1. Temat

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

2. Cel zadania

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 – 110 pkt).

Algorytmy te należy zaimplementować dla obu poniższych reprezentacji grafu w pamięci komputera:
- reprezentacja macierzowa (macierz incydencji),
- reprezentacja listowa (lista następników/poprzedników).

3. 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 największej możliwej liczby krawędzi – grafu pełnego). 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`
- 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`, `Python`),
- 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 7),
- 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.
- Wszystkie ścieżki powinny być względne a najlepiej do katalogu bieżącego. Program powinien być przenośny!

4. Sprawdzenie poprawności zbudowanej struktury/operacji

- 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:
  — w pierwszej linii zapisana jest liczba krawędzi oraz liczba wierzchołków (rozdzielone spacją),
  — wierzchołki numerowane są w sposób ciągły od zera,
  — 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),
  — 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.

5. Opcje programu

W pliku konfiguracyjnym powinny być opcje dające możliwość:

- wyboru jednego z dwóch problemów:
  - — MST – minimalne drzewo rozpinające (ang. Minimal Spanning Tree),
  - — SP – wyznaczenie najkrótszej ścieżki.
- wczytania grafu z pliku,
- wygenerowania grafu losowego,
- wyświetlenia grafu w formie listowej i macierzowej na ekranie,
- `Algorytm 1` (np. Prima) na reprezentacji macierzowej i listowej z wyświetleniem na ekranie,
- `Algorytm 2` (np. Kruskula) na reprezentacji macierzowej i listowej z wyświetleniem na ekranie,

6. Sprawozdanie

- Ogólne wytyczne znajdują się w regulaminie.
- 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 Dijkstra/Bellman) i gęstość grafu – czyli $2 \cdot 3 = 6$ 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.

7. Ocena projektu

0 – 69 – po jednym algorytmie z każdego problemu (możliwość wykorzystania biblioteki `STL`.

0 – 89 – po dwa algorytmy z każdego problemu (możliwość wykorzystania biblioteki `STL`, wersja obiektowa).

0 – 100 – po dwa algorytmy z każdego problemu (bez korzystania z bibliotek np. `STL`, wersja obiektowa).

0 – 110 – po 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).

8. Literatura

1. Cormen T., Leiserson C.E., Rivest R.L., Stein C., Wprowadzenie do algorytmów, WNT
2. Drozdek A., C++. Algorytmy i struktury danych, Helion