Algorytm Dijkstry

Do czego służy:

-do znajdowania najkrótszej ścieżki z pojedynczego wierzchołka do innych wierzchołków w grafie o krawędziach z nieujemnymi wagami.

Jak działa:

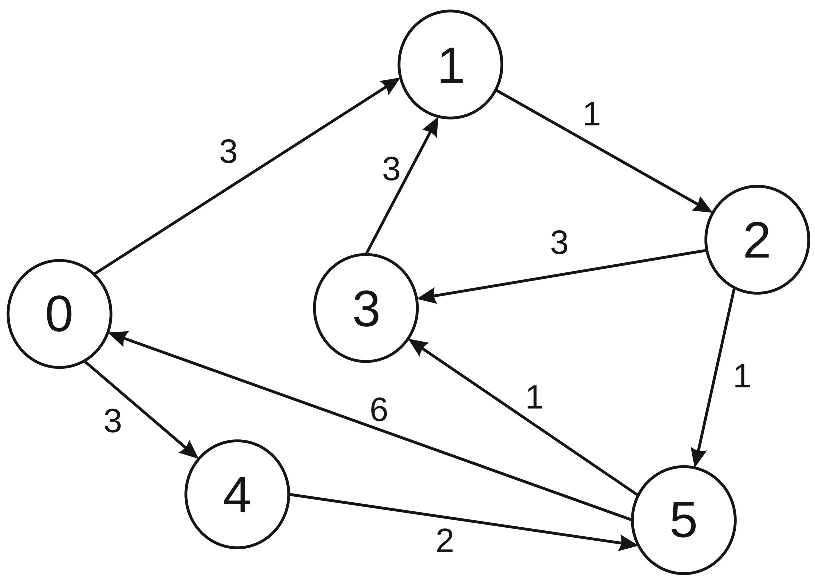
Przyjmijmy, że s to wierzchołek początkowy (źródło). w_{ij} to waga krawędzi łączącej wierzchołki o indeksach i oraz j

1) Najpierw tworzymy sobie tablicę odległości pomiędzy źródłem, a wszystkimi innymi wierzchołkami w grafie. Tablicę inicjalizujemy w ten sposób, że $d[s] = 0$ dla źródła (wierzchołka s) oraz $d[v] = +\infty$ dla pozostałych wierzchołków
2) Tworzymy kolejkę priorytetową Q dla wszystkich wierzchołków w grafie. Priorytetem będzie odległość od źródła (czyli to co aktualnie trzymamy w tablicy d). Im mniejsza odległość (mniejsze $d[x]$, tym wyższa pozycja w kolejce wierzchołka x).
3) Dopóki kolejka nie jest pusta to usuwamy z niej poszczególne elementy w następujący sposób:
   a) Usuwamy element u o najniższym priorytecie.
   b) Po usunięciu każdego elementu uaktualniamy tablicę d (a tym samym priorytety w kolejce) w taki sposób, że dla każdego sąsiada u (czyli wierzchołka v) sprawdzamy czy $d[v] > d[u] + w_{ij}$, jeśli tak to zastępujemy aktualną wartość $d[v]$ nową: $d[v] = d[u] + w_{ij}$. Jest to tzw. procedura relaksacji.

Jeśli chcemy znać najkrótszą ścieżkę od źródła do wierzchołków P to potrzebujemy dodatkowej tablicy, w której będziemy trzymać indeksy poprzedników (bezpośrednich wierzchołków poprzedzających na najkrótszej ścieżce), przy relaksacji, przy każdorazowym nadpisaniu tablicy d w tablicy P poprzednikiem wierzchołka v staje się wierzchołek u. Początkowo tablica poprzedników (za wyjątkiem źródła) jest zainicjowana np. -1.

Przykład:

Zróbmy przykład dla grafu o 6 wierzchołkach (o indeksach od 0 do 5) i następujących wagach



Załóżmy, że **źródłem będzie wierzchołek 1**. Początkowo mamy tablice *d* oraz *P*:

Wierzchołek	0	1	2	3	4	5
d	∞	0	∞	∞	∞	∞
P	-1	1	-1	-1	-1	-1

Kolejka Q zawiera wierzchołki 1 , 0 , 2, 3, 4, 5

Usuwamy z kolejki wierzchołek nr 1.

Z 1 możemy przejść tylko do 2, droga z 1 do 2 ma wagę 1 (co jest mniejsze od nieskończoności dokonujemy więc relaksacji). Nowa tabelka wygląda tak (na czerwono wierzchołki już usunięte z kolejki, na zielono elementy aktualizowane w danym kroku)

Wierzchołek	0	1	2	3	4	5
d	∞	0	1	∞	∞	∞
P	-1	1	1	-1	-1	-1

Uaktualniamy priorytety w kolejce Q, teraz zawiera ona wierzchołki 2 ,0, 3, 4, 5

W kolejnym kroku usuwamy z kolejki wierzchołek nr 2.

Z 2 możemy przejść do 3 i 5. Obie drogi są korzystniejsze od aktualnych, zatem uaktualniamy tabelkę i kolejkę

Wierzchołek	0	1	2	3	4	5
d	∞	0	1	$3+1=4$	∞	$1+1$=2
P	-1	1	1	2	-1	2

Q: 5, 3,0,4

W kolejnym kroku usuwamy 5. Uwaga, tym razem musimy zrelaksować drogę do 3, która przez 5 okazuje się „lepsza” niż aktualna.

Wierzchołek	0	1	2	3	4	5
d	8	0	1	3	∞	2
P	5	1	1	5	-1	2

Q: 0,3,4

W kolejnym kroku usuwamy 0:

Wierzchołek	0	1	2	3	4	5
d	8	0	1	3	11	2
P	5	1	1	5	0	2

Q: 3,4

Teraz z kolejki usuwamy 3 (tym razem nie ma żadnej relaksacji)

Wierzchołek	0	1	2	3	4	5
d	8	0	1	3	11	2
P	5	1	1	5	0	2

Q: 4

Usuwamy z kolejki ostatni wierzchołek numer 4 (teraz też nie ma żadnej relaksacji)

Wierzchołek	0	1	2	3	4	5
d	8	0	1	3	11	2
P	5	1	1	5	0	2

W tabelce (wiersz) d mamy zatem koszty dojścia z wierzchołka 1 do poszczególnych wierzchołków.

I tak dojście do

0 kosztuje 8

1 kosztuje 0

2 kosztuje 1

3 kosztuje 3

4 kosztuje 11

5 kosztuje 2

Teraz zajmiemy się najkrótszymi ścieżkami. Odczytujemy je „od tyłu”. Powiedzmy, że szukamy dojścia do wierzchołka 0. Jego bezpośredni poprzednik to 5 (z tabelki wiersz P). Następnie bierzemy poprzednik 5 (czyli 2) potem poprzednik 2 (czyli 1). Jedynka jest już źródłem, czyli kończymy. Zatem najkrótsza ścieżka do 0 wiedzie przez wierzchołki **1 2 5** (odwracamy kolejność odczytywania).

Inny przykład:

Dla wierzchołka 4 ścieżka „odwórcona” ścieżka byłaby 0,5,2,1; zatem sama ścieżka to **1 2 5 0.**

Uwaga:

Może się zdarzyć, że z danego źródła jakieś wierzchołki będą nieosiągalne. Wtedy w tabelce d będzie dla nich nadal występować nieskończoność (przykład takiego grafu będzie podany w materiałach).

Jeśli kolejkę zaimplementujemy przez kopiec to algorytm ma złożoność czasową $O(E \cdot log\ V)$.