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

AiZO - zadanie projektowe nr 2

Autor : Arkadiusz Błasiak

Nr albumu: 273022

Data: 13.06.2024 godz. 11.15

# Wprowadzenie

### Pojęcie złożoności obliczeniowej

Złożoność obliczeniowa algorytmu określa, jak wydajny jest algorytm, ile musi on wykonać operacji w zależności ilości danych oraz ile potrzebuje do tego pamięci. Często zdarza się, że dany problem algorytmiczny można rozwiązać kilkoma metodami, czyli algorytmami o różnej złożoności obliczeniowej. Złożoność obliczeniową dzielimy na złożoność pamięciową oraz złożoność czasową.

W tym projekcie pomijamy złożoność pamięciową i skupiamy się w pełnie na złożoności czasowej.

### Kilka ważnych terminów:

* złożoność czasowa

ilość czasu potrzebnego do wykonania zadania, wyrażona jako funkcja ilości danych.

* złożoność obliczeniowa

ilość zasobów komputerowych potrzebnych do wykonania zadania.

* złożoność oczekiwana

inaczej złożoność średnia; ilość zasobów potrzebna do zrealizowania zadania dla statystycznie oczekiwanych danych; zapisywana za pomocą notacji theta –

* złożoność pesymistyczna

ilość zasobów potrzebna do zrealizowania zadania w przypadku najgorszych danych; zapisywana za pomocą notacji „duże O” –

### Reprezentacje grafów używane przez program

**Macierz incydencji** (ang. incidence matrix) jest macierzą **A** o wymiarze n × 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:

Obraz zawierający tekst, Czcionka, biały, pismo odręczne

Opis wygenerowany automatycznie

Znalezienie wszystkich sąsiadów wierzchołka vi nie jest bezpośrednie.

Operacja ta ma złożoność O (|V||E|).

**lista sąsiedztwa** (ang. adjacency lists) jest tablicą 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.

Znalezienie sąsiadów wierzchołka ma złożoność O (|E|).

# Algorytmy grafowe

## Algorytm Prima

algorytm zachłanny wyznaczający tzw. minimalne drzewo rozpinające. Mając do dyspozycji graf nieskierowany i spójny, tzn. taki w którym krawędzie grafu nie mają ustalonego kierunku oraz dla każdych dwóch wierzchołków grafu istnieje droga pomiędzy nimi, algorytm oblicza podzbiór E′ zbioru krawędzi E, dla którego graf nadal pozostaje spójny, ale suma kosztów wszystkich krawędzi zbioru E′ jest najmniejsza możliwa.

Złożoność czasowa:

## Algorytm Dijkstry

Algorytm opracowany przez holenderskiego informatyka Edsgera Dijkstrę, służy do znajdowania najkrótszej ścieżki z pojedynczego źródła w grafie o nieujemnych wagach krawędzi. jest przykładem algorytmu zachłannego. Jest często używany w sieciach komputerowych, np. przy trasowaniu.

Złożoność czasowa:

## Algorytm Bellmana-Forda

algorytm służący do wyznaczania najkrótszych ścieżek w grafie. Wyznacza najkrótsze ścieżki z jednego wierzchołka (zwanego wierzchołkiem źródłowym) do pozostałych wierzchołków. W odróżnieniu od algorytmu Dijkstry, algorytm Bellmana-Forda dopuszcza krawędzie o ujemnych wagach, nie mogą istnieć jednak ujemne cykle osiągalne z wierzchołka źródłowego. Algorytm może być również wykorzystywany do sprawdzania, czy w grafie występują ujemne cykle.

Złożoność czasowa:

# Metody

1. Program zapisany został w języku C++ wykorzystując programowanie obiektowe,
2. Tablice do posortowania tworzone są dynamicznie,
3. Tablice są zapełnione liczbami typu int bądź też specjalnie zaimplementowaną strukturą dla reprezentacji listy sąsiedztwa,
4. Generowanie gafu rozpoczyna się od stworzenia grafu pełnego. Następnie wykonywany jest na nim znajdujący MST, by ostatecznie rozwinąć graf o dodatkowe krawędzie do wymaganej gęstości grzewa.
5. W algorytmach Prima oraz Dijkstry zamiast kolejki użyto tablicy ze względu na jej nie poprawne działania (mowa o liście z biblioteki STL)
6. Program uruchamiany był na laptopie HP Pavilion – 15-bc402nw z procesorem Intel Core i5-8300H, kartą graficzną Nvidia GeForece GTX 1050 oraz 16 GB pamięcią RAM,
7. Czas mierzony był, kiedy laptop podłączony był do prądu, w trybie pełnej wydajności, minimalizując udział niepotrzebnych procesów w tle,
8. Do odmierzania czasu program posługuje się funkcją std::chrono::high\_resolution\_clock z biblioteki chrono,
9. Czas mierzony był mikrosekundach i zaokrąglany do pełnych milisekund.

# Pomiary

## Minimalne Drzewo Spinające

#### Algorytm Prima

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | czas [ms] dla zadanej gęstości [%] | | | | | |
| macierz incydencji | | | lista sąsiedztwa | | |
| liczba wierzchołków | 25% | 50% | 99% | 25% | 50% | 99% |
| 10 | 0,006 | 0,008 | 0,012 | 0,004 | 0,005 | 0,007 |
| 25 | 0,029 | 0,046 | 0,074 | 0,013 | 0,016 | 0,019 |
| 50 | 0,144 | 0,248 | 0,59 | 0,047 | 0,058 | 0,069 |
| 100 | 1,04 | 2,038 | 3,812 | 0,198 | 0,238 | 0,33 |
| 200 | 8,537 | 29,956 | 60,701 | 0,931 | 1,19 | 1,622 |
| 400 | 168,074 | 358,685 | 701,495 | 3,335 | 5,101 | 8,054 |
| 800 | 1582,38 | 2919,92 | 6074,92 | 18,218 | 30,644 | 51,15 |

## Najkrótsza ścieżka

#### Algorytm Dijkstry

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | czas [ms] dla zadanej gęstości [%] | | | | | |
| macierz incydencji | | | lista sąsiedztwa | | |
| liczba wierzchołków | 25% | 50% | 99% | 25% | 50% | 99% |
| 10 | 0,011 | 0,013 | 0,017 | 0,005 | 0,006 | 0,008 |
| 25 | 0,034 | 0,051 | 0,082 | 0,014 | 0,015 | 0,21 |
| 50 | 0,148 | 0,252 | 0,461 | 0,048 | 0,052 | 0,064 |
| 100 | 1,047 | 2,1 | 3,831 | 0,181 | 0,2119 | 0,278 |
| 200 | 8,273 | 29,529 | 82,747 | 0,806 | 1,018 | 1,468 |
| 400 | 175,004 | 361,91 | 704,723 | 3,708 | 4,667 | 6,847 |
| 800 | 1502,38 | 2777,76 | 5997,65 | 19,699 | 25,47 | 53,856 |

#### Algorytm Bellmana-Forda

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | czas [ms] dla zadanej gęstości [%] | | | | | |
| macierz incydencji | | | lista sąsiedztwa | | |
| liczba wierzchołków | 25% | 50% | 99% | 25% | 50% | 99% |
| 10 | 0,004 | 0,006 | 0,01 | 0,001 | 0,02 | 0,002 |
| 25 | 0,045 | 0,086 | 0,151 | 0,005 | 0,006 | 0,008 |
| 50 | 0,34 | 0,582 | 1,404 | 0,012 | 0,019 | 0,028 |
| 100 | 3,661 | 7,667 | 14,073 | 0,046 | 0,087 | 0,198 |
| 200 | 31,864 | 101,822 | 294,149 | 0,238 | 0,488 | 1,498 |
| 400 | 711,468 | 1732,85 | 3154,561 | 1,689 | 4,409 | 8,913 |
| 800 | 7225,15 | 16339,8 | 40965,9 | 10,576 | 22,834 | 64,393 |

#### Porównanie

# Podsumowanie i wnioski

Widać zdecydowaną przewagę implementacji grafu w postaci listy sąsiadów względem macierzy incydencji. Pomimo że obie reprezentacje na wykresach wykazują taką samą złożoność, to przy bezpośrednim porównaniu reprezentacja listy sąsiedztwa jest o rzędy wielkości szybsza od reprezentacji macierzy incydencji.

Implementacja algorytmów Prima oraz Dijkstry jest bardzo podobna, dlatego nie dziwi fakt, że wyniki pomiarów są zbliżone, a wykresy bliźniacze.

Algorytm Dijkstry jest szybszy w porównaniu do algorytmu Bellmana-Forda. Jednak jest [pojedynczy przypadek](#bf_wygrywa), gdzie to algorytm Bellmana forda jest lepszy od algorytmu Dijkstry.

Wszystkie algorytmy zachowują złożoność zgodną z przewidywaniami.

# Źródła

* <https://wikipedia.org/>
* <https://eduinf.waw.pl/inf/alg/001_search/0124.php#P3>
* <http://algorytmy.ency.pl/artykul/algorytm_bellmana_forda>