Algorytm Kruskala – minimalne drzewo rozpinające

Autorzy: Jakub Szych, Jakub Orchowski

# 1. Wstęp

Celem zadania jest zaimplementowanie algorytmu Kruskala w celu odnalezienia minimalnego drzewa rozpinającego (MST) w spójnym, nieskierowanym grafie z wagami dodatnimi. MST minimalizuje łączny koszt połączeń, co ma zastosowania m.in. w projektowaniu sieci komputerowych i infrastruktury transportowej.

# 2. Opis algorytmu

1. Posortuj wszystkie krawędzie rosnąco według wag.  
2. Zainicjuj strukturę zbiorów rozłącznych (Disjoint Set Union, DSU), w której każdy wierzchołek tworzy osobny zbiór.  
3. Przeglądaj krawędzie w kolejności rosnących wag:  
 a. Jeśli wierzchołki krawędzi należą do różnych zbiorów, dodaj krawędź do MST i połącz zbiory.  
4. Zakończ, gdy MST zawiera V−1 krawędzi.

Pseudokod:

sort(E) // E – zbiór krawędzi rosnąco po wadze  
MST = {}  
forall v ∈ V: make\_set(v)  
for (u,v,w) in E:  
 if find(u) ≠ find(v):  
 MST ← MST ∪ {(u,v,w)}  
 union(u,v)  
 if |MST| = V-1: break

# 3. Złożoność obliczeniowa

· Sortowanie krawędzi: O(E log E).  
· Operacje find/union z kompresją ścieżek: prawie liniowe O(E α(V)).  
Łącznie: \*\*O(E log E)\*\*, pamięć: O(V + E).

# 4. Przykład działania

Rozważmy graf o 6 wierzchołkach (0–5) i 9 krawędziach:  
(0,1,4), (0,2,4), (1,2,2), (1,3,6), (2,3,8), (2,4,9), (3,4,9), (3,5,8), (4,5,5).

|  |  |  |  |
| --- | --- | --- | --- |
| k | Krawędź (u,v) | Waga | MST po kroku k |
| 1 | 1–2 | 2 | {1–2} |
| 2 | 0–1 | 4 | {1–2, 0–1} |
| 3 | 0–2 | 4 | {1–2, 0–1} (odrzucona – cykl) |
| 4 | 4–5 | 5 | {1–2, 0–1, 4–5} |
| 5 | 1–3 | 6 | {1–2, 0–1, 4–5, 1–3} |
| 6 | 3–5 | 8 | {1–2, 0–1, 4–5, 1–3, 3–5} |

Koszt całkowity MST = 25.

# 5. Sposób wprowadzania danych

Dane wejściowe znajdują się w pliku tekstowym, np. 'graph.txt'. Pierwsza linia zawiera dwie liczby całkowite: V (liczba wierzchołków) oraz E (liczba krawędzi). Każda z kolejnych E linii zawiera trzy liczby całkowite: u v w – oznaczające krawędź między u i v o wadze w.

# 6. Instrukcja uruchomienia aplikacji

1. Przygotuj plik 'graph.txt' w opisanym formacie.  
2. Uruchom polecenie:  
 python AlgKruskal.py graph.txt  
3. Wynik zostanie zapisany w pliku 'mst.txt'.  
  
\*\*Kompilacja do pliku .exe (opcjonalnie)\*\*  
Zainstaluj \*pyinstaller\* i wykonaj:  
 pyinstaller --onefile AlgKruskal.py  
Powstanie folder 'dist' z plikiem 'AlgKruskal.exe'.

# 7. Wyniki działania programu

Po uruchomieniu aplikacji AlgKruskal.exe z plikiem wejściowym graph.txt otrzymaliśmy plik mst.txt z następującą zawartością:  
  
1 2 2  
0 1 4  
4 5 5  
1 3 6  
3 5 8  
Total = 25

Dokument został przygotowany zgodnie z wymaganiami zadania nr 10.