**Struktury danych i złożoność obliczeniowa**

**Zadanie projektowe nr 2**

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

Radosław Zimoch 263963

1. **Wstęp**

W ramach projektu należało zaimplementować określone algorytmy grafowe oraz dokonać pomiarów czasu potrzebnego do ich wykonania w zależności od ilości wierzchołków i gęstości grafu.

Zrealizowane algorytmy obejmowały minimalne drzewo rozpinające, gdzie zastosowano Algorytm Prima oraz Algorytm Kruskala. Dodatkowo, badano problem najkrótszej ścieżki przy użyciu Algorytmu Dijkstry oraz Algorytmu Bellmana Forda.

Badania zostały przeprowadzone dla wszystkich możliwych kombinacji liczby wierzchołków, tj. 10, 25, 50, 75 i 100, oraz różnych poziomów gęstości grafu: 25%, 50%, 75% oraz 99%. Ta szeroka gama wartości pozwoliła na kompleksową analizę zachowania algorytmów w różnych warunkach.

Dokładne pomiary czasu wykonania dla każdej kombinacji liczby wierzchołków i gęstości grafu umożliwiły otrzymanie konkretnych danych dotyczących efektywności algorytmów. Wyniki zostały przedstawione w postaci tabel oraz wykresów, które pozwoliły na łatwą analizę zależności czasu wykonania od rozmiaru grafu i jego gęstości.

1. **Pomiary**
2. Minimalne drzewo rozpinające:

**Algorytm Prima**

Teoretyczna złożoność obliczeniowa *O(ElogV)*

Do sortowania rozpatrywanych krawędzi malejąco względem wagi został wykorzystany kopiec typu minimum. Wykorzystanie tej struktury daje lepszą (niższą) pesymistyczną złożoność obliczeniową, niż kolejka priorytetowa, która skutkowałaby wyższą złożonością. Algorytm Prima rozpoczyna od jednego wierzchołka i stopniowo rozbudowuje drzewo rozpinające, dodając kolejne wierzchołki i krawędzie o najmniejszej wadze, które łączą już istniejące drzewo z nowym wierzchołkiem. Jest on często wydajny dla gęstych grafów (o dużym stosunku liczby krawędzi do liczby wierzchołków), gdzie E jest bliskie V^2.

Wyniki pomiarów:

Wartości podane są w mikrosekundach.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Liczba wierzchołków | | | | |
| Gęstość [%] | 10 | 25 | 50 | 75 | 100 |
| 25 | 4,08 | 22,19 | 88,58 | 229,36 | 500,09 |
| 50 | 5,66 | 29,73 | 141,36 | 408,85 | 852,58 |
| 75 | 5,83 | 33,68 | 176,76 | 534,00 | 1114,52 |
| 99 | 6,29 | 38,31 | 210,95 | 651,32 | 1382,79 |

Tabela 1 Implementacja macierzowa

Tabela 2 Implementacja listowa

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Liczba wierzchołków | | | | |
| Gęstość [%] | 10 | 25 | 50 | 75 | 100 |
| 25 | 3,13 | 18,22 | 79,22 | 206,90 | 459,53 |
| 50 | 4,89 | 27,15 | 132,42 | 387,13 | 828,06 |
| 75 | 5,91 | 36,36 | 180,31 | 565,57 | 1196,04 |
| 99 | 7,77 | 44,12 | 230,08 | 807,35 | 1572,09 |

**Algorytm Kruskala**

Teoretyczna złożoność obliczeniowa *O(ElogE)*

Do sortowania krawędzi względem wag wykorzystany został kopiec typu minimum. Algorytm Kruskala sortuje wszystkie krawędzie grafu według ich wag i następnie iteruje po posortowanej liście, dodając krawędzie do drzewa rozpinającego, jeśli nie tworzą one cyklu z już dodanymi krawędziami. Algorytm Kruskala jest często wydajny dla rzadkich grafów (o małym stosunku liczby krawędzi do liczby wierzchołków), gdzie E jest bliskie V lub mniejsze.

Wyniki pomiarów:

Wartości podane są w mikrosekundach.

Tabela 3 Implementacja macierzowa

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Liczba wierzchołków | | | | |
| Gęstość [%] | 10 | 25 | 50 | 75 | 100 |
| 25 | 3,12 | 16,70 | 74,96 | 146,89 | 274,01 |
| 50 | 5,41 | 31,15 | 129,26 | 295,53 | 585,43 |
| 75 | 7,13 | 43,23 | 209,88 | 438,12 | 1191,25 |
| 99 | 8,96 | 54,29 | 251,84 | 575,48 | 1264,28 |

Tabela 4 Implementacja listowa

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Liczba wierzchołków | | | | |
| Gęstość [%] | 10 | 25 | 50 | 75 | 100 |
| 25 | 6,06 | 61,19 | 1052,15 | 2977,12 | 9200,38 |
| 50 | 13,19 | 193,25 | 2609,61 | 11700,20 | 39487,70 |
| 75 | 21,30 | 431,15 | 6324,97 | 26247,80 | 130807,00 |
| 99 | 32,09 | 672,57 | 10007,40 | 46495,10 | 167723,00 |

Wykresy typu I:  
Wykresy typu II:

1. Problem najkrótszej ścieżki

**Algorytm Dijkstry**

Teoretyczna złożoność obliczeniowa *O(E + V logV)*

Algorytm Dijkstry znajduje najkrótsze ścieżki od jednego wierzchołka do wszystkich innych w grafie, przy założeniu, że wagi krawędzi są nieujemne. Algorytm rozpoczyna od wierzchołka startowego, a następnie stopniowo rozbudowuje ścieżki do innych wierzchołków, wybierając krawędzie o najmniejszej wadze. Algorytm Dijkstry jest często wydajny dla grafów o małej gęstości (mały stosunek liczby krawędzi do liczby wierzchołków) i jest szczególnie efektywny, gdy graf jest reprezentowany przez kopiec binarny.

Wyniki pomiarów:

Wartości podane są w mikrosekundach

Tabela 5 Implementacja macierzowa

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Liczba wierzchołków | | | | |
| Gestość [%] | 10 | 25 | 50 | 75 | 100 |
| 25 | 49,26 | 154,22 | 442,11 | 874,39 | 1573,01 |
| 50 | 46,88 | 180,99 | 597,43 | 1260,78 | 2468,15 |
| 75 | 53,52 | 225,96 | 799,41 | 1664,35 | 3148,33 |
| 99 | 57,87 | 259,88 | 936,05 | 2322,59 | 4585,34 |

Tabela 6 Implementacja listowa

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Liczba wierzchołków | | | | |
| Gestość [%] | 10 | 25 | 50 | 75 | 100 |
| 25 | 36,53 | 142,35 | 405,42 | 810,63 | 1536,80 |
| 50 | 39,54 | 174,96 | 536,25 | 1259,53 | 2549,64 |
| 75 | 47,55 | 220,63 | 836,86 | 1725,83 | 3453,63 |
| 99 | 51,00 | 254,18 | 970,12 | 2573,98 | 5061,50 |

**Algorytm Bellmana - Forda**

Teoretyczna złożoność obliczeniowa *O(E ∗ V)*

Powyższa złożoność wynika z faktu, że w najgorszym wypadku algorytm musi “odwiedzić” wszystkie krawędzie w grafie V razy (gdzie V jest liczbą wierzchołków).

W przeciwieństwie do algorytmu Dijkstry, algorytm Bellmana - Forda jest skuteczny w wyszukiwaniu ścieżek w grafach zawierających cykle ujemne, gdyż kończy się w momencie zakończenia propagacji cykli ujemnych. Przykładowo, algorytm Dijkstry nie jest w stanie rozwiązać problemu najkrótszej ścieżki dla grafu z ujemnymi cyklami (mógłby się nigdy nie zakończyć).

Wyniki pomiarów:

Wartości podane są w mikrosekundach

Tabela 7 Implementacja macierzowa

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Liczba wierzchołków | | | | |
| Gęstość [%] | 10 | 25 | 50 | 75 | 100 |
| 25 | 80,12 | 1045,58 | 6763,69 | 22559,99 | 49081,01 |
| 50 | 107,90 | 1507,57 | 11064,28 | 39122,75 | 89486,49 |
| 75 | 158,53 | 1956,62 | 15105,56 | 54991,74 | 116037,97 |
| 99 | 172,80 | 2557,56 | 21828,06 | 59952,23 | 149188,88 |

Tabela 8 Implementacja listowa

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Liczba wierzchołków | | | | |
| Gęstość [%] | 10 | 25 | 50 | 75 | 100 |
| 25 | 77,16 | 715,05 | 5175,21 | 18413,76 | 45941,56 |
| 50 | 96,40 | 1287,44 | 11367,86 | 34396,84 | 87206,78 |
| 75 | 132,02 | 1767,97 | 14505,88 | 51903,36 | 126170,96 |
| 99 | 155,79 | 2222,84 | 18893,88 | 73563,44 | 181436,48 |

Wykresy typu I:

Wykresy typu II:

Wnioski:

Na podstawie przeprowadzonych eksperymentów oraz analizy wyników można wyciągnąć następujące wnioski dotyczące efektywności poszczególnych struktur w rozwiązywaniu problemów Minimalnego Drzewa Rozpinającego (MST) oraz Najkrótszej Ścieżki:

* Reprezentacja grafu ma istotny wpływ na czas wykonania algorytmów. W przypadku problemu MST, reprezentacja macierzowa okazała się bardziej efektywna niż listowa. Dla problemu Najkrótszej Ścieżki, obie reprezentacje dały podobne wyniki.
* W przypadku problemu MST, algorytm Prima osiągał lepsze wyniki czasowe niż algorytm Kruskala. Dla problemu Najkrótszej Ścieżki, algorytm Dijkstry był zdecydowanie szybszy od algorytmu Bellmana-Forda.
* Wraz ze wzrostem gęstości grafu, czas wykonania algorytmów również wzrastał. Problem MST wykazywał większą wrażliwość na gęstość grafu niż problem Najkrótszej Ścieżki.
* Analizując złożoności teoretyczne a uzyskane wyniki eksperymentalne, można stwierdzić, że istnieją pewne rozbieżności. Przyczyny takich różnic mogą wynikać z czynników takich jak implementacja algorytmów, specyfika generowanych grafów, a także sposób pomiaru czasu wykonania.

Zakończenie:

W trakcie przeprowadzonych eksperymentów analizujących efektywność algorytmów rozwiązujących problemy Minimalnego Drzewa Rozpinającego (MST) oraz Najkrótszej Ścieżki, uzyskane wyniki potwierdzają istotność wyboru odpowiedniej reprezentacji grafu oraz algorytmu w celu optymalizacji czasu wykonania.

Algorytm Prima okazał się efektywniejszy niż algorytm Kruskala w przypadku problemu MST, podczas gdy algorytm Dijkstry był znacznie szybszy od algorytmu Bellmana - Forda w problemie Najkrótszej Ścieżki.

W przypadku gęstości grafu, obserwowano wzrost czasu wykonania algorytmów. Istotne jest zatem odpowiednie dostosowanie algorytmów do specyfiki problemu i oczekiwanej gęstości grafu.

Różnice między złożonościami teoretycznymi a uzyskanymi wynikami eksperymentalnymi mogą wynikać z różnych czynników, takich jak implementacja algorytmów czy specyfika generowanych grafów.