Projektowanie Algorytmów i Metody Sztucznej Inteligencji
Projekt 2
Grafy

Krzysztof Górski, 245079
prowadzący Mgr Marta Emirsajłow

grupa piątek, 13:15-15:00

1 Wstęp

1.1 Cel ćwiczenia

Celem ćwiczenia było zapoznanie się, implementacją oraz eksperymentalne testowanie algorytmu najkrótszej, polegające na badaniu złożoności czasowej algorytmu podczas przeszukiwania 100 instancji grafu o różnych rozmiarach oraz róznych stopniach gęstości. Eksperyment polega na znalezieniu w grafie ważonym skierowanym najkrótszego połaczenia pomiędzy dwoma wierzchołkami.Graf zaimplementowano obiektowo, z dwoma typami reprezentacji - listą sąsiedztwa oraz macierzą sąsiedztwa. Zastosowano algorytm Bellmana-Forda.

Algorytm zaimplementowano w formie szablonowej z zachowaniem zasad programowania obiektowego. Program pozwala na wprowadzenie grafu z pliku, jak również zapis wyników poszukiwania najkrótszej ścieżki, zapis grafu do pliku, oraz metody potrzebne do generowania, oraz manipulowania parametrami grafu. Algorytm Bellmana-Forda został zaimplementowany dla dwóch reprezentacji grafu, w postaci macierzy sąsiedztwa oraz listy sąsiedztwa.

2 Algorytm

2.1 Algorytm Bellmana-Forda

Algorytm polega na relaksacji, tzn. następuję przeszukiwanie wszystkich połączeń dwóch wierzchołków, w których ścieżka jest zastępowana lepszymi połączeniami, aż do otrzymania najkrószej możliwej drogi. Każda nowe bardziej optymalne połączenie zastępowane jest sumą poprzednich połączeń oraz wagą nowej krawędzi. Z każdym wykonaniem pętli relaksacji algorytm znajduje najkrószą scieżkę do nowego wierzchołka.

Algorytm posiada następującą złożoność obliczeniowa:

- **średni przypadek** - $O(|V| \cdot |E|)$
  - V - liczba wierzchołków
  - E - liczba krawędzi

3 Przebieg eksperymentu oraz wyniki pomiarów

ekpseryment polegał na porównaniu złożoności czasowej algorytmu dla dwóch reprezentacji grafu(lista sąsiedztwa oraz macierz sąsiedztwa), przeprowadzono pomiary czasu wykonania dla 100 instancji grafu, każda o następujących rozmiarach:
100, 200, 500, 750, 1000
oraz różnych stopniach gęstości:
$25\%, 50\%, 75\%, 100\%$ .Zatem wykonano algorytm 100x5x4=20000 razy.

do odmierzenia czasu zastosowano bibliotekę *chrono*. Następnie obliczono średni czas sorotwania jednej tablicy.

Algorytm BF dla 100 instancji Grafu, lista sąsiedztwa				
[s]	25%	50%	75%	100%
100	0,146431	0,347803	0,44896	0,63152
200	1,45096	2,95419	4,42406	5,93968
500	28,0846	40,58325	83,04	132,1738
750	168,776	197,6385	407,078	540,322
1000	212,367	344,829	926,902	1247,954

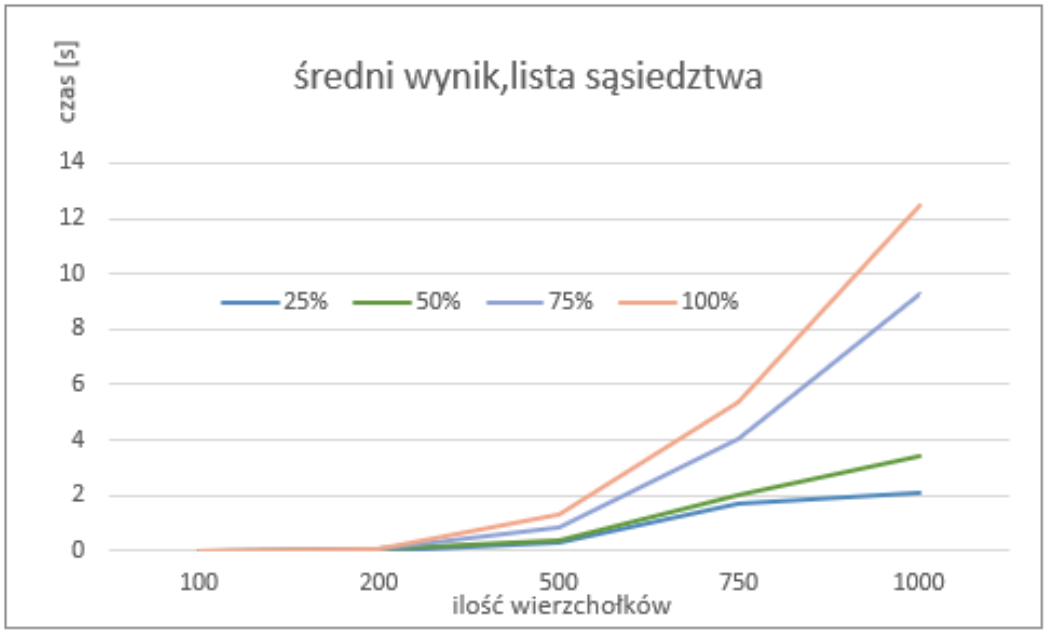
Algorytm BF dla 100 instancji Grafu, macierz sąsiedztwa				
[s]	25%	50%	75%	100%
100	0,82236	1,338	0,90858	0,4406
200	6,46552	10,34928	7,24334	3,43844
500	101,6518	160,1416	115,2624	63,926
750	352,268	558,764	399,662	229,318
1000	956,588	1361,374	1069,596	859,946

Rysunek 1: Wyniki pomiaru czasu dla 100 instancji

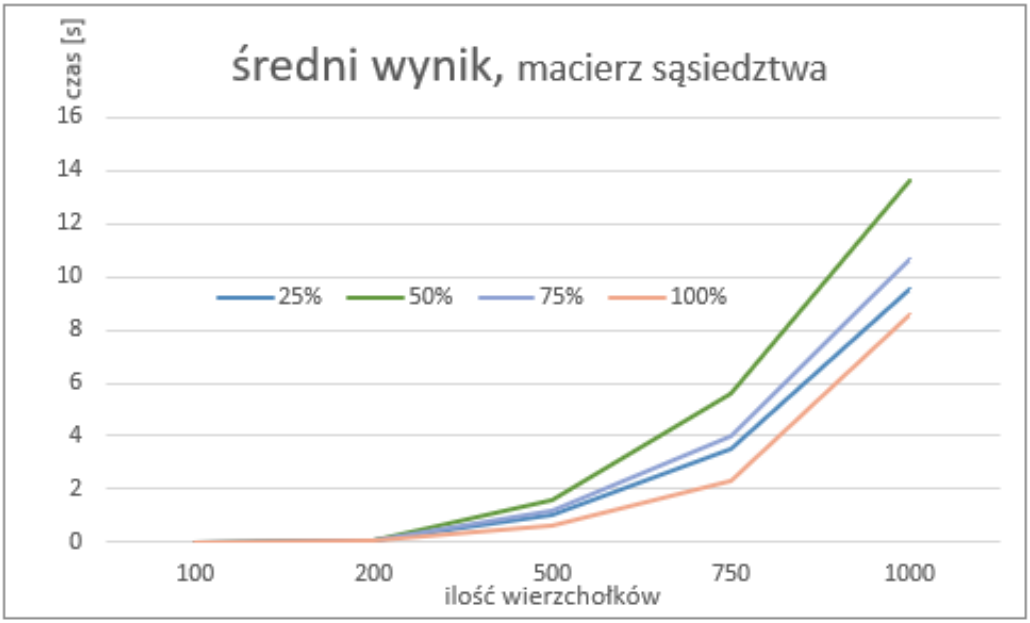
Wynik średni, macierz sąsiedztwa				
[s]	25%	50%	75%	100%
100	0,0082236	0,01338	0,0090858	0,004406
200	0,0646552	0,1034928	0,0724334	0,0343844
500	1,016518	1,601416	1,152624	0,63926
750	3,52268	5,58764	3,99662	2,29318
1000	9,56588	13,61374	10,69596	8,59946

Wynik średni, lista sąsiedztwa				
[s]	25%	50%	75%	100%
100	0,0014643	0,00347803	0,0044896	0,0063152
200	0,0145096	0,0295419	0,0442406	0,0593968
500	0,280846	0,4058325	0,8304	1,321738
750	1,68776	1,976385	4,07078	5,40322
1000	2,12367	3,44829	9,26902	12,47954

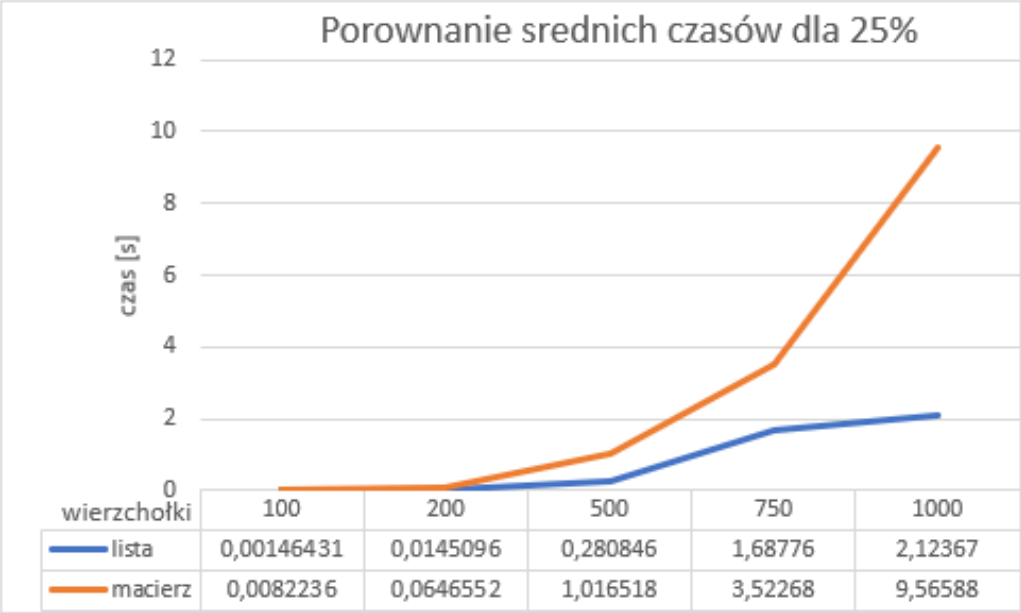
Rysunek 2: Sredni czas szukania ścieżki dla jednego grafu



Rysunek 3: Wykres średnich czasów - Lista

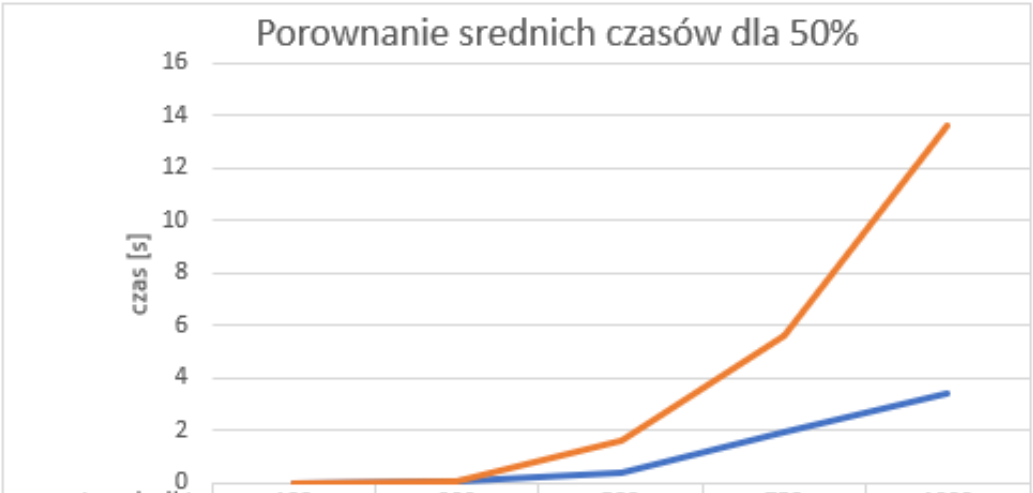


Rysunek 4: Wykres średnich czasów - Macierz



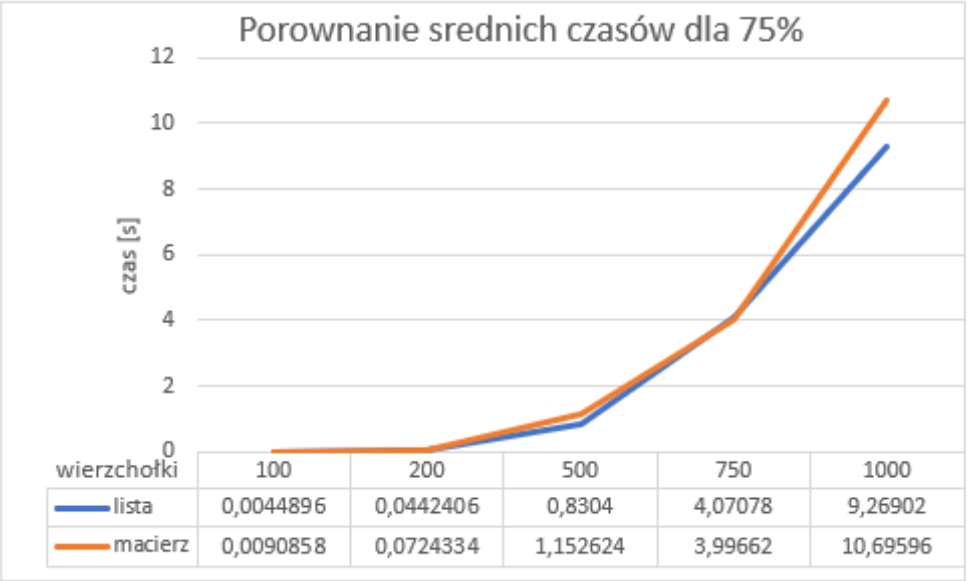
wierzchołki	100	200	500	750	1000
lista	0,00146431	0,0145096	0,280846	1,68776	2,12367
macierz	0,0082236	0,0646552	1,016518	3,52268	9,56588

Rysunek 5: Porownanie czasów dla listy oraz macierzy przy 25% gęstości grafu



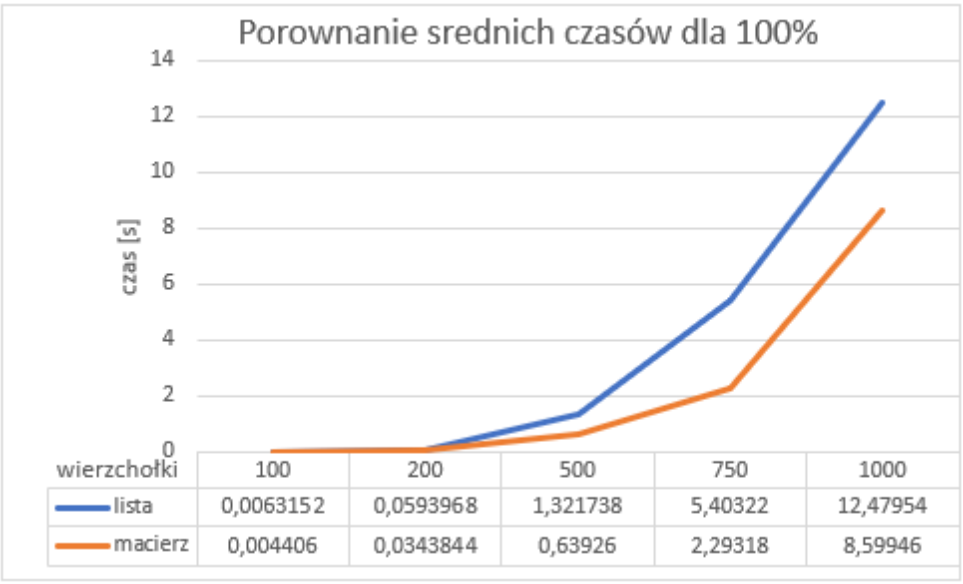
wierzchołki	100	200	500	750	1000
lista	0,00347803	0,0295419	0,4058325	1,976385	3,44829
macierz	0,01338	0,1034928	1,601416	5,58764	13,61374

Rysunek 6: Porownanie czasów dla listy oraz macierzy przy 50% gęstości grafu



wierzchołki	100	200	500	750	1000
lista	0,0044896	0,0442406	0,8304	4,07078	9,26902
macierz	0,0090858	0,0724334	1,152624	3,99662	10,69596

Rysunek 7: Porownanie czasów dla listy oraz macierzy przy 75% gęstości grafu



wierzchołki	100	200	500	750	1000
lista	0,0063152	0,0593968	1,321738	5,40322	12,47954
macierz	0,004406	0,0343844	0,63926	2,29318	8,59946

Rysunek 8: Porownanie czasów dla listy oraz macierzy przy 100% gęstości grafu

4 Wnioski

Algorytm Bellmana-Forda jest doskonałym algorytmem poszukiwania najkrótszej ścieżki, może on operować na różnych typach danych, dopuszcza wagi ujemne, oraz bada istnienie cyklu ujemnego. Algorytm ma zdecydowanie lepszy (mniejszy) czas przeszukiwania dla reprezentacji grafu w postaci listy sąsiedztwa. Różnica zachowania algorytmu między reprezentacjami jest szczególnie widoczna dla wiekszych Grafów, gdzie czas otrzymywany dla reprezentacji macierzowej jest o 200% do nawet 450% większy niż w reprezentacji listowej. Wyjątkiem był przypadek 100% gęstości grafu, czyli takiej, w której każdy wierzchołek ma krawędź łączącą go z każdym innym. W tym przpadku algorytm dla macierzy okazał się do 150% szybszy. Wynika to najprawdopodobniej od wybranego zakresu wag w eksperymencie tj. 1-10. W takim przpadku z dużym prawodopodobieństwem najkrótsza ścieżka będzie miała maksymalnie długość 2 krawędzi. Ze względu na konstrukcje macierzową, istnieje bezpośredni dostęp do krawędzi łączącej dany wierzchołek ze startowym, która stanowi w dużej liczbie par jedyne potrzebne połączenie, zatem algorytm nie wykonuje dalszych poszukiwań dla danej krawędzi. W przypadku listy, ze względu na konstrukcję liniową, należy przejść wszystkich sąsiadów znajdujących się na liście przed poszukiwanym połączeniem, co w przypadku pesymistycznym, oznacza przjrzenie wszystkich pozostałych krawędzi wychodzących dla danego wierzchołka. Podsumowując algorytm zadziałał zgodnie z przewidywaniem, poza przypadkiem 100ęstości. Dzięki implementacji szablonowej oraz obiektowej, algorytm można wykorzystać dla róznych typów danych, z zadowalającą złożonością czasową.

Literatura

[1] Cormen T., Leiserson C.E., Rivest R.L., Stein C., *Wprowadzenie do algorytmów, WNT*

[2] Drozdek A., *C++. Algorytmy i struktury danych, Helion*

[3] http://users.v-lo.krakow.pl/~toma/algorytmy/Algorytmy%20grafowe.pdf

[4] https://eduinf.waw.pl/inf/alg/001_search/0138a.php

[5] https://eduinf.waw.pl/inf/alg/001_search/0086.php

[6] https://pl.wikipedia.org/wiki/Algorytm_Bellmana-Forda