**Sprawozdanie**

**Projektowanie Algorytmów i Metody Sztucznej Inteligencji**

25.03.2020

Kacper Grabowski 248930

Dr inż. Andrzej Rusiecki

Czwartek 13:15

1. **Wprowadzenie.**

Zadaniem napisanego programu, jest wyliczenie, ile czasu zajmuje sporządzenie rozwiązanie problemu najkrótszej drogi w grafie między dwoma wierzchołkami. Do rozwiązania tego problemu wykorzystywany jest algorytm Dijkstry. Rozróżniamy grafy o różnych gęstościach. Graf o gęstości 100%, jest to taki, gdzie istnieje każda możliwa krawędź. Zadane parametry grafów, to wszystkie kombinacje wielkości równej 100, 200, 500, 1000 oraz 2000 z gęstością 25%, 50%, 75% i 100%. Mierzony czas, to czas działania metody Dijk(int).

Wykorzystujemy 2 warianty algorytmu Dijkstry. Pierwsza z nich wykorzystuje listę, druga macierz. W dalszej części sprawozdania porównujemy działanie tych algorytmów.

1. **Opis wykorzystanego algorytmu.**

Algorytm Dijkstry- dla danego źródła w grafie algorytm wyszukuje po kolei najkrótszą dostępną krawędź i dodaje ją do grafu, jeśli nie łączy ona już połączonym wierzchołkiem. Działanie to jest powtarzane, aż wszystkie wierzchołki zostaną połączone z grafem startowym.

Złożoność algorytmu z wykorzystaniem macierzy:

Złożoność algorytmu z wykorzystaniem listy:

1. **Omówienie przebiegu eksperymentów oraz uzyskane wyniki.**

Program ma trzy tryby działania. Pierwszym jest główny, to jest stu krotne policzenie najkrótszych ścieżek dla losowych grafów o 5 wielkościach i 4 gęstościach oraz policzenie czasu działania dla każdego wyznaczania oddzielnie po czym zapisanie go do pliku. Dodatkowymi opcjami jest policzenie jednego grafu o wielkości 100 i gęstości 50% przy użyciu wybranego algorytmu i wyświetlenie ścieżek dla każdego wierzchołka oraz wczytanie danych z pliku i ścieżek do pliku.

Do wykonania eksperymentów wykorzystywałem tylko pierwszą opcję, która utworzyła dwa pliki, które znajdują się w załącznikach (Wyniki\_L.txt oraz Wyniki\_M.txt). Otrzymane dane przekopiowałem do Excel ’a i sporządziłem wykresy.

1. **Tabele i wykresy z wynikami.**

Szczegółowe dane znajdują się w dodatkowym pliku znajdującym się w załącznikach w pliku

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Gęstość 25% | | | | | | | | | | |
|  | Lista | | | | |  | Macierz | | | | |
| Gestosc [%] | 100 | 200 | 500 | 1000 | 2000 |  | 100 | 200 | 500 | 1000 | 2000 |
| Średnia [ms] | 0,33642 | 2,186648 | 10,8835 | 62,15168 | 282,7459 |  | 0,33642 | 1,797939 | 11,59525 | 66,49808 | 294,6235 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Gęstość 50% | | | | | | | | | | |
|  | Lista | | | | |  | Macierz | | | | |
| Gestosc [%] | 100 | 200 | 500 | 1000 | 2000 |  | 100 | 200 | 500 | 1000 | 2000 |
| Średnia [ms] | 0,636576 | 2,884727 | 26,8832 | 131,7748 | 622,4364 |  | 0,636576 | 3,000113 | 28,40453 | 136,5448 | 636,6234 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Gęstość 75% | | | | | | | | | | |
|  | Lista | | | | |  | Macierz | | | | |
| Gestosc [%] | 100 | 200 | 500 | 1000 | 2000 |  | 100 | 200 | 500 | 1000 | 2000 |
| Średnia [ms] | 0,968732 | 4,914792 | 45,17299 | 208,071 | 975,8378 |  | 0,968732 | 5,116046 | 47,37083 | 213,4669 | 1002,699 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Gęstość 100% | | | | | | | | | | |
|  | Lista | | | | |  | Macierz | | | | |
| Gestosc [%] | 100 | 200 | 500 | 1000 | 2000 |  | 100 | 200 | 500 | 1000 | 2000 |
| Średnia [ms] | 1,370225 | 7,28649 | 63,12452 | 289,1287 | 1365,561 |  | 1,370225 | 7,255417 | 64,91877 | 296,1348 | 1441,529 |

1. **Wnioski.**

Co pokazują powyższe tabele oraz wykresy, algorytm Dijkstry z wykorzystaniem Listy działa lekko szybciej niż algorytm wykorzystujący macierz. Tendencja ta powinna być znacznie lepiej zauważalna dla wielkich zestawów danych, jednak program, który napisałem nie był w stanie wykonywać obliczeń na grafach o ilości wierzchołków większej niż 9000. Najprawdopodobniej wynika to z zbyt dużej ilości pamięci RAM potrzebnej do przechowywania zmiennych (około 81 milionów gałęzi, dla gęstości 100%). Ilość wykorzystywanej pamięci przy 9000 wierzchołków dochodziła do około 1.5 GB.

1. **Zródła:**
2. <https://eduinf.waw.pl/inf/alg/001_search/0138.php>
3. <https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm>
4. <https://pl.wikipedia.org/wiki/Algorytm_Dijkstry>
5. <https://stackoverflow.com/questions/12039728/inheriting-base-class-defining-the-base-class-prototype-method-but-calling-chi>
6. <https://www.w3schools.com/>
7. <http://www.cplusplus.com/>