**Struktury danych i złożoność obliczeniowa – projekt**

**Sprawozdanie**

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

Autor: Michał Bańka, 235051

Termin zajęć: Wtorek TP, 15:15

Prowadzący: Dr inż. Dariusz Banasiak

1. Wstęp

Istnieje wiele reprezentacji grafów w komputerze. Najpopularniejsze z nich to macierz sąsiedztwa, macierz incydencji, lista następników/poprzedników i *forward star*. Na potrzeby projektu zostały użyte dwie z nich, mianowicie macierz incydencji i lista następników/poprzedników.

Macierz incydencji to macierz o rozmiarze [E,V]. Każdy wiersz przetrzymuje informację o początku i końcu krawędzi.

Lista następników/poprzedników to tablica list dwukierunkowych. Kolejne indeksy tablicy oznaczają numer wierzchołka, natomiast to co znajduje się w każdej z list dwukierunkowych to krawędzie wychodzące z wierzchołków.

W projekcie zostały zaimplementowane po dwa algorytmy dla każdego z dwóch problemów:

1. Problem MST – minimalne drzewo rozpinające
   1. Algorytm Prima

Złożoność obliczeniowa: – dla implementacji kolejki priorytetowej opartej na kopcu binarnym

* 1. Algorytm Kruskala

Złożoność obliczeniowa: – dla implementacji kolejki priorytetowej opartej na kopcu binarnym

1. Problem najkrótszej ścieżki w grafie
   1. Algorytm Dijkstry:
   2. Algorytm Bellmana-Forda:
2. Założenia pomiarów

Założeniem pomiarów ma być wyznaczenie średniego czasu wykonywania powyższych algorytmów. Zgodnie z wytycznymi projektu, każdy algorytm wykonany został po 100 razy dla każdej możliwej kombinacji gęstości *d* (25%, 50%, 75%, 99%) i ilości wierzchołków *V* (20, 40, 60, 80, 100). Pomiędzy każdym wykonaniem algorytmu dane w obu reprezentacjach były zmieniane, jednak tak aby grafy przedstawiane przez te reprezentacje były takie same.

Po wykonaniu każdego algorytmu czas wykonania był dodawany do odpowiedniej zmiennej. Na sam koniec wszystkie dane zapisywane były do plików tekstowych,  
a w procesie tworzenia dokumentacji podzielone przez 100 tak aby otrzymać średni czas.

Losowe generowanie grafu nie pozwala na utworzenie grafu niespójnego. W pierwszym etapie dodawania danych krawędzią łączone są wszystkie wierzchołki *n* i *n+1*,  
gdzie *n = { 0, 1, … , V-1}.* Zapobiega to występowaniu błędów podczas wykonywania algorytmów, które muszą być wykonywane na grafach spójnych.

1. Wyniki pomiarów
2. Problem MST – reprezentacja macierzowa

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Średni czas algorytmu Prima - macierz [ms] | | | | | |
| V  d | 20 | 40 | 60 | 80 | 100 |
| 25% | 0,13 | 0,79 | 2,34 | 5,22 | 9,84 |
| 50% | 0,18 | 1,07 | 3,24 | 7,24 | 13,7 |
| 75% | 0,22 | 1,34 | 4,11 | 9,22 | 17,52 |
| 99% | 0,26 | 1,59 | 4,94 | 11,22 | 21,76 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Średni czas algorytmu Kruskala - macierz [ms] | | | | | |
| V  d | 20 | 40 | 60 | 80 | 100 |
| 25% | 0,31 | 2,5 | 9,61 | 25,65 | 55,38 |
| 50% | 0,56 | 5,64 | 24,24 | 69,47 | 158,53 |
| 75% | 0,89 | 10,46 | 47,05 | 139 | 326,85 |
| 99% | 1,27 | 16,52 | 76,59 | 231,57 | 550,56 |

1. Problem MST – reprezentacja listowa

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Średni czas algorytmu Prima - lista [ms] | | | | | |
| V  d | 20 | 40 | 60 | 80 | 100 |
| 25% | 0,13 | 0,77 | 2,23 | 4,92 | 9,17 |
| 50% | 0,18 | 1,02 | 3 | 6,61 | 12,33 |
| 75% | 0,22 | 1,26 | 3,74 | 8,25 | 15,44 |
| 99% | 0,26 | 1,48 | 4,45 | 9,89 | 18,68 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Średni czas algorytmu Kruskala - lista [ms] | | | | | |
| V  d | 20 | 40 | 60 | 80 | 100 |
| 25% | 0,31 | 2,5 | 9,61 | 25,65 | 55,38 |
| 50% | 0,56 | 5,64 | 24,24 | 69,47 | 158,53 |
| 75% | 0,89 | 10,46 | 47,05 | 139 | 326,85 |
| 99% | 1,27 | 16,52 | 76,59 | 231,57 | 550,56 |

1. Wyznaczanie najkrótszej ścieżki – reprezentacja macierzowa

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Średni czas algorytmu Dijkstra - macierz [ms] | | | | | |
| V  d | 20 | 40 | 60 | 80 | 100 |
| 25% | 0,71 | 9,36 | 44,47 | 136,38 | 326,36 |
| 50% | 1,33 | 18,27 | 87,34 | 269,45 | 647,86 |
| 75% | 1,95 | 27,02 | 130,33 | 402,04 | 970,39 |
| 99% | 2,53 | 35,48 | 171,55 | 530,88 | 1280,32 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Średni czas algorytmu Bellmana-Forda - macierz [ms] | | | | | |
| V  d | 20 | 40 | 60 | 80 | 100 |
| 25% | 0,2 | 2,56 | 11,84 | 35,43 | 83,54 |
| 50% | 0,41 | 5,17 | 23,79 | 71,21 | 167,59 |
| 75% | 0,61 | 7,75 | 35,8 | 106,66 | 253 |
| 99% | 0,81 | 10,23 | 47,31 | 141,56 | 334,77 |

1. Wyznaczanie najkrótszej ścieżki – reprezentacja listowa

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Średni czas algorytmu Dijkstra - lista [ms] | | | | | |
| V  d | 20 | 40 | 60 | 80 | 100 |
| 25% | 0,11 | 0,58 | 1,69 | 3,78 | 7,12 |
| 50% | 0,12 | 0,61 | 1,76 | 4 | 7,41 |
| 75% | 0,12 | 0,63 | 1,85 | 4,22 | 7,65 |
| 99% | 0,13 | 0,66 | 2 | 4,41 | 7,92 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Średni czas algorytmu Bellmana-Forda - lista [ms] | | | | | |
| V  d | 20 | 40 | 60 | 80 | 100 |
| 25% | 0,04 | 0,18 | 0,43 | 0,82 | 1,34 |
| 50% | 0,07 | 0,34 | 0,79 | 1,56 | 2,64 |
| 75% | 0,11 | 0,47 | 1,17 | 2,31 | 3,81 |
| 99% | 0,14 | 0,6 | 1,46 | 2,93 | 4,79 |

1. Porównanie czasów dla gęstości 25% - reprezentacja macierzowa
2. Porównanie czasów dla gęstości 25% - reprezentacja macierzowa
3. Porównanie czasów dla gęstości 75% - reprezentacja macierzowa
4. Porównanie czasów dla gęstości 99% - reprezentacja macierzowa
5. Porównanie czasów dla gęstości 25% - reprezentacja listowa
6. Porównanie czasów dla gęstości 50% - reprezentacja listowa
7. Porównanie czasów dla gęstości 75% - reprezentacja listowa
8. Porównanie czasów dla gęstości 99% - reprezentacja listowa

1. Wnioski
   * Problem MST wykonuje się dłużej na reprezentacji listowej niż na macierzowej.
   * Przeciwnie jest podczas wyszukiwania najkrótszej ścieżki, te algorytmy na liście wykonują się niemal błyskawicznie, a na macierzy względnie długo.
   * Algorytm Bellmana-Forda jest szybszy od algorytmu Dijkstry.
   * Algorytm Kruskala jest wolniejszy od algorytmu Prima.
   * Podczas wyszukiwania najkrótszej ścieżki w reprezentacji listowej algorytm Dijkstry zdaje się być bardzo mało lub wcale zależny od ilości wierzchołków.
   * W reprezentacji listowej niezależnie od gęstości algorytm Kruskala jest znacznie wolniejszy od pozostałych trzech.