Projekt w języku Python- Algorytm Dijkstry - znajdowanie najkrótszej ścieżki z pojedynczego źródła w grafie nieskierowanym o nieujemnych wagach krawędzi.

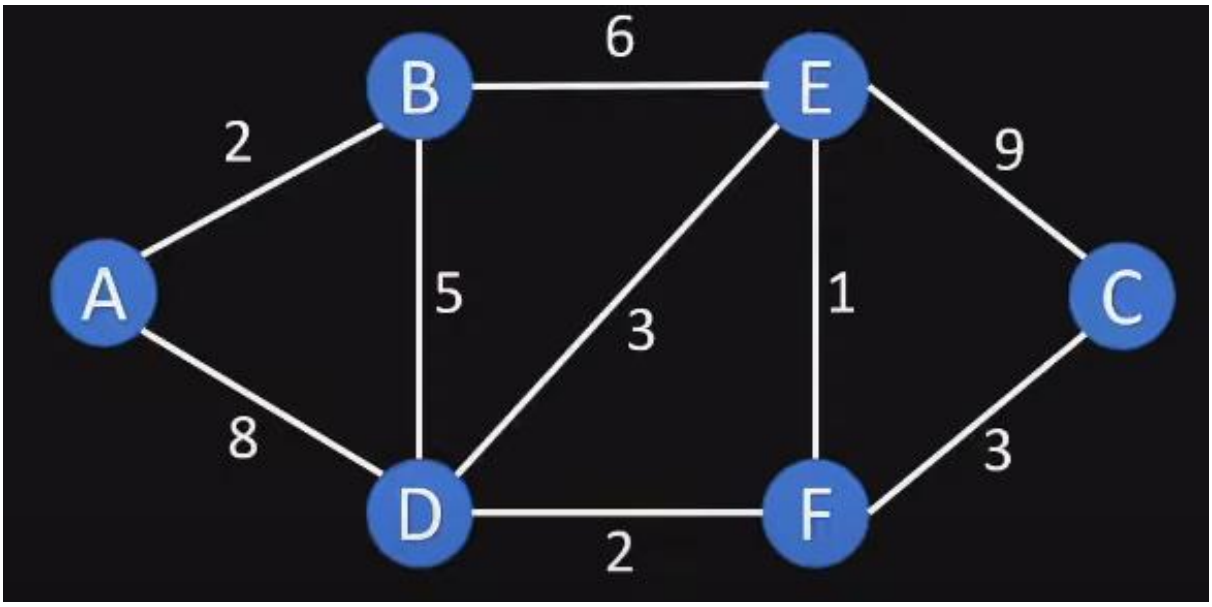
1. Wstęp teoretyczny

Do czego służy algorytm Dijkstry?

Algorytm Dijkstry jest używany do znajdowania najkrótszej ścieżki w grafie skierowanym lub nieskierowanym z wagami na krawędziach. Opracowany przez Edsgera W. Dijkstrę w 1956 roku, algorytm ten ma zastosowanie w różnych dziedzinach, takich jak systemy nawigacyjne, sieci telekomunikacyjne, czy planowanie tras. Algorytm Dijkstry zwraca tablicę najkrótszych odległości od źródłowego wierzchołka do pozostałych wierzchołków oraz informacje o poprzednikach na tych ścieżkach.

Jak działa ten algorytm?

Mamy dany graf nieskierowany o nieujemnych wagach krawędzi:



Wybieramy wierzchołek startowy (wierzchołek od którego chcemy sprawdzić najkrótsze odległości do innych wierzchołków). Dla przykładu wybieramy wierzchołek A.

Wierzchołek	Najkrótszy dystans	Poprzedni wierzchołek
A	0	-
B	∞	-
C	∞	-
D	∞	-
E	∞	-
F	∞	-

Teraz tworzymy tabelę:

Jako że, zaczynamy od wierzchołka A nie znamy jeszcze dystansów do innych wierzchołków, dlatego wszędzie wpisujemy znak ∞.

Wierzchołek	Najkrótszy dystans	Poprzedni wierzchołek
A	0	-
B	2	A
C	∞	-
D	8	A
E	∞	-
F	∞	-
Odwiedzone wierzchołki: A	Nieodwiedzone wierzchołki: B, C, D, E, F	

Następnie porównujemy trasy do nieodwiedzonych sąsiadów wierzchołkla A. Aktualizujemy „Najkrótszy dystans" do wierzchołka B i D. Jako „Poprzedni wierzchołek" wpisujemy A. Następnie wybieramy kolejny wierzchołek dla którego dystans jest najkrótszy w tym przypadku jest to wierzchołek B.

Wierzchołek	Najkrótszy dystans	Poprzedni wierzchołek
A	0	-
B	2	A
C	∞	-
D	7	B
E	8	B
F	∞	-
Odwiedzone wierzchołki: A, B	Nieodwiedzone wierzchołki: C, D, E, F	

Przyglądamy się teraz nieodwiedzonym sąsiadom wierzchołka B. Aktualizujemy dystans z wierzchołka A do wierzchołka E przez wierzchołek B (2+6). Jako że, dystans od wierzchołka D przez wierzchołek B jest krótszy (2+5) aktualizujemy też ten dystans. Następnie wybieramy kolejny wierzchołek z nieodwiedzonych wierzchołków, który ma najmniejszy dystans. W tym przypadku wierzchołek D.

Wierzchołek	Najkrótszy dystans	Poprzedni wierzchołek
A	0	-
B	2	A
C	∞	-
D	7	B
E	8	B
F	9	D
Odwiedzone wierzchołki: A, B, D	Nieodwiedzone wierzchołki: C, E, F	

Przeglądamy się teraz nieodwiedzonym sąsiadom wierzchołka D. Aktualizujemy dystans z wierzchołka A do wierzchołka F przez wierzchołek D(7+2). Jako że dodany dystans z wierzchołka D do wierzchołka E jest większy (7+3) nic nie zmieniamy. Następnie wybieramy kolejny wierzchołek z nieodwiedzonych wierzchołków, który ma najmniejszy dystans. W tym przypadku wierzchołek E.

Wierzchołek	Najkrótszy dystans	Poprzedni wierzchołek
A	0	-
B	2	A
C	17	E
D	7	B
E	8	B
F	9	D
Odwiedzone wierzchołki: A, B, D, E		Nieodwiedzone wierzchołki: C, F

Przeglądamy się teraz nieodwiedzonym sąsiadom wierzchołka E. Aktualizujemy dystans z wierzchołka A do wierzchołka C przez wierzchołek E (8+9). Jako że dodany dystans z wierzchołka E do wierzchołka F jest taki sam nic nie zmieniamy. Następnie wybieramy kolejny wierzchołek z nieodwiedzonych wierzchołków, który ma najmniejszy dystans. W tym przypadku wierzchołek F.

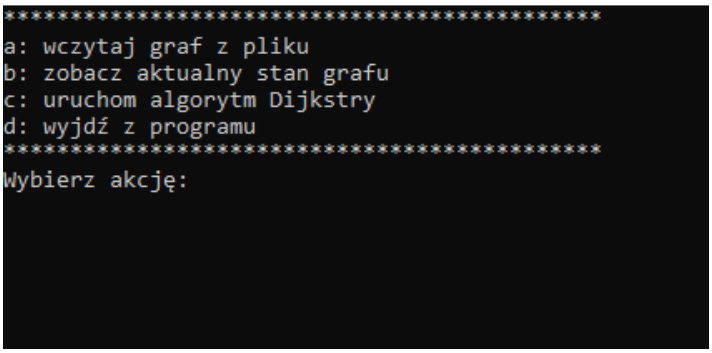
Wierzchołek	Najkrótszy dystans	Poprzedni wierzchołek
A	0	-
B	2	A
C	12	F
D	7	B
E	8	B
F	9	D
Odwiedzone wierzchołki: A, B, D, E, F		Nieodwiedzone wierzchołki: C

Przeglądamy się teraz nieodwiedzonym sąsiadom wierzchołka F. Jako że dodany dystans z wierzchołka A do wierzchołka C przez wierzchołek F jest mniejszy, zmieniamy ten wynik. Następnie wybieramy kolejny wierzchołek z nieodwiedzonych wierzchołków, który ma najmniejszy dystans. W tym przypadku wierzchołek C. Jako że wierzchołek C nie ma nieodwiedzonych sąsiadów, kończymy działanie algorytmów.

Otrzymaliśmy informację na temat najkrótszej odległości wierzchołka startowego do wszystkich innych wierzchołków. Możemy też prześledzić przez jakie wierzchołki idzie najkrótsza droga.

2. Działanie programu

Aby uruchomić program trzeba mieć zainstalowane odpowiednie biblioteki matplotlib, networkx oraz interpreter Pythona. W konsoli piszemy python main.py.



Do wyboru mamy kilka akcji. Wybieramy akcję i wciskamy klawisz Enter.

Prześledźmy teraz działanie programu dla tego samego grafu co poprzednio. Graf wczytujemy z pliku akcją ‘a’.

W pliku definiujemy wierzchołki i krawędzie między nimi,

W: A B C D E F (Po literze W możemy dodawać wierzchołki do naszego grafu)

K: A B 2 (po literze K definiujemy krawędzie, wierzchołek1 wierzchołek2 waga)

K: A D 8

K: B D 5

K: B E 6

K: D E 3

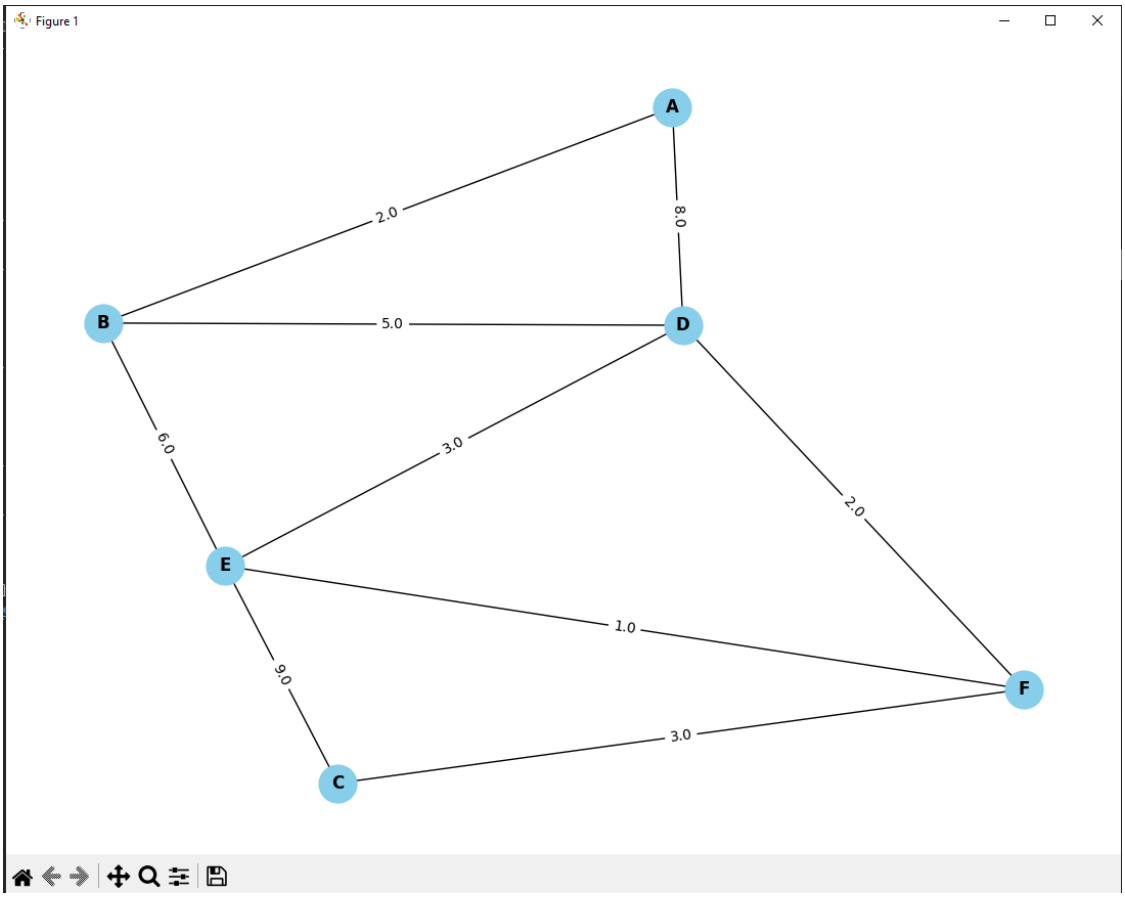
K: D F 2

K: E F 1

K: E C 9

K: F C 9

Akcją b) możemy zobaczyć jak wygląda nasz graf:



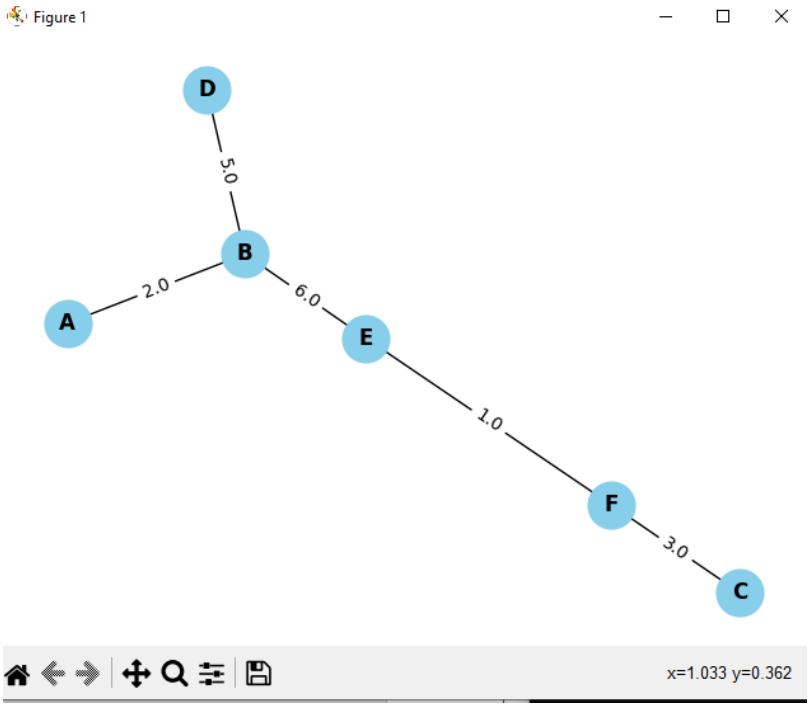
Akcją c) uruchomimy algorytm Dijkstry:

```
Wierzchołek, z którego rozpocząć algorytm Dijkstry:
```

Wpisujemy od którego wierzchołka chcemy zacząć. Tak jak w poprzednim przykładzie zaczniemy od wierzchołka A.

```
Wierzchołek, z którego rozpocząć algorytm Dijkstry: A
Najkrótsza odległość od A do A: 0
Najkrótsza odległość od A do B: 2.0
Najkrótsza odległość od A do C: 12.0
Najkrótsza odległość od A do D: 7.0
Najkrótsza odległość od A do E: 8.0
Najkrótsza odległość od A do F: 9.0
```

Otrzymaliśwy wyniki. Jak widzimy są takie same jak dla przykładu wykonanego po kolei. Wyświetlił się też graf



Możemy prześledzić najkrótszą ścieżkę od A do każdego wierzchołka. Widzimy przez jakie wierzchołki przechodzi.