_0) - h(v_k).$$

To oznacza, że $w(P') < w(P)$ i prowadzi do sprzeczności.

Lemat *Graf G ma ujemny cykl przy funkcji wagowej w wtedy i tylko wtedy, gdy ma ujemny cykl przy funkcji w' .*

Dowód:

Niech C będzie dowolnym cyklem w G zawierającym wierzchołki $v_0, \dots, v_k = v_0$. Wówczas:

$$w'(C) = w(C) + h(v_0) - h(v_k) = w(C),$$

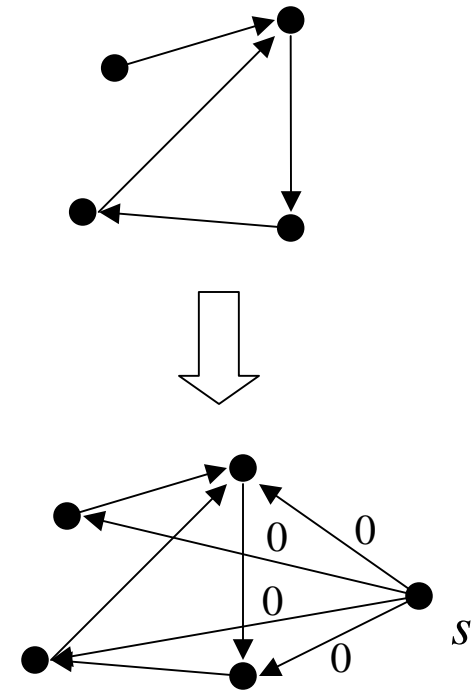
co kończy dowód.

Konstrukcja funkcji h

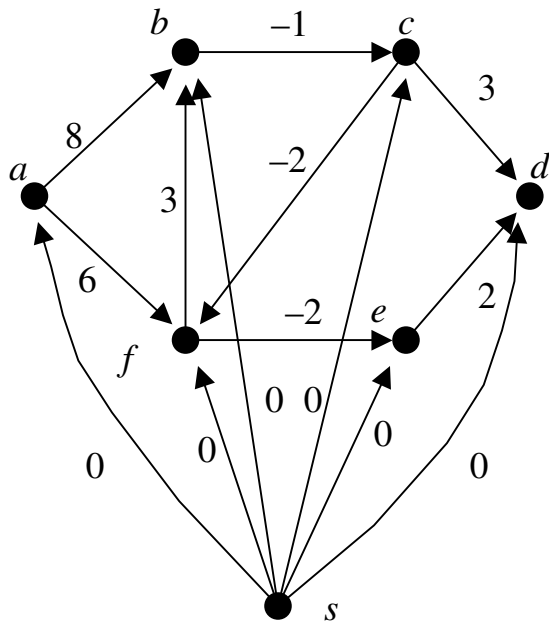
Uwaga Celem jest dobranie takiej funkcji h , aby $w'(e) \geq 0$ dla każdej krawędzi e . W tym celu tworzymy nowy graf G' powstały z G poprzez dodanie wierzchołka s , z którego wychodzi łuk o wadze 0 do każdego innego wierzchołka grafu G .

Uwagi:

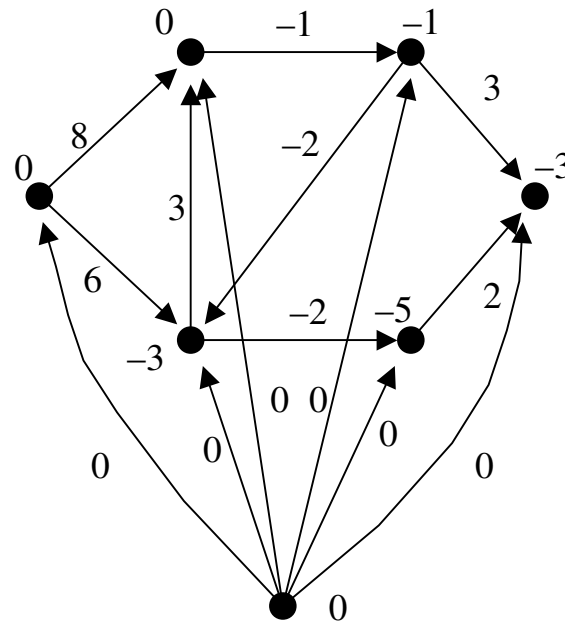
- G nie ma cykli o ujemnych wagach $\Leftrightarrow G'$ nie ma cykli o ujemnych wagach
- definiujemy, dla każdego wierzchołka v
 $h(v) = d(v)$ – długość najkrótszej ścieżki z s do v
- Własność: $h(v) \leq h(u) + w((u,v))$ dla wszystkich krawędzi (u,v)
- Definiujemy: $w'((u,v)) = w((u,v)) + h(u) - h(v) \geq 0$



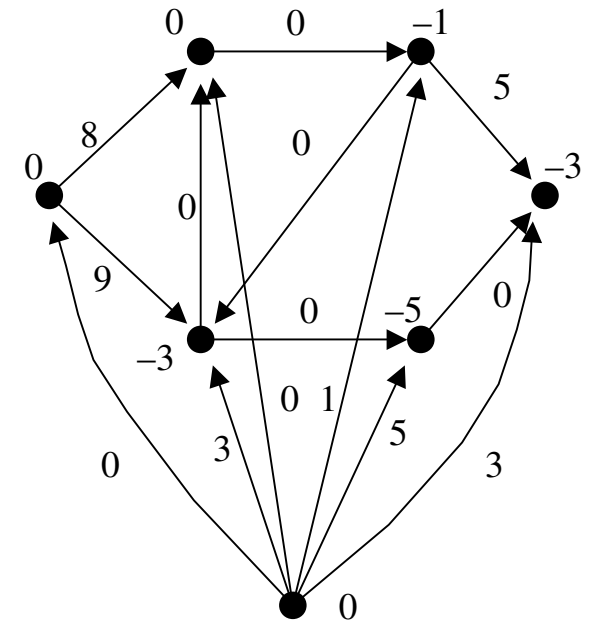
Przykład



Rys. 1: Utworzenie grafu G' poprzez dodanie wierzchołka s .



Rys. 2: Wyznaczenie długości najkrótszych ścieżek z s do pozostałych wierzchołków za pomocą alg. Bellmana-Forda.



Rys. 3: Utworzenie funkcji wagowej w' dla grafu G' wg wzoru:
 $w'((u,v)) = w((u,v)) + d(u) - d(v)$.

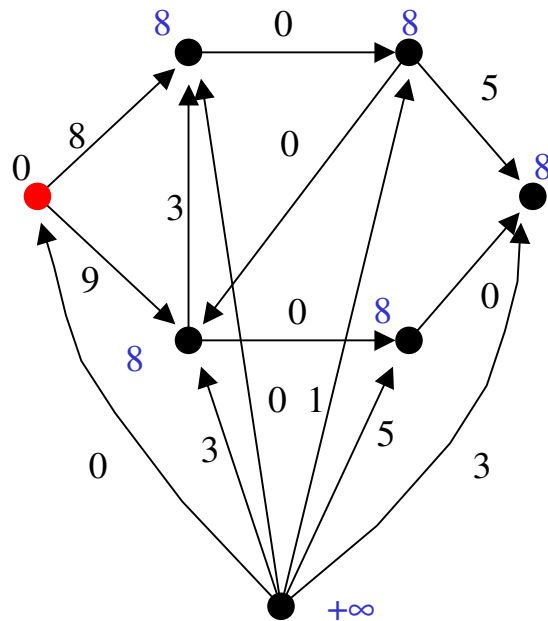
Algorytm Johnsona

```

procedure Johnson(  $G$  )
begin
  wyznacz graf  $G'$ ;
  if Bellamn-Ford(  $G'$ ,  $w$ ,  $s$  ) = false then
    return false;
  else begin
    for each  $v \in V(G')$  do
       $h(v) := d(v)$ ; /* obliczone w algorytmie Bellmana-Forda */
    for each  $(u,v) \in E(G')$  do
       $w'((u,v)) := w((u,v)) + h(u) - h(v)$ ;
    for each  $v \in V(G)$  do
      Dijkstra(  $G$ ,  $w'$ ,  $v$  );
    for each  $u \in V(G)$  do
       $d[u,v] := [\text{dł. najkr. ścieżki z } v \text{ do } u \text{ w } G'] + h(v) - h(u)$ ;
  end
end

```

Przykład c.d.



Wyznaczyliśmy wcześniej funkcję pomocniczą h :

	a	b	c	d	e	f	s
h	0	0	-1	-3	-5	-3	0

Skonstruujemy jeden wiersz tablicy wyjściowej, mianowicie $d[a,*]$, co odpowiada wykonaniu jednego obiegu odpowiedniej pętli alg. Johnsona:

- wierzchołek a – kolor czerwony
- niebieskie etykiety – najkrótsza droga z wierzchołka a
- obliczone wartości dla grafu G' :

	a	b	c	d	e	f	s
a	0	8	8	8	8	8	∞

- po przekształceniu odwrotnym wynik dla G :

	a	b	c	d	e	f
a	0	8	7	5	3	5