Projektowanie Efektywnych Algorytmów

Projekt

18/10/2021

264067 Mateusz Waszczuk

(1) Brute Force

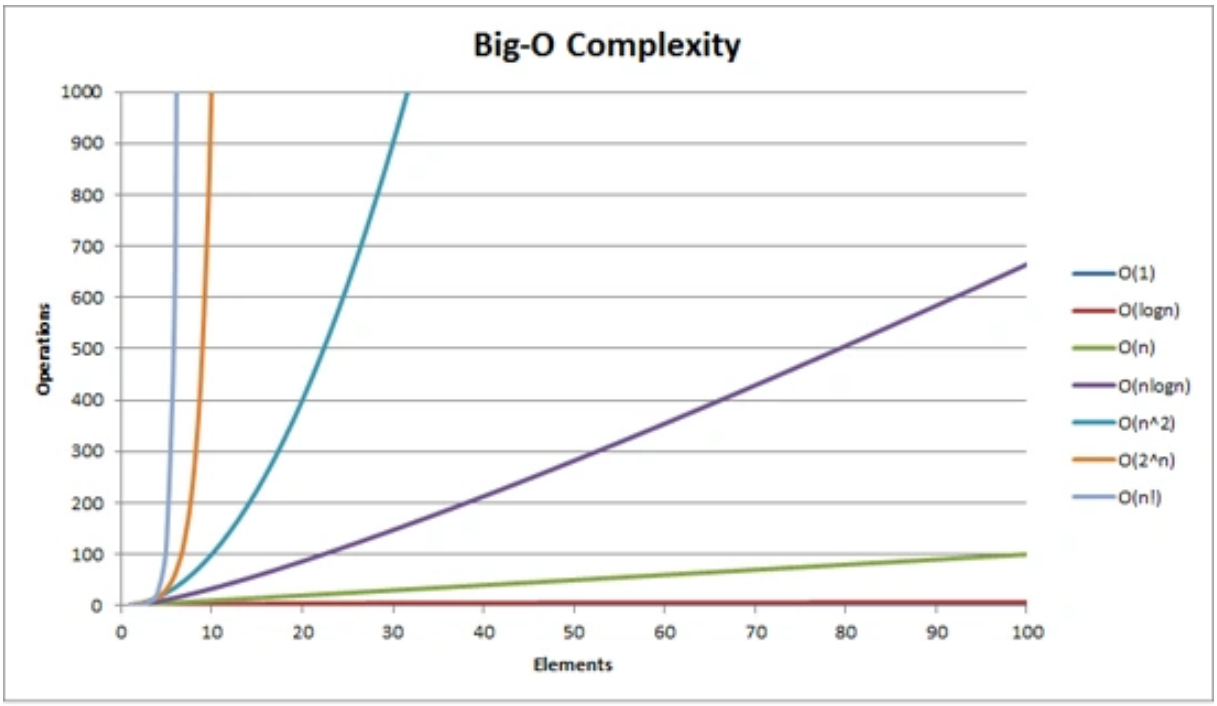
spis treści	*strona*

1.Sformułowanie zadania

Zadanie polega na opracowaniu, implementacji i zbadaniu efektywności algorytmu przeglądu zupełnego rozwiązującego problem komiwojażera w wersji optymalizacyjnej. Oznacza to że celem niniejszego zadania jest stworzenie programu który znajdzie wszystkie możliwe cykle Hamiltona w zadanych grafach oraz sprawdzi który ze znalezionych cykli ma najmniejszy koszt, tzn. sumę wag tworzących go krawędzi. Program ma mierzyć czas wykonania algorytmu brute force dla każdej instancji. Dane wejściowe dotyczące ilości powtórzeń dla każdej instancji oraz poprawne minimalne cykle Hamiltona wraz z właściwymi kosztami zostaną umieszczone w pliku inicjującym. Minimalne cykle Hamiltona i koszty mają pełnić jedynie funkcję weryfikacji poprawności działania algorytmu, aby mieć pewność, że pomiary czasu będą miarodajne. Mimo tego zabezpieczenia stworzona implementacja ma samodzielnie porównywać ze sobą wszystkie istniejące cykle Hamiltona w danym grafie i zwrócić ten najmniej kosztowny. Wyniki mają zostać zwrócone w pliku .csv. Po zaimplementowaniu algorytmu przeglądu zupełnego następnym zadaniem będzie przeprowadzanie testów poprawności jego działania oraz dostrojenie go poprzez określenie liczby pomiarów dla każdej wybranej instancji. Przy dostrajaniu należy uwzględnić czas potrzebny do przeprowadzenia algorytmu dla danej instancji tak, aby stosunek potrzebnego czasu do ilości pomiarów był jak najlepszy. Następnie należy przeprowadzić właściwe pomiary dla odpowiednio skonfigurowanego już pliku inicjującego. Po zakończeniu pomiarów należy przeprowadzić ich analizę, a następnie wyciągnąć wnioski i porównać otrzymane wyniki z oczekiwanymi.

2.Metoda

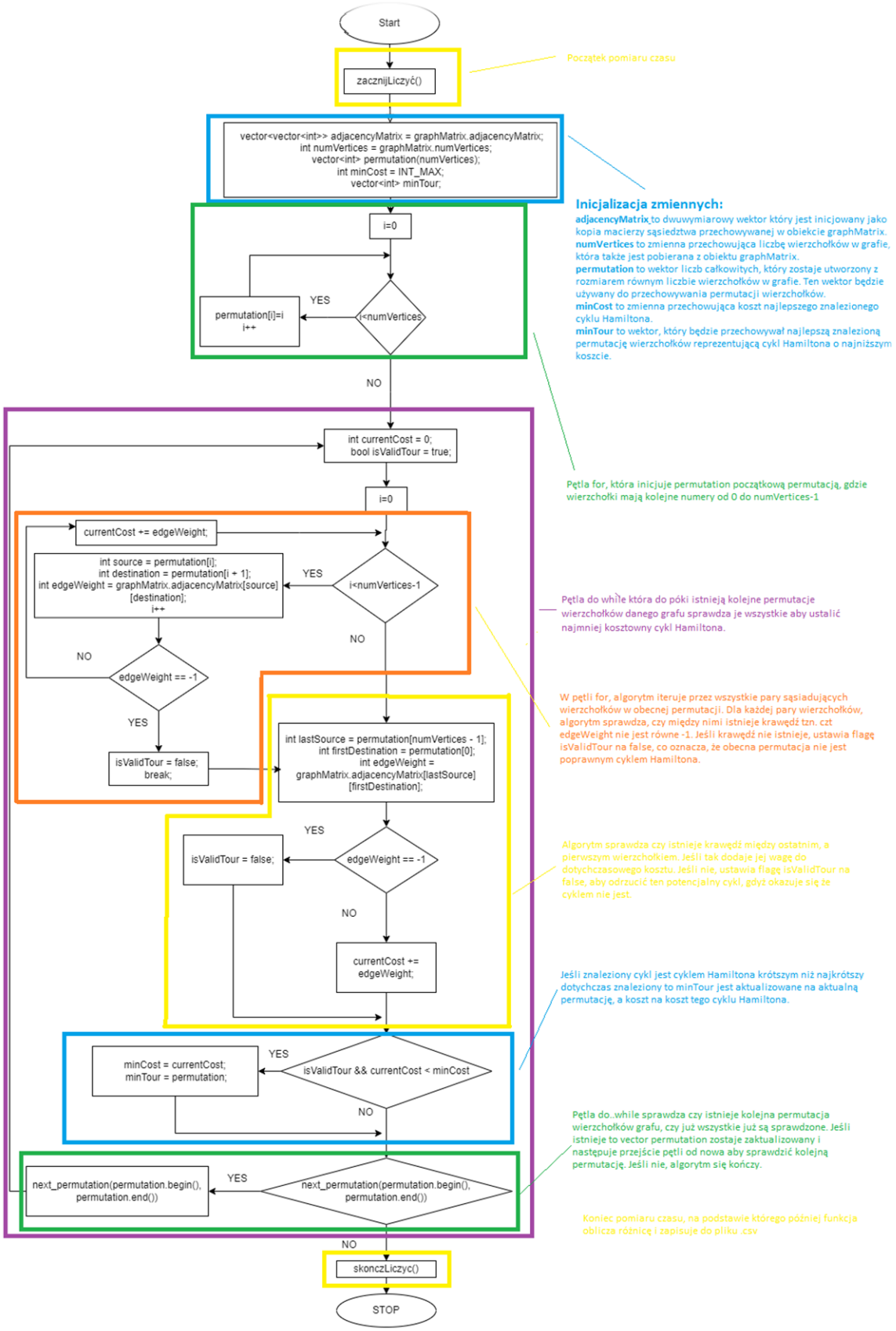
Metoda przeglądu zupełnego, tzw. przeszukiwanie wyczerpujące (eng. exhaustive search) bądź metoda siłowa (eng. brute force), polega na znalezieniu i sprawdzeniu wszystkich rozwiązań dopuszczalnych problemu, wyliczeniu dla nich wartości funkcji celu i wyborze rozwiązania o ekstremalnej wartości funkcji celu – najniższej (problem minimalizacyjny) bądź najwyższej (problem maksymalizacyjny). Zgodnie z obecnym stanem wiedzy metoda siłowa jest jedyną możliwą która daje gwarancję optymalności rozwiązania. Zastosowanie tej metody jest bardzo kosztowne obliczeniowo i oczekiwana złożoność czasowa wynosi O(n!). Oznacza to, że przy takiej złożoności dla instancji n+1 elementów wykona się n+1 razy więcej operacji niż dla instancji n elementów.



Rysunek 1: Wykres poglądowy porównujący złożoność O(n!) do innych złożoności

3.Algorytm

Na poniższym rysunku (Rysunek 2) znajduje się schemat blokowy algorytmu brute force w opracowanej przeze mnie implementacji. Schemat obejmuje jedynie sam algorytm, a nie całą metodę bruteForce, gdyż metoda ta zawiera także inne elementy, jak pomiar czasu i wyświetlanie wyników użytkownikowi programu. Wspomniane elementy nie mają wpływu na sam algorytm ani wyniki pomiarów czasu, dlatego nie są ujęte na schemacie. Analogiczne komentarze znajdują się w pliku Algorytmy.cpp który zawiera metodę bruteForce, a w niej sam algorytm brute force.



Rysunek 2: Schemat blokowy zaimplementowanego algorytmu brute force

4.Dane testowe

Do sprawdzenia poprawności działania algorytmu oraz do dostrojenia, wybrano następujący zestaw instancji:

-tsp_6_1.txt

-tsp_10.txt

-tsp_12.txt

http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/

Do przeprowadzania pomiarów wybrano następujący zestaw instancji:

-tsp_6_1.txt

-tsp_10.txt

-tsp_12.txt

-tsp_13.txt

-tsp_14.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: (format pliku: liczba_wykonań, optymalny_koszt_cyklu_Hamiltona, nazwa_pliku_z_grafem, [ścieżka optymalna]). Wyniki były zapisywane w pliku pomiary.csv. Poniżej przedstawiono zawartość pliku inicjującego:

50 132 tsp_6_1.txt 0 1 2 3 4 5 0

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

50 264 tsp_12.txt 0 1 8 4 6 2 11 9 7 5 3 10 0

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

1 282 tsp_14.txt 0 10 3 5 7 9 13 11 2 6 4 8 1 12 0

1 291 tsp_15.txt 0 12 1 14 8 4 6 2 11 13 9 7 5 3 10 0

1 39 tsp_17.txt 0 11 13 2 9 10 1 12 15 14 5 6 3 4 7 8 16 0

Każda instancji rozwiązywana była zgodnie z liczbą jej wykonań, np. tsp_6_1.txt wykonana została 50 razy, a tsp_14 raz. Tsp_15 i tsp_17 nie zostały wykonane ani razu, gdyż czas ich wykonania był zbyt długi. Czas pojedynczego wykonania tsp_14.txt wynosił ponad 5 godzin, stąd podjęto decyzję o zatrzymaniu programu po wykonaniu obliczeń dla tsp_14.txt. Do pliku wyjściowego pomiary.csv zapisywany był czas wykonania pomiarów. W przypadku niezgodności kosztu lub ścieżki optymalnej, program wyświetlał na ekranie komunikat o błędzie. W przypadku przeprowadzonych pomiarów nie miało to miejsca ani razu, a co za tym idzie ścieżki optymalne i ich ścieżki były zgodne z oczekiwanymi. Z tego powodu program nie zapisywał optymalnych ścieżek i kosztów do pliku, a jedynie wyświetlał je każdorazowo na ekranie. Plik wyjściowy zapisywany był w formacie csv. Dane znajdują się w następującym formacie nazwa_instancji; czas_1_pomiaru; czas_2_pomiaru ; … ; czas_n_pomiaru. Program czas dla instancji (0,8> oblicza w nanosekundach, instancji (8,10> w milisekundach, instancji (10, +∞) w sekundach. Program wyświetla taką informację po zakończeniu obliczeń. Poniżej przedstawiono fragment zawartości pliku wyjściowego (jednostki w nawiasach kwadratowych zostały dopisane przeze mnie przykład był przejrzysty):

tsp_6_1.txt;0;0;0;0;0;0;0 [nanosekundy]

tsp_10.txt;673;707;687;712 [milisekundy]

tsp_12.txt;104;104;104;106;104 [sekundy]

tsp_13.txt ;1411;1411;1410;1409 [sekundy]

tsp_14.txt;20552 [sekundy]

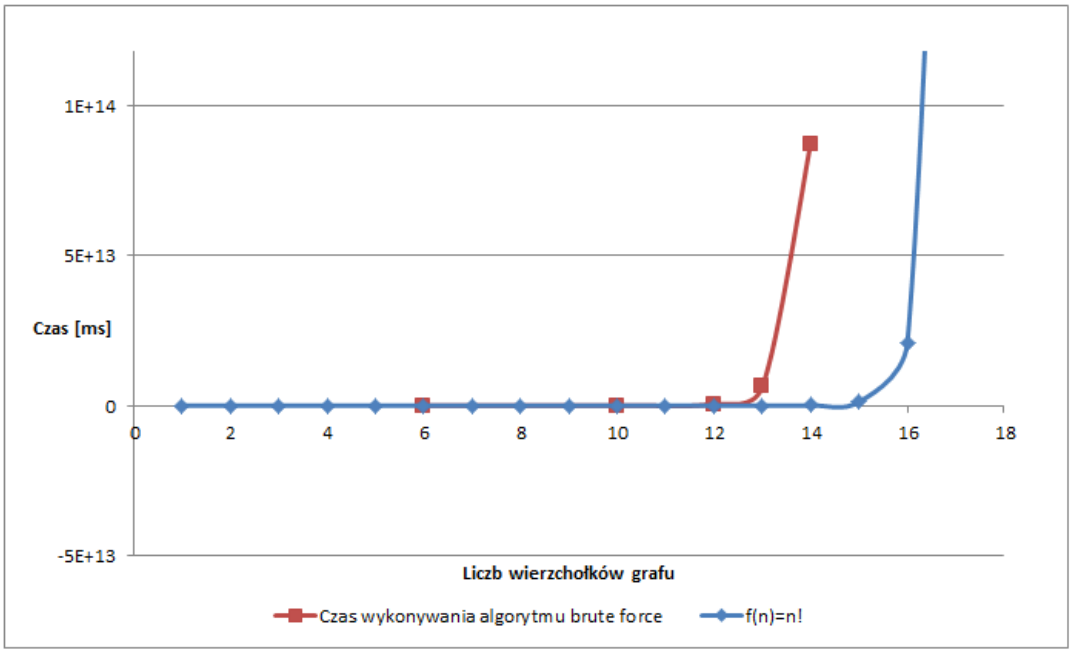
Wyniki zostały opracowane w MS Excel.

6.Wyniki

Wyniki zgromadzone zostały w pliku pomiary.csv. Plik został dołączony do raportu i znajduje się na dysku Google pod adresem:

https://drive.google.com/drive/folders/1BzIs4vg5FYJQlVRcNnvYcAPUtsAkRDOR?usp=drive_link.

Wyniki przedstawione zostały w postaci wykresu zależności czasu uzyskania rozwiązania problemu od wielkości instancji (rysunek 3). Obok wykresu czasu od wielkości instancji zamieszczono także wykres funkcji f(n)=n! celem późniejszego porównania wyników z oczekiwaną ich złożonością czasową.



Rysunek 3: Wykres czasu wykonania algorytmu brute force od ilości wierzchołków grafu

7.Analiza wyników i wnioski

Funkcja czasu wykonania algorytmu brute force szukającego najkrótszego cyklu Hamiltona w grafie jest zbliżona klasą złożoności do nałożonej na wykres funkcji f(n)=n!, co jest widoczne na rysunku 3. Zaimplementowany przeze mnie algorytm rozwiązujący problem komiwojażera dla badanych przeze mnie instancji ma klasę złożoności czasowej O(n!). Jest to w pełni zgodne ze spodziewanymi przeze mnie rezultatami. Hipotezę można uznać za potwierdzoną.