PROJEKTOWANIE ALGORYTMÓW I METODY SZTUCZNEJ INTELIGENCJI – GRAFY

Wtorek 15:15

Rafał Rzewucki 248926

Prowadzący:

Mgr inż. Marcin Ochman

# Wstęp

Celem projektu było zbadać efektywność algorytmu Bellmana-Forda w zależności od sposobu reprezentacji grafu.

# Opis Algorytmu Bellmana-Forda

# Algorytm służący do znalezienia najkrótszej ścieżki w grafie wazony z wierzchołka źródłowego do wszystkich pozostałych wierzchołków. Działanie algorytmu opiera się na metodzie relaksacji (sprawdzaniu, czy przy przejściu dana krawędzią grafu, nie otrzymamy krótszej ścieżki niż dotychczasowa).

# Algorytm Bellmana-Forda w porównaniu z algorytmem Dijkstry jest wolniejszy, jednak bardziej uniwersalny. Algorytm ten jest bowiem w stanie obsługiwać grafy z ujemnymi wartościami krawędzi oraz wykrywać ujemne cykle, które być może zostały stworzone przez te ujemne wartości.

# 3. Opis programu

Program składa się z pliku głównego main.cpp oraz dodatkowych plików nagłówkowych w tym z plików:

* GraphGenerator.h – zawiera klasę, która pozwala wygenerować losowy graf o zadanej liczbie wierzchołków, oraz gęstości,
* Edge.h – zawiera klasę, która definiuje obiekt krawędzi w grafie,
* Vertex.h – zawiera klasę, która definiuje obiekt wierzchołka w grafie,
* ListGraph.h – zawiera klasę definiującą podstawowe operacje grafu oraz wszystkie metody grafu. Definiuje listę sąsiedztwa, dziedziczy po klasie GraphGenerator,
* MatrixGraph.h – zawiera klasę dziedziczącą po klasie ListGraph. Przeciąża ona niektóre metody, aby zamiast listy sąsiedztwa uzyskać macierz sąsiedztwa.

Do badania czasu wykonywania algorytmu została wykorzystana biblioteka „chrono” a dokładniej zegar w niej zawarty o nazwie „high\_resolution\_clock”, który gwarantuje największą możliwą na danym komputerze rozdzielczość pomiaru czasu.

W pliku main.cpp został również zaimplementowany mechanizm umożliwiający uruchomienie testowania każdego algorytmu z osobna równocześnie na osobnych wątkach, co umożliwia pełne wykorzystanie możliwości obliczeniowych komputera i znacznie przyśpiesza testowanie zaimplementowanych klas grafów. Prezentacja wyników

Dla jak najlepszego wyeliminowania błędów pomiarów związanych z uruchomionymi dodatkowymi programami w tle, pomiary zostały powtórzone trzy razy i została wyciągnięta z nich średnia arytmetyczna.

# Otrzymane rezultaty

1. Wyniki efektywności zaimplementowanego algorytmu Bellmana-Forda dla wszystkich testowanych gęstości grafu.
   1. Lista sąsiedztwa

|  |  |  |
| --- | --- | --- |
| **Liczba wierzchołków** | **Gęstość grafu [%]** | **Czas [ms]** |
| 10 | 25% | 0,0067 |
| 10 | 50% | 0,0069 |
| 10 | 75% | 0,0070 |
| 10 | 100% | 0,0070 |
| 50 | 25% | 0,1460 |
| 50 | 50% | 0,1471 |
| 50 | 75% | 0,1456 |
| 50 | 100% | 0,1415 |
| 100 | 25% | 0,5721 |
| 100 | 50% | 0,5761 |
| 100 | 75% | 0,5823 |
| 100 | 100% | 0,5839 |
| 500 | 25% | 15,700 |
| 500 | 50% | 17,728 |
| 500 | 75% | 17,065 |
| 500 | 100% | 16,738 |
| 1000 | 25% | 77,533 |
| 1000 | 50% | 83,290 |
| 1000 | 75% | 80,433 |
| 1000 | 100% | 80,311 |

* 1. Macierz sąsiedztwa

|  |  |  |
| --- | --- | --- |
| **Liczba wierzchołków** | **Gęstość grafu [%]** | **Czas [ms]** |
| 10 | 25% | 0,0148 |
| 10 | 50% | 0,0152 |
| 10 | 75% | 0,0149 |
| 10 | 100% | 0,0152 |
| 50 | 25% | 0,3418 |
| 50 | 50% | 0,3437 |
| 50 | 75% | 0,3415 |
| 50 | 100% | 0,3424 |
| 100 | 25% | 1,6457 |
| 100 | 50% | 1,8230 |
| 100 | 75% | 1,6960 |
| 100 | 100% | 1,3944 |
| 500 | 25% | 42,960 |
| 500 | 50% | 41,703 |
| 500 | 75% | 43,350 |
| 500 | 100% | 42,722 |
| 1000 | 25% | 159,08 |
| 1000 | 50% | 161,43 |
| 1000 | 75% | 166,69 |
| 1000 | 100% | 151,33 |
|  |  |  |

# Wyniki efektywności dla poszczególnych gęstości grafu:

* 1. Gęstość 25%

|  |  |  |  |
| --- | --- | --- | --- |
| Liczba wierzchołków | Czas [ms] | | Stosunek czasów (Macierz/Lista) |
| Lista | Macierz |
| 10 | 0,0067 | 0,0148 | 2,22 |
| 50 | 0,1460 | 0,3418 | 2,34 |
| 100 | 0,5721 | 1,6457 | 2,88 |
| 500 | 15,6998 | 42,9602 | 2,74 |
| 1000 | 77,5333 | 159,0820 | 2,05 |

* 1. Gęstość 50%

|  |  |  |  |
| --- | --- | --- | --- |
| Liczba wierzchołków | Czas [ms] | | Stosunek czasów (Macierz/Lista) |
| Lista | Macierz |
| 10 | 0,0069 | 0,0152 | 2,22 |
| 50 | 0,1471 | 0,3437 | 2,34 |
| 100 | 0,5761 | 1,8230 | 3,16 |
| 500 | 17,7276 | 41,7033 | 2,35 |
| 1000 | 83,2901 | 161,4320 | 1,94 |

* 1. Gęstość 75%

|  |  |  |  |
| --- | --- | --- | --- |
| Liczba wierzchołków | Czas [ms] | | Stosunek czasów (Macierz/Lista) |
| Lista | Macierz |
| 10 | 0,0070 | 0,0149 | 2,15 |
| 50 | 0,1456 | 0,3415 | 2,34 |
| 100 | 0,5823 | 1,6960 | 2,91 |
| 500 | 17,0647 | 43,3501 | 2,54 |
| 1000 | 80,4331 | 166,6940 | 2,07 |

* 1. Pełny graf (100%)

|  |  |  |  |
| --- | --- | --- | --- |
| Liczba wierzchołków | Czas [ms] | | Stosunek czasów (Macierz/Lista) |
| Lista | Macierz |
| 10 | 0,0070 | 0,0152 | 2,18 |
| 50 | 0,1415 | 0,3424 | 2,42 |
| 100 | 0,5839 | 1,3944 | 2,39 |
| 500 | 16,7377 | 42,7219 | 2,55 |
| 1000 | 80,3114 | 151,3310 | 1,88 |

# Podsumowanie

Uzyskane czasy przetwarzania zaimplementowanego algorytmu dla różnych liczb wierzchołków i gęstości grafu wyraźnie wskazują, że algorytm dla reprezentacji grafu za pomocą listy sąsiedztwa jest bardziej efektywny niż dla reprezentacji za pomocą macierzy sąsiedztwa. Może być to spowodowane faktem, że w macierzy sąsiedztwa, przeszukiwana jest cała macierz co o wiele bardziej wydłuża czas dostępu do danych niż w zaimplementowanej liście sąsiedztwa, gdzie przeszukiwana jest tylko jej część.

Należy również zaznaczyć, że reprezentacja grafu za pomocą macierzy sąsiedztwa potrzebuje znacznie więcej pamięci niż lista sąsiedztwa.

# Literatura

* <https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm>
* <https://eduinf.waw.pl/inf/alg/001_search/0124.php>
* <https://eduinf.waw.pl/inf/alg/001_search/0138a.php>