

## Wydział Informatyki i Telekomunikacji

## Sztuczna Inteligencja i Inżynieria Wiedzy

## Lista nr 1

## *Web security 2*

# Jakub Krupiński 255356

### Wprowadzenie

#### Algorytm Dijkstry

Funkcja kosztu, która została tutaj użyta to czas pomiędzy czasem przyjazdu do stacji końcowej a czasem obecnym (w sekundach).

#### Algorytm A\* - kryterium czasu

Funkcja kosztu, która została została tu użyta to czas pomiędzy czasem przyjazdu do stacji końcowej a czasem obecnym (w sekundach), do którego doliczona została wartość odległości Manhattan pomiędzy przystankiem początkowym a końcowym pomnożona o 1000 w celu odpowiedniego wyskalowania wyniku.

#### Algorytm A\* - kryterium przesiadek

Funkcja kosztu, która została tu użyta to czas pomiędzy czasem przyjazdu do stacji końcowej a czasem obecnym (w sekundach), do którego doliczana jest wartość ‘50’, jeśli wymagana będzie przesiadka.

#### Algorytm A\* - optymalizacje

* Zaimplementowanie kolejki priorytetowej w oparciu o stertę
* Zaimplementowanie grafu w oparciu o słowniki, w celu uzyskania stałego czasu wyszukiwania wartości
* Odrzucanie tych krawędzi, których czas odjazdu jest wcześniejszy niż obecny czas

### Klasy użyte w implementacji

#### Node – wierzchołek grafu, który zawiera w sobie nazwę przystanku, oraz jego długość i szerokość geograficzną.

#### Edge – krawędź grafu, określa połączenie między dwoma wierzchołkami. Zawiera w sobie indeks, nazwę ‘firmy’ (np. MPK Wrocław), wierzchołki: startowy i końcowy, czas odjazdu oraz czas przyjazdu.

#### Graph – klasa reprezentująca pełen graf, zawiera w sobie słowniki, które przypisują wierzchołki oraz krawędzie do odpowiadającym im nazwom przystanków.

#### DataReader – klasa, która służy mi do odczytania i odpowiedniego wypakowania zawartości pliku ‘connection\_graph.csv’ do obiektów klas ‘Node’, ‘Edge’ oraz ‘Graph’.

### Założenia i problemy

Dla ułatwienia założyłem, że wszystkie przystanki o tej samej nazwie można połączyć w jeden przystanek.

Podczas wykonywania symulacji zaimplementowanych przeze mnie algorytmów odkryłem błąd, który powodował, że za każdym razem, kiedy algorytmy natrafiały na przystanek „Zorawina – Niepodległosci (Mostek)”, program zatrzymywał się pod wpływem błędu KeyError – z racji braku czasu nie zdążyłem poświęcić temu wystarczająco wiele czasu, żeby dojść do źródła tego problemu, tak więc do implementowanych algorytmów dodałem warunek, który sprawia, że jeśli miałyby one poprowadzić do tego właśnie przystanku, omijana jest jedna iteracja pętli.

Ze względu na brak czasu nie udało mi się zaimplementować przeszukiwania Tabu.

### Analiza wyników

### ???