PEA – projekt nr 1

Temat: Implementacja i analiza efektywności algorytmu podziału i ograniczeń i programowania dynamicznego .

Należy zaimplementować oraz dokonać analizy efektywności algorytmu przeglądu zupełnego, podziału i ograniczeń (B&B) i/lub programowania dynamicznego (DP) dla asymetrycznego problemu komiwojażera (ATSP).

Podczas realizacji zadania należy przyjąć następujące założenia:

- używane struktury danych powinny być alokowane dynamicznie (w zależności od aktualnego rozmiaru problemu),
- program powinien umożliwić weryfikację poprawności działania algorytmu. W tym celu powinna istnieć możliwość wczytania danych wejściowych z pliku tekstowego,
- po zaimplementowaniu i sprawdzeniu poprawności działania algorytmu należy dokonać pomiaru czasu jego działania w zależności od rozmiaru problemu N (badania należy wykonać dla minimum 7 różnych reprezentatywnych wartości N),
- dla każdej wartości N należy wygenerować po 100 losowych instancji problemu (w sprawozdaniu należy umieścić tylko wyniki uśrednione – pamiętać, aby nie mierzyć czasu generowania instancji),
- implementacje algorytmów należy dokonać zgodnie z obiektowym paradygmatem programowania,
- używanie „okienek” nie jest konieczne i nie wpływa na ocenę (wystarczy wersja konsolowa),
- kod źródłowy powinien być komentowany.

Sprawozdanie powinno zawierać:

- wstęp teoretyczny zawierający opis rozpatrywanego problemu, oraz oszacowanie jego złożoności obliczeniowej na podstawie literatury dla badanych algorytmów,
- przykład praktyczny - opis działania algorytmu „krok po kroku” dla przykładowej instancji danego problemu o małej wartości N – minimum N=4 (w przypadku dwóch metod przykład robimy tylko dla metody B&B, dla przeglądu zupełnego nie robimy przykładu)
- opis implementacji algorytmu (dokładny opis funkcji obliczającej ograniczenia w przypadku B&B, wykorzystane struktury danych istotne dla działania algorytmu),
- plan eksperymentu (rozmiar używanych struktur danych, sposób generowania danych, metoda pomiaru czasu, itp.),

2024-10-07

- wyniki eksperymentów (w postaci tabel i wykresów) – należy też porównać wyniki dla różnych metod (najlepiej na wspólnym wykresie),
- wnioski dotyczące otrzymanych wyników,
- kod źródłowy w formie elektronicznej wraz z wersją wykonywalną programu (kopiowany na dysk google'a po akceptacji prowadzącego).

Sprawdzenie poprawności zaimplementowanego algorytmu:

Aby sprawdzić poprawność działania algorytmu musi być możliwość wczytania danych z pliku i tekstowego i wykonania na nich obliczeń. Menu programu powinno umożliwiać

1.Wczytanie danych z pliku

2.Wygenerowanie danych losowych

3.Wyświetlenie ostatnio wczytanych lub wygenerowanych danych

4.Uruchomienie danego algorytmu dla ostatnio wczytanych lub wygenerowanych danych i wyświetlenie wyników (należy wyświetlić długość ścieżki, ciąg wierzchołków oraz **czas wykonania algorytmu**)

Dane dla których będzie testowana poprawność algorytmu umieszczone są na stronie prowadzącego i/lub na dysku google.

Format danych w pliku jest następujący:

- w pierwsze linii jest podana ilość miast,
- w pozostałych liniach macierz kosztów: w każdej linii wiersz macierzy (liczby przedzielone spacją),
- dane na przekątnej mają wartość równą -1.

Ocena (maksimum): zaczynamy od oceny 2.0 i dodajemy do tego wyniku punkty jak poniżej:

a) obowiązkowo przegląd zupełny (Brute Force):

bez bibliotek– 0.5 (ale bez korzystania z funkcji rekurencyjnej)

z bibliotekami (i/lub rekurencją) - 0.3

b) Branch and Bound – 1.2 (jedna metoda przeszukiwania), 1.4 dwie metody liczenia ograniczenia, 1.5- algorytm Little'a

c) programowanie dynamiczne – 1.0

Aby zaliczyć ćwiczenie należy wybrać przegląd zupełny i co najmniej jedną z dwóch metod.

Jeżeli wybrano B&B to należy przyjąć dopuszczalny czas wykonania dla B&B – np. pięć minut i jeśli problem nie został rozwiązany w tym czasie, to go przerwać - policzyć ile (w %) w zależności od N(rozmiar problemu) zostało przerwanych.

Dla każdego algorytmu wyznaczyć maksymalne N (w tym celu przyjąć dla wszystkich czas graniczny (np. 2 minuty).

Języki programowania

Wskazane jest korzystanie z języków kompilujących się do kodu maszynowego (np. C++).

Uwaga!!!

1.Jeżeli korzystamy z Visual Studio (lub innego środowiska), to testy prowadzimy na wersji RELEASE (a nie DEBUG)

2.Należy wgrać na dysk google'a przed oddaniem wersję exe programu, gdyż w tym zadaniu algorytmy innych osób będą testowane w tych samych warunkach – ich czas działania będzie wpływał na ocenę końcową.

3.Przy metodzie przeglądu zupełnego proszę nie skracać przebiegu algorytmu poprzez pomijanie niektórych permutacji. Nie jest dozwolone jest przerwanie sumowania przy obliczaniu drogi dla konkretnego rozwiązania jeśli dotychczasowy fragment drogi jest gorszy od dotychczasowego najlepszego rozwiązania – należy zrobić klasyczny przegląd zupełny.

Dodatkowe materiały internetowe:

https://www.ii.uni.wroc.pl/~prz/2011lato/ah/opracowania/met_podz_ogr.opr.pdf- plecakowy i TSP

http://cs.pwr.edu.pl/zielinski/lectures/om/mow10.pdf - TSP(Little'a) oraz B&B

https://www.youtube.com/watch?v=-cLsEHP0qt0 – wprowadzenie do TSP

https://www.youtube.com/watch?v=nN4K8xA8ShM&t=927s – TSP B&B

https://www.youtube.com/watch?v=JE0JE8ce1V0 – TSP && programowanie dynamiczne

https://www.youtube.com/watch?v=XaXsJJh-Q5Y – TSP && programowanie dynamiczne

https://dspace.mit.edu/bitstream/handle/1721.1/46828/algorithmfortrav00litt.pdf?sequence=1&isAllowed=y - algorytm Little'a

2024-10-07