PSZT - przeszukiwanie

Stawczyk Przemysław 293153, Piotr Zmyślony 268833

1 Opis Zagadnienia

1.1 Treść zadania

Zaimplementować i przetestować algorytm $A*$ dla zadania znalezienia ścieżki o najmniejszej wadze od punktu A do B. Wejściem aplikacji jest plik z listą krawędzi grafu (dla każdej krawędzi zdefiniowany jest punkt początkowy, końcowy I waga krawędzi). Wyjściem aplikacji jest najkrótsza ścieżka od punktu A do punktu B. Porównać działanie algorytmu $A*$ z *brutalnym* przeszukiwaniem grafu. Zastosowanie dodatkowego algorytmu będzie dodatkowym atutem przy ocenie projektu.

1.2 Narzędzia

Skrypty oraz algorytm zostały zaimplementowane w Pythonie 3.
Wykorzystano biblioteki: *xml, math, time, json, requests, heapq*

1.3 Realizacja

Po konsultacji wybrane dane wejściowe to grafy *Polska* oraz *Germany50* ze strony sndlib.zib.de. Dane przedstawione w formacie XML zostają przeprocesowane na plik XML, sformatowany w nieco inny sposób: wierzchołki (miasta) kodowane identycznie, krawędzie nie zawierają id, pojemności i kosztu, a odległość pomiędzy miastami, pominięte zostają również *zapotrzebowania* pomiędzy poszczególnymi miastami. Dane o odległości drogowej pobierane są z wykorzystaniem API serwisu *here.com*, a w przypadku błędu w odpowiedzi API, odległość wyliczana jest formułą haversine, która słóży również jako heurystyka dla algorytmu $A*$.

1.4 Heurystyka dla A*

Heurystyka używana przez algorytm $A*$ musi pozwalać na oszacowanie prawdopodobieństwa wyniku pozostałej części procesu analizy składniowej, przy ustalonym już, uprzednio przeanalizowanym fragmencie. Musi być problemem prostszym niż problem ogólny *[nie może być to np. rozwiązanie podproblemu algorytmem wymagajacym czasowo w stosunku do pierwotnego]*. Wynik działania funkcji powinien stanowić ograniczenie dolne kosztu dotarcia do celu *[nie da się tam dotrzeć szybciej, ale może być tak, że da się wolniej]*. W szczególnym przypadku, gdy heurystyka zwraca wartość 0 algorytm sprowadza się do algorytmu Dijkstry.
Wymagania spełnia formuła haversine zastosowana w naszej implementacji tego algorytmu.

Formuła haversine Formuła ta pozwala na otrzymanie przybliżonej długości łuku, który łączy dwa punktu na kuli, używając jako dane wejściowe szerokości i długości geograficznych obu punktów.

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

W powyższym wzorze r to promień Ziemi, φ i λ to odpowiednio szerokości i długości geograficzne.

2 Implementacja

2.1 Program

Skrypt zawierający algorytmy przyjmuje 4 argumenty :
<graf> <tryb> <początek> <cel>
przy czym tryby pracy programu są następujące:

1. alg. brutalny (przeszukiwanie wszystkich możliwych ścieżek)
2. algorytm A*
3. algorytm Dijkstry
4. A* oraz algorytm Dijkstry
5. Wszystkie 3 algorytmy
6. Tryb testowy, wymaga jedynie podania argumentów w postaci *<graf> 6 <ilość prób testowych>*. Porówuje wydajności A* oraz algorytmu Dijkstry na podstawie określonej ilości danych testowych (generowanych automatycznie).

Wynikiem działania dla wszystkich opcji są ścieżki, jej koszt (odległość) oraz czas działania. Czas działania jest mierzony tylko w momencie działania algorytmu. Funkcja heurystyczna korzysta z formuły haversine obliczającą odległość w linii prostej po powierzchni sfery pomiędzy miastami. Funkcja ta spełnia wymagania heurystyki w algorytmie A* gdyż stanowi dolne ograniczenie odległości pomiędzy miastami *[nie da się dotrzeć szybciej niż w linii prostej]*

2.2 Testy działania

Poniższa tabelka przedstawia czasy działania algorytmu dla jednej procesu wyszukiwania jednej ścieżki, uśrednione na podstawie 10000 powtórzeń dla A* i algorytmu Dijkstry.

Uśrednione wyniki		
Algorytm	średni czas dla grafu Polska	średni czas dla Germany50
Brutalny	375 μs	>15 minut
Dijkstra	19 μs	87 μs
A*	25 μs	41 μs

2.3 Analiza wyników

Algorytm Dijkstry oraz $A*$ są zdecydowanie szybsze od podejścia brutalnego, którego użycie dla grafów o większej ilości krawędzi i wierzchołków może prowadzić do złożoności *O(n!)*. Dodatkowo dzięki zastosowaniu algorytmu heurystycznego algorytm $A*$ sprawdza jedynie potencjalnie najlepsze ścieżki co ogranicza *rozprzestrzenianie* się algorytmu i zdecydowanie przyspiesza jego pracę dla odpowiednio dużych grafów. Jako, że funkcja haversine jest umiarkowanie skomplikowaną formułą to dla mapy Polski z 12 miastami i 18 krawędziami wzrost wydajności jest znikomy, ale już dla bardziej realistycznego zastosowania, w grafie dla mapy Niemiec, o 50 wierzchołkach znajduje dłuższe trasy szybciej.