

Projekt: „Algorytm Kruskala - minimalne drzewo rozpinające dla grafu nieskierowanego ważonego spójnego”

Autor: Łukasz Maruszak

Spis treści

| | |
|------------------------------------|---|
| Projekt | 1 |
| Autor..... | 1 |
| Minimalne drzewo rozpinające | 1 |
| Algorytm Kruskala | 1 |
| Opis Implementacji..... | 1 |
| Wyniki działania programu..... | 3 |
| Źródła..... | 6 |

Minimalne drzewo rozpinające

Drzewo rozpinające grafu składa się ze wszystkich wierzchołków grafu oraz niektórych krawędzi. Minimalne drzewo rozpinające jest drzewem rozpinającym którego suma krawędzi jest najmniejsza z możliwych. Istnieje kilka algorytmów, które rozwiązują ten problem np. Algorytm Kruskala.

Algorytm Kruskala

Algorytm polega na tym, że:

1. Utwórz listę K wszystkich krawędzi grafu. Następnie sortujemy ją w porządku rosnącym według wartości wag krawędzi.
2. Tworzymy las L z wierzchołków grafu, każdy wierzchołek na początku jest osobnym drzewem.
3. Utwórz pusty graf G, który będzie minimalnym drzewem rozpinającym
4. Pobierz z K krawędź o najmniejszej wadze
5. Jeżeli krawędź łączy dwa różne drzewa w lesie L to dodaj ją do grafu G oraz połącz dwa odpowiadające drzewa w jedno
6. W przeciwnym wypadku odrzuć
7. Jeżeli w G jest (ilość wierzchołków grafu początkowego – 1) krawędzi zakończ algorytm

Opis Implementacji

Program składa się z 3 plików:

1. Graph.py
2. Visualization.py
3. Main.py

Oraz katalogu „grafy”, w którym znajdują się pliki tekstowe, z których odczytywany jest wygląd grafu. Pliki te składają się z trzech kolumn. Odpowiednio oznaczają – początek krawędzi, koniec krawędzi, waga krawędzi.

W pliku „Graph.py” znajduje się klasa Edge, która reprezentuje krawędź w grafie. Przyjmuje ona trzy wartości, które odpowiadają wierzchołkowi, który jest początkiem krawędzi, końcem oraz wagę krawędzi. W tej klasie znajduje się również metoda `__repr__` służąca do wyświetlania krawędzi w konsoli.

Klasa Graph służy to reprezentacji grafu w programie. W konstruktorze inicjalizowane są trzy słowniki, `graph` służy do słownikowej reprezentacji grafu, `parent` oraz `rank` wykorzystywane są w algorytmie Kruskala. `parent` przechowuje informację o korzeniu danego wierzchołka oraz `rank`, gdzie zapisywana jest informacja o ilości węzłów z jakimi ma połączenie dany węzeł.

Klasa Graph posiada metody:

`add_node(self, node)` – dodanie wierzchołka do grafu, czyli nowego klucza do słownika

`add_edge_undirected(self, edge)` – dodanie nowej krawędzi do grafu, metoda sprawdza czy początek i koniec krawędzi to nie ten sam wierzchołek aby zapobiec pętlom. Dla odpowiedniego klucza tworzony jest pusty słownik do którego dodawane są wierzchołki z jakimi jest połączony wierzchołek.

`list_nodes(self)` – zwraca listę wierzchołków grafu.

`list_edges(self)` – zwraca listę krawędzi, czyli listę krotek o trzech wartościach.

`graph_size(self)` – zwraca wielkość grafu, ilość wierzchołków

`print_graph(self)` – wypisuje postać grafu nieskierowanego ważonego na ekran

Metody wykorzystywane w algorytmie Kruskala

`sort_edges(self)` – sortuje rosnąco krawędzie w grafie na podstawie ich wag

`FindParent(self, node)` – znajduje rodzica wierzchołka w grafie. Sprawdza wartość w słowniku `parent`.

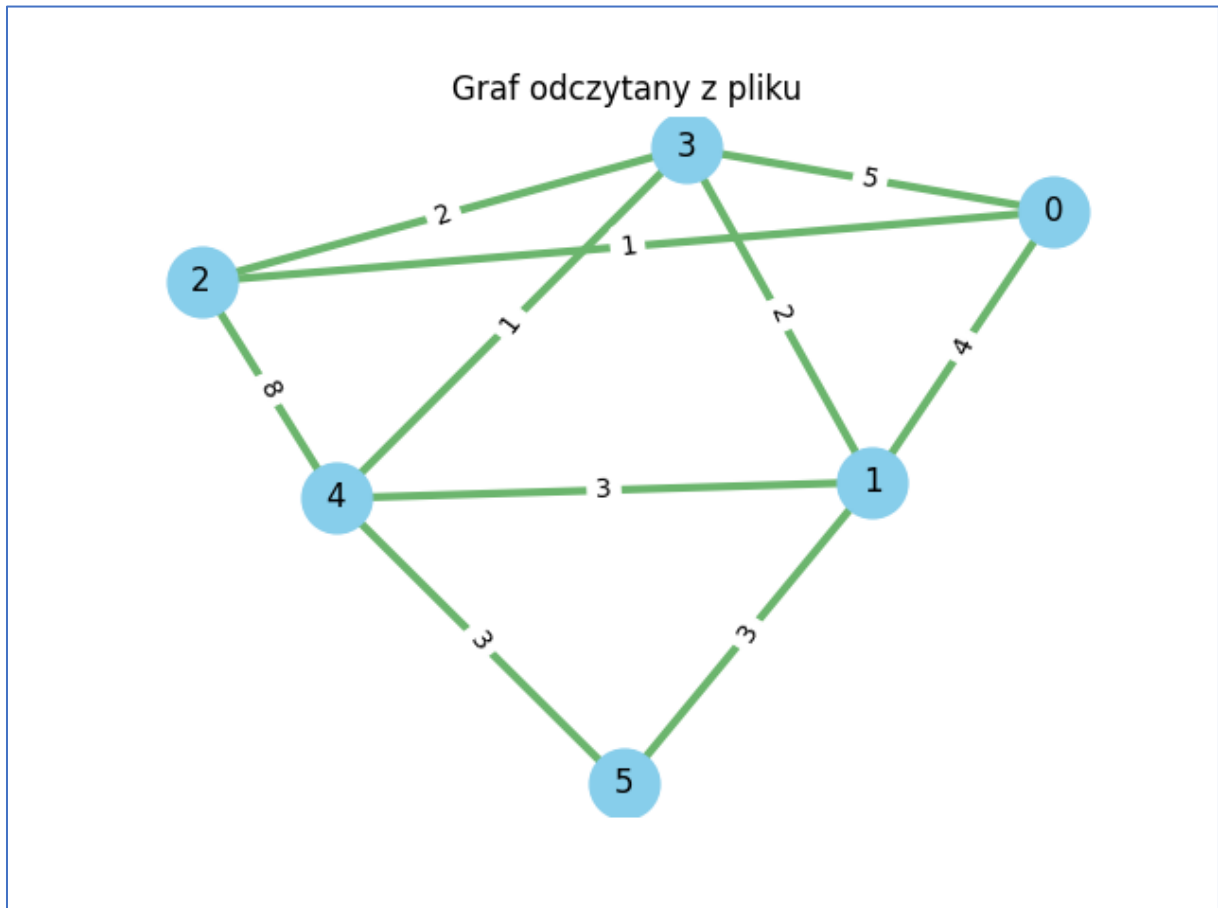
`Kruskal_Algorithm(self)` – główna metoda wykonująca algorytm Kruskala. Na początku tworzony jest pusty graf (`mst`), który będzie zwrócony jako Minimalne Drzewo Rozpinające. Inicjalizowane są słowniki `rank` oraz `parent`. Początkowo każdy węzeł jest rodzicem dla samego siebie, ponieważ tworzymy las drzew, gdzie każde składa się z jednego węzła, dlatego też wartości w słowniku `rank` dla każdego wierzchołka wynoszą 0 ponieważ nie jest ono połączone z innym wierzchołkiem. Główna pętla przechodzi tyle razy ile znajdują się krawędzie w grafie początkowym. Pobieram krawędź o minimalnej wadze i szukam rodziców dla wierzchołka początkowego i końcowego. Jeżeli rodzice są różni, czyli wierzchołki należą do różnych drzew, krawędź dodaje do grafu `mst`. Następnie sprawdzam które drzewo jest większe na podstawie wartości w słowniku `rank` i przyłączam mniejsze drzewo do większego, czyli dla wierzchołka ustawiam nowego rodzica.

Plik `visualization.py` odpowiada za wygenerowanie obrazów przedstawiających podany graf oraz minimalne drzewo rozpinające. Przy generowaniu, wykorzystane są metody z `matplotlib` oraz `networkx`.

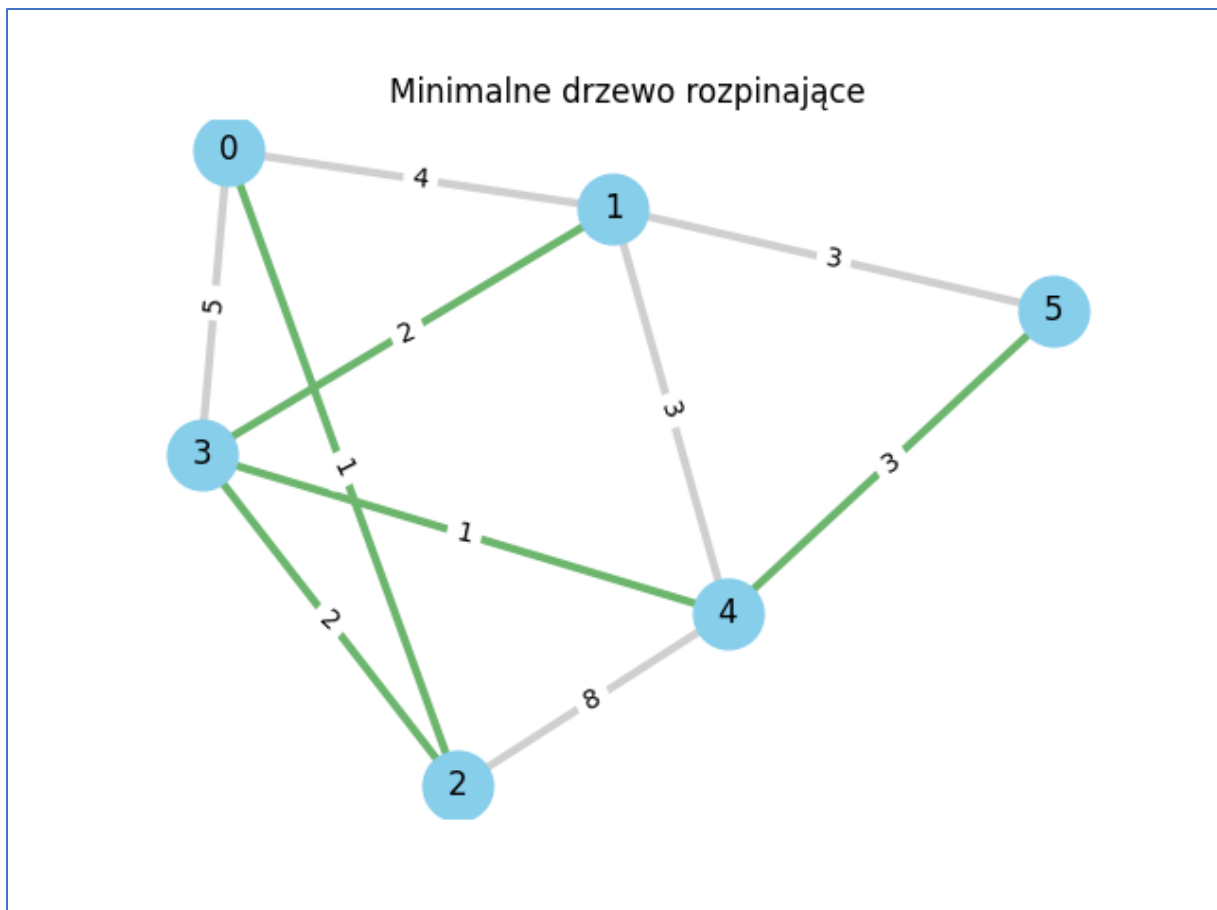
Plik main.py znajduje się metoda do odczytania danych z pliku. Przyjmuje ona jako argument nazwę pliku. Domyślnie odczytywany jest plik „grafy/graf.txt”. Plik odczytywany jest linia po linii a do grafu dodawane utworzone krawędzie.

Wyniki działania programu

Wierzchołki w wygenerowanym pliku za każdym razem mogą znajdować się w innym miejscu, dlatego może się zdarzyć sytuacja, że obrazek będzie nie czytelny, ponieważ krawędzie mogą nachodzić na siebie. W takiej sytuacji zalecam uruchomienie programu od nowa.

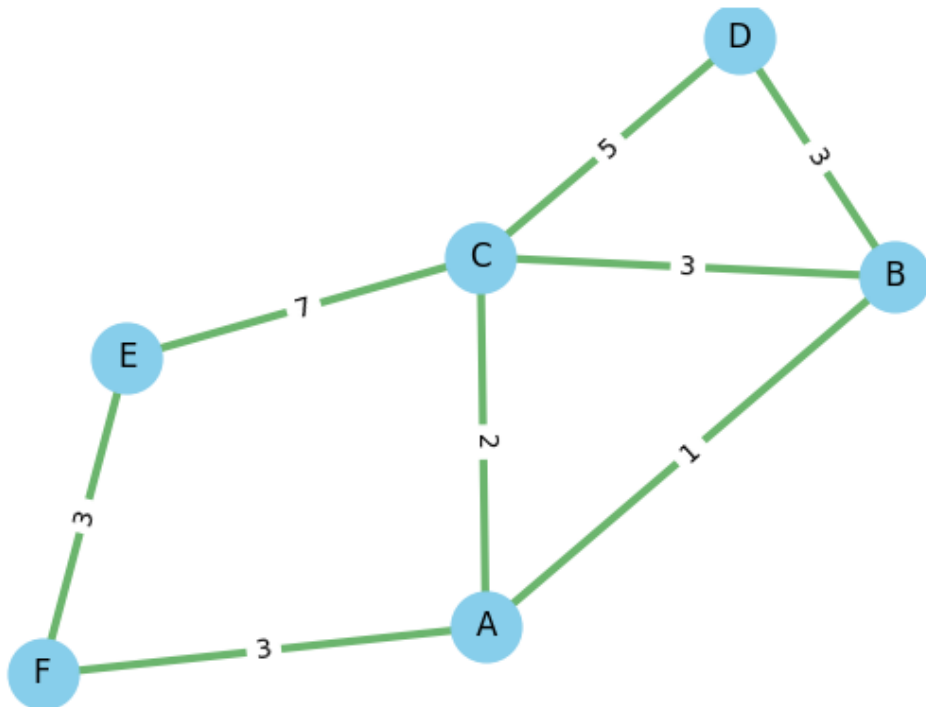


Wyniki działania programu po odczytaniu pliku graf1.txt

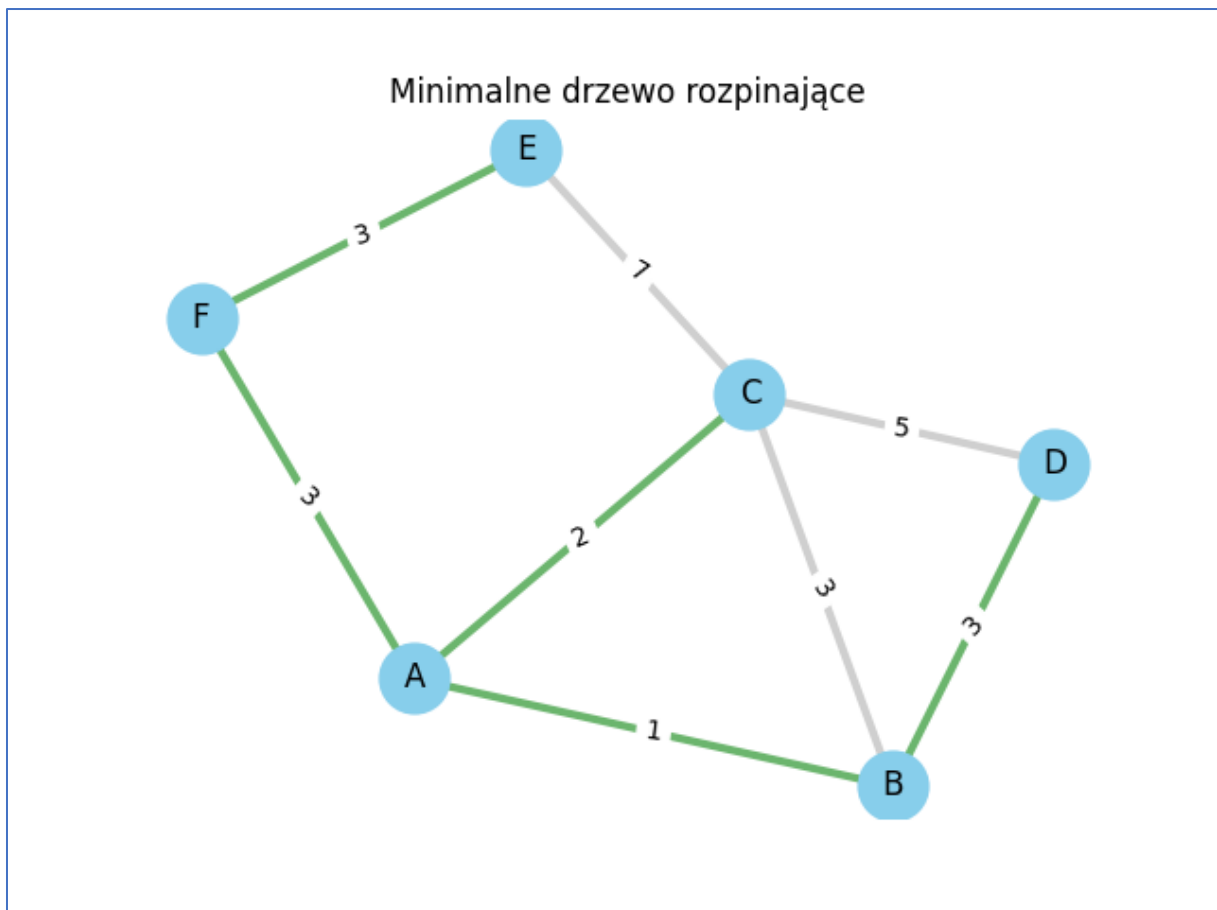


Wyniki działania algorytmu Kruskala dla grafu z pliku graf1.txt, zielone krawędzie oznaczają krawędzie występujące w Minimalnym Drzewie Rozpinającym, szare krawędzie to pozostałe krawędzie które nie wchodzą w Minimalne Drzewo Rozpinające grafu.

Graf odczytany z pliku



Wyniki działania programu po odczytaniu pliku graf.txt



Wyniki działania algorytmu Kruskala dla grafu z pliku graf.txt, zielone krawędzie oznaczają krawędzie występujące w Minimalnym Drzewie Rozpinającym, szare krawędzie to pozostałe krawędzie które nie wchodzą w Minimalne Drzewo Rozpinające grafu.

Źródła

1. https://pl.wikipedia.org/wiki/Algorytm_Kruskala
2. https://eduinf.waw.pl/inf/alg/001_search/0141.php