|  |  |  |  |
| --- | --- | --- | --- |
|  | **Nazwa kursu:** Struktury Danych i Złożoność Obliczeniowa (projekt nr 2.) | | |
| **Temat projektu:** „ Badanie efektywności operacji dodawania, usuwania oraz wyszukiwania elementów w różnych strukturach danych.” | | |
| **Autor:**  Piotr Ławniczak 209775 | **Data oddania:** 12.05.2015 | |
| **Grupa:** wtorek 11:15 | |
| **Prowadzący:**  Dr inż. Jarosław Mierzwa | **Ocena:** |

# Cel projektu

Celem projektu jest analiza wydajności algorytmów wyznaczających w grafie: minimalne drzewo rozpinające (algorytm Prima) oraz najkrótszą ścieżkę (algorytm Dijkstry). Polegać ona będzie na badaniach wykazujących zależność ich szybkości wykonania w zależności od rozmiaru, gęstości grafu, a także od rodzaju jego reprezentacji w pamięci komputera.

# Wstęp teoretyczny

Minimalnym drzewem rozpinającym (w skrócie ang. MST) nazywa się drzewo rozpinające danego grafu o najmniejszej z możliwych wag, tj. takie, że nie istnieje dla tego grafu inne drzewo rozpinające o mniejszej sumie wag krawędzi. Sposobów jego wyznaczania jest kilka. Jednym z nich jest algorytm Prima.

Algorytm Prima należy do grupy algorytmów zachłannych. Dysponując grafem spójnym i nieskierowanym, obliczany jest podzbiór ze zbioru krawędzi, dla którego graf nadal pozostaje spójny, jednak suma wag wszystkich takich krawędzi w owym podzbiorze jest najmniejsza z możliwych.

# Najważniejsze założenia

-wagi krawędzi są liczbami całkowitymi większymi od zera, przechowywane jako unsigned int  
-

# Reprezentacja grafów w pamięci komputera

Na potrzeby projektu zostały zaimplementowane dwa sposoby reprezentacji grafu:

1. Macierz sąsiedztwa

Sposób ten zakłada stworzenie dwuwymiarowej tablicy liczb całkowitych nieujemnych. Każdy jej element inicjalizowany jest wartością „0”, co oznacza, że między wierzchołkiem reprezentowanym przez numer wiersza a wierzchołkiem reprezentowanym przez numer kolumny nie występuje krawędź. W przypadku kiedy w danej komórce znajduje się liczba różna od zera – oznacza to występowanie krawędzi. Liczba ta jednocześnie jest jej wagą. Wymagania pamięciowe to O(V2). Operacje przejrzenia wszystkich krawędzi, sąsiadów danego wierzchołka i sprawdzenie, czy dana krawędź istnieje wykonują się w czasie O(V).

1. Lista sąsiedztwa

Reprezentacja listowa opiera się na utworzeniu w pierwszej kolejności tablicy zawierającej tyle list ile graf ma wierzchołków. Indeks tablicy odpowiada numerowi wierzchołka, a każdy element listy znajdującej się pod nim – wskazuje na jego sąsiadów (z którymi tworzy krawędź). Sąsiedzi przechowywani są w specjalnie stworzonej do tego celu strukturze, która oprócz numeru sąsiada zawiera kompletne informacje o krawędzi (wierzchołek początkowy, wierzchołek końcowy, waga). Złożoność pamięciowa wynosi O(E). Przejrzenie wszystkich krawędzi ma złożoność czasową O(E), natomiast przeglądanie sąsiadów wierzchołka i sprawdzanie istnienia krawędzi wykonuje się w czasie O(E).

# Implementacja algorytmów

1. Algorytm Prima – wyznaczanie minimalnego drzewa rozpinającego (MST)

Na początku algorytm dodaje do zbioru A reprezentującego drzewo krawędź o najmniejszej wadze, łączącą wierzchołek początkowy v z dowolnym wierzchołkiem. W każdym kolejnym kroku procedura dodaje do A najlżejszą krawędź wśród krawędzi łączących wierzchołki już odwiedzone z nieodwiedzonymi. Jeśli struktura A jest kolejką priorytetową opartą na kopcu binarnym, czasowa złożoność obliczeniowa operacji wynosi O(m \* log2n).

1. Algorytm Dijkstry – znajdowanie najkrótszych ścieżek w grafie

Algorytm Dijkstry znajduje w grafie najkrótsze ścieżki pomiędzy wybranym wierzchołkiem początkowym a wszystkimi pozostałymi, wyliczając również koszt przejścia każdej z tych ścieżek, czyli sumę wag krawędzi na ścieżce. Algorytm ma złożoność czasową O(n2) przy wykorzystaniu wyszukiwania liniowego podczas szukania wierzchołków o najmniejszym koszcie dojścia. Czas ten może być zmniejszony dzięki wykorzystaniu kolejki priorytetowej opartej na kopcu binarnym. Wtedy w korzeniu przechowywany jest wierzchołek o najmniejszej wartości kosztu dojścia. Złożoność czasowa upraszcza się do O(n \* log2n) – tyle, ile wynosi czas przywracania własności kopca.

# Plan eksperymentu

* Czas potrzebny do wykonania danej operacji będzie mierzony przy użyciu licznika cykli procesora wykonanych od jego uruchomienia - TSC (Time Stamp Counter). Dostęp do niego umożliwia funkcja QueryPerformanceCounter. Jego wartość będzie odczytywana i zapisywana tuż przed rozpoczęciem mierzonego polecenia, oraz zaraz po. Różnica podzielona przez ilość cykli jakie procesor wykonuje w czasie jednej sekundy (odczytaną przy użyciu funkcji QueryPerformanceFrequency) daje szczegółowy czas w sekundach do sześciu miejsc po przecinku.
* Pomiary wykonane będą dla grafów o rozmiarach 50, 75, 100, 125 i 150 wierzchołków. Dla każdego z rozmiarów rozpatrywane będą gęstości 25%, 50%, 75% i 99%. Aby pomiary były jak najbardziej wiarygodne, każdy przypadek będzie wykonywany po 100 razy, a wyniki uśredniane.
* W przypadku algorytmu Prima krawędzie są traktowane jako nieskierowane, a w przypadku algorytmu Dijkstry – skierowane.
* Krawędzie wielokrotne oraz pętle nie występują w grafach generowanych przez program.

# Wyniki eksperymentu

1. Algorytm Dijkstry
2. Macierz sąsiedztwa
3. Lista sąsiedztwa
4. Algorytm Prima
5. Macierz sąsiedztwa
6. Lista sąsiedztwa

# Wnioski

Na podstawie uzyskanych wyników nasuwają się jednoznaczne wnioski:

* Złożoności obliczeniowe zaimplementowanych przeze mnie algorytmów w większości zgadzają się z pierwotnymi założeniami na podstawie danych znalezionych w literaturze.
* Różnice w czasie wykonywania algorytmów pomiędzy reprezentacjami listowymi a macierzowymi zwłaszcza w przypadku dużych grafów – są bardzo duże.

# Bibliografia

[1] Robert L. Kruse – „Data Structures and Program Design in C++”

[2] Piotr Wróblewski – „Struktury Danych i Techniki Programowania – Algorytmy”

[3] Thomas H. Cormen – „Wprowadzenie do Algorytmów”

[4] Big-O Cheat Sheet (wersja angielska) - <http://bigocheatsheet.com/>