

**Zagadnienie komiwojażera**

**w sieci miejskiej**

**Bartosz Dec  
Michał Chudzik  
Aleksander Maciejewski**

Spis treści:

1.Wstęp

1.1 Podstawowe informacje

Zadaniem programu jest zwizualizowanie problemu zagadnienia komiwojażera w sieci miejskiej z uwzględnieniem dróg jednokierunkowych. Jest to problem polegający na znalezieniu optymalnej trasy między punkami, zaczynającą się oraz kończącą na konkretnym punkcie.

1.2 Wykorzystane technologie

Program został opracowany w środowisku programistycznym Visual Studio z wykorzystaniem języka c# oraz WPF.

2. Podstawy tematyczne

2.1 Opis zagadnienia

Zagadnienie komiwojażera to zagadnienie optymalizacyjne, polegające na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważony. Nazwa zagadnienia pochodzi od ilustracji problemu, przedstawiającej go z punktu widzenia wędrownego sprzedawcy nazwanego komiwojażerem. W tym problemie występuje dana liczba miejsc, przez które tytułowy komiwojażer musi przejść. Celem programu jest znalezienie najkrótszej/najszybszej trasy łączącej wszystkie punkty, zaczynając z podanego miejsca oraz powrotu do niego. Podczas obierania trasy komiwojażer musi również uwzględniać drogi jednokierunkowe występujące w sieci miejskiej, powodujące, że odległość od punktu A do punktu B, nie musi być taka sama jak odległość od punktu B do punktu A.

Początek badań nad tym problemem nie jest jasny. Pierwszy raz wspomniano o nim w podręczniku z 1832 roku. Była tam zawarta przykładowa trasa po Niemczech i Szwajcarii. Niestety w tym podręczniku brakowało matematycznych uzasadnień problemu.

Następnie w 1859 roku irlandzki matematyk William Rowan Hamilton sformułował problem ostatnia cyklu o długości n w grafie n-wierzchołkowym.

Za pierwszego autora problemu komiwojażera uznaje się austriackiego matematyka Karla Mengera, który jako pierwszy matematycznie sformalizował ten problem w 1930 roku. Zwrócił on szczególną uwagę na trudności wynikające z obliczania rozwiązania.

Pierwsza próba rozwiązania tego problemu miała miejsce w 1937 roku, gdy Merrill Flood pracował nad rozwiązaniem problemu wyznaczenia tras przez, które będą przejeżdżać autobusy szkolne.

Opis problemu jest bardzo prosty, jednak ma on opinię o bardzo trudnym obliczeniowo procesie optymalizacji. Mimo to problem komiwojażera stał się bardzo popularny. Zainteresowanie tym tematem trwa od lat pięćdziesiątych XX wieku do dziś.

Przykład:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Punkt A | Punkt B | Punkt C | Punkt D |
| Punkt A | 0 | 100 | 200 | 150 |
| Punkt B | 75 | 0 | 50 | 75 |
| Punkt C | 220 | 75 | 0 | 250 |
| Punkt D | 130 | 100 | 280 | 0 |

Zadaniem jest wyznaczenie najkrótszej/najszybszej trasy zaczynającej się np. w punkcie A, przechodząc jednokrotnie przez pozostałe punkty i wrócić z powrotem do punktu początkowego.

3. Opis programu

3.1 Podział programu na 2 wersje

Program zawiera dwie wersje: demonstracyjną oraz pełną.

Zarówno wersja demonstracyjna jak i pełna wersja zawierają zmodyfikowane, na potrzebę lepszej wizualizacji problemu, mapy Manhattanu, jednego z okręgów Nowego Yorku. Na ulicach widoczne są strzałki informujące, że dana ulica jest jednokierunkowa w kierunku zgodnym z kierunkiem strzałki. W sytuacjach, gdy na danej ulicy nie występuje żadna strzałka, na tej drodze jest stosowany ruch w obu kierunkach.

Program został podzielony na dwie części w celu testowania rozwiązań/algorytmów na wersji demonstracyjnej, aby następnie łatwiej zaimplementować uzyskane rozwiązania problemów w pełnej wersji.

Mapa do wersji demonstracyjnej jest dużo mniejsza co znacznie ułatwiało wprowadzanie danych potrzebnych do utworzenia macierzy umożliwiającej szukanie dróg między skrzyżowaniami.

Rozwiązania, które umożliwiły poprawne działanie wersji demonstracyjnej, zostały przeniesione i odpowiednio zmodyfikowane do pełnej wersji.

Mapa do wersji demonstracyjnej:



Mapa do pełnej wersji:



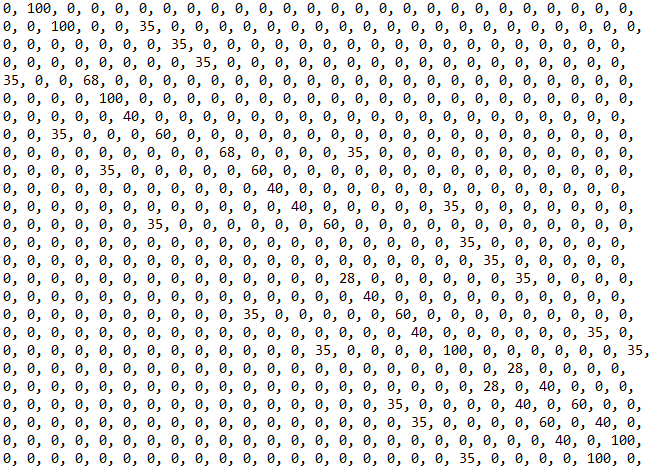
3.2 Utworzenie macierzy na podstawie mapy

Każde skrzyżowanie w obu wersjach programu ma swój unikatowy numer. W programie zapisane są położenia każdego skrzyżowania wraz z ich indeksem. Na ich podstawie powstała macierz incydencji, z której program będzie odczytywać kierunki dróg, w przypadku ulic jednokierunkowych, odległość z jednego skrzyżowania do drugiego czy możliwość przejścia z danego miejsca do innego.

Indeksy skrzyżowań w wersji demonstracyjnej:



Macierz incydencji stworzona na podstawie mapy z wersji demonstracyjnej



W macierzy są zapisane odległości między skrzyżowaniami. W przypadku, kiedy odległość z jednego skrzyżowania do drugiego jest równa zero, oznacza, że z danego miejsca nie można przejść do drugiego, ponieważ nie ma między nimi drogi lub jest tam droga jednokierunkowa, która nie pozwala przejść w danym kierunku. Jeśli w macierzy jest liczba inna od zera, to oznacza, że z danego skrzyżowania można przejść do drugiego i jest to odległość między nimi.

Takie samo rozwiązanie zostało zastosowane dla pełnej wersji programu, gdzie macierz jest większa, niż w wersji demonstracyjnej, ponieważ mapa wykorzystana w wersji pełnej zawiera więcej skrzyżowań.

3.3 Interfejs

3.3.1 Okno startowe

Przy uruchomieniu programu wyświetla się okno startowe, gdzie znajduje się nazwa zagadnienia. Dodatkowo na oknie są widoczne dwa duże przyciski decydujące o tym, z której wersji programu chce skorzystać użytkownik oraz mniejszy przycisk „Pomoc”, który ma pomóc użytkownikowi z wyborem co powinien wybrać, aby przejść do dalszej części programu.

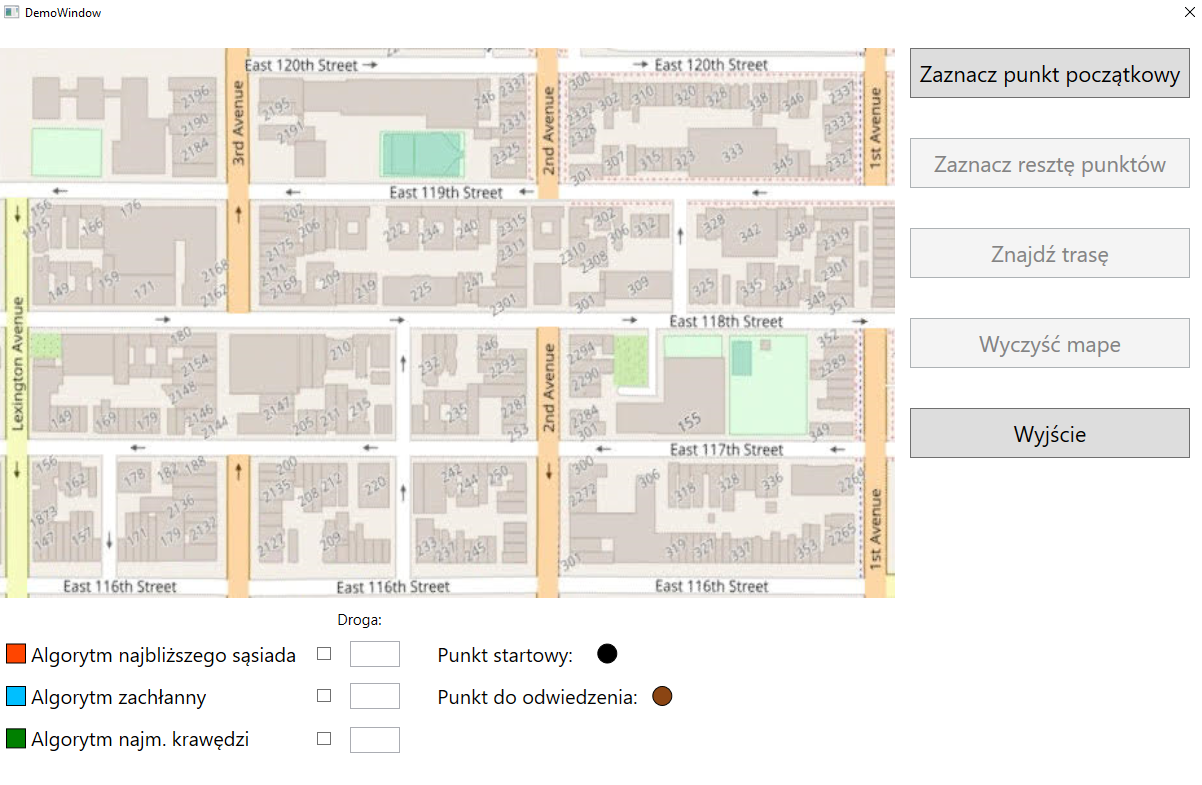
Obraz zawierający tekst

Opis wygenerowany automatycznie

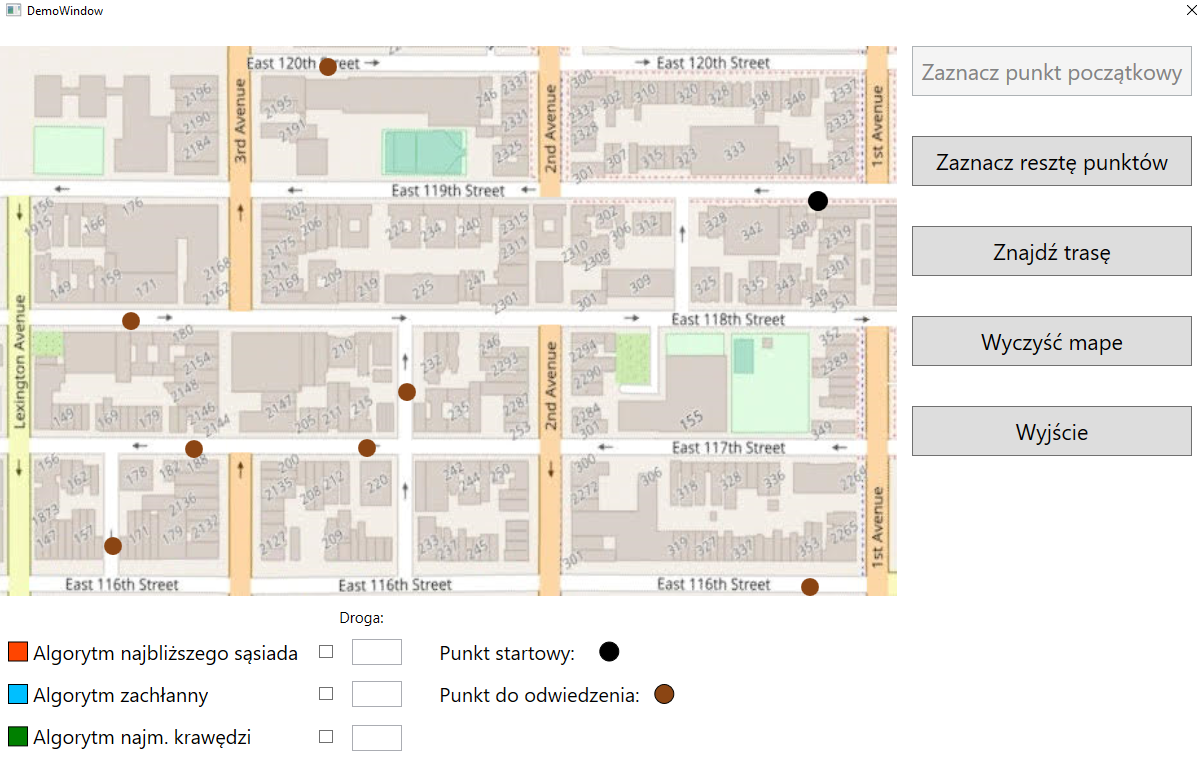
3.3.2 Wersja demonstracyjna

Gdy użytkownik wybierze wersję demonstracyjną, otworzy się okno zawierającą przeznaczoną dla niej mapę, przyciski służące do obsługi programu oraz legendę ilustrującą kolory punktu startowego, punktów do odwiedzenia oraz dróg wygenerowanych przez konkretne algorytmy. Przy algorytmach znajdują się okienka do zaznaczenia, które odpowiadają za to, z których algorytmów program ma skorzystać, aby znaleźć drogę pomiędzy zaznaczonymi wcześniej punktami. Użytkownik może wybrać wszystkie algorytmy jednocześnie, co poskutkuje narysowaniem dróg w kolejności, najpierw narysuje drogę z pierwszego, następnie z drugiego i na koniec z trzeciego algorytmu. Obok znajdują się okienka, w których, po narysowaniu trasy, wyświetli się długość drogi, która przeszedł dany algorytm.

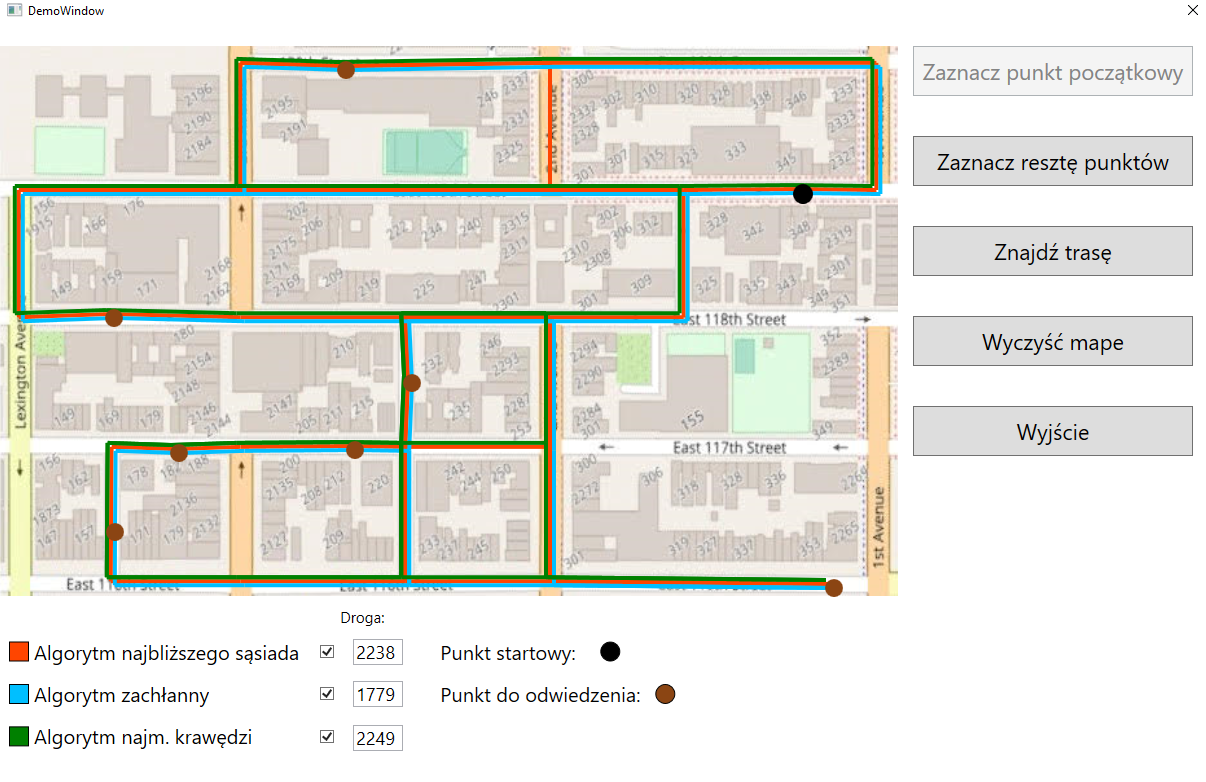
Okno bez wybrania punktów



Okno po zaznaczeniu punktów



Okno po użyciu przycisku „Znajdź trasę”



3.3.3 Wersja pełna

Wybór wersji pełnej na oknie startowym spowoduje wyświetlenie dużej mapy, przycisków oraz legendy jak w wersji demonstracyjnej. Różnicą między interfejsem wersji demonstracyjnej, a wersji pełnej jest zastosowanie innych przełączników przy wyborze algorytmu. Te w pełnej wersji pozwalają na zaznaczenie maksymalnie jednego algorytmu, z którego program ma skorzystać, aby wyznaczyć trasę między punktami.

Początkowe okno pełnej wersji programu

Obraz zawierający stół

Opis wygenerowany automatycznie

3.4 Przyciski w programie

3.4.1 Okno startowe

3.4.1.1 Wersja pełna

Wybór tej opcji spowoduje otworzenie okna zawierającego pełną wersję.

3.4.1.2 Wersja demonstracyjna

Wybór tej opcji spowoduje otworzenie okna zawierającego wersję demonstracyjną.

3.4.1.3 Pomoc

Otworzenie małego okienka zawierające informację na temat obsługi programu.

3.4.2 Wersja pełna i demonstracyjna

3.4.2.1 Zaznacz punkt początkowy

Pozwala ustawić na mapie jeden punkt początkowy na drodze, za pomocą dowolnego przycisku myszki, z którego algorytmy będą zawsze zaczynać szukanie trasy. W przypadku, gdy użytkownik będzie chciał zmienić jego położenie należy użyć przycisku „Wyczyść mapę” i zaznaczyć go na nowo.

3.4.2.2 Zaznacz resztę punktów

Pozwala zaznaczyć punkty na drogach, za pomocą dowolnego przycisku myszki, przez które komiwojażer musi przejść swoją trasę zaczynając i kończąc na punkcie startowym.

3.4.2.3 Przełączniki algorytmów

Przełączniki pozwalają wybrać algorytmy, z których skorzysta program podczas szukania trasy. W wersji demonstracyjnej zastosowane są przyciski wyboru, pozwalające na zaznaczenie wszystkich algorytmów. Z kolei w pełnej wersji zastosowane zostały przyciski radiowe, które pozwalają na zaznaczenie tylko jednego z algorytmów.

3.4.2.4 Znajdź trasę

Ten przycisk uruchamia zaznaczone algorytmy do szukania tras. Algorytmy, które nie zostały zaznaczone zostaną pominięte.

3.4.2.5 Wyczyść mapę

Usuwa z mapy wszystkie punkty oraz trasy wygenerowane przez przycisk „Znajdź trasę”. Usuwa również zaznaczenia przełączników algorytmów i długości tras, pokonanych przez algorytmy.

3.4.2.6 Wyjście

Kończy pracę programu i zamyka okno.

3.4.2.7 Blokowanie przycisków

Na początkowych oknach użytkownik ma możliwość skorzystania z przycisków „Zaznacz punkt początkowy”, „Wyjście” i do przełączników algorytmów. Po zaznaczeniu punktu początkowego użytkownik uzyskuje dostęp kolejnych przycisków „Zaznacz resztę punktów” i „Wyczyść mapę” oraz zablokowana zostaje możliwość kliknięcia przycisku „Zaznacz punkt początkowy”. Aby odblokować możliwość korzystania z ostatniego przycisku „Znajdź trasę” trzeba zaznaczyć chociaż jeden punkt poza startowym. Wtedy użytkownik może skorzystać z ostatniego przycisku i narysować trasę. Gdy użytkownik zdecyduje się narysować trasę, przycisk ponownie zostaje zablokowany i będzie musiał wyczyścić mapę za pomocą przeznaczonego do tego przycisku i rozpocząć od ponownego stawiania punktów i zaznaczenia algorytmów.

3.5 Dodawanie punktów na drogach

3.6 Algorytmy

3.6.1 Algorytm Dijkstry

Jest to algorytm służący do znajdowania najkrótszych ścieżek między węzłami w grafie. Został opracowany przez informatyka Edsgera W. Dijkstrę w 1956 roku i opublikowany trzy lata później. Dla danego węzła źródłowego w grafie algorytm wyznacza najkrótszą ścieżkę między tym węzłem, a pojedynczym węzłem docelowym w grafie, poprzez zatrzymanie algorytmu po określeniu najkrótszej ścieżki do węzła docelowego. Algorytm na początku swojego działania oznacza wszystkie węzły jako nieodwiedzone. Następnie uzupełnia odległość do każdego węzła. Wszystkie odległości są początkowo ustawione na nieskończoność, z wyjątkiem węzła początkowego, którego odległość zostaje zapisana jako zero. Wstępną odległość węzła v to długość najkrótszej odkrytej dotychczas ścieżki między węzłem v, a węzłem początkowym. Następnie algorytm dla bieżącego węzła rozważa jego wszystkich nieodwiedzonych sąsiadów i oblicza odległość do nich. Algorytm następnie porównuje nowo obliczone wartość z bieżąco wpisanymi i wybiera mniejszą. Kiedy zakończone zostanie rozpatrywanie wszystkich sąsiadów bieżącego węzła, zostaje on oznaczony jako odwiedzony i usunięty ze zbioru punktów nieodwiedzonych. Taki węzeł nie rozstanie już ponownie sprawdzony. Algorytm przechodzi tak przez wszystkie nieodwiedzone punkty, a następnie kończy swoją pracę.

Dane wejściowe:

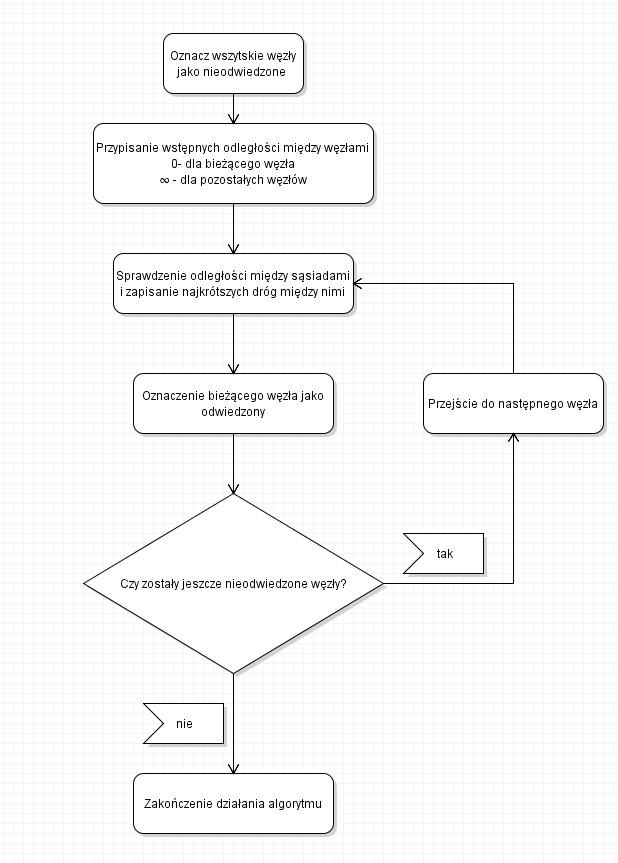
-macierz incydencji

Dane wyjściowe:

-najkrótsze odległości między węzłami

Złożoność czasowa: Θ(nlog n)

Złożoność pamięciowa: Θ(n+nlog n)



3.6.2 Algorytm najbliższego sąsiada

Ten algorytm wykorzystuje strategie zachłanną. Rozpoczyna swoją pracę od wierzchołka oznaczonego jako startowy. Ten wierzchołek zostaje oznaczony jako odwiedzony i aktualnie odwiedzany. Następnie algorytm sprawdza, który z pozostałych nieodwiedzonych punktów zaznaczonych przez użytkownika jest najbliżej za pomocą algorytmu Dijkstry. Po wyborze najkrótszej drogi zostaje ona dołączona do rozwiązania, a punkt do którego prowadzi ta droga zostaje zaznaczony jako odwiedzony i jest on aktualną pozycją wyjściową do szukania następnej drogi do punktów nieodwiedzonych. Algorytm powtarza swoje działanie do momentu, aż wszystkie punkty zaznaczone przez użytkowania zostaną odwiedzone. Gdy dojdzie do tego momentu wyznaczona zostaje trasa od ostatnio odwiedzonego punktu do punktu startowego. Po dodaniu tej drogi, algorytm zwraca pełną marszrutę.

Dane wejściowe:

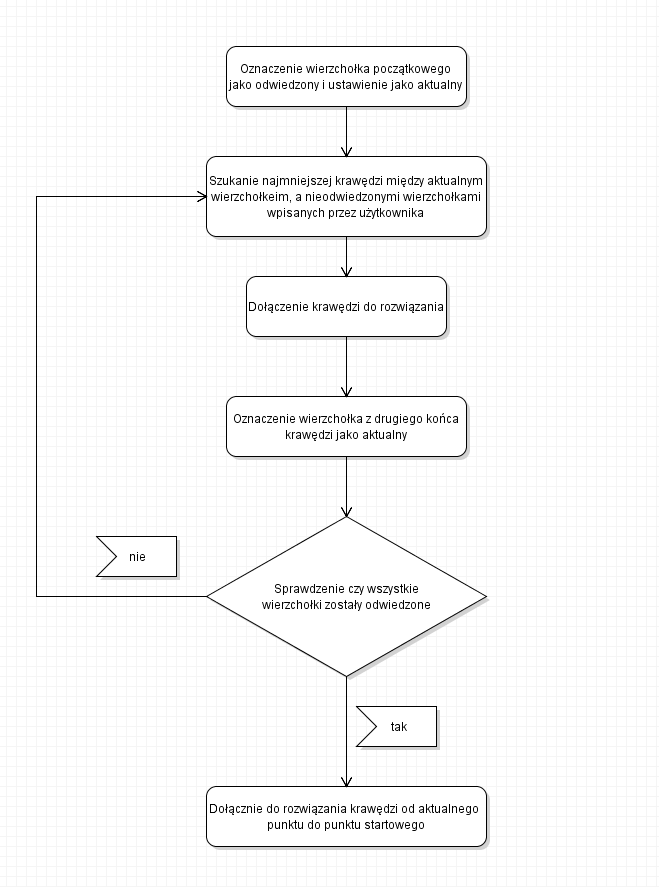
-macierz incydencji

Dane wyjściowe:

-Marszruta stworzona z punktów docelowych

Złożoność czasowa: Θ(N2)

Złożoność pamięciowa: Θ(N)



3.6.3 Algorytm Brute-force

Jest to algorytm szukający wszystkich możliwych permutacji zbioru punktów początkowych. Algorytm w trakcie szukania rozwiązań korzysta z indeksów punktów docelowych, zapisując ich każda kombinację. Następnie na początku i na końcu każdej permutacji dodawany jest indeks punktu początkowego, aby uzyskać pełną trasę. Kolejnym krokiem algorytmu jest porównanie długości każdej drogi w celu wybrania najkrótszej z nich. Algorytm do porównania długości marszrut korzysta z algorytmu Dijkstry, który zwraca najkrótszą drogę między 2 punktami. Na początku porównywania długości algorytm wybiera pierwszą trasę jako najkrótszą, a następnie porównuje do niej kolejne drogi. Jeśli dana trasa jest krótsza od aktualnie oznaczonej jako najkrótsza zastępuje jej miejsce, a algorytm przechodzi tak przez dalsze drogi. Po wybraniu najkrótszej marszruty otrzymujemy ją w postaci listy punktów.

Obraz zawierający akcesorium, parasol

Opis wygenerowany automatycznie

Zakładając wierzchołek z numerem 1 jako początkowy z powyższego grafu możemy wyszczególnić marszruty:

-1,2,3,4

-1,2,4,3

-1,3,2,4

-1,3,4,2

-1,4,2,3

-1,4,3,2

Dane wejściowe:

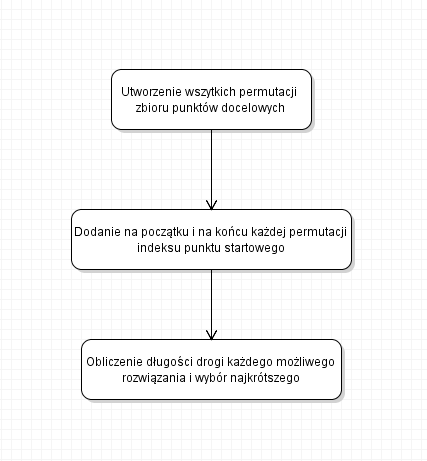
-macierz incydencji

Dane wyjściowe:

-Marszruta stworzona z punktów docelowych

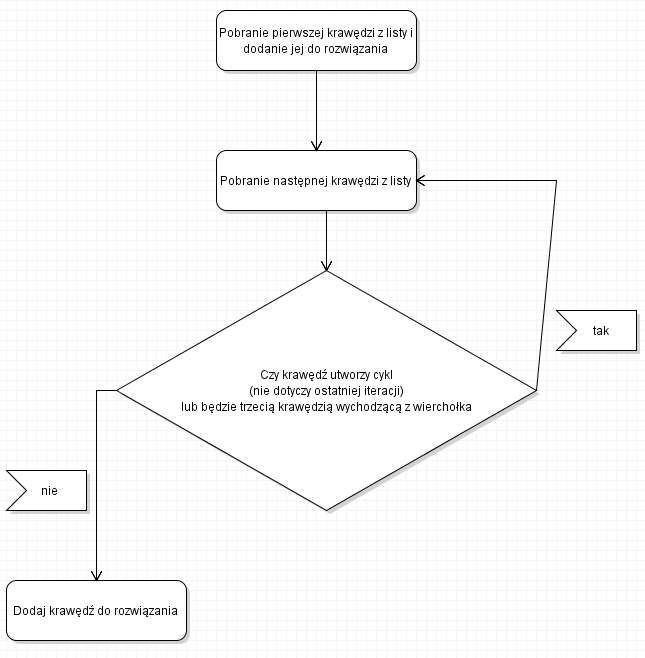
Złożoność czasowa: Θ(N!)

Złożoność pamięciowa: Θ(N!)

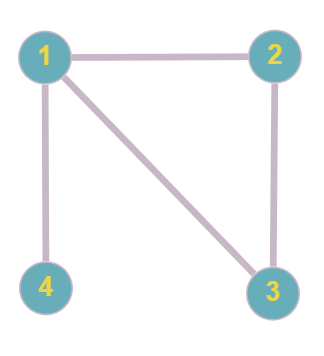
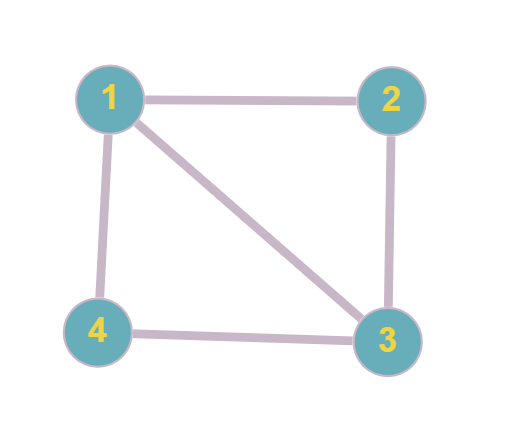


3.6.4 Algorytm najmniejszej krawędzi

Ten algorytm polega na kolejnym dołączaniu do rozwiązania najkrótszych spośród dopuszczalny krawędzi. Początkowo algorytm zapisuje wszystkie krawędzie z ich długościami między wszystkimi punktami docelowymi w liście, a następnie sortuje je rosnąco względem długości. Algorytm wybiera kolejne krawędzi z listy i próbuje dołączyć je do rozwiązania. Podczas próby dołączenia krawędzi do rozwiązania algorytm sprawdza czy nie spowoduje ona utworzenia cyklu (nie dotyczy ostatniej iteracji) lub powstania wierzchołka, z którego wychodzą trzy krawędzie, jeśli dana krawędź spełnia chociaż jeden z tych warunków nie będzie brana pod uwagę w rozwiązaniu, a algorytm przejdzie do następnej krawędzi z listy. Po utworzeniu połączeń między punktami docelowymi algorytm ustawia je tak, aby pierwszym punktem na liście wynikowej był punkt początkowy, a następnie kolejne punkty które łączą krawędzie. Po wykonaniu tego wszystkiego algorytm zwraca marszrutę stworzoną z punktów docelowych.

Schemat krokowy:

Niedopuszczlane sytuacje:

Obraz zawierający tekst, zegar

Opis wygenerowany automatycznie

Dane wejściowe:

-macierz incydencji

Dane wyjściowe:

-Marszruta stworzona z punktów docelowych

Złożoność czasowa: Θ(n2log n)

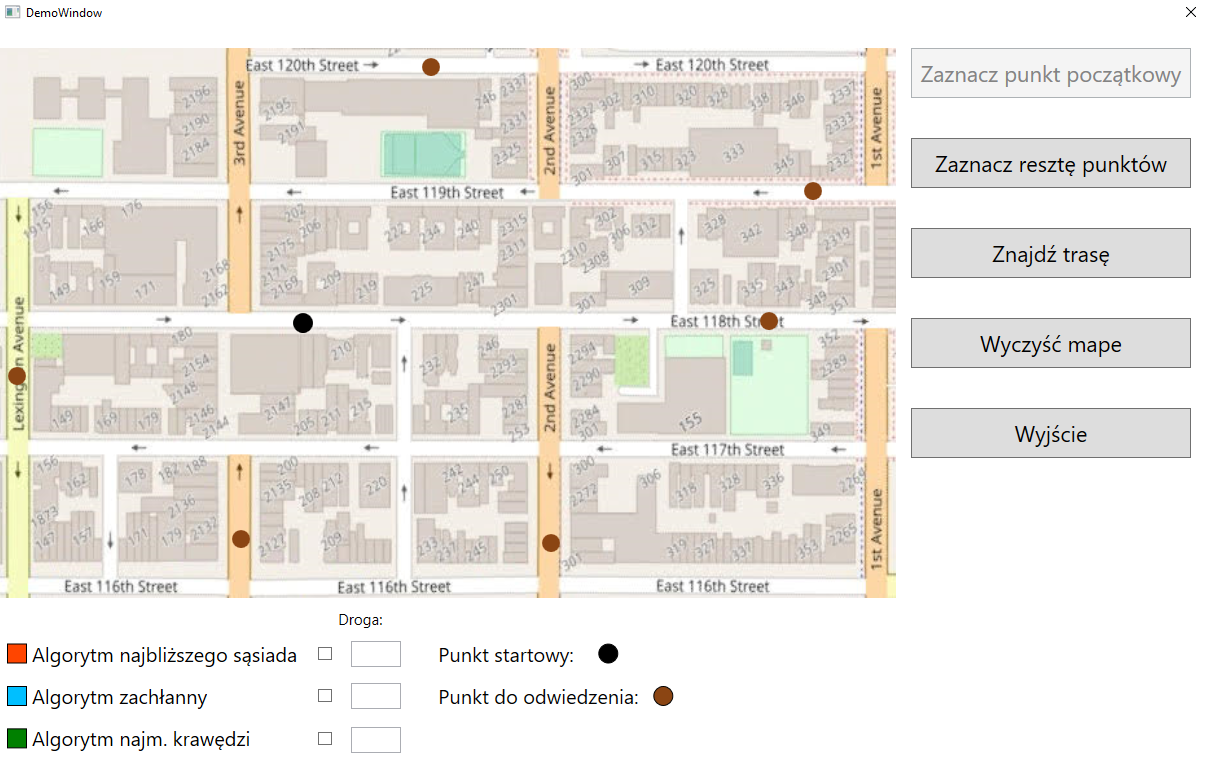
Złożoność pamięciowa: Θ(n2log n)

4. Wymagania sprzętowe

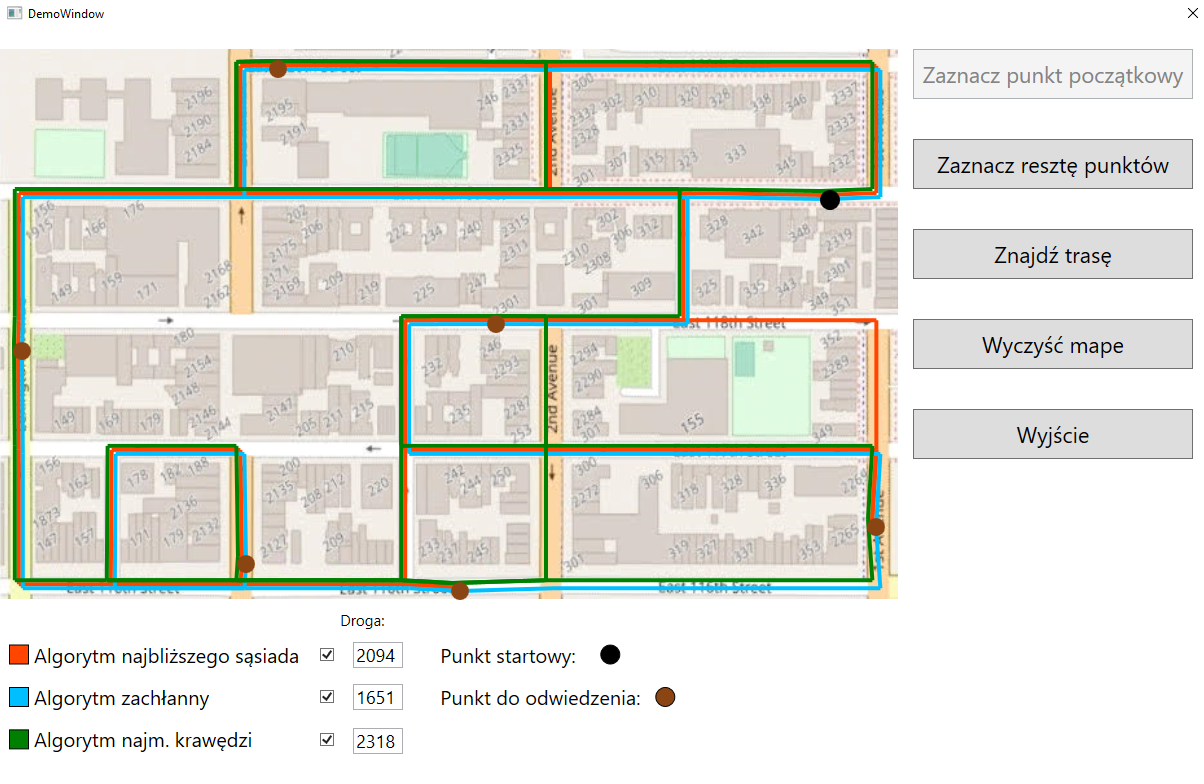
5. Instrukcja obsługi

5.1 Przykładowe rozmieszczenie punktów

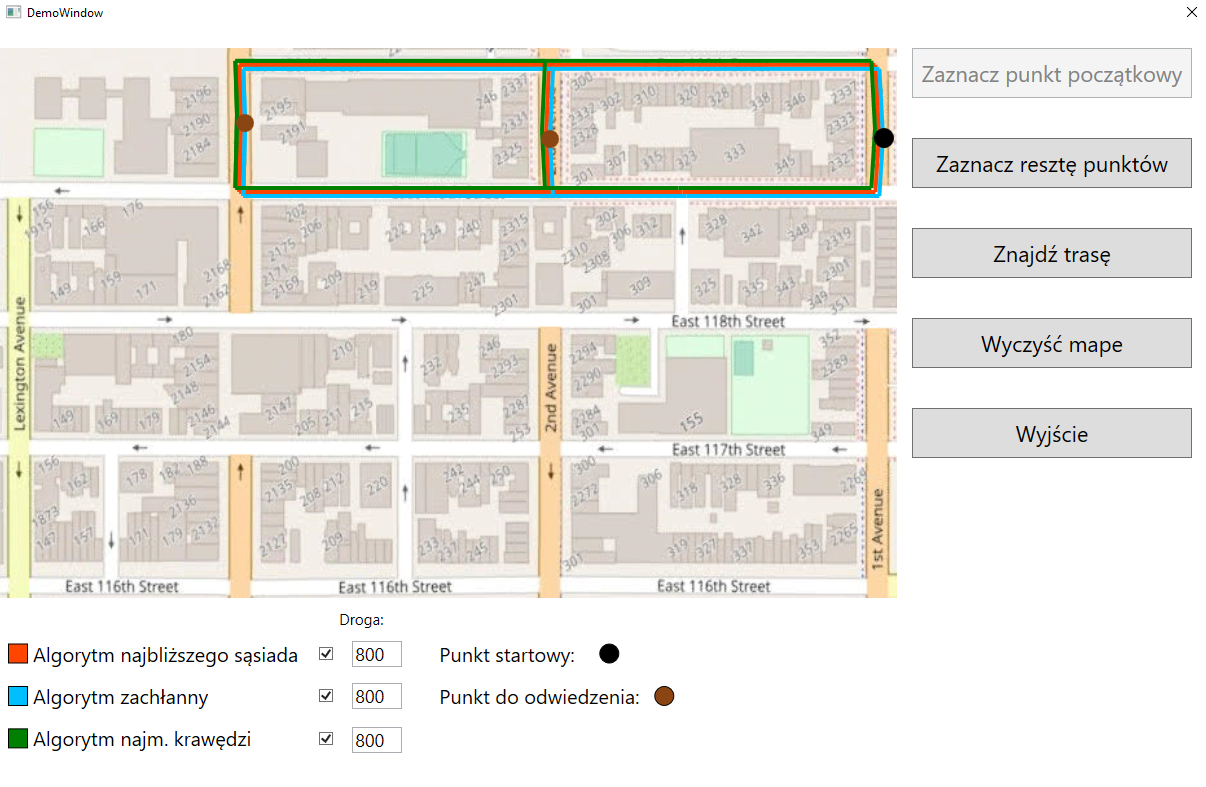
Rozmieszczenie punktów



Uzyskanie różnych tras przez algorytmy



Uzyskanie takich samych tras przez wszystkie algorytmy



6. Bibliografia

-Zagadnie komiwojażera:

<https://pl.wikipedia.org/wiki/Problem_komiwoja%C5%BCera>

<https://en.wikipedia.org/wiki/Travelling_salesman_problem>

-Algorytm Dijkstry

<https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm>

<https://www.programiz.com/dsa/dijkstra-algorithm>

-Algorytm najbliższego sąsiada

<https://en.wikipedia.org/wiki/Nearest_neighbour_algorithm>

<http://algorytmy.ency.pl/artykul/algorytm_najblizszego_sasiada>

-Algorytm zachłanny (Brute-force)

<https://en.wikipedia.org/wiki/Brute-force_search>

-Algorytm najmniejszej krawędzi

<http://algorytmy.ency.pl/artykul/algorytm_najmniejszej_krawedzi>