Zadanie projektowe nr 3

Implementacja i analiza efektywności algorytmów optymalnych o pseudowielomianowej złożoności obliczeniowej dla wybranych problemów kombinatorycznych

Należy zaimplementować oraz dokonać pomiaru czasu działania algorytmów dla następujących problemów kombinatorycznych:

a) dyskretnego problemu plecakowego (w wersji optymalizacyjnej),

b) asymetrycznego problemu komiwojażera (w wersji optymalizacyjnej).

1. Dyskretny problem plecakowy

Parametrami zadania są: skończony zbiór elementów A = $\{a_1, a_2, ..., a_n\}$, z których każdy ma określony rozmiar $s(a_i)>0$ i wartość $w(a_i)>0$ oraz pojemność plecaka $b>0$. Rozwiązaniem jest taki podzbiór elementów $A' \subset A$, który maksymalizuje łączną wartość wybranych elementów ($\sum_{a_i \epsilon A'} w(a_i)$) przy warunku nie przekroczenia dopuszczalnej pojemności plecaka ($\sum_{a_i \epsilon A'} s(a_i) \leq b$). Należy przyjąć, że wszystkie parametry zadania są liczbami naturalnymi.

Dla tak zdefiniowanego problemu plecakowego należy opracować następujące algorytmy:

- przegląd zupełny,
- algorytm zachłanny (jako kryterium wyboru elementu przyjąć jego wartość $w(a_i)$ oraz stosunek wartości i rozmiaru $w(a_i)/s(a_i)$),
- algorytm oparty na programowaniu dynamicznym.

<u>Podczas realizacji projektu należy przyjąć następujące założenia:</u>

- używane struktury danych powinny być alokowane dynamicznie (w zależności od aktualnego rozmiaru problemu),
- program powinien umożliwić weryfikację poprawności działania poszczególnych algorytmów. W tym celu powinna istnieć możliwość wczytania danych z pliku tekstowego (należy przyjąć następującą strukturę pliku: w pierwszej linii podajemy pojemność plecaka oraz liczbę przedmiotów, które chcemy w nim umieścić, w kolejnych liniach podajemy rozmiar i wartość dla poszczególnych przedmiotów – w jednej linii dla jednego przedmiotu),
- wynikiem podczas testowania ma być lista włożonych przedmiotów (poprzez podanie wartości i wagi przedmiotu) oraz sumaryczna cena i waga włożonych przemiotów,
- po zaimplementowaniu i sprawdzeniu poprawności działania każdego z algorytmów należy dokonać pomiaru czasu działania algorytmów w zależności od liczby przedmiotów N oraz rozmiaru plecaka B. Badania należy wykonać dla 5 różnych (reprezentatywnych) liczb przedmiotów N. Dla

każdej wartości N badania trzeba wykonać dla 3 różnych pojemności plecaka B (jest to szczególnie ważne dla algorytmu wykorzystującego programowanie dynamiczne),

- dla każdego zestawu: algorytm, liczba przedmiotów N i pojemność plecaka B należy wygenerować po 100 losowych instancji (w sprawozdaniu należy umieścić tylko wyniki uśrednione),
- przy generowaniu danych testowych należy zapewnić warunek, aby sumaryczny rozmiar przedmiotów był większy niż pojemność plecaka,
- przy badaniach algorytmu wykonującego przegląd zupełny określić wartość N, dla którego algorytm wykonuje się w „rozsądnym" czasie (przyjąć t_{max} np. 10, 30 lub 60 minut),
- używanie okienek nie jest konieczne i nie wpływa na ocenę (wystarczy wersja konsolowa),
- do przechowywania danych nie należy korzystać z gotowych bibliotek np. STL (algorytmy i struktury danych muszą być zaimplementowane samodzielnie przez studenta – oprócz pewnych wyjątków zapisanych przy sposobie oceniania),
- implementacja projektu powinna być wykonana w formie jednego programu,
- kod źródłowy powinien być komentowany.

2. Asymetryczny problem komiwojażera

Parametrami zadania są: skończony zbiór miast C = {c_1, c_2, ..., c_n} oraz odległości d_{ij} z miasta c_i do miasta c_j (nie ma wymogu $d_{ij} = d_{ji}$). Należy określić kolejność odwiedzania wszystkich miast (ich permutację) <$c_{i[1]}$, $c_{i[2]}$, …, $c_{i[n]}$>, aby sumaryczna trasa była jak najkrótsza ($\sum_{j=1}^{n-1} d_{i[j],i[j+1]} + d_{i[n],i[1]}$) przy założeniu, że każde miasto zostało odwiedzone dokładnie jeden raz. Należy przyjąć, że wszystkie parametry zadania są liczbami naturalnymi.

Dla tak zdefiniowanego problemu komiwojażera należy opracować następujące algorytmy:

- przegląd zupełny,
- algorytm zachłanny,
- algorytm przeszukiwania lokalnego (w najprostszej wersji można zastosować algorytm 2-opt. Algorytm zaczyna od losowej permutacji miast i w kolejnych krokach próbuje ją ulepszyć. Dla danej permutacji należy rozpatrzyć wszystkie permutacje uzyskane na jej podstawie przez zamianę dwóch niesąsiadujących ze sobą krawędzi tzw. zamiana dwukrawędziowa. Spośród wszystkich uzyskanych w ten sposób permutacji wybieramy permutację dającą najlepszą poprawę. Permutacja ta staje się aktualnym rozwiązaniem. Algorytm jest kontynuowany do momentu, gdy możliwe jest uzyskanie lepszych rozwiązań. Szczegółowy opis algorytmu można znaleźć w literaturze np. [2]).

Podczas realizacji projektu należy przyjąć następujące założenia:

- używane struktury danych powinny być alokowane dynamicznie (w zależności od aktualnego rozmiaru problemu),
- do reprezentacji odległości między miastami należy użyć macierz sąsiedztwa,
- program powinien umożliwić weryfikację poprawności działania poszczególnych algorytmów. W tym celu powinna istnieć możliwość wczytania danych z pliku tekstowego (należy przyjąć następującą strukturę pliku: w pierwszej linii podajemy liczbę miast, w kolejnych liniach podajemy wiersze macierzy odległości pomiędzy miastami. Na przekątnej czyli odległości miasta i do i jest zero). Miasta numerujemy od zero. Wynik ma być podany w formie sumarycznej drogi (proszę nie zapomnieć o dodaniu odległości od miasta ostatniego do pierwszego) oraz listy po kolei odwiedzanych miast.
- po zaimplementowaniu i sprawdzeniu poprawności działania każdego z algorytmów należy dokonać pomiaru czasu działania algorytmów w zależności od liczby miast N. Badania należy wykonać dla 5 różnych (reprezentatywnych) liczb miast N,
- dla każdego zestawu: algorytm i liczba miast N należy wygenerować po 100 losowych instancji (w sprawozdaniu należy umieścić tylko wyniki uśrednione),
- przy badaniach algorytmu wykonującego przegląd zupełny przyjąć wartości N równe 5, 10 i 20. Następnie określić średni czas rozpatrywania 1 permutacji. Na tej podstawie określić przewidywany czas działania algorytmu dla N równego 50, 100 oraz 1000,
- pozostałe założenia są identyczne jak dla problemu plecakowego,
- wszystkie badania przeprowadzać w wersji RELEASE.

Sprawozdanie powinno zawierać:

- krótki wstęp zawierający opis zastosowanych algorytmów i oszacowanie ich złożoności obliczeniowej na podstawie literatury,
- plan eksperymentu, czyli założenia co do wielkości struktur, sposobu generowania ich elementów, sposobu pomiaru czasu, itp.
- wyniki - należy przedstawić w tabelach oraz w formie wykresów dla każdego problemu osobno. Dla problemu plecakowego zamieścić dwa typy wykresów: ilustrujące zależność czasu wykonania algorytmu od liczby przedmiotów N (przy stałej pojemności plecaka B) oraz zależność czasu wykonania algorytmu od pojemności plecaka B (przy stałej liczbie przedmiotów).
- wnioski dotyczące otrzymanych wyników. Wskazać (jeżeli występują) przyczyny rozbieżności pomiędzy złożonościami teoretycznymi a uzyskanymi eksperymentalnie,

- załączony kod źródłowy w formie elektronicznej (cały projekt wraz z wersją skompilowaną programu).

Ocena projektu:

3.0 – po jednym algorytmie z każdego problemu (możliwość korzystania z biblioteki STL)

4.0 – po dwa algorytmy z każdego problemu (możliwość wykorzystania z biblioteki STL)

4.5 – po dwa algorytmy z każdego problemu (bez STL)

5.0 – po trzy algorytmy z każdego problemu (bez STL w wersji obiektowej)

5.5 – wymagania jak na ocenę 5.0 + realizacja algorytmu symulowanego wyżarzania lub algorytmu mrówkowego dla problemu komiwojażera.

Literatura:

1. T. H. Cormen, C. E. Leiserson, R. L. Rivest, Wprowadzenie do algorytmów, WNT, Warszawa, 1997
2. M. Sysło, N. Deo, J. S. Kowalik, Algorytmy optymalizacji dyskretnej z programami w języku Pascal, WNT, Warszawa 1999
3. http://iair.mchtr.pw.edu.pl/~bputz/aisd_cpp/lekcja2/segment5/main.htm [problem plecakowy]