

## DOKUMENTACJA PROJEKT ZALICZENIOWY PYTHON – DANIEL KUŁAGA

### TEMAT PROJEKTU:

**Algorytm Prima - minimalne drzewo rozpinające dla grafu nieskierowanego ważonego spójnego.**

### Opis projektu:

W skład drzewa rozpinającego grafu wchodzi wszystkie wierzchołki, ale nie wszystkie krawędzie. Minimalne drzewo rozpinające jest drzewem rozpinającym, którego suma krawędzi jest najmniejsza z możliwych. Zaimplementowany przeze mnie algorytm Prima wyznacza wspomniane minimalne drzewo, oprócz tego algorytmu, możliwe jest wyznaczenie go za pomocą np.: algorytmu Kruskala.

### Opis algorytmu:

Algorytm Prima, ma następujące kroki:

1. Tworzę drzewo zawierające jeden dowolnie wybrany wierzchołek.
2. Tworzę kolejkę priorytetową, zawierającą wierzchołki osiągalne z MDR, o najmniejszym priorytecie (czyli najmniejszej wadze krawędzi).
3. Wśród nieodwiedzonych wierzchołków (spoza MDR) wybieram ten, dla którego koszt dojścia z obecnego MDR jest najmniejszy.
4. Dodaje do obecnego MDR wierzchołek i krawędź realizującą najmniejszy koszt.
5. Aktualizuję kolejkę priorytetową, uwzględniając nowe krawędzie wychodzące z dodanego wierzchołka.
6. Kroki od pkt. 3 do pkt.5 powtarzam dopóki drzewo nie obejmuje wszystkich wierzchołków grafu.
7. Zwracam minimalne drzewo rozpinające, co kończy algorytm.

### Szczegóły implementacji:

W skład projektu, wchodzi 4 pliki:

**Graph.py** – klasa Graph

**Edge.py** – klasa Edge

**PriorityQueue.py** – klasa PriorityQueue

**main.py** – główna funkcja programu z interfejsem konsolowym

oraz pliki **test\_edge.py**, **test\_graph.py**, **test\_priorityqueue.py** testujące metody z poszczególnych klas, oraz **Prima\_algorithm\_test.py** – testujący działanie zaimplementowanego algorytmu Prim'a.

**class PriorityQueue:** klasa opisująca kolejkę priorytetową (metody dołączania kolejnych elementów, usuwania, sposobu wyświetlania itd.).

**class Edge:** klasa reprezentująca krawędź w grafie.

**class Graph:** klasa reprezentująca graf, występują w niej następujące metody:

`add_node(self, node)` wstawianie wierzchołka do grafu

`add_edge_undirected(self, Edge)` Dodaje krawędź do grafu nieskierowanego

`list_nodes(self)` Zwraca listę wierzchołków grafu

`generator_node_edges(self, node)` Zwraca trójki (od , do , waga) wartości krawędzi wychodzących z konkretnego wierzchołka

`print_graph(self)` rysuje graf

`graph_size(self)` - zwraca rozmiar grafu

`PrimAlgorithm(self, selected_node):` metoda implementująca algorytm Prima od wybranego wierzchołka

### Instrukcja korzystania z programu:

Po uruchomieniu pliku main.py pojawia się interfejs konsolowy:

```
Witaj w programie generującym minimalne drzewo rozpinające algorytmem Prima
Wybierz: A - gotowy graf, B- chce podać swój graf :
```

Możemy wybrać A , wtedy zostanie wygenerowany graf spójny, ważony zainstalowany domyślnie.

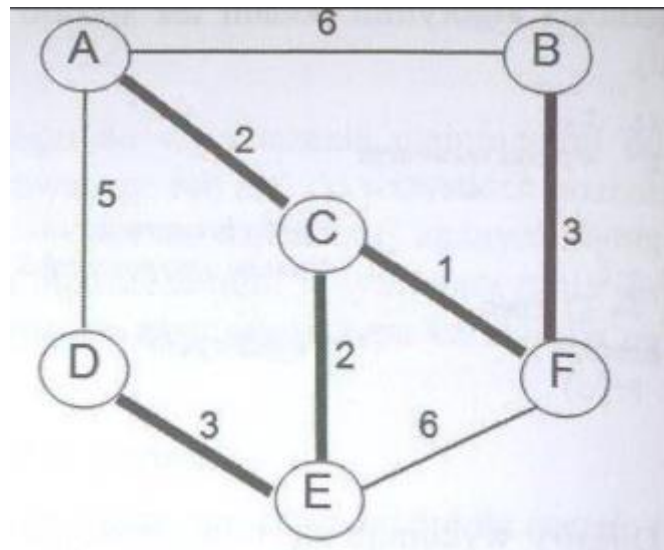
Przy wyborze B, użytkownik zostanie poproszony o podanie następujących danych:

```
Witaj w programie generującym minimalne drzewo rozpinające algorytmem Prima
Wybierz: A - gotowy graf, B- chce podać swój graf : B
Podaj ilość krawędzi twojego grafu spójnego i ważonego: 8
Podaj dane krawędzi: wierzcholek_startowy(spacja)wierzcholek_koncowy(spacja)waga_krawedzi A C 2
Podaj dane krawędzi: wierzcholek_startowy(spacja)wierzcholek_koncowy(spacja)waga_krawedzi A D 5
Podaj dane krawędzi: wierzcholek_startowy(spacja)wierzcholek_koncowy(spacja)waga_krawedzi B A 6
Podaj dane krawędzi: wierzcholek_startowy(spacja)wierzcholek_koncowy(spacja)waga_krawedzi |
```

Następnie użytkownik podaje, od którego wierzchołka algorytm ma zacząć pracę i poniżej otrzymuje wygenerowane MDR:

```
Podaj od którego z powyższych wierzchołków ma zacząć algorytm: A
A : C(2)
C : A(2) F(1) E(2)
F : C(1) B(3)
E : C(2) D(3)
D : E(3)
B : F(3)
```

Poniższy graf: (pogrubione linie to minimalne drzewo rozpinające) – algorytm zwrócił identyczne minimalne drzewo rozpinające



Źródła:

[https://pl.wikipedia.org/wiki/Algorytm\\_Prima](https://pl.wikipedia.org/wiki/Algorytm_Prima)

<https://docs.python.org/3/library/heapq.html>

P.Wróblewski – Minimalne drzewa rozpinające