Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

ALHE

Dokumentacja końcowa projektu

Mateusz Śledź, Maciej Paszylka

Warszawa, 2020

1.Treść projektu

SK.ALHE.1

Zaimplementowac i przetestowac algorytm A∗ dla zadania komiwojazera. Wejsciem aplikacji jest plik z lista współrzednych kolejnych punktów. Wyjsciem aplikacji jest najkrótsza sciezka. Porównac działanie algorytmu A∗ z ”brutalnym”przeszukiwaniem grafu. Zastosowanie dodatkowego algorytmu bedzie do- datkowym atutem. Dane pobrac ze strony http://sndlib.zib.de/home.action, dla sieci cost266.

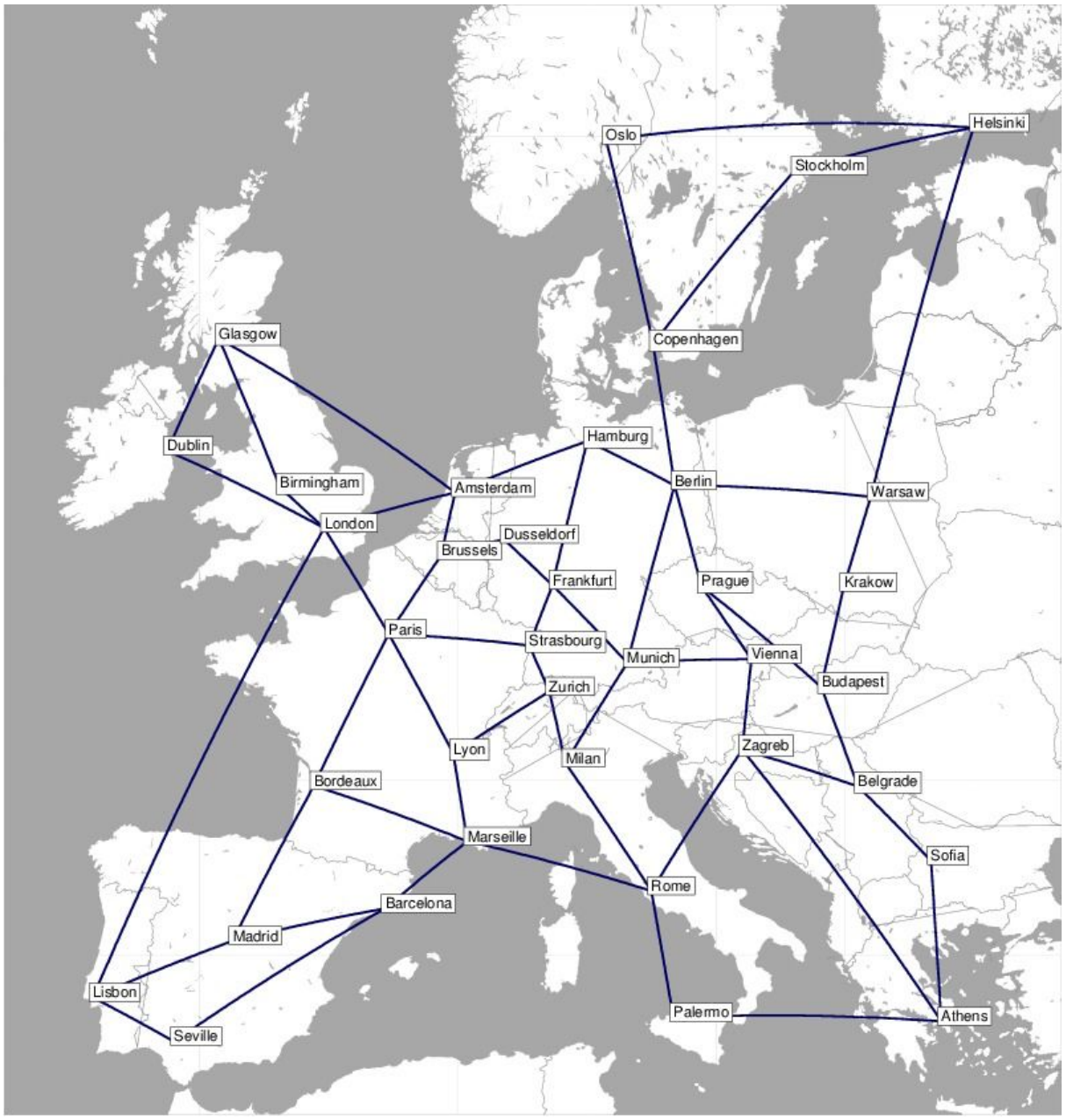
2. Analiza problemu

Do rozwiązania problemu przyjęliśmy następujące założenia:

1. Plik z danymi zostanie pobrany ze wskazanej strony, następnie zostaną z niego wyłuskane informacje o współrzędnych miast.
2. Z pozyskanych współrzędnych zostaną policzone oraz zapisane odległości pomiędzy miastami.
3. Sieć połączeń jest pełnym grafem nieskierowanym.

3. Dane

Zgodnie z treścią projektu dane zostały pobrane ze strony http://sndlib.zib.de/home.action. W dokumentacji wstępnej założyliśmy, że do realizacji algorytmu utworzymy graf pełny z wykorzystaniem współrzędnych miast podanych w pliku cost266.txt, a pominiemy siatkę połączeń, która jest tam załączona. Poniżej mapa miast, które brały udział w naszym rozwiązaniu:



4. Opis algorytmu oraz realizacja.

Do reprezentacji grafu wybraliśmy macierz sąsiedztwa, która zostanie zmodyfikowana tak, aby od razu wskazywała wagę przejścia z punktu do punktu zamiast standardowego podejścia - 0 świadczącego o braku krawędzi i 1 w przypadku, gdy krawędź łączy dwa punkty. W naszym programie waga wpisana w daną komórkę tabeli informuje o połączeniu między tymi punktami.

W naszym zadaniu głównym punktem było rozwiązanie problemu komiwojażera wykorzystując algorytm A*. Nasza funkcja decyzyjna ma następującą postać:

$$f(n) = g(n) + h(n)$$

gdzie:
g(n) - suma wag krawędzi, które należą już do ścieżki plus waga krawędzi łączącej aktualny węzeł z n.
h(n) - funkcja heurystyczna na którą składają się 3 czynniki:

- odległość od n do najbliższego nieodwiedzonego węzła
- odległość od najbliższego nieodwiedzonego węzła do punktu startowego
- suma wag krawędzi minimalnego drzewa rozpinającego utworzonego na węzłach nieodwiedzonych (bez n). Do wyliczenia heurystyki został użyty algorytm Prima - algorytm zachłanny, który z każdego odwiedzonego wierzchołka wybiera kolejny, nieodwiedzony, o najniższym koszcie dotarcia

Jest to algorytm szukający najkrótszej ścieżki w grafie ważonym, który na bieżąco wylicza również dolne ograniczenie kosztu dotarcia od aktualnego punktu do końca zadanej ścieżki.

Algorytm „brute force” będzie przeszukiwał każdą możliwą drogę od zadanego punktu startowego i wybierał tę o najmniejszym koszcie.
Jako że rozmiar problemu jest duży (dla 37 miast złożoność wynosi O(37!)) algorytm brute force na pewno nie wykona się w rozsądnym czasie, dlatego będzie on wykonywał obliczenia do momentu osiągnięcia liczby tras podanej jako parametr wywołania.

5. Sposób testowania

Program będzie testowany dla różnych punktów początkowych z dostępnych miast. Oto przykładowe rozwiązania łącznie z trasą i kosztem przejazdu.

ALGORYTM A *

```
Path found:
Warsaw
Budapest
Krakow
Vienna
Prague
Berlin
Hamburg
Frankfurt
Strasbourg
Zurich
Milan
Munich
Zagreb
Belgrade
Sofia
Athens
Palermo
Marseille
Lyon
Paris
Brussels
Amsterdam
Dusseldorf
Oslo
Copenhagen
Stockholm
Barcelona
Bordeaux
Madrid
Seville
Lisbon
London
Birmingham
Dublin
Glasgow
Rome
Helsinki
Warsaw
Score: 223.9051643915529
```

BRUTE FORCE

```
Calculating 500000 routes...
Best found route is:
Warsaw
Sofia
Madrid
Strasbourg
Vienna
Krakow
Seville
Zagreb
Barcelona
Amsterdam
Athens
Dublin
Munich
Marseille
Birmingham
London
Lisbon
Belgrade
Oslo
Glasgow
Prague
Helsinki
Stockholm
Brussels
Dusseldorf
Lyon
Palermo
Zurich
Paris
Hamburg
Rome
Frankfurt
Copenhagen
Budapest
Milan
Berlin
Bordeaux
Warsaw
With cost 460.90092381635486
```

```
Path found:
Amsterdam
Brussels
Dusseldorf
Frankfurt
Strasbourg
Zurich
Milan
Marseille
Lyon
Paris
London
Birmingham
Dublin
Glasgow
Hamburg
Copenhagen
Berlin
Prague
Vienna
Zagreb
Budapest
Krakow
Warsaw
Belgrade
Sofia
Athens
Munich
Rome
Palermo
Barcelona
Bordeaux
Madrid
Seville
Lisbon
Oslo
Stockholm
Helsinki
Amsterdam
Score: 208.4432255197842
```

```
Calculating 500000 routes...
Best found route is:
Amsterdam
Lyon
Zurich
Warsaw
Brussels
Dusseldorf
Glasgow
Sofia
Zagreb
Berlin
Palermo
Stockholm
Copenhagen
Athens
Lisbon
Marseille
Budapest
Vienna
Strasbourg
Prague
Madrid
Seville
Hamburg
London
Oslo
Barcelona
Birmingham
Helsinki
Dublin
Munich
Milan
Paris
Rome
Bordeaux
Krakow
Belgrade
Frankfurt
Amsterdam
With cost 441.826225624312
```

6. Wnioski:

Zgodnie z przewidywaniami, algorytm A* pozwolił na szybkie odnalezienie trasy dla zadanego problemu. Dodatkowo, heurystyka wykorzystująca minimalne drzewo rozpinające jako składową dzięki, której przewidujemy następny krok naszego algorytmu, pomogła jeszcze bardziej zmniejszyć koszt przejścia po całym grafie i powrotu do punktu startowego. Algorytm brute force w zależności od parametru, który jest liczbą wyszukanych tras daje lepsze lub gorsze wyniki, jednak nadal nieporównywalnie gorsze do trasy znalezionej przez drugi algorytm.

7.Wykorzystane technologie

Do implementacji został użyty język Python.

8. Sposób uruchomienia

Uruchamianie programu w konsoli poleceniem:

```
./python3 program.py miasta.txt
```

miasta.txt - plik ze współrzędnymi miast, w naszym przypadku cost266.txt