Algorytmy do zestawu nr 3[1]

1. Algorytmy pomocnicze

Tablice d_s i p_s mają n elementów (n – liczba wierzchołków grafu G).

1.1. Nadanie wartości początkowych atrybutów d oraz p dla wierzchołków grafu G

Algorytm 1: `init`(G, s) $\triangleright$ *G – graf wejściowy, s – wybrany wierzchołek-źródło*

for każdy wierzchołek v należący do grafu G **do**
  $d_s[v] \leftarrow \infty$
  $p_s[v] \leftarrow$ NIL
end for
$d_s[s] \leftarrow 0$

1.2. Relaksacja krawędzi (u, v)

Algorytm 2: `relax`(u, v, w) $\triangleright$ *u, v – wierzchołki połączone krawędzią (u, v) poddaną relaksacji; w – macierz wag*

if $d_s[v] > d_s[u] + w[u][v]$ **then**
  $d_s[v] \leftarrow d_s[u] + w[u][v]$
  $p_s[v] \leftarrow u$
end if

2. Algorytm Dijkstry

Algorytm 3: `dijkstra`(G, w, s)

`init`(G, s)
$S \leftarrow \varnothing$ $\triangleright$ *S - zbiór "gotowych" wierzchołków; na początku jest pusty.*
while $S \neq$ zbiór wszystkich wierzchołków G **do**
  $u \leftarrow$ wierzchołek o najmniejszym $d_s[u]$ spośród niegotowych wierzchołków, $u \notin S$
  $S \leftarrow S \cup u$
  for każdy wierzchołek $v \notin S$ będący sąsiadem u **do**
    `relax`(u, v, w)
  end for
end while

Po zakończeniu działania algorytmu Dijkstry tablica atrybutów d_s przechowuje długości najkrótszych ścieżek z wierzchołka startowego s do wszystkich wierzchołków grafu. Tablica p_s przechowuje poprzed-

[1] Na podstawie: Cormen Thomas H., Leiserson Charles E., Rivest Ronald L., *Wprowadzenie do algorytmów*, Wyd. 4, Warszawa, Wydawnictwo Naukowo – Techniczne, 2001, ISBN 83-204-2665-0.

nika każdego z wierzchołków na jego najkrótszej ścieżce z s, dzięki czemu można odtworzyć każdą ze ścieżek. Podsumowując:

- $d_s[u]$ = długość najkrótszej ścieżki z wierzchołka startowego s do wierzchołka u,
- $p_s[u]$ = poprzednik wierzchołka u na najkrótszej ścieżce z s do u.

Natomiast w trakcie działania algorytmu Dijkstry tablicę d_s możemy rozumieć jako górne oszacowanie długości najkrótszych ścieżek z s. Oszacowanie to, początkowo nieskończone (patrz algorytm 1), jest poprawiane przez algorytm Dijkstry (alg. 3) przy pomocy relaksacji (alg. 2).

3. Minimalne drzewo rozpinające[2]

- G – graf wejściowy,
- n – liczba wierzchołków grafu G,
- (u, v) – krawędź z wierzchołka u do v,
- T – minimalne drzewo rozpinające.

3.1. Algorytm Prima

Początkowo wierzchołki grafu dzielimy na dwie części: dowolny startowy wierzchołek dodajemy do pustego drzewa T, pozostałe wierzchołki tworzą zbiór W. W każdym kroku **analizujemy tylko krawędzie łączące T z W:**

1. Wybieramy krawędź (u, v) o najmniejszej wadze (tzw. krawędź lekka).
2. Dodajemy (u, v) do drzewa T, jednocześnie usuwając odpowiedni wierzchołek z W.

Pętlę wykonujemy póki T nie zawiera wszystkich n wierzchołków grafu wejściowego.

3.2. Algorytm Kruskala

Wiadomo, że do drzewa T będą należały wszystkie wierzchołki. Początkowo nie są połączone (należą do różnych drzew). Sortujemy krawędzie grafu G według niemalejących wag. Dla każdej krawędzi (u, v) od najmniejszej wagi: jeżeli dodanie krawędzi (u, v) nie spowoduje powstania cyklu (czyli: krawędź (u, v) łączy różne drzewa), to dodajemy ją do T.
Algorytm można zakończyć przed końcem pętli, o ile T zawiera już $n - 1$ krawędzi.

[2]Na podstawie: V. K. Balakrishnan, *Schaum's Outline of Theory and Problems of Graph Theory*, McGraw-Hill Education – Europe, 1997, ISBN 0-07-005489-4.