Politechnika Wrocławska

Algorytmy wyznaczania najkrótszych ścieżek

w rzeczywistych sieciach drogowych

Tomasz Strzałka
Wydział Podstawowych Problemów Techniki

14 listopada 2014

Czy jest najkrótsza ścieżka?

Najkrótszą ścieżką nazywamy...

...taką ścieżkę $p = \langle v_0, v_1, \cdots, v_k \rangle$ z wierzchołka v_0 do v_k, że dla każdej pary $(i, j) : 1 \leq i \leq j \leq k$ ścieżka $p' = \langle v_i, v_{i+1}, \cdots, v_j \rangle$ jest najkrótszą ścieżką.

- Własność optymalnej podstruktury

W szczególności:

jeśli istnieje ścieżka $p^{(0)} = \langle v_0, v_1, \cdots, v_k \rangle$ i jest ona najkrótsza to rekurencyjnie ścieżka $p^{(1)} = \langle v_0, v_1, \cdots, v_{k-1} \rangle$ także ma tę własność.

Czy jest najkrótsza ścieżka?

Najkrótszą ścieżką nazywamy...

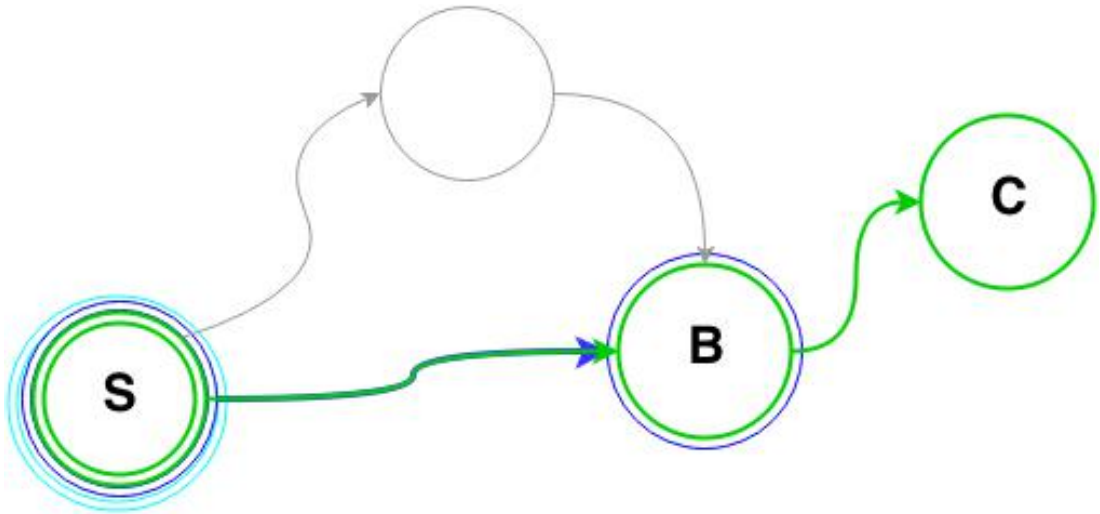
...taką ścieżkę $p = \langle v_0, v_1, \cdots, v_k \rangle$ z wierzchołka v_0 do v_k, że dla każdej pary $(i, j) : 1 \leq i \leq j \leq k$ ścieżka $p' = \langle v_i, v_{i+1}, \cdots, v_j \rangle$ jest najkrótszą ścieżką.

- Własność optymalnej podstruktury

W szczególności:

jeśli istnieje ścieżka $p^{(0)} = \langle v_0, v_1, \cdots, v_k \rangle$ i jest ona najkrótsza to rekurencyjnie ścieżka $p^{(1)} = \langle v_0, v_1, \cdots, v_{k-1} \rangle$ także ma tę własność.

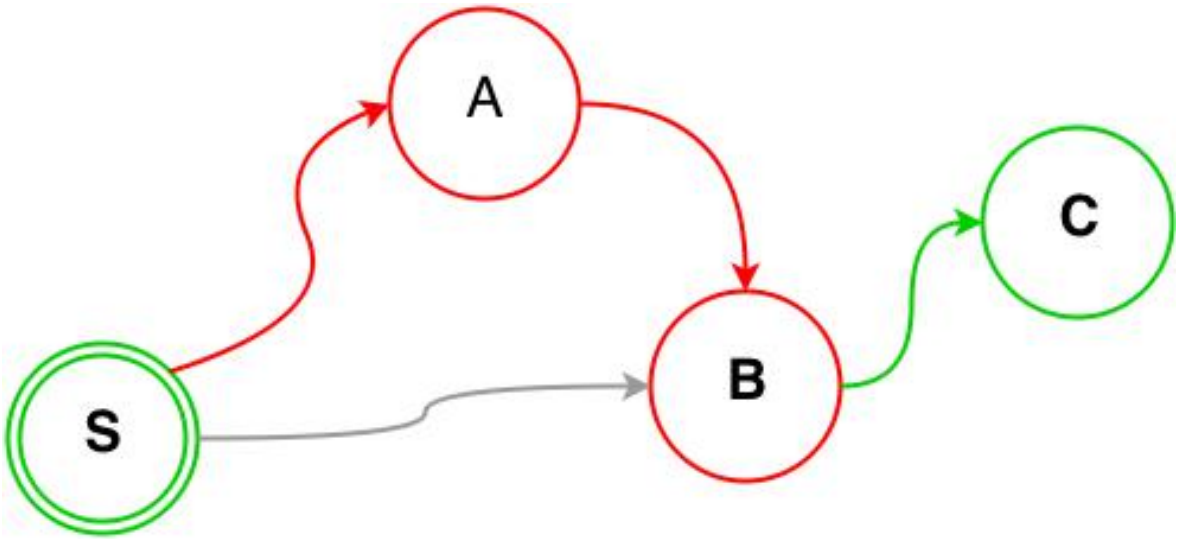
Czy jest najkrótsza ścieżka?



Opis

Najkrótsza ścieżka: $p^{(0)} = \langle S, B, C \rangle$. Najkrótszą ścieżką jest zatem także $p^{(1)} = \langle S, B \rangle$ oraz $p^{(2)} = \langle S \rangle$.

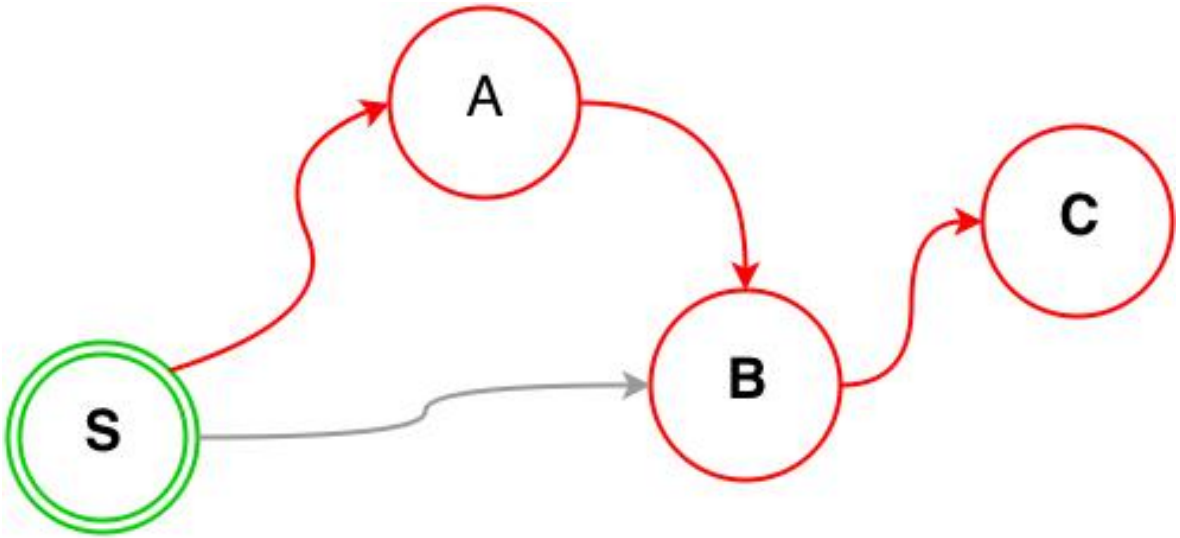
Czy jest najkrótsza ścieżka?



Opis

Najkrótszą ścieżką nie jest ścieżka: $p^{(0)} = \langle S, A, B \rangle$ zatem...

Czy jest najkrótsza ścieżka?

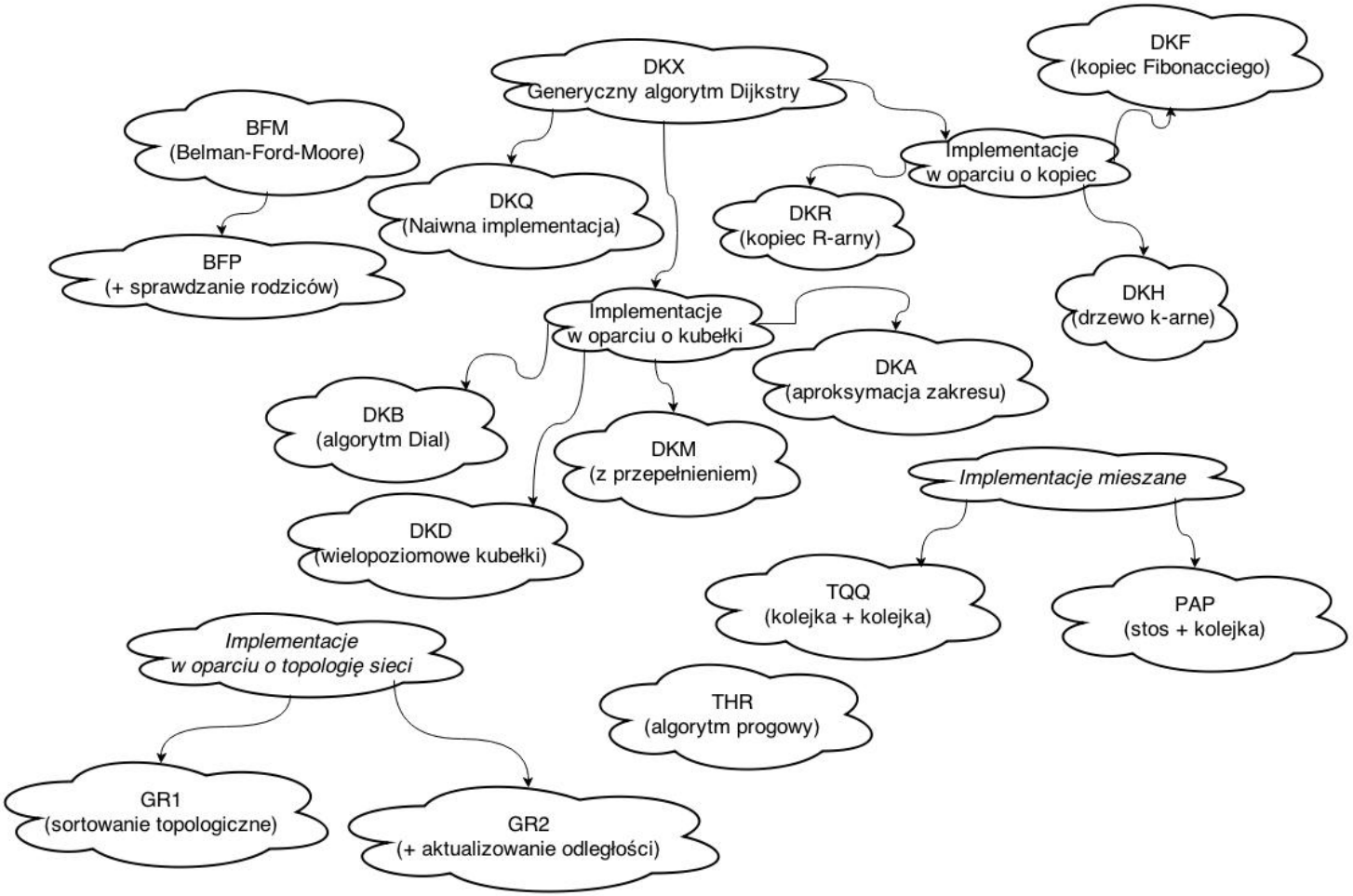


Opis

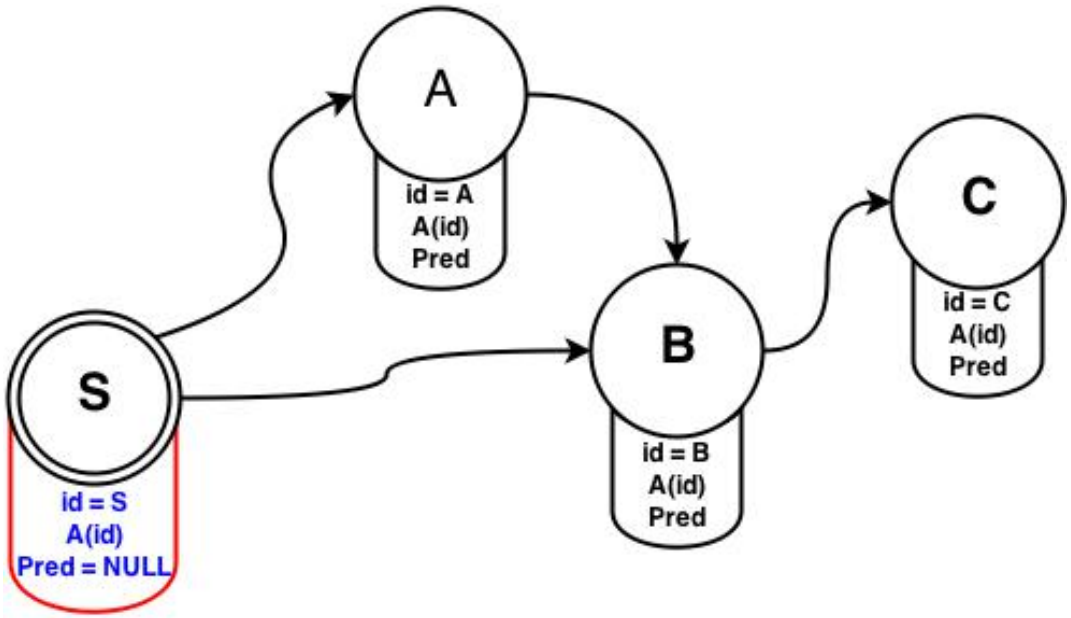
... nie jest nią także ścieżka: $p^{(0)} = \langle S, A, B, C \rangle$.

Jak rozwiązać problem...

...najkrótszych ścieżek?



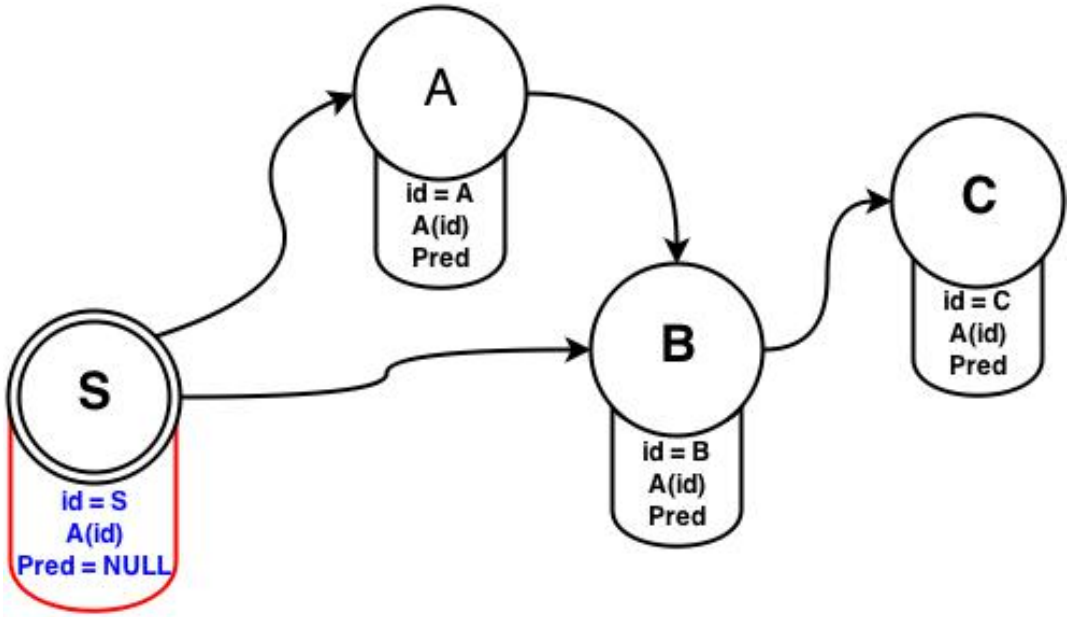
Potrzebne oznaczenia



Opis

- *id* - identyfikator,

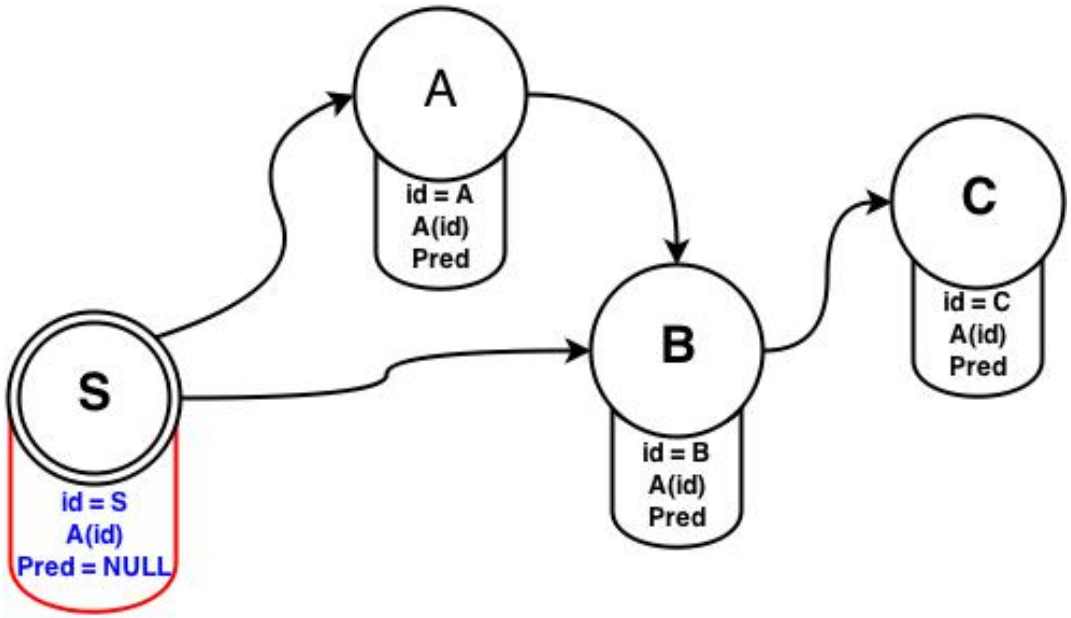
Potrzebne oznaczenia



Opis

- $A(id)$ - zbiór $\{i : v_{id} \rightsquigarrow i\}$,

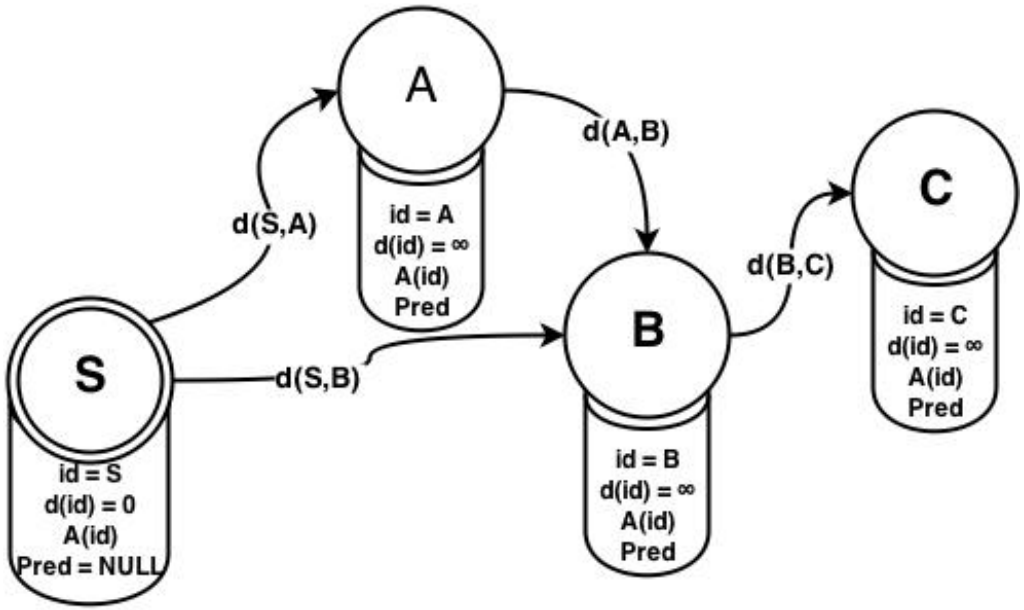
Potrzebne oznaczenia



Opis

- *Pred* - poprzedzający węzeł w najkrótszej ścieżce.

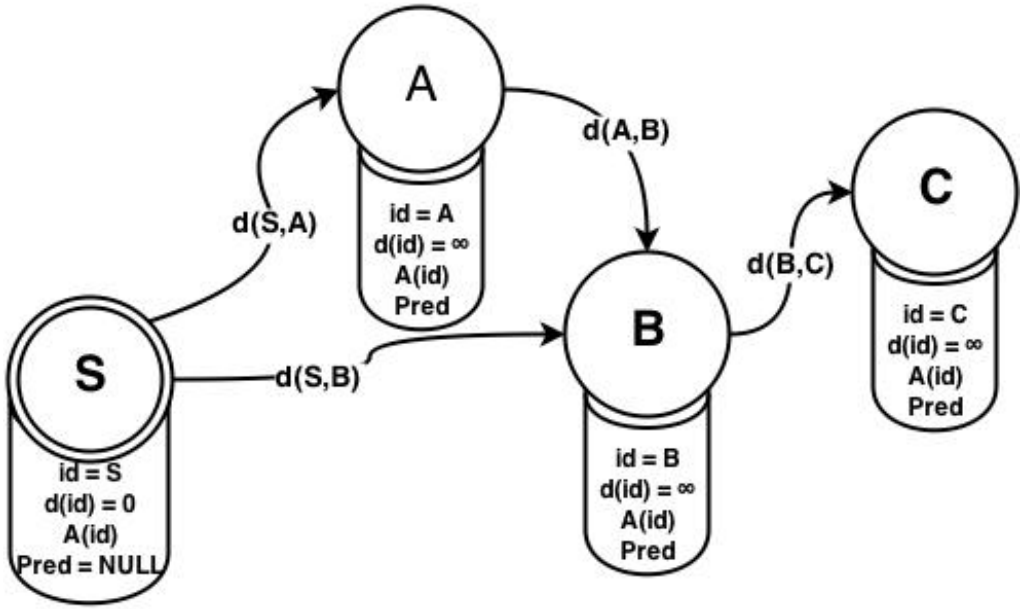
Potrzebne oznaczenia



Opis

- $d(i,j)$ - koszt przebycia drogi z V_i do V_j.

Potrzebne oznaczenia



Opis

- $d\,(id)$ - górne ograniczenie na koszt ścieżki od V_S do V_{id}.

Potrzebne oznaczenia

Właściwości odległości:

- $d(id) = \begin{cases} 0 & id = S \\ \infty & id \neq S \end{cases}$
- Jeśli ścieżka $p = \langle v_0, v_1, \cdots, v_k \rangle$ jest najkrótsza wtedy dla każdego węzła v_i w tej ścieżce:
  - $d(i) = \sum_{j=0}^{i-1} d(j, j+1)$

Sens $d(id) = \infty$

- Relaksacja wierzchołków

Potrzebne oznaczenia

Właściwości odległości:

- $d(id) = \begin{cases} 0 & id = S \\ \infty & id \neq S \end{cases}$
- Jeśli ścieżka $p = \langle v_0, v_1, \cdots, v_k \rangle$ jest najkrótsza wtedy dla każdego węzła v_i w tej ścieżce:
  - $d(i) = \sum_{j=0}^{i-1} d(j, j+1)$

Sens $d(id) = \infty$

- Relaksacja wierzchołków

Relaksacja wierzchołków

Operacja relaksacji:

Jeśli jesteśmy w stanie znaleźć taką ścieżkę $p = \langle v_0, v_1, \cdots, v_k \rangle$, że $d\left(v_k^{ID}\right) + d\left(k, i\right) < d\left(v_i^{ID}\right)$ to znaczy, że do $d\left(v_i^{ID}\right)$ istnieje krótsza ścieżka, niż wyliczona dotychczas. W takim przypadku aktualizuj $d\left(v_i^{ID}\right)$ i ustaw $Pred\left(v_i^{ID}\right) = v_k$.

Sens $d\left(id\right) = \infty$

Przez powyższe równanie wyrażamy fakt, że nie znamy jeszcze drogi do wierzchołka o identyfikatorze id (jego najkrótsza ścieżka ma koszt nieskończony - nigdy do danego węzła nie dojdziemy na podstawie posiadanych informacji).

Relaksacja wierzchołków

Operacja relaksacji:

Jeśli jesteśmy w stanie znaleźć taką ścieżkę $p = \langle v_0, v_1, \cdots, v_k \rangle$, że $d\left(v_k^{ID}\right) + d\left(k, i\right) < d\left(v_i^{ID}\right)$ to znaczy, że do $d\left(v_i^{ID}\right)$ istnieje krótsza ścieżka, niż wyliczona dotychczas. W takim przypadku aktualizuj $d\left(v_i^{ID}\right)$ i ustaw $Pred\left(v_i^{ID}\right) = v_k$.

Sens $d\left(id\right) = \infty$

Przez powyższe równanie wyrażamy fakt, że nie znamy jeszcze drogi do wierzchołka o identyfikatorze id (jego najkrótsza ścieżka ma koszt nieskończony - nigdy do danego węzła nie dojdziemy na podstawie posiadanych informacji).

Algorytm Bellmana - Forda - Moore'a

Idea:

Wykonać wielokrotnie relaksację dla wszystkich wierzchołków (jeśli konieczne).

Przykład:

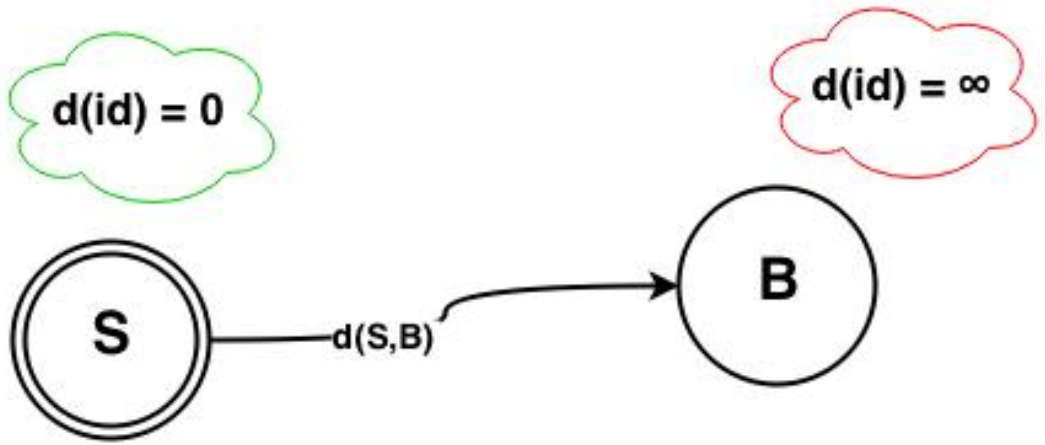
Nie mamy czasu na przykłady.

Algorytm Bellmana - Forda - Moore'a

Czemu $|V| - 1$?

Mniej niż $|V| - 1$

- Niech $|V| = 2$
- Algorytm nie wykona żadnych iteracji!



Algorytm Bellmana - Forda - Moore'a

Czemu $|V| - 1$?

Więcej niż $|V| - 1$

- Każda iteracja próbuje przeprowadzić relaksacje każdego z węzłów sieci,
- najkrótsza ścieżka w sieci o $|V| = n$ wierzchołkach ma maksymalnie n wierzchołków składowych,
- węzeł, będący źródłem, "już jest zrelaksowany".

Więcej niż $|V| - 1$

Jeśli po wykonaniu $|V| - 1$ iteracji nadal jest możliwa do wykonania operacja relaksacji znaczy to, że w podanej sieci istnieje cykl o ujemnej długości.

Algorytm Bellmana - Forda - Moore'a

Czemu $|V| - 1$?

Więcej niż $|V| - 1$

- Każda iteracja próbuje przeprowadzić relaksacje każdego z węzłów sieci,
- najkrótsza ścieżka w sieci o $|V| = n$ wierzchołkach ma maksymalnie n wierzchołków składowych,
- węzeł, będący źródłem, "już jest zrelaksowany".

Więcej niż $|V| - 1$

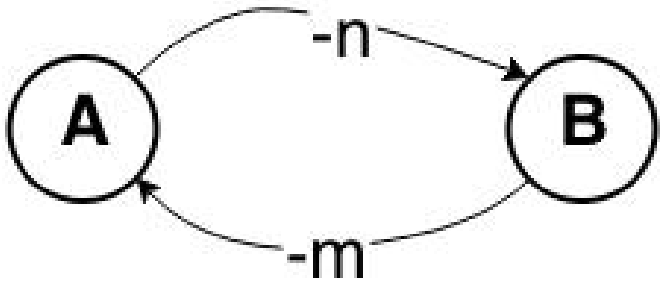
Jeśli po wykonaniu $|V| - 1$ iteracji nadal jest możliwa do wykonania operacja relaksacji znaczy to, że w podanej sieci istnieje cykl o ujemnej długości.

Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,

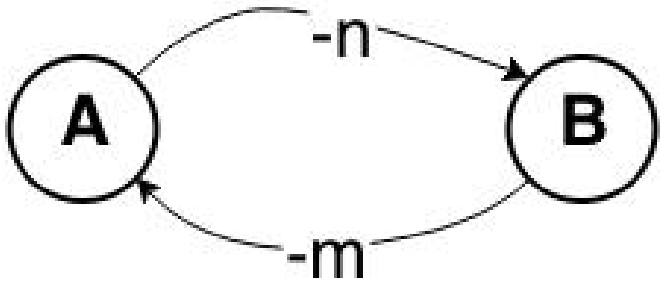


Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,

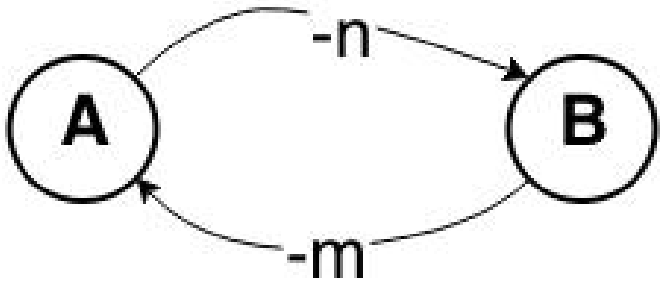


Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,

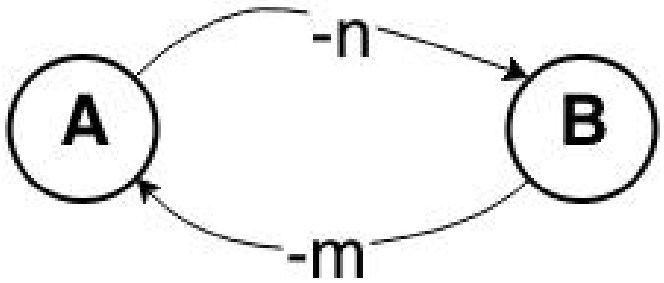


Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,

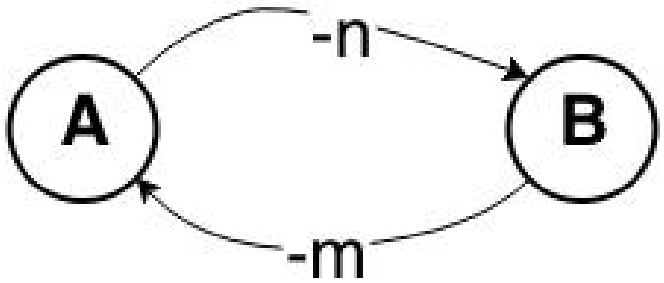


Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,

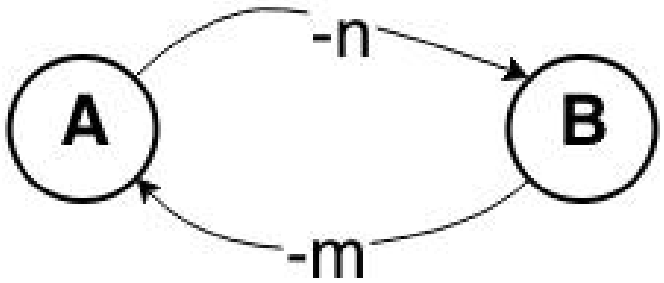


Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,

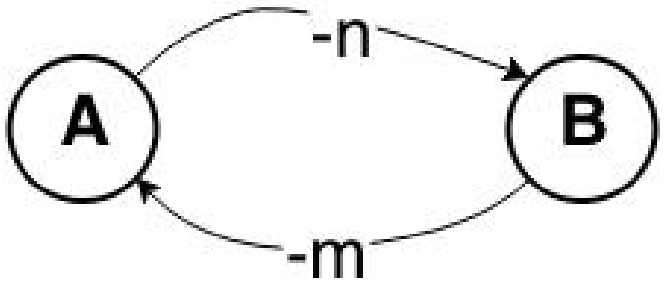


Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,

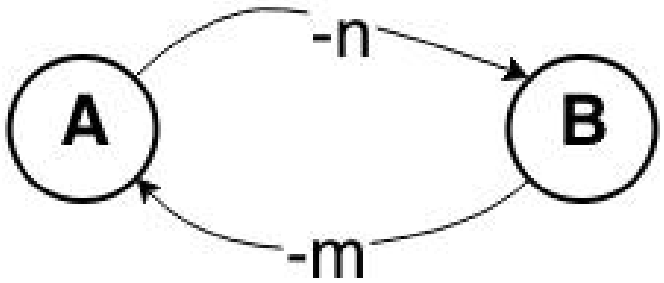


Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,

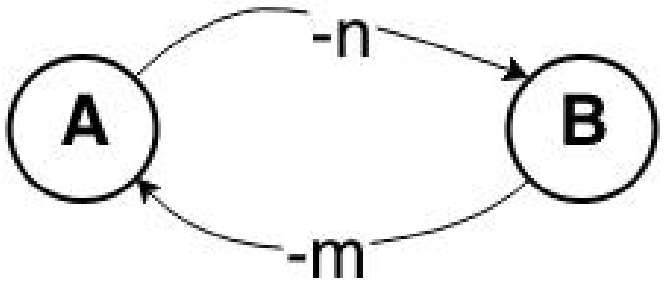


Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,

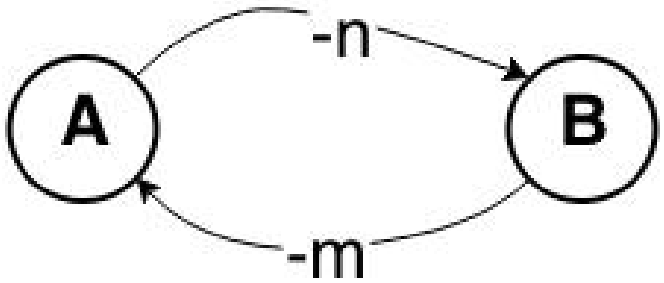


Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,

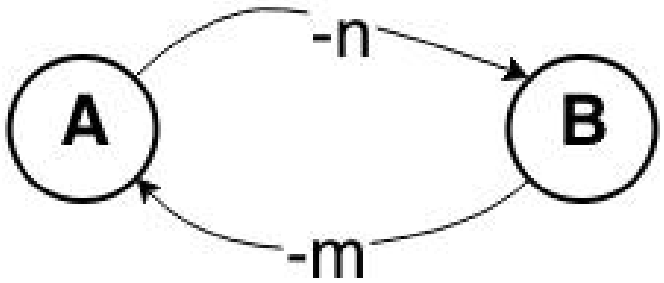


Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,

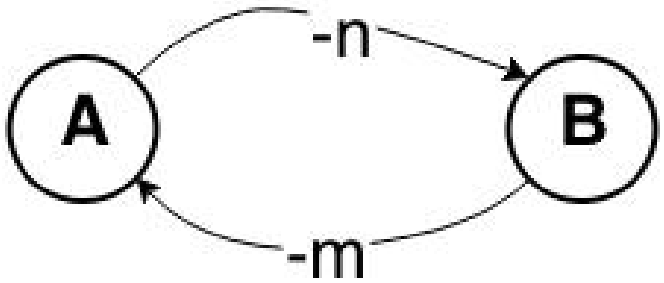


Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,

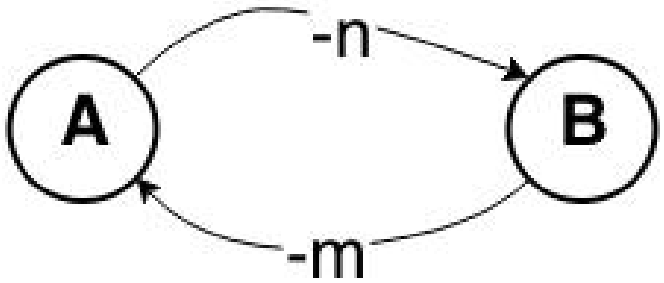


Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,

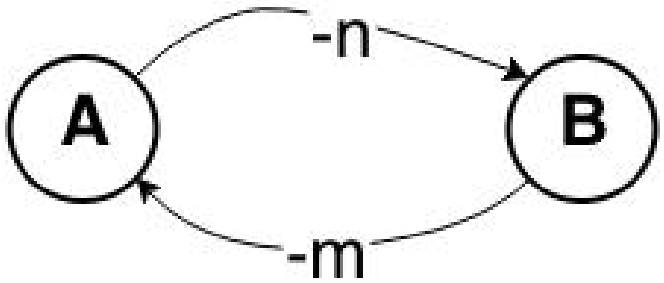


Algorytm Bellmana - Forda - Moore'a

Relaksacja w "ujemnym cyklu"

Dla dowolnie dużej liczby iteracji:

- A jest ostatnim węzłem na "najkrótszej ścieżce". B posiada $d\left(v_B^{ID}\right)$ większe o wyrażenie $m+n$ (ustalone w poprzednim cyklu). Następuje relaksacja B.,
- po relaksacji B, jest ono ostatnim węzłem na "najkrótszej ścieżce". A posiada $d\left(v_A^{ID}\right)$ większe o wyrażenie $n+m$ (ustalone w poprzednim cyklu). Następuje relaksacja A.,



Generyczny algorytm Dijkstry

Pseudokod

Algorithm 1: $DIJKSTRA(G, w, s)$

begin

$G =$ ustaw $d(i) = \infty$ oraz $Pred(i) = NULL$ dla każdego węzła o $id = i$. Dodatkowo dla źródła $d(i) = 0$.

$S = \emptyset$

$Q = G.V$

while $Q \neq \emptyset$ **do**

usuń z Q element o najniższym $d(i)$ i dodaj go do S

for *każdy wierzchołek* $v \in A(i)$ **do**

wykonaj relaksację

Generyczny algorytm Dijkstry

Przykład

Nie mamy czasu na przykłady :<

Wpływ struktury

na efektywność algorytmu

Złożoność (generyczny algorytm Dijkstry)

- inicjalizujemy $|V|$ wierzchołków,
- usuwamy łącznie $|V|$ wierzchołków,
- wykonujemy relaksację dla $|E|$ krawędzi (każdy wierzchołek odwiedzamy tylko raz),

Zatem:

- $|V| \cdot O(INIT(?))$,
- $+ |V| \cdot O(EXTRACT\_MIN(Q))$,
- $+ |E| \cdot O(UPDATE(?))$

Wpływ struktury

na efektywność algorytmu

Złożoność (generyczny algorytm Dijkstry)

- inicjalizujemy $|V|$ wierzchołków,
- usuwamy łącznie $|V|$ wierzchołków,
- wykonujemy relaksację dla $|E|$ krawędzi (każdy wierzchołek odwiedzamy tylko raz),

Zatem:

- $|V| \cdot O(INIT(?))$,

$+ \; |V| \cdot O(EXTRACT\_MIN(Q))$,

$+ \; |E| \cdot O(UPDATE(?))$

Wpływ struktury

na efektywność algorytmu

Złożoność (generyczny algorytm Dijkstry)

- inicjalizujemy $|V|$ wierzchołków,
- usuwamy łącznie $|V|$ wierzchołków,
- wykonujemy relaksację dla $|E|$ krawędzi (każdy wierzchołek odwiedzamy tylko raz),

Zatem:

- $|V| \cdot O(INIT(?))$,
- $+ \; |V| \cdot O(EXTRACT\_MIN(Q))$,
- $+ \; |E| \cdot O(UPDATE(?))$

Wpływ struktury

na efektywność algorytmu

Złożoność (generyczny algorytm Dijkstry)

- inicjalizujemy $|V|$ wierzchołków,
- usuwamy łącznie $|V|$ wierzchołków,
- wykonujemy relaksację dla $|E|$ krawędzi (każdy wierzchołek odwiedzamy tylko raz),

Zatem:

- $|V| \cdot O(INIT(?))$,
- $+ |V| \cdot O(EXTRACT\_MIN(Q))$,
- $+ |E| \cdot O(UPDATE(?))$

Tablice, listy, stosy, kolejki, kubełki...

sposoby podejścia do problemu SSSP

Tablice, listy, stosy, kolejki, kubełki...

sposoby podejścia do problemu SSSP

Złożoność (generyczny algorytm Dijkstry)

$$|V| \cdot [O(INSERT(Struct, Node)) + O(EXTRACT\_MIN(Struct))] + |E| \cdot O(UPDATE\_STRUCT(Struct, Node))$$

	Tablica	Lista	Stos	k. R-arny	k. Fibonacciego
I	$O(n)$	$O(1)$	$O(1)$	$O(\log_R(n))$	$O(1)$
E	$O(n)$	$O(1)$	$O(1)$	$O(R \cdot \log_R(n))$	$O(\log(n))$
U	$O(n)$	$O(n)$	$O(n)$	$O(\log_R(n))$	$O(1)$

Koszty amortyzacyjne

czyli jak oszukać przeznaczenie

	Sortowanie Szybkie		kopiec Fibonacciego	
Analiza:	WCA	randomizacja	WCA	amortyzacyjna
S	$O(n^2)$	$O(n \cdot log(n))$	—	$O(n \cdot log(n))$
D	—	—	$O(n \cdot log(n))$	$O(n \cdot 1)$

Struktura sieci

Wpływ struktury sieci na czas działania

Oprócz samego algorytmu, struktur danych w nim wykorzystanych, ważnym elementem, wpływającym na zachowywanie się implementacji, jest także sposób reprezentacji samej sieci.

Struktura sieci

Reprezentacje

- Macierz incydencji
  - węzeł-węzeł
  - węzeł-krawędź
- listy sąsiedztwa
  - pośrednie
  - bezpośrednie
- Star Representation
  - forward
  - reverse

Struktura sieci

Reprezentacje

- Macierz incydencji
  - węzeł-węzeł
  - węzeł-krawędź
- listy sąsiedztwa
  - pośrednie
  - bezpośrednie
- Star Representation
  - forward
  - reverse

Struktura sieci

Reprezentacje

- Macierz incydencji
  - węzeł-węzeł
  - węzeł-krawędź
- listy sąsiedztwa
  - pośrednie
  - bezpośrednie
- Star Representation
  - forward
  - reverse

Struktura sieci

Reprezentacje

- Macierz incydencji
  - węzeł-węzeł
  - węzeł-krawędź
- listy sąsiedztwa
  - pośrednie
  - bezpośrednie
- Star Representation
  - forward
  - reverse

Struktura sieci

Reprezentacje

- Macierz incydencji
  - węzeł-węzeł
  - węzeł-krawędź
- listy sąsiedztwa
  - pośrednie
  - bezpośrednie
- Star Representation
  - forward
  - reverse

Struktura sieci

Reprezentacje

- Macierz incydencji
  - węzeł-węzeł
  - węzeł-krawędź
- listy sąsiedztwa
  - pośrednie
  - bezpośrednie
- Star Representation
  - forward
  - reverse

Struktura sieci

Reprezentacje

- Macierz incydencji
  - węzeł-węzeł
  - węzeł-krawędź
- listy sąsiedztwa
  - pośrednie
  - bezpośrednie
- Star Representation
  - forward
  - reverse

Struktura sieci

Reprezentacje

- Macierz incydencji
  - węzeł-węzeł
  - węzeł-krawędź
- listy sąsiedztwa
  - pośrednie
  - bezpośrednie
- Star Representation
  - forward
  - reverse

Struktura sieci

Reprezentacje

- Macierz incydencji
  - węzeł-węzeł
  - węzeł-krawędź
- listy sąsiedztwa
  - pośrednie
  - bezpośrednie
- Star Representation
  - forward
  - reverse

Struktura sieci

Reprezentacje

- Macierz incydencji
  - węzeł-węzeł
  - węzeł-krawędź
- listy sąsiedztwa
  - pośrednie
  - bezpośrednie
- Star Representation
  - forward
  - reverse

Opisy struktur reprezentacji sieci

Listy sąsiedztwa

Motywacja

Najbardziej intuicyjna, szybka, niewiele ustępuje złożonością FSR:

- pamięciową (bogata struktura danych)
- obliczeniową (czas operacji na listach)

Motywacja

Ustępuje funkcjonalnością cF&RSR (compact Forward and Revesre Star Representation):

- brak listy poprzedników

Opisy struktur reprezentacji sieci

Listy sąsiedztwa

Motywacja

Najbardziej intuicyjna, szybka, niewiele ustępuje złożonością FSR:

- pamięciową (bogata struktura danych)
- obliczeniową (czas operacji na listach)

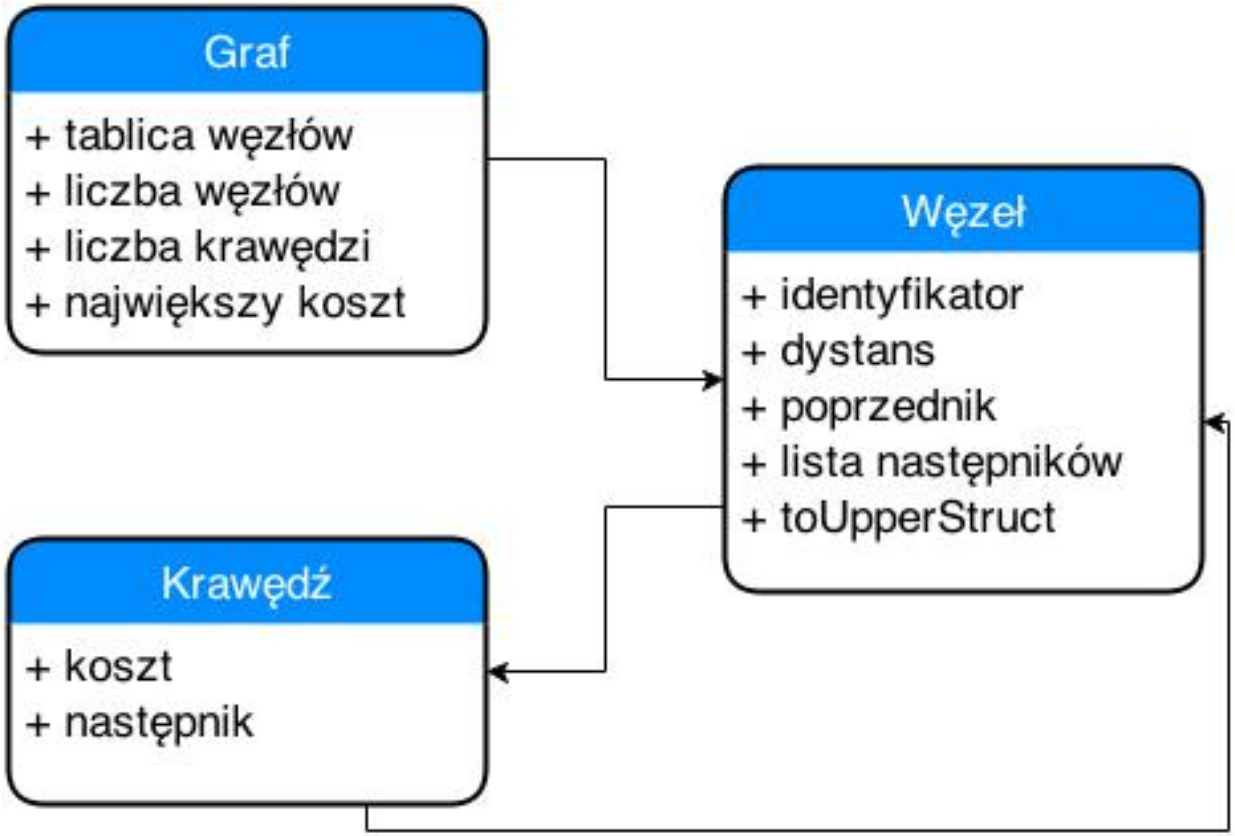
Motywacja

Ustępuje funkcjonalnością cF&RSR (compact Forward and Revesre Star Representation):

- brak listy poprzedników

Opisy struktur reprezentacji sieci

Listy sąsiedztwa



Algorytm sortowania topologicznego

Jak posortować topologicznie?

Motywacja

Otrzymując węzły grafu w porządku topologicznym mamy pewność, że dla każdego takiego wierzchołka wymagana jest co najwyżej jedna operacja relaksacji

- wyszukiwanie najkrótszej ścieżki w czasie liniowym

Algorytm sortowania topologicznego

Jak posortować topologicznie?

Kroki

- zaczynamy od źródła
- usuwamy krawędzie, po których dotarliśmy do następnego węzła.
  - Jeśli lista poprzedników węzła jest pusta to dodaj węzeł na koniec, posortowanej topologiczne, listy
  - w przeciwnym przypadku skanuj dalej
- Jeśli przeszedłeś po wszystkich wierzchołkach, a w grafie nadal są krawędzie - istnieje cykl

Algorytm sortowania topologicznego

Jak posortować topologicznie?

Kroki

- zaczynamy od źródła
- usuwamy krawędzie, po których dotarliśmy do następnego węzła.
  - Jeśli lista poprzedników węzła jest pusta to dodaj węzeł na koniec, posortowanej topologiczne, listy
  - w przeciwnym przypadku skanuj dalej
- Jeśli przeszedłeś po wszystkich wierzchołkach, a w grafie nadal są krawędzie - istnieje cykl

Algorytm sortowania topologicznego

Jak posortować topologicznie?

Kroki

- zaczynamy od źródła
- usuwamy krawędzie, po których dotarliśmy do następnego węzła.
  - Jeśli lista poprzedników węzła jest pusta to dodaj węzeł na koniec, posortowanej topologiczne, listy
  - w przeciwnym przypadku skanuj dalej
- Jeśli przeszedłeś po wszystkich wierzchołkach, a w grafie nadal są krawędzie - istnieje cykl

Algorytm sortowania topologicznego

Jak posortować topologicznie?

Kroki

- zaczynamy od źródła
- usuwamy krawędzie, po których dotarliśmy do następnego węzła.
  - Jeśli lista poprzedników węzła jest pusta to dodaj węzeł na koniec, posortowanej topologiczne, listy
  - w przeciwnym przypadku skanuj dalej
- Jeśli przeszedłeś po wszystkich wierzchołkach, a w grafie nadal są krawędzie - istnieje cykl

Algorytm sortowania topologicznego

Jak posortować topologicznie?

Kroki

- zaczynamy od źródła
- usuwamy krawędzie, po których dotarliśmy do następnego węzła.
  - Jeśli lista poprzedników węzła jest pusta to dodaj węzeł na koniec, posortowanej topologiczne, listy
  - w przeciwnym przypadku skanuj dalej
- Jeśli przeszedłeś po wszystkich wierzchołkach, a w grafie nadal są krawędzie - istnieje cykl

Algorytm sortowania topologicznego

Jak posortować topologicznie?

Kroki

- zaczynamy od źródła
- usuwamy krawędzie, po których dotarliśmy do następnego węzła.
  - Jeśli lista poprzedników węzła jest pusta to dodaj węzeł na koniec, posortowanej topologiczne, listy
  - w przeciwnym przypadku skanuj dalej
- Jeśli przeszedłeś po wszystkich wierzchołkach, a w grafie nadal są krawędzie - istnieje cykl

Algorytm sortowania topologicznego

Jak posortować topologicznie?

Problemy

- brak list poprzedników w wybranej reprezentacji
- graf nie może mieć cykli

Najszybsze algorytmy SSSP

Implementacje

- z dwoma kolejkami (TQQ)
- Dijkstry z kubełkami:
  - aproksymacyjnymi (DKA)
  - wielopoziomowymi (DKD - dwu)

Najszybsze algorytmy SSSP

Algorytm Dijkstry z kubełkami - aproksymacyjny

Implementacje

Szkic na tablicy, jak będzie czas :

Najszybsze algorytmy SSSP

Algorytm Dijkstry z kubełkami - dwupoziomowy

Implementacje

Szkic na tablicy, jak będzie czas :

Biblioteka Take Me Home

API

Wymagania:

- pliki z danymi zgodne z formatem narzuconym podczas 9th DIMACS Implementation Challenge
- wierzchołki numerowane od 1

Podstawowe API

- logi oparte na idei biblioteki Log4j
- podstawowe struktury:
  - TMHConfig - konfiguracja algorytmów
  - TMH - konfiguracja biblioteki
- pseudo-obiektowość

Biblioteka Take Me Home

API

TMHConfig

- createTMHConfig(ścieżka do pliku z poleceniami)
  - createTMHConfig(USA-road-d.USA.ss);
- setAllowInterrupt(config,false);
- setCheckConfig(config,false);
- setGraphOrder(config,NONE);
  - NONE - z typu wyliczeniowego GraphOrder
- setGraphStruct(config,ADJACENCY_LIST);
  - ADJACENCY_LISTNE - z typu wyliczeniowego GraphStructAbbreviation

Biblioteka Take Me Home

API

TMHConfig

- setAlgorithm(config,BFM);
  - BFM - z typu wyliczeniowego AlgorithmAbbreviation

TMH_API

- ins = createTMHAlgorithmInstance(config,"*.gr");
- runTMHAlgorithm(config->algorithm,ins);
- destroyTMHAlgorithmInstancje(alg,ins,false);
  - false - czy zresetować kofiguracje