22.05.2018r.

Struktury danych i złożoność obliczeniowa

Projekt drugi

Prowadzący: dr inż.. Dariusz Banasiak  
Autor: Wojciech Wójcik

235621

Wtorek 15:15 TN

1. Wstęp

W programie zostały zaimplementowane grafy w postaci macierzy incydencji oraz listy następników. Do każdej postaci zostały zaimplementowane metody wyszukiwania najkrótszej drogi w grafie oraz wyznaczanie minimalnego drzewa rozpinającego.

**Graf** jest powszechnie stosowaną strukturą danych we współczesnej informatyce. Pozwala on modelować wszelkiego rodzaju sieci oraz układy, w których występują skomplikowane zależności pomiędzy składnikami. Powstała cała teoria matematyczna, zwana **Teorią Grafów,** która zajmuje się tymi strukturami, ich własnościami oraz zasadami wykonywania na nich różnych działań.

**Macierzy incydencji** – jest macierzą A o wymiarze *n* x *m*, gdzie *n* oznacza liczbę wierzchołków grafu, a *m* liczbę jego krawędzi. Każdy wiersz tej macierzy odwzorowuje jeden wierzchołek grafu. Każda kolumna odwzorowuje jedną krawędź. Zawartość komórki A[i,j] określa powiązanie (incydencję) wierzchołka *vi* z krawędzią *ej* w sposób następujący:

**Lista następników** – do reprezentacji grafu wykorzystujemy tablicę *n* elementową A, gdzie *n* oznacza liczbę wierzchołków. Każdy element tej tablicy jest listą. Lista reprezentuje wierzchołek startowy. Na liście są przechowywane numery wierzchołków końcowych, czyli sąsiadów wierzchołka startowego, z którymi jest on połączony krawędzią.

Złożoność obliczeniowa dla algorytmów wyszukiwania najkrótszej drogi oraz wyznaczania minimalnego drzewa rozpinającego w grafie:

* Algorytm Dijkstra(najkrótsza droga w grafie) : **O(E\*logV)**
* Algorytm Prima(minimalne drzewo rozpinające) : **O(E\*logV)**

1. Plan eksperymentu

Wykorzystane środowisko programistyczne: Visual Studio 2017

Język: C++

W programie zostały zaimplementowane struktury grafu oraz metody szukania najkrótszej ścieżki w grafie i wyznaczanie minimalnego drzewa rozpinającego.  
Dane do programów są wczytywane z pliku.

Testy wykonane dla 5 różnych liczb wierzchołków V(10, 20, 50, 80,100) oraz następujących gęstości grafu: 25%, 50%, 75% oraz 99%.

Waga krawędzi jest określana przez użytkownika podczas procesu generowania grafu.

Na potrzeby pomiarów maksymalna wagę ustaliłem jako 10.  
  
Mierzenie czasu zostało wykonane za pomocą QueryPerformanceCounter.   
Pomiary mierzone w ms. Funkcja do mierzenia czasu uruchamiana przed daną funkcją oraz zatrzymywana zaraz po zakończeniu funkcji.

1. Wyniki czasowe dla podanych algorytmów

Użyte oznaczenia na wykresach: MI-macierz incydencji, LD-lista sąsiedztwa.

Algorytm Prima:

Algorytm Kruskala

Tabele z wynikami

Tabela Wyniki dla Prima [ms]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Czasy uśrednione dla | Ilosc wierzchołków | Gęstość | algorytm Prima listowo | algorytm Prima macierz |
| 100 powtórzeń [ms] | 10 | 25% | 9 | 0,184 |
|  | 50% | 11,18 | 0,216 |
|  | 75% | 13,56 | 0,251 |
|  | 99% | 16,39 | 0,286 |
|  | 20 | 25% | 90,25 | 1,266 |
|  |  | 50% | 126 | 1,706 |
|  |  | 75% | 164 | 2,152 |
|  |  | 99% | 201 | 2,823 |
|  | 30 | 25% | 364 | 4,961 |
|  |  | 50% | 561 | 7,265 |
|  |  | 75% | 778 | 9,052 |
|  |  | 99% | 952 | 12,19 |
|  | 40 | 25% | 1032 | 11,93 |
|  |  | 50% | 1665 | 20,06 |
|  |  | 75% | 2302 | 28,54 |
|  |  | 99% | 2893 | 35,06 |
|  | 50 | 25% | 2364 | 29,27 |
|  |  | 50% | 3981 | 49,20 |
|  |  | 75% | 5477 | 66,68 |
|  |  | 99% | 7080 | 83,63 |

Tabela Wyniki dla postaci Dijkstry [ms]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Czasy uśrednione dla | Ilosc wierzchołków | Gęstość | algorytm Djikstra listtowo | algorytm Djikstra macierzowo |
| 100 powtórzeń [ms] | 10 | 25% | 0,277 | 0,102 |
|  | 50% | 0,333 | 0,128 |
|  | 75% | 0,371 | 0,122 |
|  | 99% | 0,417 | 0,148 |
|  | 20 | 25% | 0,556 | 0,203 |
|  |  | 50% | 0,792 | 0,386 |
|  |  | 75% | 1,08 | 0,605 |
|  |  | 99% | 1,131 | 0,542 |
|  | 30 | 25% | 0,834 | 0,312 |
|  |  | 50% | 1,72 | 1,38 |
|  |  | 75% | 1,58 | 0,89 |
|  |  | 99% | 1,87 | 1,26 |
|  | 40 | 25% | 1,64 | 1,27 |
|  |  | 50% | 2,21 | 2,00 |
|  |  | 75% | 4,02 | 3,92 |
|  |  | 99% | 3,64 | 3,37 |
|  | 50 | 25% | 2,23 | 2,70 |
|  |  | 50% | 3,91 | 5,66 |
|  |  | 75% | 4,47 | 6,01 |
|  |  | 99% | 6,12 | 7,86 |

1. Analiza wyników

Algorytm Dejikstry pokrywa się z założeniami. Wraz ze wzrostem rozmiaru grafu lista następników wyraźnie zaczyna dawać lepsze rezultaty. Wszystko to jednak zależy od przypadku grafu.

Algorytm Prima mimo poprawnego zaimplementowania i bliźniaczej zasadzie działania daje niebywale gorsze rezultaty w postaci listy następników. Różnica ta, jest tak duża, że wskazuje na błąd w zaimplementowanej klasie odpowiedzialnej za przechowywanie grafu w formie listowej, albo na niepoprawne zaimplementowanie algorytmu.  
Wyniki algorytmu Prima w formie macierzowej są dużo bardziej prawdopodobne, dlatego uważam , że ten algorytm został częściowo poprawnie zaimplementowany.

1. Bibliografia

* Materiał z ćwiczeń dr inż. Jarosława Mierzwy
* <http://eduinf.waw.pl/inf/>