Projektowanie algorytmów i metody sztucznej inteligencji

Projekt 2

Termin zajęć: środa 18.55-20.35

1. **Wstęp**

Celem projektu było zaimplementowanie grafów w oparciu o listę sąsiedztwa oraz macierzy sąsiedztwa. Zbadanie wydajności jednego z algorytmów wyszukującego najkrótszą drogę (użyto algorytmu Djikstry)

1. **Wykonanie**

Zaimplementowano grafy dwoma metodami. Program został napisany obiektowo wszystkie klasy i ich funkcje zostały dodatkowo zapisane w różnych plikach co ułatwia analizę kodu. W algorytmie Djikstry zostało zastosowane wyszukiwanie liniowe (bez kopca) przez co jego pesymistyczna czasowa złożoność obliczeniowa wynosi

Program posiada możliwość wczytania grafu z pliku jak i możliwość generowania losowych grafów, zapisywania ich do pliku oraz umożliwia zapisanie wyniku w pliku tekstowym.

Testy przeprowadzane były dla 4 różnych gęstości wypełnienie grafu 25%, 50%, 75%

Oraz 100% dla 5 różnych ilości wierzchołków 10, 50, 100, 200 i 300 wykonano je na komputerze stacjonarnym z podzespołami:

Procesor - AMD Ryzen 5 1500X 3.60GHz

RAM - 2666MHz 16Gb

System operacyjny - Windows 10 64bit

1. **Wyniki badań**

Poniżej przedstawiona jest tabela zawierająca czas (w mikrosekundach[us]) wykonywania się algorytmu w zależności od ilości wierzchołków oraz od gęstości wypełnienia grafu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ilość wierzchołków | Lista | | | |
| Gęstość | 25% | 50% | 75% | 100% |
| 10 | 7,8 | 7,7 | 7,4 | 7,9 |
| 50 | 20,2 | 21 | 20,7 | 20,8 |
| 100 | 38,1 | 40,3 | 40,9 | 41,3 |
| 200 | 81,3 | 84,8 | 88,6 | 99,2 |
| 300 | 130,2 | 157,1 | 153,5 | 164 |
|  | Macierz | | | |
| Gęstość | 25% | 50% | 75% | 100% |
| 10 | 7,7 | 7,6 | 7,5 | 7,3 |
| 50 | 20,9 | 21,1 | 21,1 | 20,7 |
| 100 | 38,5 | 38,8 | 40,3 | 39,7 |
| 200 | 83,1 | 81,4 | 80,8 | 85,9 |
| 300 | 127,3 | 128,8 | 131,6 | 133,3 |

Wykresy pierwszego typu przedstawiają ile czasu wykonywał się algorytm w zależności od gęstości, każdy wykres przedstawia inna implementacje grafu.

Wykresy drugiego typu pokazują zależność miedzy implementacjami grafu dla określonych gęstości.

1. **Wnioski**

Analizując otrzymane dane, można stwierdzić że gęstość grafu nie ma aż tak znaczącego wpływu na czas wykonywania się algorytmu Djikstry różnicę można zaobserwować jedynie przy implementacji z użyciem listy sąsiedztwa przy macierzy różnica nie jest prawie widoczna, największy wpływ na czas działania ma ilość wierzchołków.

Implementacja grafu ma duży wpływ na czas wykonywania się algorytmu widać to na wykresach drugiego typu gdzie w większości przypadków algorytm dla grafu macierzy sąsiedztwa wykonywał się szybciej, może być to spowodowane użyciem liniowego przeszukiwania w algorytmie Djikstry.

Ważnym elementem programu jest również możliwość generowania losowych grafów o zadanych parametrach, należy jednak zwrócić uwagę iż przy generowaniu grafów pełnych czas działania całego programu znacznie się zwiększa, przy grafach o ilości wierzchołków większych niż 300 czas działania był nie współmierny do wartości uzyskanych danych. Kolejną uwagą powinien być fakt, iż w niektórych wypadkach graf wypełniony w 75% był wykonywany szybciej niż ten wypełniony w 50% wynika to prawdopodobnie i danych wylosowanych do grafu i z sieci połączeń wytworzonych przez to losowanie.

Podsumowując wszystkie funkcje programu działają poprawnie jest napisany w pełni obiektowo   
z zachowaniem wszelkich standardów. Nie jest to jednak najbardziej optymalne rozwiązanie zadanego problemu. Algorytm działałby szybciej gdyby przeszukiwanie było zaimplementowane przy użyciu kopca. Funkcja generująca graf również powinna zostać dopracowana jest słabo wydajna dla grafów pełnych.

1. **Literatura**

[1] <https://eduinf.waw.pl/inf/alg/001_search/index.php>

[2] <https://pl.wikipedia.org/wiki/Graf_(matematyka)>

[3] <https://pl.wikipedia.org/wiki/Graf_skierowany>

[4] <https://pl.wikipedia.org/wiki/Graf_pe%C5%82ny>

[5] <http://lukasz.jelen.staff.iiar.pwr.wroc.pl/styled-2/page-2/index.php> (wykłady)