

Szukanie najkrótszych dróg z jednym źródłem

Algorytm Dijkstry

Założenia:

- dany jest spójny graf prosty G z wagami na krawędziach
- waga $w(e)$ dla każdej krawędzi e jest nieujemna
- dany jest wyróżniony wierzchołek s

Wyjście: dla każdego wierzchołka v algorytm znajduje $d[v]$, długość najkrótszej drogi z wierzchołka s do v . Przez długość drogi rozumiemy sumę wag krawędzi należących do drogi.

Uwaga: Algorytm można łatwo zmodyfikować tak, aby oprócz długości drogi wyznaczał krawędzie do niej należące oraz tak, aby działał w przypadku grafów skierowanych.

Pseudokod

Uwaga S jest pomocniczym zbiorem wierzchołków (nazywany *zbiorem pewności*). O wierzchołku v należącym do S wiadomo, że jego etykieta $d[v]$ jest równa długości najkrótszej drogi z s do v .

Procedure Dijkstra(G, s)

begin

$S := \emptyset$; $d[s] = 0$; $d[v] = +\infty$ dla $v \neq s$;

for $i := 1$ **to** n **do begin**

znajdź $v \in \mathcal{V} \setminus S$, posiadający minimalną etykietę $d[v]$;

$S := S \cup \{v\}$;

for każdy sąsiad $u \in \mathcal{V} \setminus S$ **do begin**

$d[u] := \min\{d[u], d[v] + w(\{u,v\})\}$;

end

end

end

Przykład

Procedure Dijkstra(G, s)

begin

$S := \emptyset$; $d[s] = 0$; $d[v] = +\infty$ dla $v \neq s$;

for $i := 1$ **to** n **do begin**

znajdź $v \in \mathbb{V}S$, o minimalnym $d[v]$;

$S := S \cup \{v\}$;

for każdy sąsiad $u \in \mathbb{V}S$ **do begin**

$d[u] := \min\{d[u], d[v] + w(\{u,v\})\}$;

end

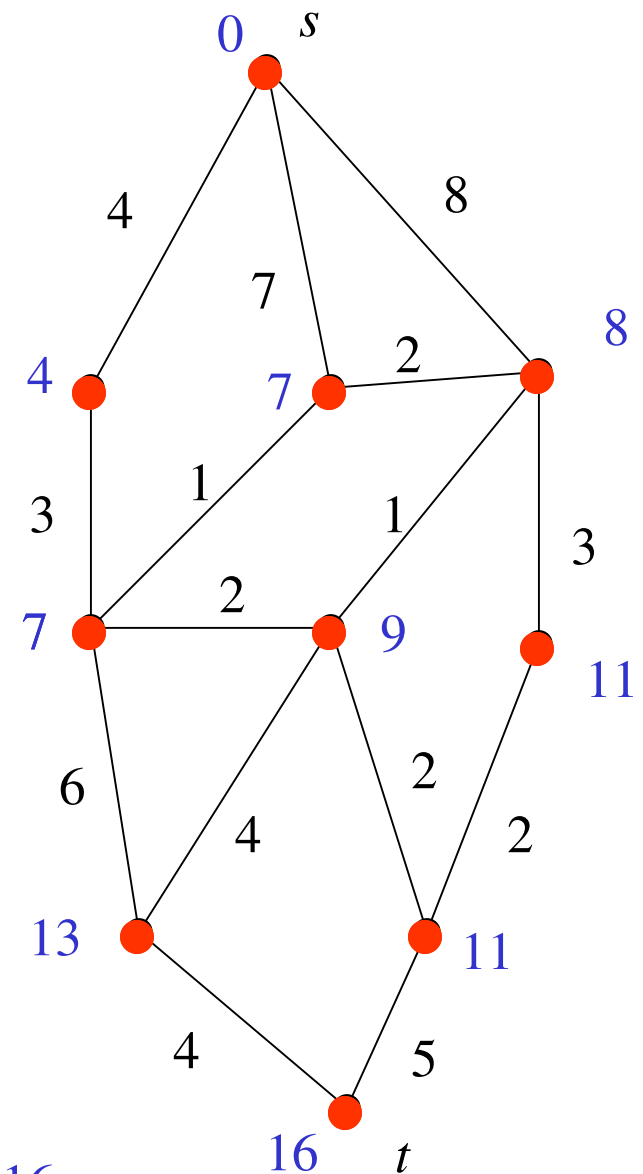
end

end

Szukamy długości najkrótszej drogi z s do t .

Elementy zbioru S oznaczamy kolorem czerwonym.

Odpowiedź: najkrótsza droga z s do t ma długość 16.



Uwagi o implementacji

- operacje „znajdź $v \in V \setminus S$, o minimalnym $d[v]$; $S := S \cup \{v\}$;" są wykonywane $O(n)$ razy
- operacja „ $d[u] := \min\{d[u], d[v] + w(\{u,v\})\}$;" jest wykonywana $O(m)$ razy
- do wydajnej implementacji wykorzystujemy kopiec binarny o tej własności, że klucz zapamiętany w danym węźle jest mniejszy od kluczy jego synów
- elementami kopca są wierzchołki $V \setminus S$, a klucze to liczby $d[v]$
- w ogólnym przypadku czas budowy kopca to $O(n \log n)$, lecz w algorytmie Dijkstry (wykorzystując fakt, że inicjalnie wszystkie klucze, z wyjątkiem $d[s]$ są identyczne) czas ten jest liniowy
- wyszukiwanie wierzchołka w zbiorze $V \setminus S$ o minimalnym kluczu, dzięki własności kopca, może być wykonana w czasie stałym
- usunięcie elementu o minimalnym kluczu (czyli instrukcja „ $S := S \cup \{v\}$;" wymaga czasu $O(\log(|V \setminus S|)) = O(\log n)$
- operacja „ $d[u] := \min\{d[u], d[v] + w(\{u,v\})\}$;" wymaga czasu $O(\log(|V \setminus S|))$
- ostatecznie, czas działania algorytmu Dijkstry to $O((n+m) \log n)$

Algorytm Bellmana-Forda

- Wejście: obciążony spójny digraf G .
- Dozwolone ujemne wagi na krawędziach.
- Algorytm stwierdza czy istnieje cykl o ujemnej sumie wag.
- Jeśli taki cykl istnieje, to algorytm zwraca informację o błędzie
- Jeśli powyższego cyklu nie ma, to algorytm znajduje długości najkrótszych dróg ze źródła s do wszystkich pozostałych wierzchołków grafu

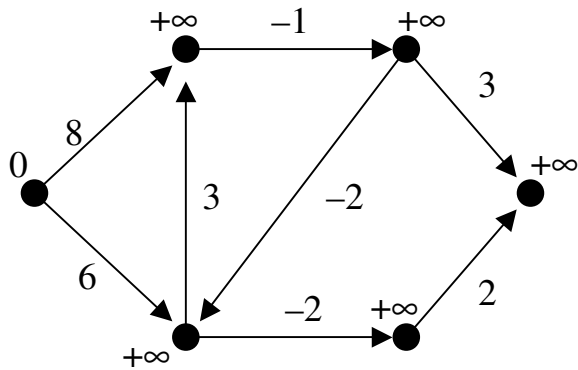
Szkic algorytmu

- algorytm szuka długości najkrótszych dróg z wyróżnionego wierzchołka s do wszystkich pozostałych wierzchołków grafu
- długość znalezionej drogi z s do v jest zapamiętana jako $d[v]$
- inicjalnie $d[s] = 0$ oraz $d[v] = +\infty$ dla $s \neq v$
- główna pętla algorytmu jest wykonywana $n - 1$ razy
- każdy przebieg głównej pętli polega na wykonaniu relaksacji każdej krawędzi
- relaksacja krawędzi (u, v) jest zmniejszeniem oszacowania $d[v]$ jeśli wartość dotychczas zapamiętana jest większa niż $d[u] + w((u, v))$, gdzie $w((u, v))$ to waga krawędzi (u, v)
- po zakończeniu obliczeń w głównej pętli algorytmu następuje sprawdzenie czy w digrafie istnieje cykl o ujemnej sumie wag, co jest realizowane poprzez sprawdzenie czy można dokonać relaksacji dowolnej krawędzi
- jeśli relaksacja jest możliwa, to graf zawiera cykl o ujemnej sumie wag, co oznacza, że najkrótszych dróg nie można obliczyć
- jeśli relaksacja nie jest możliwa, to zapamiętane wartości $d[v]$ są szukanymi długościami najkrótszych dróg z s

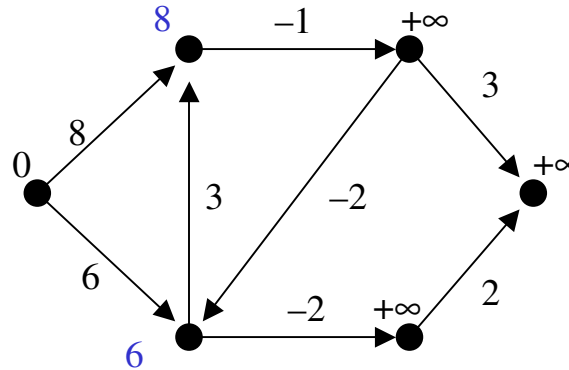
Pseudokod

```
procedure Bellman-Ford(  $G, w, s$  )  
begin  
     $d[s] = 0$ ;  
    for each  $v \in V(G) \setminus \{s\}$  do  
         $d[v] = +\infty$ ;  
    for  $i := 1$  to  $n - 1$  do  
        for each  $(u, v) \in E(G)$  do  
            if  $d[u] + w((u, v)) < d[v]$  then  
                 $d[v] := d[u] + w((u, v))$ ;  
        for each  $(u, v) \in E(G)$  do  
            if  $d[u] + w((u, v)) < d[v]$  then  
                return false;  
    return true;  
end
```

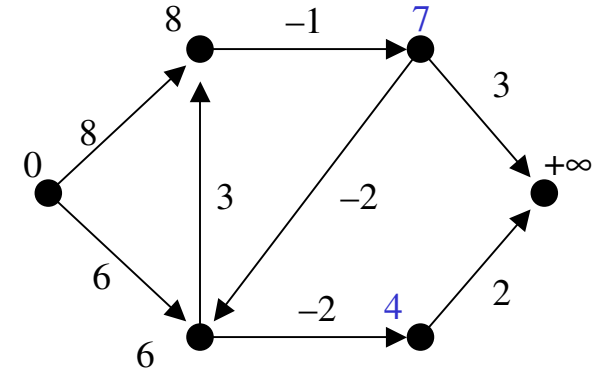
Przykład



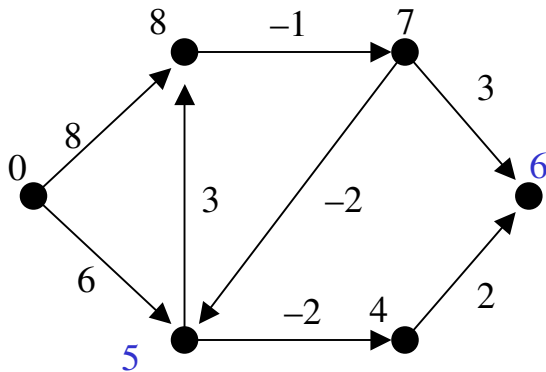
Rys. 1: Sytuacja po inicjalizacji.



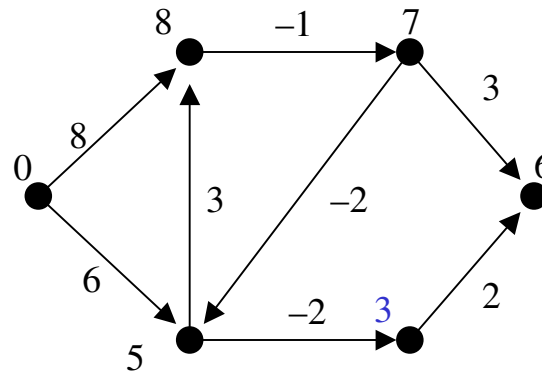
Rys. 2: Sytuacja po wykonaniu pętli dla $i = 1$



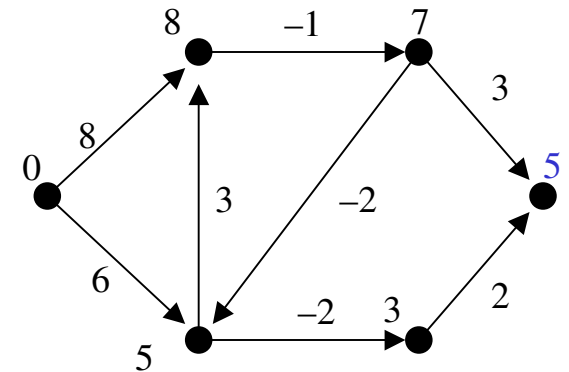
Rys. 3: Sytuacja po wykonaniu pętli dla $i = 2$



Rys. 4: Sytuacja po wykonaniu pętli dla $i = 3$



Rys. 5: Sytuacja po wykonaniu pętli dla $i = 4$



Rys. 6: Sytuacja po wykonaniu pętli dla $i = 5$

Poprawność

Tw *Jeśli digraf G nie posiada ujemnych cykli osiągalnych ze źródła s , to po zakończeniu algorytmu Bellmana-Forda $d[v]$ jest równe długości najkrótszej ścieżki z s do v dla każdego wierzchołka v osiągalnego z s .*

Dowód:

- niech $s=v_0, \dots, v_k=v$ będzie najkrótszą ścieżką z s do v
- mamy $k < |V(G)|$
- dowodzimy indukcyjnie, że $d[v_i]$ jest równe długości najkrótszej ścieżki z s do v_i po i -tym przebiegu „drugiej” pętli for
- jeśli $i=0$, to własność wynika z faktu, że w fazie inicjalizacji algorytmu podstawiamy $d[s]=0$ i równość ta nie ulega zmianie podczas działania algorytmu
- niech $i > 0$. Po dokonaniu relaksacji krawędzi (v_{i-1}, v_i) w i -tym przebiegu pętli liczba $d[v_i]$ przyjmuje wartość najkrótszej ścieżki z s do v_i .

Poprawność

Tw. *Jeśli digraf G zawiera cykl o ujemnej wadze osiągalny z s , to algorytm Bellmana-Forda zwraca wartość *false*.*

Dowód:

- niech $v_0, \dots, v_k = v_0$ będzie cyklem o ujemnej sumie wag, tzn.

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

- Gdyby algorytm zwrócił wartość *true*, to dla każdego wierzchołka cyklu mamy $d[v_i] \leq d[v_{i-1}] + w((v_{i-1}, v_i))$.
- Sumując nierówności parami otrzymujemy

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

- Wartości pierwszych dwóch sum w powyższym wyrażeniu są sobie równe więc

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

co prowadzi do sprzeczności.