AZO – zadanie projektowe nr 2

Badanie efektywności algorytmów grafowych w zależności od rozmiaru instancji oraz sposobu reprezentacji grafu w pamięci komputera.

Należy zaimplementować oraz dokonać pomiaru czasu działania wybranych algorytmów grafowych rozwiązujących następujące problemy:

a. wyznaczanie minimalnego drzewa rozpinającego (MST) - algorytm Prima oraz algorytm Kruskala,

b. wyznaczanie najkrótszej ścieżki w grafie – algorytm Dijkstry oraz algorytm Forda-Bellmana,

c. wyznaczanie maksymalnego przepływu – algorytm Forda-Fulkersona (na ocenę 5.5).

Algorytmy te należy zaimplementować dla obu poniższych reprezentacji grafu w pamięci komputera:

- reprezentacja macierzowa (macierz incydencji),

- reprezentacja listowa (lista następników/poprzedników).

Należy przyjąć następujące założenia:

- wszystkie struktury danych powinny być alokowane dynamicznie,
- przepustowość/koszt krawędzi jest liczbą całkowitą (dodatnią),
- po zaimplementowaniu każdego z algorytmów dla obu reprezentacji grafu należy dokonać pomiaru czasu działania algorytmów w zależności od rozmiaru grafu oraz jego gęstości (liczba krawędzi w stosunku do liczby krawędzi dla grafu pełnego, czyli zawierającego wszystkie możliwe krawędzie). Badania należy wykonać dla 7 różnych (reprezentatywnych) liczb wierzchołków V oraz następujących gęstości grafu: 25%, 50% oraz 99%. Dla każdego zestawu: reprezentacja, liczba wierzchołków i gęstość należy wygenerować serię losowych instancji (np. 50 ze względu na możliwy rozrzut poszczególnych wyników), zaś w sprawozdaniu umieścić wyłącznie wyniki uśrednione z danej serii,
- sposób generowania grafu powinien zapewnić jego spójność (np. można najpierw wygenerować drzewo rozpinające, a dopiero potem pozostałe krawędzie do wymaganej gęstości grafu),
- do dokładnego pomiaru czasu w systemie Windows w C++ można skorzystać z funkcji QueryPerformanceCounter lub std::chrono::high_resolution_clock (opis na stronie http://cpp0x.pl/forum/temat/?id=21331)
- program powinien umożliwić sprawdzenie poprawności zaimplementowanych operacji i zbudowanej struktury grafu (dodatkowe informacje w dalszej części dokumentu),
- zalecanymi językami programowania są języki kompilowane do kodu natywnego (np. C, C++), a nie interpretowane lub uruchamiane na maszynach wirtualnych (np. JAVA, .NET, Phyton, w wyjątkowej sytuacji można je wykorzystać, o ile nie wpływa to istotnie na uzyskane wyniki),
- używanie okienek nie jest konieczne i nie wpływa na ocenę (wystarczy wersja konsolowa),

- korzystanie z gotowych bibliotek np. STL, Boost lub innych powoduje obniżenie oceny za projekt (dotyczy to podstawowych dla projektu struktur, szczegóły zostały omówione w części dotyczącej oceny projektu),
- wszystkie algorytmy i struktury muszą być zaimplementowane samodzielnie (nie należy kopiować gotowych rozwiązań),
- implementacja projektu powinna być wykonana w formie jednego programu,
- kod źródłowy powinien być komentowany.

Sprawdzenie poprawności zbudowanej struktury/operacji obejmuje:

- wczytanie struktury grafu z pliku tekstowego (należy umożliwić wprowadzenie dowolnej nazwy pliku). Plik zawiera opis poszczególnych krawędzi według wzoru: początek krawędzi, koniec krawędzi oraz waga/przepustowość. Struktura pliku jest następująca:
  a. w pierwszej linii zapisana jest liczba krawędzi oraz liczba wierzchołków (rozdzielone spacją),
  b. wierzchołki numerowane są w sposób ciągły od zera,
  c. w kolejnych liniach znajduje się opis krawędzi (każda krawędź w osobnej linii) w formie trzech liczb przedzielonych spacjami (wierzchołek początkowy, wierzchołek końcowy oraz waga/ przepustowość),
  d. dla problemu MST pojedynczą krawędź traktuje się jako nieskierowaną, natomiast dla algorytmów najkrótszej drogi i maksymalnego przepływu jako skierowaną,
- losowe wygenerowanie grafu (jako dane podaje się liczbę wierzchołków oraz gęstość w procentach). Do przechowywania struktury grafu wczytanej z pliku lub wygenerowanej losowo należy wykorzystać tą samą zmienną/obiekt (ostatnia operacja generowania danych losowo lub wczytywania z pliku nadpisuje poprzednią),
- możliwość wyświetlenia na ekranie wczytanego lub wygenerowanego grafu w formie reprezentacji listowej i macierzowej,
- uruchomienie algorytmu dla obu reprezentacji i wyświetlenie wyników na ekranie (wyświetlane wyniki powinny w pełni reprezentować rozwiązanie danego problemu np. dla algorytmów Dijkstry oraz Forda-Bellmana należy wyświetlić zarówno znalezioną ścieżkę jak i całkowity koszt tej ścieżki). Dla problemu najkrótszej drogi w grafie i maksymalnego przepływu musi być możliwość podania wierzchołka początkowego i końcowego.

Poniższe operacje należy zrealizować w formie menu dla każdej struktury i dla każdego problemu z osobna:

1. Wczytaj dane z pliku.
2. Wygeneruj graf losowo.

3. Wyświetl graf listowo i macierzowo na ekranie.
4. Algorytm 1 (np. Prima) macierzowo i listowo z wyświetleniem wyników.
5. Algorytm 2 (np. Kruskala) macierzowo i listowo z wyświetleniem wyników.

Uwaga! Po uruchomieniu program powinien zapytać, który problem chcemy rozwiązywać i przejść do odpowiedniego podmenu.

Sprawozdanie powinno zawierać:

- krótki wstęp zawierający oszacowanie złożoności obliczeniowej poszczególnych problemów na podstawie literatury,
- plan eksperymentu czyli założenia co do wielkości struktur, sposobu generowania ich elementów, sposobu pomiaru czasu, itp.
- opis metody generowania grafu (sposób powinien zapewnić spójność oraz zmienną strukturę grafu),
- wyniki należy przedstawić w tabelach oraz w formie wykresów dla każdego problemu osobno (oddzielnie MST i najkrótsza droga w grafie). Dla każdego problemu (MST oraz najkrótsza ścieżka) należy przygotować następujące wykresy:
  a. wykresy typ1 (osobne wykresy dla każdej reprezentacji grafu) - w formie linii (połączonych punktów), których parametrem jest typ algorytmu (Kruskal/Prim lub Dijsktra/Bellman) i gęstość grafu - czyli 2x3 = 8 linii na rysunek (2 algorytmy x 3 gęstości grafu),
  b. wykresy typ2 (osobne wykresy dla każdej gęstości grafu) – w formie linii których parametrem jest typ algorytmu i typ reprezentacji grafu (czyli 4 linie na każdy rysunek – 2 algorytmy x 2 reprezentacje).
  c. analogicznie jest dla algorytmu maksymalnego przepływu.
- wszystkie wykresy należy przedstawić w następującym układzie współrzędnych: czas wykonania danego algorytmu (oś Y) w funkcji liczby wierzchołków grafu (oś X) – osie należy odpowiednio opisać i koniecznie podać jednostki (dotyczy to głównie zmierzonego czasu),
- nie umieszczać na jednym rysunku wyników działania algorytmów z różnych problemów,
- wnioski dotyczące efektywności poszczególnych struktur. Wskazać (jeśli są) przyczyny rozbieżności pomiędzy złożonościami teoretycznymi a uzyskanymi eksperymentalnie,
- załączony kod źródłowy w formie elektronicznej (cały projekt wraz z wersją skompilowaną programu) oraz sprawozdanie w formie elektronicznej.

Ocena projektu:

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

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

5.0 – pod dwa algorytmy z każdego problemu (bez korzystania z bibliotek np. STL, wersja obiektowa),

5.5 – pod dwa algorytmy z każdego problemu (bez korzystania z bibliotek np. STL, wersja obiektowa), dodatkowo algorytm wyznaczania maksymalnego przepływu Forda-Fulkersona (określanie ścieżek metodą przeszukiwania grafu w głąb i wszerz).