Imię i nazwisko: Patryk Ostrowski
Numer indeksu: 240671
Dane prowadzącego: Marta Emirsajłow
Termin zajęć: Piątek 7:30

Projektowanie algorytmów i metody sztucznej inteligencji

Projekt 2: Grafy – badanie efektywności algorytmu Bellmana - Forda

1. Wprowadzenie:

1.1 Opis projektu:

Celem projektu było zbadanie efektywności jednego z algorytmów wyznaczającego najkrótszą ścieżkę między wybranym wierzchołkiem do pozostałych wierzchołków grafu. Do wyboru był algorytm Dijkstry i Bellmana - Forda. W projekcie testowany był algorytm Bellmana – Forda w zależności od sposobu reprezentacji grafu (w postaci macierzy i listy) oraz gęstości grafu. W obu przypadkach reprezentacji grafu, testowane grafy były nieskierowane.
Badania zostały wykonane dla 5 różnych liczb wierzchołków:
• 10
• 25
• 50
• 100
• 250

oraz dla następujących gęstości grafu:
• 25%
• 50%
• 75%
• 100% (graf pełny)

1.2 Budowa programu:

Dla każdego zestawu: reprezentacja grafu, liczba wierzchołków i gęstość wygenerowano po 100 losowych grafów i zbadano czas działania algorytmu Bellmana-Forda.
Program został napisany obiektowo. Stworzono abstrakcyjną klasę Graph, która została opisana w pliku graph.hpp. Po klasie matce dziedziczą klasa GraphList oraz klasa GraphMatrix. Definicja klasy GraphList została opisana w pliku graph_list.hpp, a implementacja metod tej klasy jest w pliku graph_list.cpp. Definicja klasy GraphMatrix jest w pliku graph_matrix.hpp, a implementacja metod tej klasy w pliku graph_matrix.cpp. Odpowiednio klasa GraphList pozwala nam stworzyć reprezentację grafu jako lista sąsiedztwa a GraphMatrix jako macierz sąsiedztwa. Dodatkowo stworzono klasę vectorClass na przykładzie kontenera danych vector z biblioteki STL w celu łatwiejszego obsługi algorytmu Bellmana - Forda dla macierzy sąsiedztwa. Plik bellmanford.hpp zawiera definicję funkcji pozwalającej zbadać algorytm Bellmana – Forda dla listy i macierzy sąsiedztwa. W pliku bellmanford.cpp znajduje się implementacja tych funkcji. Program dodatkowo pozwala wczytać graf z pliku dla macierzy sąsiedztwa i listy sąsiedztwa oraz zapisać wyniki działania algorytmu dla listy i macierzy do pliku.

1.3 Opis algorytmu Bellmana – Forda:

Algorytm służy do znalezienia najkrótszej ścieżki w grafie ważonym z wierzchołka źródłowego do wszystkich pozostałych wierzchołków. Stosuję on metodę relaksacji krawędzi, czyli sprawdzaniu czy przy przejściu przez daną krawędź grafu, nie otrzymamy krótszej ścieżki niż dotychczas.
Algorytm Bellmana – Forda jest wolniejszy w swoim działaniu w porównaniu do algorytmu Dijkstry. Jednakże jest on bardziej uniwersalny, ponieważ może on być stosowany w grafach z ujemnymi wartościami wag krawędzi oraz wykrywać cykle ujemne, które mogły by powstać przez ujemne wagi w grafie.
Złożoność obliczeniowa:
• Dla reprezentacji w postaci listy sąsiedztwa -> O(VE)
• Dla reprezentacji w postaci macierzy sąsiedztwa -> $O(V^3)$

Złożoność pamięciowa algorytmu przy założeniu, że bierzemy pod uwagę również graf:
• Dla reprezentacji w postaci listy sąsiedztwa -> O(V+E)
• Dla reprezentacji w postaci macierzy sąsiedztwa -> $O(V^2)$

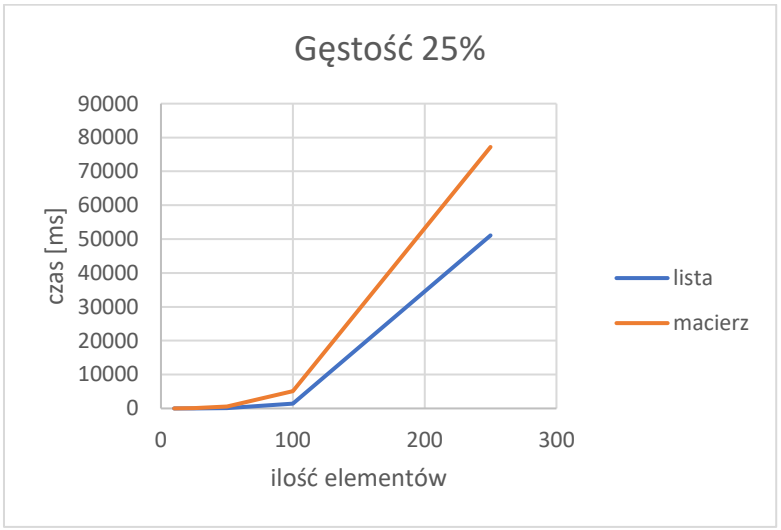
2. Testy

2.1 Przewidywane wyniki:

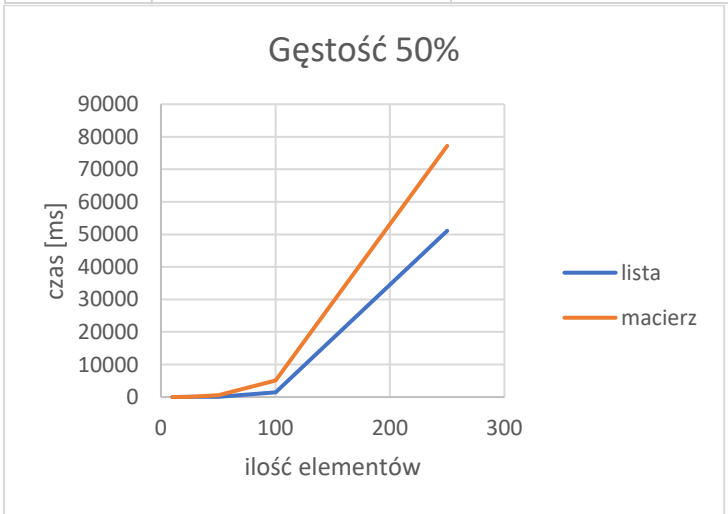
Złożoność obliczeniowa macierzy sąsiedztwa zależy tylko od liczby węzłów i wynosi $O(V_3)$.
Złożoność obliczeniowa listy sąsiedztwa zależy od liczby węzłów oraz gęstości i wynosi O(VE).
Porównując obie złożoności widać, że wraz ze wzrostem badanych liczby węzłów czas dla macierzy powinien być co raz dłuższy w porównaniu do badania algorytmu dla listy. Dla 250 wierzchołków różnica powinna być największa. Nawet dla najbardziej pesymistycznego przypadku lista powinna być szybsza niż macierz.

2.2 Przebieg testów:

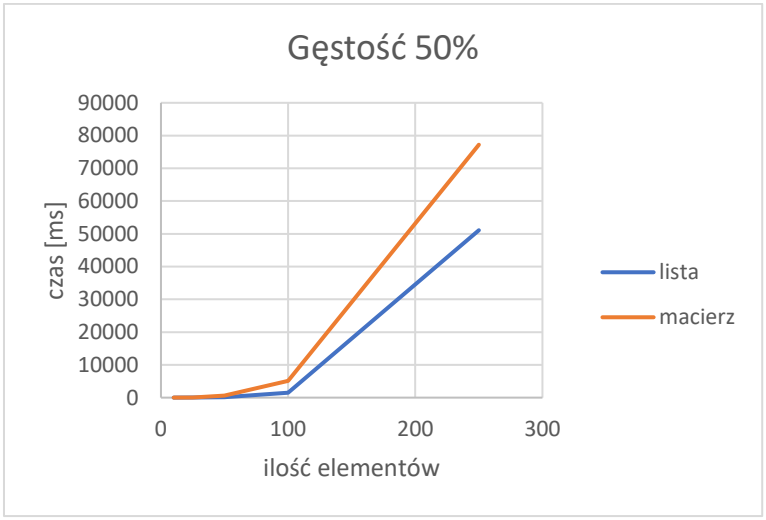
gęstość 25%		
wierzchołki	czas dla różnych reprezentacji	
	lista[ms]	macierz [ms]
10	0,53	2,44
25	7,68	30,68
50	39,77	262,86
100	264,68	2074,67
250	7611,49	36377,2



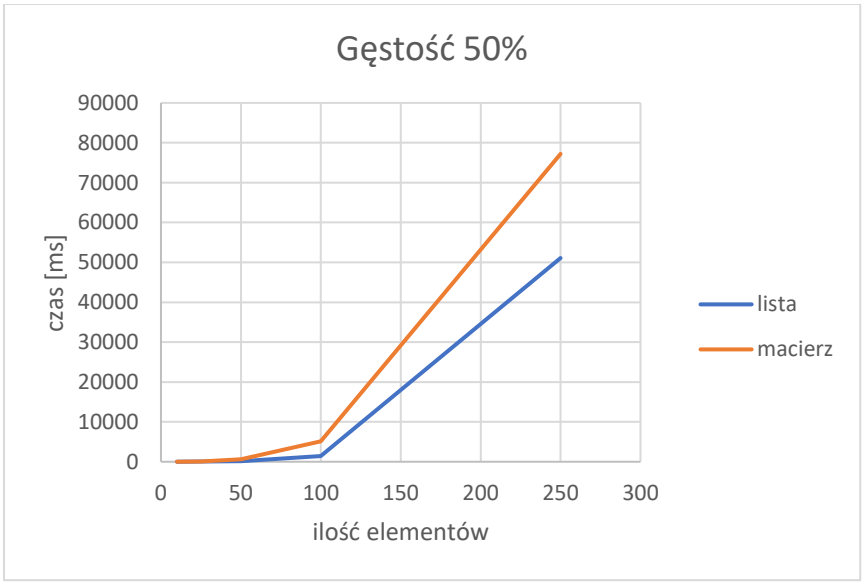
gęstość 50%		
wierzchołki	czas dla różnych reprezentacji	
	lista[ms]	macierz [ms]
10	0,97	2,89
25	16,45	56,45
50	59,45	394,62
100	557,62	3191,91
250	18321,6	50090,7



gęstość 75%		
wierzchołki	czas dla różnych reprezentacji	
	lista[ms]	macierz [ms]
10	1,42	3,58
25	21,21	58,21
50	93,21	512,8
100	940,15	4357,4
250	29542,5	65956,8



gęstość 75%		
wierzchołki	czas dla różnych reprezentacji	
	lista[ms]	macierz [ms]
10	1,78	8,71
25	26,63	77,63
50	121,63	607,04
100	1460,56	5136,85
250	51094,4	77206,1



3. Wnioski:

3.1 Problemy:

Problemem okazało się zaimplementowanie macierzy sąsiedztwa z wagami. Macierz jako tablica 2d może przechowywać tylko jedną wagę. Na początku próbowałem rozwiązać ten problem implementując macierz jako tablice 3d lecz po kilku próbach zrezygnowałem z tego pomysłu. Rozwiązaniem okazało się stworzenie klasy vectorClass, która działa podobnie jak kontener vector z biblioteki STL ale może przechowywać tylko inty. Rozwiązanie to pozwoliło mi na przechowywanie wielu krawędzi o różnych wagach z jednego wierzchołka do drugiego, a nie tylko jednej krawędzi z jednego wierzchołka do drugiego.

3.2 Podsumowanie:

Uzyskane wyniki pokazują, że algorytm Bellmana - Forda działa szybciej dla listy niż dla macierzy. Różnica w szybkości jest co raz bardziej zauważalna wraz ze wzrostem wierzchołków zgodnie z przewidywaniami. Reprezentacja macierzowa potrzebuje więcej pamięci niż lista. Na czas może wpływać również budowa listy i macierzy. Przeszukiwanie całej tablicy jest wolniejsze niż dostęp do danych w zaimplementowanej liście. Algorytm Bellmana – Forda dla macierzy musi wykonać więcej pętli podczas działania niż dla listy sąsiedztwa co również może mieć znaczenie.

4. Literatura:

1. Drozdek A. C++ Algorytmy i struktury danych, Helion
2. Forum Stackoverflow
3. Wikipedia
4. Forum GeeksForGeeks