Projektowanie Efektywnych Algorytmów

Projekt

19.10.2021

256423 Maciej Radziszewski

(1) Brute force

1. Treść zadania

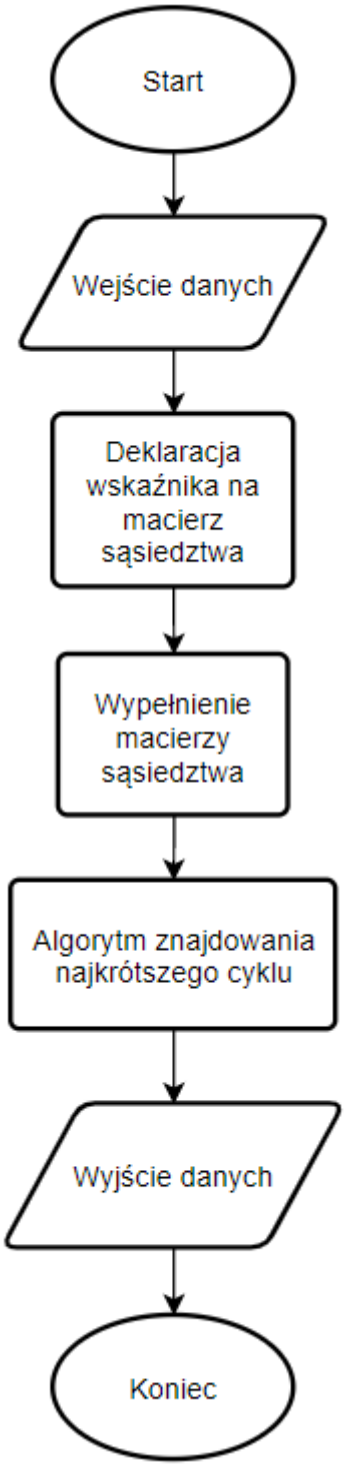
Zadane polega na opracowaniu, implementacji i zbadaniu efektywności algorytmu przeglądu zupełnego rozwiązującego problem komiwojażera w wersji przeglądu zupełnego. Problem komiwojażera polega na odnalezieniu minimalnego lub maksymalnego (w zależności od wymagań) cyklu Hamiltona w grafie. Graf, który jest brany pod uwagę musi być pełny, oraz jego krawędzie muszą mieć wagi.

2. Metoda

Metoda przeglądu zupełnego, tzw. przeszukiwanie wyczerpujące (eng. exhaustive search) bądź metoda siłowa (eng. brute force), polega na znalezieniu i sprawdzeniu wszystkich rozwiązań dopuszczalnych problemu, wyliczeniu dla nich wartości funkcji celu i wyborze rozwiązania o ekstremalnej wartości funkcji celu – najniższej (problem minimalizacyjny) bądź najwyższej (problem maksymalizacyjny). Metoda zawsze znajduje najlepszą drogę, jednak jest bardzo wolna. Spowodowane jest to tym, że złożoność algorytmu to $O(n!)$, co sprawia,że już przy grafach o 15 wierzchołkach algorytm wykonuje się bardzo długo. Zaletą tej metody jest także prostota implementacji.

3. Algorytm

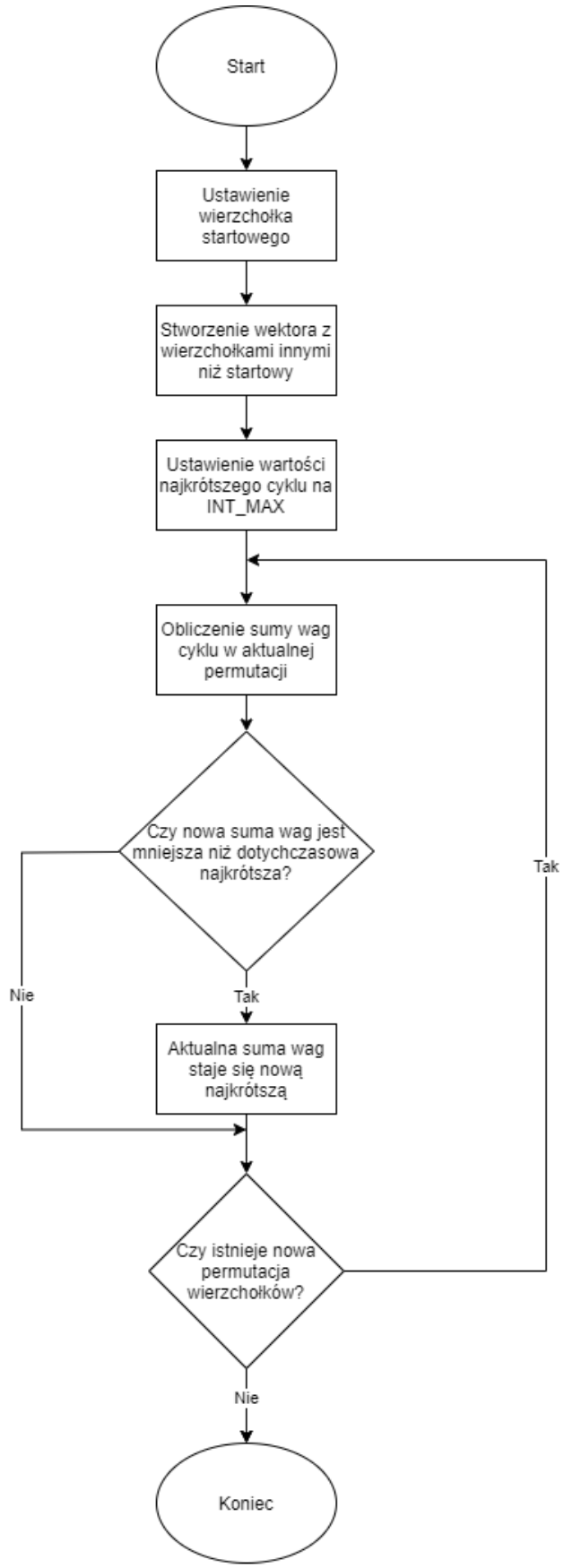
Schemat blokowy programu wykorzystanego do implementacji algorytmu:



Rys. 1. – Schemat blokowy programu

Na początku program wczytuje dane z pliku *plik.ini*. Następnie ma miejsce deklaracja wskaźnika na macierz sąsiedztwa ,w której będą przechowywane wagi krawędzi grafu. Macierz ta wypełniana jest danymi. W następnej kolejności ma miejsce algorytm znajdowania najkrótszego cyklu. W wyjściu danych czas operacji zapisywany jest do pliku *wyniki.csv*.

Algorytm znajdowania najkrótszego cyklu opisuje schemat blokowy:



Rys. 2. – Schemat blokowy algorytmu

4. Dane testowe

Do sprawdzenia poprawności działania algorytmu wybrano następujący zestaw instancji:

1. tsp_6_1.txt,
2. tsp_6_2.txt,
3. tsp_10.txt,
4. tsp_12.txt,
5. tsp_13.txt,
5. tsp_14.txt,
6. tsp_15.txt, http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/

5. Procedura badawcza

Należało zbadać zależność czasu rozwiązania problemu od wielkości instancji. W przypadku algorytmu realizującego przegląd zupełny przestrzeni rozwiązań dopuszczalnych nie występowały parametry programu, które mogły mieć wpływ na czas i jakość uzyskanego wyniku. W związku z tym procedura badawcza polegała na uruchomieniu programu sterowanego plikiem inicjującym .INI.

Dla każdej instancji czas mierzony był 5 razy, oprócz grafu o 15 wierzchołkach, gdzie pomiar był wykonany 2 razy ze względu na ogromną ilość czasu potrzebną do wykonania pomiaru. Pomiary zostały wykonane na komputerze ze specyfikacją: Procesor AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz. Zainstalowana pamięć RAM 8,00 GB (dostępne: 7,37 GB). Typ systemu 64-bitowy system operacyjny, procesor x64. Do pomiaru czasu została użyta biblioteka <chrono>, wraz z narzędziem high_resolution_clock. Fragment kodu w języku C++ wykonujący pomiar czasu:

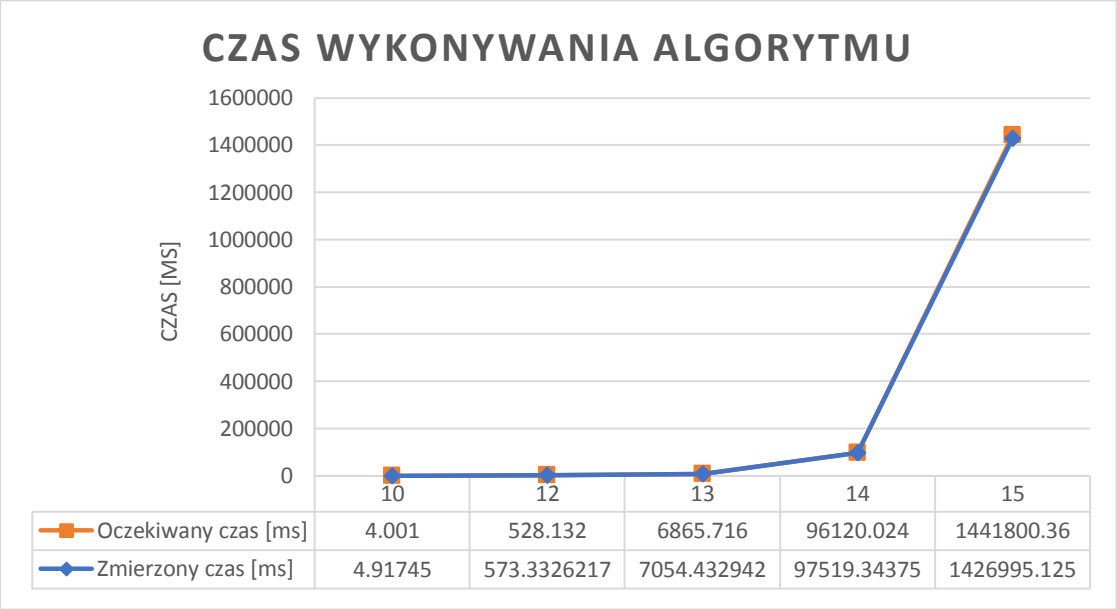
```
auto start = high_resolution_clock::now();

suma_cyklu = najkrotsza_sum(matrix, num_wierzch);

auto stop = high_resolution_clock::now();

auto duration = duration_cast<nanoseconds>(stop - start);

float time = (duration.count())/(1e+9);

(…)

csvFile << "Time [ms] = ;" << std::fixed << std::setprecision(5) << time*1000  << endl << endl;
```

6. Wyniki

Wyniki zgromadzone zostały w pliku: Wykres.xlsx

Pomiar czasu dla każdej instancji:

Nazwa pliku	Średnia czasu [ms]
tsp_6_1.txt,	0.0008333
tsp_6_2.txt,	0.0009067
tsp_10.txt,	4.91745
tsp_12.txt,	573.33262
tsp_13.txt,	7054.4329
tsp_14.txt,	97519.344
tsp_15.txt,	1426995.1



Rys 3. – Wykres czasu wykonywania algorytmu w zależności od wielkości instancji

7. Analiza wyników i wnioski

Krzywa wzrostu czasu (niebieska) względem wielkości instancji ma charakter wykładniczy (rysunek 3). Nałożenie krzywej $t(n) = t(n-1)*n$ (pomarańczowa) potwierdza, że badany algorytm wyznacza rozwiązania problemu komiwojażera dla badanych instancji w czasie n! zależnym względem wielkości instancji. Złożoność czasowa opracowanego algorytmu wynosi $O(n!)$.