Projekt zespołowy
Problem szukania najkrótszej drogi przy pomocy algorytmu dijkstra
Dokumentacja końcowa

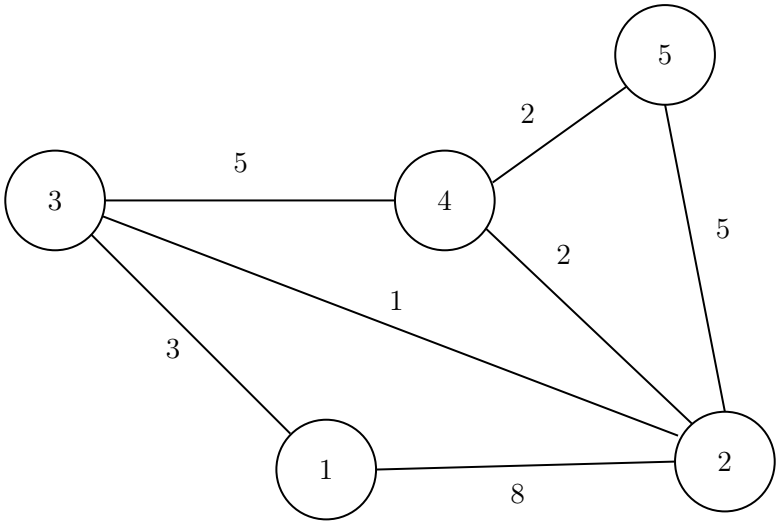
Skład zespołu:
Krzysztof Brzózka, Jakub Dimtruk, Oskar Werner

15.01.2025

1 Opis problemu

Problem, którym się zajmowaliśmy polega na znalezieniu najkrótszej ścieżki między dwoma wierzchołkami w grafie ważonym.

Przykładowo dla poniższego grafu i wierzchołków 1 i 5 najkrótsza ścieżka prowadzi przez wierzchołki $\{1, 3, 2, 4, 5\}$. Ta ścieżka składa się z 4 krawędzi, mimo tego, że z wierzchołka 1 do 5 można przejść korzystając z trzech krawędzi.



2 Dane wejściowe i wyjściowe algorytmu

Danymi wejściowymi algorytmu jest graf ważony czyli lista wierzchołków oraz krawędzi z wagami, a także wierzchołek startowy z którego szukamy najkrótszej ścieżki.

Danymi wyjściowymi algorytmu jest lista wierzchołków wraz z długościami najkrótszych ścieżek z danego źródła.

3 Algorytm dijkstra

Algorytm dijkstra znajduje długość najkrótszej ścieżki z wybranego wierzchołka źródłowego do każdego innego w danym grafie, o ile wszystkie wagi krawędzi są nieujemne.

3.1 Opis algorytmu

Poniżej przedstawione są kolejne kroki algorytmu:

1. Stwórz zbiór wszystkich nieodwiedzonych wierzchołków U.
2. Przypisz każdemu wierzchołkowi odległość od wierzchołka początkowego. Dla wierzchołka początkowego odległość wynosi 0, a dla pozostałych ∞.

3. Ze zbioru U wybierz jako aktualny wierzchołek ten, który ma najmniejszą odległość. Początkowo jest to wierzchołek startowy. Jeśli zbiór U jest pusty lub zawiera tylko wierzchołki o nieskończonej odległości to algorytm się kończy przechodząc do kroku 6. Jeśli szukamy ścieżki jedynie między dwoma wierzchołkami, to algorytm kończy działanie, gdy aktualny wierzchołek jest naszym wierzchołkiem docelowym. W przeciwnym wypadku program przechodzi do punktu 4.

4. Dla aktualnego wierzchołka rozważ wszystkich jego nieodwiedzonych sąsiadów i policz ich odległości prowadzące przez aktualny wierzchołek. Następnie porównaj nowo obliczone odległości z aktualnie przypisanymi do danego sąsiada i przypisz mu mniejszą z wartości.

5. Po rozpatrzeniu wszystkich nieodwiedzonych sąsiadów aktualnego wierzchołka jest on usuwany ze zbioru U. Następnie wróć do kroku 3.

6. Po zakończeniu pętli (kroki 3-5), każdy odwiedzony wierzchołek zawiera najmniejszą odległość od wierzchołka źródłowego

W trakcie wykonywania powyższego algorytmu można również każdemu wierzchołkowi przypisywać poprzedni wierzchołek na najkrótszej ścieżce prowadzącej do niego, co pozwoli na odtworzenie tej ścieżki.

3.2 Złożoność obliczeniowa

Rozważmy graf o $|V|$ wierzchołkach i $|E|$ krawędziach.

W przypadku gdy algorytm implementowany jest na strukturze listy złożoność obliczeniowa to $O(|V|^2)$.

Korzystając ze struktury binarnego kopca złożoność zmniejsza się do $O((|E| + |V|)\log|V|)$.

Przy pomocy kopca Fibinacciego jesteśmy w stanie zmniejszyć złożoność jeszcze bardziej, do $O(|E| + |V|\log|V|)$.

4 Opis aplikacji i jej funkcjonalności

Stworzony przez nas program rozwiązuje problem znalezienia najkrótszej ścieżki w gafie ważonym między dwoma wybranymi wierzchołkami. Nasza implementacja algorytmu dijkstra jest oparta o strukturę listy.

Frontend aplikacji został napisany korzystając z React, a backend aplikacji w Pythonie.

4.1 Dane wejściowe

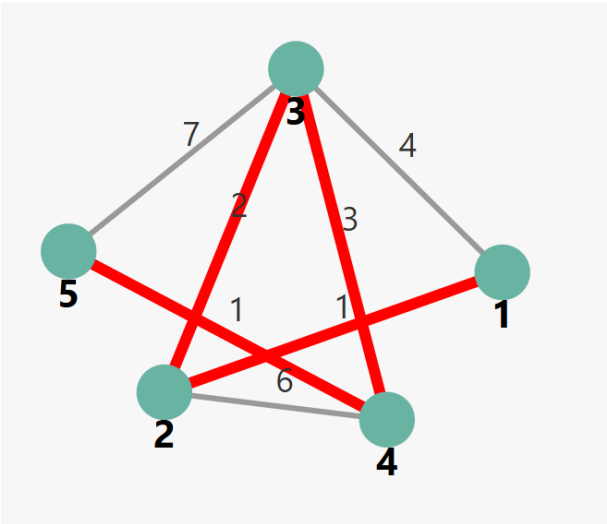
Aplikacja jako dane wejściowe przyjmuje plik w formacie *.csv*, w którym w pierwszej linijce znajdują się wybrane wierzchołki między którymi szukamy najkrótszej ścieżki, a w każdej kolejnej para wierzchołków definiująca krawędź oraz waga tej krawędzi.

4.2 Dane wyjściowe

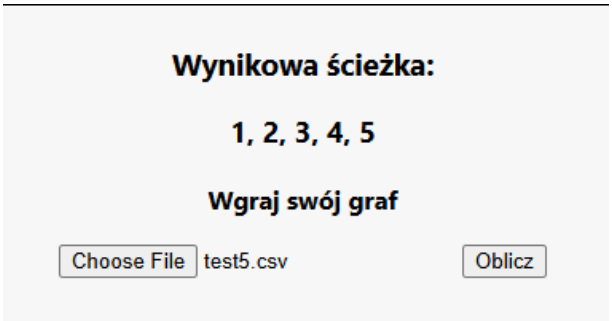
Na wyjściu aplikacja zwraca ścieżkę w postaci kolejnych wierzchołków grafu oraz rysuje graf z zaznaczoną na nim szukaną ścieżką.

4.3 Interfejs i funkcjonalności

Aplikacja posiada podstawowy interfejs graficzny. Główną część strony zajmuje obszar w którym rysowany jest wynikowy graf razem z zaznaczoną optymalną ścieżką.

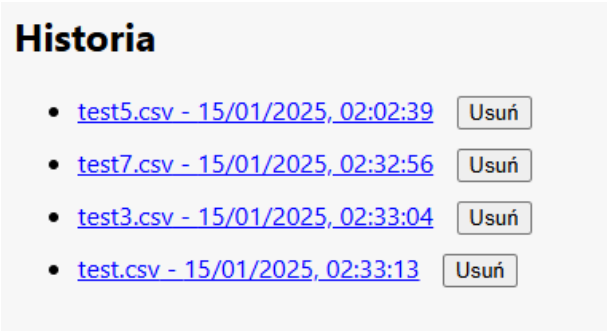


Poniżej znajduje się pole w którym wypisywana jest znaleziona ścieżka, a także przycisk Choose File służący do wgrywania plików z danymi początkowymi oraz przycisk Oblicz służący do uruchamiania algorytmu na przesłanym pliku.



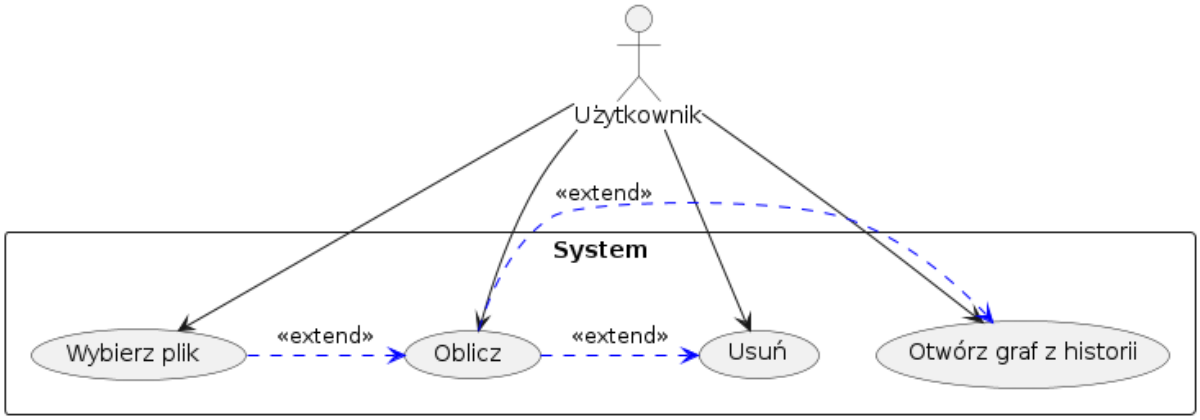
Na samym dole znajduje się sekcja **Historia**, w której zapisane są poprzednie użycia aplikacji.

Wybierając dany wpis w historii można odtworzyć wybrany graf razem z otrzymanym wynikiem, a korzystając z przycisku Usuń można usunąć dany rejestr z historii.



5 Diagram przypadków użycia

Poniżej przedstawiony jest diagram przypadków użycia naszej aplikacji:



6 Testy

W trakcie testów sprawdzaliśmy poprawność wyznaczania najkrótszej ścieżki przy pomocy algorytmu, poprzez porównanie wyniku otrzymanego z użyciem naszej implementacji z faktyczną najkrótszą ścieżką. Dla każdego z badanych przez nas przypadków testowych ścieżka wyznaczona algorytmem pokrywała się z faktyczną optymalną ścieżką, skąd wnioskujemy że algorytm został zaimplementowany poprawnie.