Struktury danych i złożoność obliczeniowa

Sprawozdanie z zadania projektowego nr 2

Imię i nazwisko	Nr indeksu
Iwo Bujkiewicz	226203

Prowadzący	Termin	Data
Dr inż. Dariusz Banasiak	wtorek nieparzysty 15:15	05.06.2018

1. Wstęp

Do zadania projektowego nr 2 zaimplementowano następujące reprezentacje grafów:

- Macierz incydencji
- Lista sąsiedztwa

Obie reprezentacje zaimplementowano bez użycia zaawansowanych struktur danych STL lub zewnętrznych bibliotek, częściowo z użyciem struktur zaimplementowanych samodzielnie w ramach zadania projektowego nr 2.

Zadanie miało na celu eksperymentalne sprawdzenie czasu wykonywania następujących algorytmów grafowych, wyszczególnionych wraz z ich złożonością czasową według ogólnodostępnych źródeł:

- Znajdowanie minimalnego drzewa spinającego
  - Algorytm Dijkstry-Jarníka-Prima (DJP) - $O(|V|^2)$
  - Algorytm Kruskala - $O(|E| \log |V|)$
- Znajdowanie ścieżki w grafie
  - Algorytm Dijkstry - $O(|V|^2)$
  - Algorytm Shimbela-Bellmana-Forda-Moore'a (SBFM) - $O(|V||E|)$

2. Plan eksperymentu

Zadanie zakładało wielokrotne powtórzenie, dla kilku różnych ilości wierzchołków w grafie i gęstości grafu, testu, polegającego na zmierzeniu czasu wykonywania czterech algorytmów na dwóch rodzajach reprezentacji grafów generowanych losowo na podstawie zadanych parametrów.

Na potrzeby eksperymentu przyjęto zestaw ilości wierzchołków, które były dostatecznie duże, aby otrzymać miarodajne wyniki, ale jednocześnie dostatecznie małe, aby obliczenia nie trwały zbyt długo: 64, 32, 16, 8 oraz 4. Gęstości grafu do przetestowania określony były z góry jako 25%, 50%, 75% oraz 99%. Dla każdej pary (ilość wierzchołków, gęstość) test powtarzany był 100 razy, a wyniki zostały uśrednione.

Grafy generowane były przy użyciu wbudowanego w standard C++11 silnika liczb pseudolosowych implementującego szeroko rozpowszechniony algorytm Mersenne Twister w wersji MT19937. Aby zapewnić spójność grafów, generowane były one według następującego algorytmu:

> 1. Wybierz losowo wierzchołek startowy i dołącz go do grafu
> 2. |V|-1 razy wybierz losowo wierzchołek jeszcze niedołączony do grafu oraz wierzchołek już dołączony do grafu; stwórz między nimi krawędź i dołącz nowy wierzchołek oraz stworzoną krawędź do grafu
> 3. Oblicz prawdopodobieństwo zawarcia w grafie krawędzi między dwoma wierzchołkami, tak, aby po

uwzględnieniu dołączonych już na początku krawędzi gęstość grafu dążyła do zadanej wartości procentowej
4. Dla każdej pary wierzchołków w grafie, jeżeli jeszcze nie ma między nimi krawędzi, wybierz losowo na podstawie obliczonego prawdopodobieństwa, czy wstawić między nimi krawędź; jeśli tak, wstaw krawędź

Czas wykonywania poszczególnych algorytmów mierzony był przy użyciu wbudowanego w standard C++11 mechanizmu czasomierza `std::chrono::high_resolution_clock`.

Program został napisany w języku C++, a następnie skompilowany i zlinkowany przez GCC 5.4.0, przy użyciu narzędzia CMake 3.10.3, dla środowiska x86-64 Linux/GNU. Testy zostały wykonane na komputerze wyposażonym w procesor AMD Ryzen 7 1700 @ 3.0~3.75GHz, pracującym pod kontrolą systemu Linux Mint 18.2 Sonya z 64-bitowym kernelem Linux 4.13.0-37-generic.

3. Wyniki

4. Wnioski