1.Analiza Algorytmu

Algorytm Dijkstry jest algorytmem wyszukiwania w grafie najkrótszej ścieżki z pojedynczego źródła, jeśli wszystkie krawędzie mają nieujemne wagi. Charakteryzuje się lepszą złożonością niż pokrewne algorytmy, m.in. algorytm Bellmana-Forda, kosztem ograniczenia do nieujemnych wartości krawędzi.

Algorytm ten świetnie się sprawdza przy wyszukiwaniu najkrótszych odległości pomiędzy punktami, a zatem we wszelkiego rodzaju systemach nawigacji, trasowaniu (routingu) danych w sieci komputerowej, m.in. protokół OSPF. Kolejnym przydatnym wykorzystaniem algorytmu jest programowanie sztucznych inteligencji, np. wytyczanie ścieżek dla postaci w grach komputerowych, tzw. pathfinding.

Następne zastosowanie to system polecania znajomych w serwisach społecznościowych (im więcej interakcji wykonują między sobą znajomi, tym są sobie "bliżej"), dzięki czemu są nam polecani ludzie, z którymi mamy największe szanse na nawiązanie więzi.

Kolejny przykład to aplikacje takie jak "jakdojade", czyli minimalizowanie czasu dotarcia do celu używając wielu środków transportu (analizując czas dojścia na przystanek, czas odjazdu, czas przyjazdu, czas oczekiwania)

2.Uruchomienie projektu i opis działania

W zipie "DijkstraProjectBL.zip" znajduje się folder o tej samej nazwie. Należy wejść do tego folderu i za pomocą Pythona uruchomić plik "main.py". Można to zrobić z terminala w Linuxie (Python jest wbudowany w większość dystrybucji), na Windowsie należy zainstalować Pythona i wtedy wystarczy kliknąć podwójnie plik "main.py".

W folderze z programem znajduje się też folder "graph_jsons", w którym są przykładowe pliki zawierające dane o grafie. Do programu można zaimportować dowolny graf, ale musi być przedstawiony w formie takiej jak grafy przykładowe w pliku.

Forma ta wygląda następująco:

- Plik składa się z nawiasu kwadratowego, w którym znajdują się oddzielone przecinkami krawędzie [e1, e2, e3]
- Każda krawędź składa się z trzech wartości oddzielonych przecinkami, z których trzecia musi być całkowitą liczbą nieujemną, całość musi być w nawiasie kwadratowym, np. e1 = [A,B,3]. Pierwsze dwie wartości to początek i koniec krawędzi, natomiast trzecia to waga.
- Przykładowo jeśli zawartość pliku to [[A,B,3],[A,C,2]], to jest to graf z 3 wierzchołkami A,B,C i z 2 krawędziami: A do B o wadze 3, A do C o wadze 2
- Program pyta czy graf jest skierowany, także nie trzeba dublować krawędzi dla grafów nieskierowanych

Plik z taką formą reprezentacji z grafu w rozszerzeniu “.json” należy umieścić w folderze “graph_jsons”. Znajdują się tam już grafy “Test”, “Test2”, “Test3”, “MyGraph”, reprezentacja graficzna trzech pierwszych znajduje się w folderze z projektem. “Test” i “Test2” są skierowane, natomiast “Test3” jest nieskierowany. MyGraph jest nieskierowanym grafem, który analizowałem w części analitycznej, wraz z dobranymi wtedy wagami.

Po uruchomieniu program poprosi o podanie nazwy pliku z grafem do analizy, bądź wpisanie “Exit”, aby go wyłączyć. Dodawanie rozszerzenia “.json” na końcu nazwy nie jest wymagane. Jeśli plik istnieje i zawiera poprawną formę grafu, program spyta, czy graf jest skierowany. Należy wpisać odpowiedź “Y” - tak lub “N” - nie. Następnie program wypisze macierz wyjściową algorytmu dijkstry (czyli tablicę najmniejszych odległości i “poprzednich” wierzchołków), a niżej najkrótsze ścieżki od “pierwszego” do każdego innego wierzchołka (gdzie kolejność wierzchołków jest alfabetyczna, zatem przy nazwach numerycznych proszę pisać “01” etc. jeśli nazwy wierzchołków przekraczają 9) wraz z odległością je dzielącą. Program działa w pętli przyjmując następne pliki aż zostanie wpisane “Exit”. Kod źródłowy programu znajduje się w plikach “main.py” oraz “graph.py”.