Projektowanie Efektywnych Algorytmow

Projekt

18/10/0222

259091 Jakub Mordalski

(1) Brute force

1. Sformułowanie zadania

Celem zadania projektowego jest opracowanie oraz implementacja algorytmu przeglądu zupełnego rozwiązującego problem komiwojażera w wersji optymalizacyjnej, a następnie zbadaniu jego efektywności poprzez pomiar czasu wykonania algorytmu. Problem komiwojażera (TSP - Traveling Salesman Problem) to zagadnienie polegające na odnalezieniu minimalnego cyklu Hamiltona w grafie pełnym ważonym.

Graf pełny ważony – między każdą parą wierzchołków istnieje połączenie, a każda z krawędzi ma swoją wagę

Cykl Hamiltona – Cykl przechodzący prze każdy wierzchołek dokładnie raz, z wyłączeniem wierzchołka początkowego, do niego wracamy po przejściu wszystkich wierzchołków.

Problem Komiwojażera polega zatem na odnalezieniu ścieżki zamkniętej, która przechodzi przez wszystkie wierzchołki, kończąc w miejscu początkowym. Koszt ścieżki powinien być jak najmniejszy.

TSP występuje w dwóch wersjach:

Symetryczny - opiera się na grafie nieskierowanym.

Asymetryczny – opiera się na grafie skierowanym.

2. Metoda

Zastosowana w zadaniu metoda to metoda przeglądu zupełnego zwana też metodą przeszukiwania wyczerpującego lub metodą siłową.

Metoda ta polega na odnalezieniu i przetestowaniu wszystkich dopuszczalnych rozwiązań problemu, a następnie wyliczniu dla nich wartości funkcji celu i wyborze rozwiązania o warości ekstremalnej:

najniższej (problem minimalizacyjny)

najwyższej (problem maksymalizacyjny)

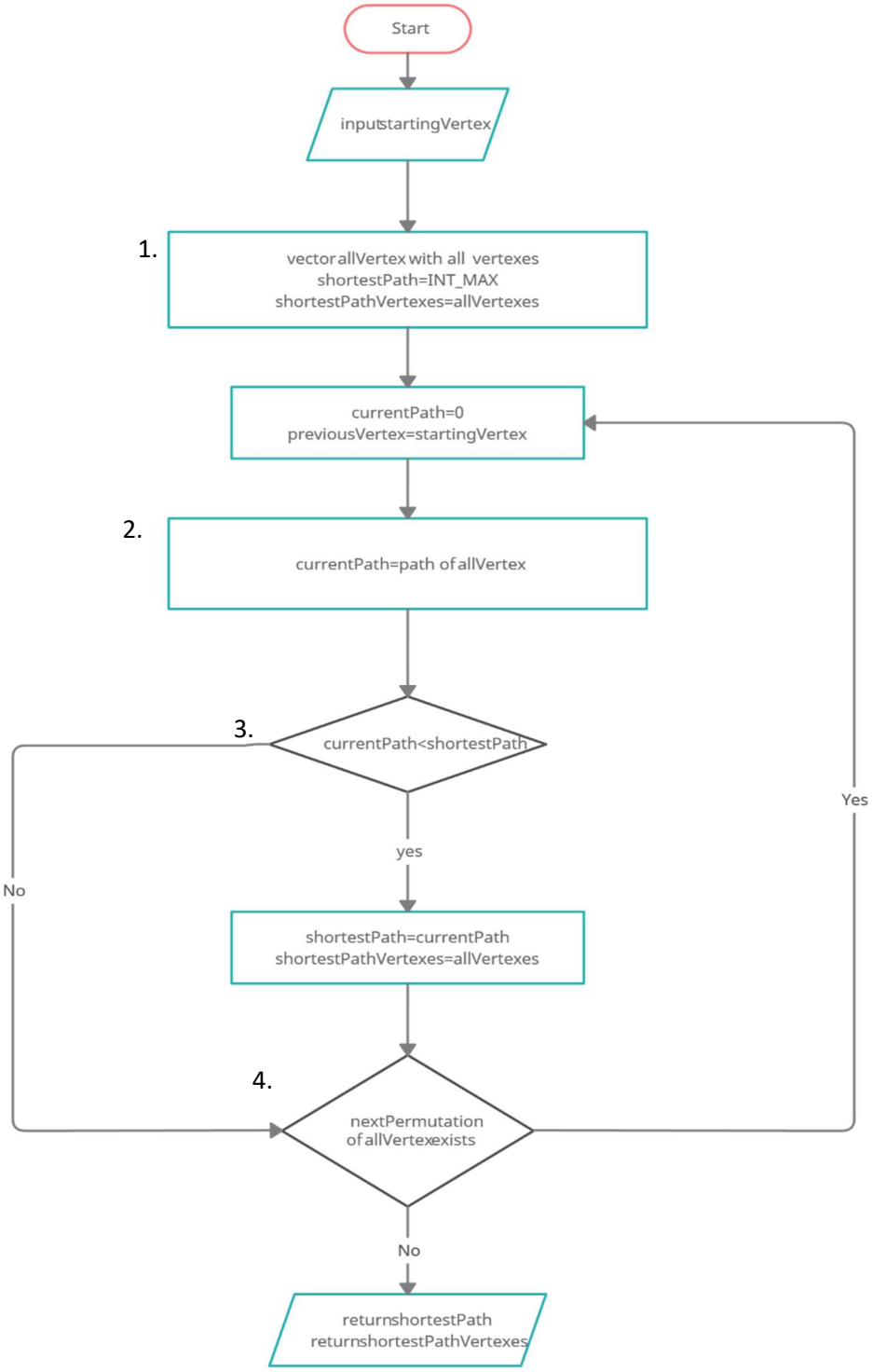
Co daje nam optymalny wynik

Minusem tej metody jest nieefektywność obliczeniowa ponieważ dla przypadku grafu nieskierowanego, złożoność obliczeniowa wynosi $\frac{(n-1)!}{2}$, natomiast dla grafu skierowanego (n-1)!.

Z racji na wysoką złożoność obliczeniową algorytm ten jest wydajny tylko dla niewielkich instancji.

3. Algorytm

Zastosowany algorytm polega na przejściu po wszystkich permutacjach zbioru wierzchołków, za każdym razem sprawdzając czy nowo odszukana ścieżka nie jest przypadkiem mniejsza od poprzedniej najmniejszej ścieżki



1.utworzenie wektora zawierającego wszystkie dostępne wierzchołki – w kolejności od najmniejszego indeksu do największego, utworzenie początkowych zmiennych dla najkrótszej wartości ścieżki oraz najkrótszej ścieżki.

2.obliczenie wartości ścieżki dla obecnie testowanej ścieżki. Poprzez iteracje przez wszystkie wierzchołki w aktualnej permutacji wektora

3. Podmiana wartości shortestPath i shortestPathVerticles jeżeli znaleziono krótszą ścieżkę

4. Tworzenie kolejnej permutacji do momentu wyczerpania wszystkich możliwości

4. Dane testowe

Do sprawdzenia poprawności działania algorytmu wybrano następujący zestaw instancji:

- tsp_6_1.txt
- tsp_6_2.txt
- tsp_10.txt
- tsp_12.txt
- tsp_13.txt

5. Procedura badawcza

Procedura badawcza polegała na uruchomieniu algorytmu dla instancji o różnych wielkościach, oraz zmierzeniu czasu działania algorytmu dla tych instancji. Dla algorytmu realizującego przegląd zupełny nie występowały parametry mające wpływ na wynik działania algorytmu.
Przy uruchomieniu używany jest plik sterujący conf.ini, zawierający następujące dane
<nazwa pliku> <ilość testów> <wielkość instancji> <optymalna długość ścieżki> <optymalna ścieżka>.
Zawartość pliku użytego podczas testów :

tsp_6_1.txt 100 6 132 0 1 2 3 4 5

tsp_6_2.txt 100 6 80 0 5 1 2 3 4

tsp_10.txt 50 10 212 0 3 4 2 8 7 6 9 1 5

tsp_12.txt 10 12 264 0 1 8 4 6 2 11 9 7 5 3 10

tsp_13.txt 1 13 269 0 10 3 5 7 9 11 2 6 4 8 1 12

Z racji na długi czas wykonywania algorytmu, test dla 13 instancji został wykonany tylko raz.

Plik wyjściowy

Wyniki Powyższego testu zostały zapisane w pliku result.csv, który zawierał następujące dane :
<długość ścieżki> <ścieżka> <czas wykonania>,

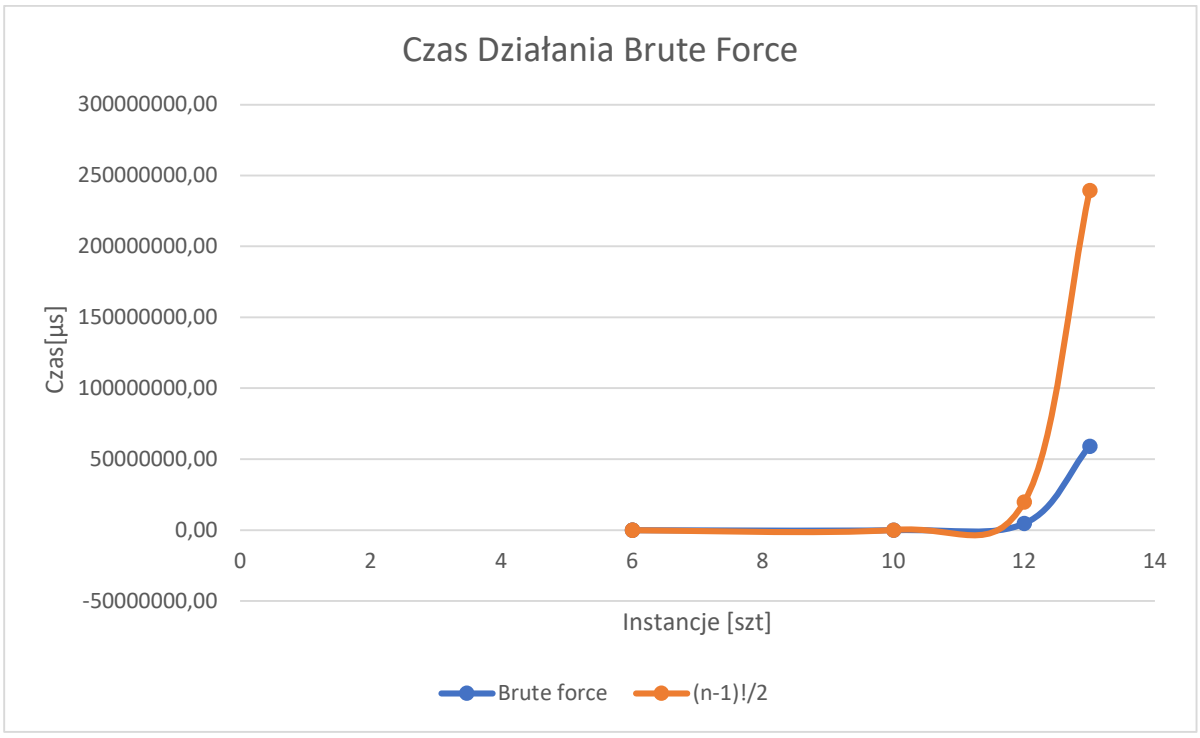
Testy dla poszczególnych plików zostały oddzielone linią zawierającą:
<nazwa pliku> <ilość testów> <wielkość instancji> <optymalna długość ścieżki> <optymalna ścieżka>

Test był przeprowadzany na platformie

- AMD Ryzen 5 3500X 4.1 GHz
- 16GB RAM DDR4 3400MHz CL16
- Samsung 500GB M.2 PCIe NVMe 970 EVO Plus

6. Wyniki

Wyniki działania programu zostały zapisane w pliku result.csv oraz zamieszczone na dysku google: https://drive.google.com/file/d/1k0Bgy-335UQ8vnZw47f5ddMiuhkN0zaj/view?usp=sharing



Rysunek 1: Wykres czasu działania algorytmu Brute Force względem ilości instancji

7. Analiza wyników i wnioski

Wykres czasu działania algorytmu BruteForce w zależności od ilości instancji jest wykładniczy.
Dla porównania dodany został wykres funkcji (n-1)!/2. Z porównania wynika że algorytm wyznacza optymalne rozwiązanie problemu komiwojażera w czasie (n-1)!/2 gdzie n oznacza liczbę instancji.
Złożoność czasowa algorytmu brute force wynosi O((n-1)!/2)

8. Źródła

https://pl.wikipedia.org/wiki/Problem_komiwoja%C5%BCera
https://pl.wikipedia.org/wiki/Cykl_Hamiltona
http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/