Tomasz Załuska

Khanh Do Van

Raport

PSZT – przeszukiwanie, temat nr 7: Odśnieżanie

1.Opis problemu

Znalezienie optymalnej trasy piaskarki, aby jak najmniejszym kosztem odśnieżyć daną sieć drogową. Zasoby pojazdu pozwalają na odśnieżenie maksymalnie N-kilometrów – po tej odległości konieczny jest powrót do wyznaczonej bazy, gdzie zasoby są uzupełniane. Piaskarka zaczyna i kończy przejazd w bazie.

2.Wstępna analiza problemu

Po głębszej analizie problemu oraz poszukiwaniach podobnego problemu w Internecie doszliśmy do wniosku, że przedstawiony problem jest analogiczny do problemu chińskiego listonosza z ograniczoną pojemnością. Wynika z tego, że jest to problem NP-trudny.

3.Rozwiązanie

Aby rozwiązać podany problem stworzyliśmy własny algorytm, który składa się z dwóch głównych części:

a) Znalezienia w grafie sieci drogowej wszystkich ścieżek, których waga będzie mniejsza lub równa zasobom piasku jakie może przewieść piaskarka w pojedynczym przejeździe.
b) Wybór ze zbioru takich ścieżek, które będą tworzyć cały graf oraz będą to ścieżki o jak najmniejszym koszcie.

4.Przedstawienie algorytmów z punktu 3

a) Algorytm szukania ścieżek o koszcie mniejszym równym pojemności piaskarki
    0. A-lista ścieżek G- graf sieci drogowych B_i – lista krawędzi sąsiednich dla i-tej ścieżki i=0 j=0
    1. Dodajemy do A wszystkie krawędzie grafu G.
    2. Tworzymy nową ścieżkę P, która składa się z A[i] i B_i[j].
    3. Sprawdzamy czy ścieżka P ma wagę, mniejszą równą pojemności piaskarki. Jeśli tak to dodaj ścieżkę P do listy A.
    4. j = j+1
    5. Jeśli j jest różne od długości listy B_i to skocz do 2.
    6. i = i+1
    7. Jeśli i jest różne od długości listy A to skocz do 2.
    8. Usuń powtarzające się ścieżki z A.
b) Algorytm wybory ścieżek z A do utworzenia finalnej trasy piaskarki
    0. C – zbiór wybranych ścieżek D – zbiór krawędzi grafu G
    1. Dzielimy listę A na listy A_i, gdzie A_i to lista ścieżek o takiej samej ilości krawędzi.
    2. Sortujemy każde A_i względem sumy wagi ścieżki, wagi ścieżki prowadzącej do początku danej ścieżki z bazy, oraz wagi ścieżki prowadzącej od końca danej ścieżki do bazy.
    3. Dodajemy do C ścieżkę z A_i , gdzie i jest maksymalne o najmniejszej wadze. Usuwamy z D krawędzi tworzące wybraną ścieżkę.

4. Sprawdzamy czy wszystkie krawędzie kolejnej ścieżki z A_i znajdują się w D. Jeśli tak to dodajemy tą ścieżkę do C i krawędzie tworzące tą ścieżkę usuwamy z D. Postępujemy tak kolejno w każdym A_i.
5. Po przejrzeniu wszystkich A_i w C mamy wszystkie ścieżki, które tworzą trasę piaskarki.
6. Tworzymy finalną trasę piaskarki ze ścieżek ze zbioru C oraz ścieżek dojazdu oraz powrotu do bazy z danej ścieżki.(ścieżki te są znajdowane algorytmem Dijkstry)

5.Rozkład obowiązków:

a) Wymyślenie rozwiązania – Tomasz Załuska
b) Implementacja algorytmu szukającego ścieżek – Tomasz Załuska
c) Implementacja algorytmu wyboru ścieżek – Khanh Do Van
d) Implementacja funkcji pobierającej dane o grafie z piku tekstowego – Tomasz Załuska
e) Implementacja funkcji, która tworzy finalny przejazd na podstawie wybranych ścieżek – Khanh Do Van
f) Testowanie – Tomasz Załuska i Khanh Do Van

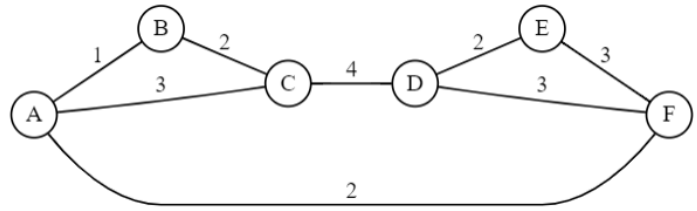
6.Implemantacja rozwiązania.

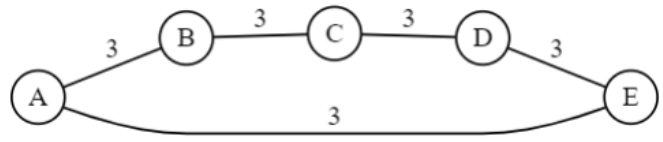
Do zaimplementowania rozwiązania użyliśmy języka Python z biblioteką NetworkX.

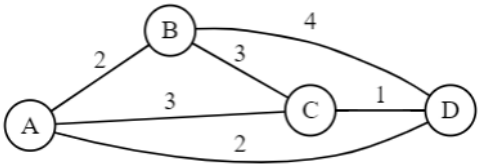
7.Testowanie.

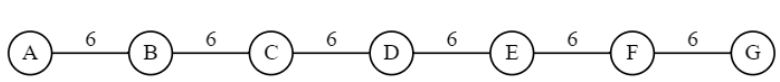
Do testowania użyliśmy prostych przykładów grafów, aby można było sprawdzić czy algorytm wybiera dobrą trasę dla piaskarki. Rozwiązania dla większych grafów ciężko by było wymiernie sprawdzić.

Przykłady testów:









Dla każdego przypadku testowego sprawdzaliśmy wybór trasy i koszt trasy w zależności od wybranego wierzchołka startowego(bazy). Dla wszystkich przedstawionych przykładów algorytm znajdował optymalną trasę. Sprawdzany był również wybór trasy w zależności od pojemności piaskarki.