Spis treści

[1. Wstęp 2](#_Toc63849737)

[2. Opis problemu 3](#_Toc63849738)

# Wstęp

Zagadnienie przejazdu z punktu A do punktu B jest ważnym zagadnieniem optymalizacyjnym ze względów ekonomicznych. W zależności od potrzeb danej osoby lub firmy możemy rozpatrywać optymalizację względem najkrótszej drogi przejazdu, najszybszego czasu przejazdu, czy też najmniejszego spalania paliwa. Niniejsza praca opisuje zagadnienie najkrótszego czasu przejazdu w sieci miejskiej, czyli zmienne w czasie wagi łuków. W tym wypadku wagi łuków, to są aktualne prędkości dróg – czasami drogi są puste, a innym razem zakorkowane. Jest to oczywiście zmienne w czasie oraz względnie losowe. Miasto posiada też ulice jednokierunkowe, co często będzie oznaczało w praktyce, że droga z A do B będzie inna niż droga z B do A.

Program odwzorowuje to zagadnienie, dla środkowej części Warszawy, dla głównych ulic w Warszawie. Został on napisany w języku zorientowanym obiektowo – C# w środowisku programistycznym Microsoft Visual Studio 2017. Interfejs użytkownika bazuje na WPF, czyli Windows Presentation Foundation - nazwa silnika graficznego i API bazującego na .NET 3. Algorytmem liczącym drogę jest algorytm Dijkstry.

# Opis problemu

## Założenia

Warszawa jest dużym miastem, więc ze względów technicznych mapa musiała zostać okrojona do prostokąta ograniczonego z góry Targówkiem mieszkaniowym, z prawej rondem wiatraczna, z dołu metrem politechnika oraz z lewej muzeum powstania warszawskiego. Co więcej, większości ulic; w szczególności tych mniejszych na mapie nie ma, ponieważ z logicznego punktu widzenia, najoptymalniejsze drogi będą ulicami głównymi, ponieważ są tam wyższe ograniczenia prędkości oraz są one najczęściej niejednopasmowe.

Ulice będą miały prędkości - 30, 40. Można też zasymulować korek sprawiając, że dana droga będzie mieć 10. Co więcej niektóre ulice są jednokierunkowe - nie tak jak w Warszawie, co jest symulacją np. jakichś robót, remontów lub innych utrudnień. Algorytm najpierw oblicza odległości węzłów (skrzyżowań) od siebie. Potem z uwzględnieniem prędkości obliczany jest algorytm Dijkstry, który wyznacza trasę A,A1,A2,,,An,B.

Następnie samochód porusza się z punktu A do punktu A1. W każdym momencie jazdy z punktu A do A1 aktualizowana jest odległość docelowa. Po dojechaniu do A1 z prawdopodobieństwem następuje ponowne losowanie prędkości dróg. Jeśli nic się nie zmieniło, to trasa jest kontynuowana. Jeśli jednak nastąpi losowanie, to wtedy algorytm Dijkstry jest ponownie wyznaczany z nowymi prędkościami – w rzeczywistości jedne drogi po pewnym czasie staną się bardziej przejezdne, a inne mniej; stąd też zmiana ich prędkości. Niezależnie od tego, czy nastąpiło losowanie, czy nie, można zasymulować korek na dowolnej drodze. Następnie algorytm Dijkstry wyznacza nową trasę od punktu A1 do B, która może być inna niż wcześniejsza, czyli A1,A2,,,,An,B

## Schemat krokowy

1. Uruchom plik wykonywalny.  
2. Znajdź na mapie indeks skrzyżowania początkowego oraz końcowego.  
3. Wprowadź indeksy skrzyżowań do algorytmu.  
4. Wyznacz trasę.  
5. (Opcjonalnie) Dodaj korki.  
6. Wykonaj ruch pojazdu.  
7. Z prawdopodobieństwem następują zmiany prędkości dróg.  
8. Jeśli skrzyżowanie końcowe nie zostało osiągnięte, wróć do punktu 4.  
9. Jeśli skrzyżowanie końcowe zostało osiągnięte, można wybrać ponowne skrzyżowania początkowe i końcowe, wracając do punku 2.