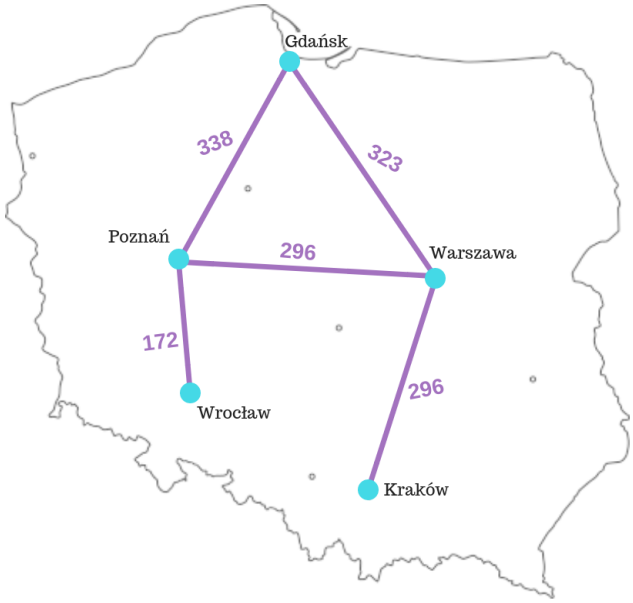
Projekt 2.	
Kierunek, nazwa wydziału: *Automatyka i Robotyka,* *Wydział Elektroniki, Fotoniki i Mikrosystemów*	**Grupa ćwiczeniowa, termin zajęć:** *Y03-51a, Pn 15:15*
Imię, nazwisko, numer albumu: *Jakub Skibiński, 259266*	**Data oddania sprawozdania:** *25.04.2022r.*
Prowadząca: *Mgr inż. Marta Emirsajłow*	**Ocena:**

Sprawozdanie

Spis treści

1 Grafy - wprowadzenie

Graf jest to struktura danych złożona z wierzchołków połączonych krawędziami. Wierzchołki grafu mogą reprezentować różnego rodzaju obiekty, natomiast krawędzie określają relacje pomiędzy wierzchołkami. Krawędzie mogą mieć określony kierunek (graf skierowany) lub posiadać wagę, która może opisywać na przykład długość drogi do danego obiektu. Struktury takie są powszechnie używane w informatyce, matematyce, biologii, itd. W komputerze grafy mogą być reprezentowane za pomocą listy sąsiedztwa lub macierzy sąsiedztwa. Na grafach można wykonywać przeróżne operacje. Do tego służą specjalne algorytmy np. Dijkstry, który odnajduje najoptymalniejszą drogę z zadanego wierzchołka startowego do innych wierzchołków grafu.

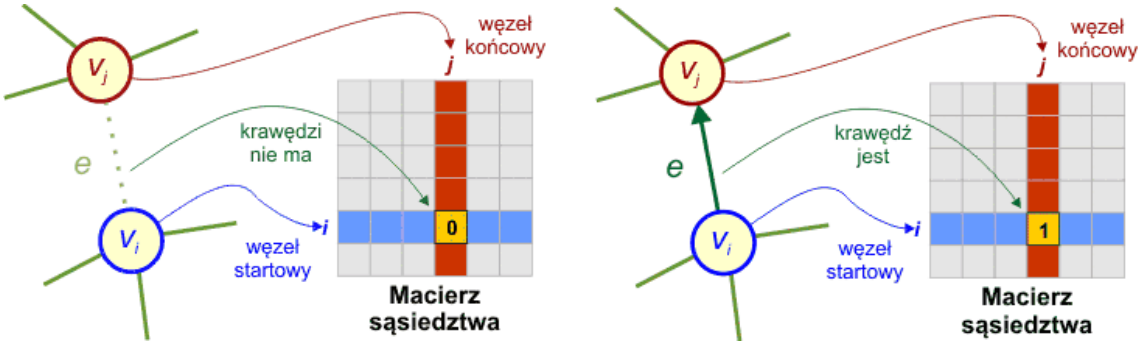


Rysunek 1: Przykładowy graf reprezentujący miasta w Polsce jako wierzchołki i odległości między nimi jako krawędzie z określoną wagą.

2 Metody reprezentacji grafu

2.1 Macierz sąsiedztwa

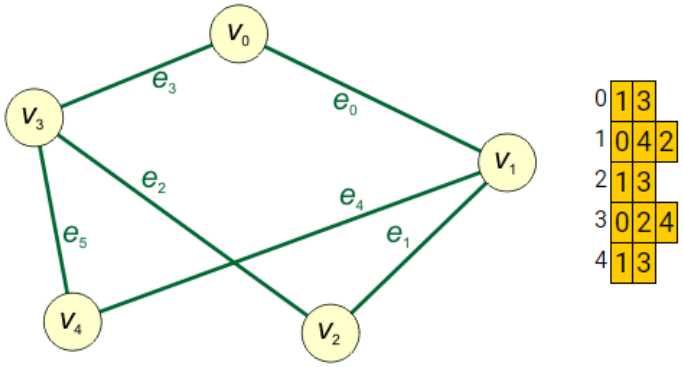
Jednym ze sposobów reprezentacji grafu jest macierz sąsiedztwa (przykładowo nazwana A). Odwzorowuje ona połączenia wierzchołków krawędziami. Wiersze macierzy sąsiedztwa odwzorowują zawsze wierzchołki startowe krawędzi, a kolumny odwzorowują wierzchołki końcowe krawędzi. Jeżeli krawędź pomiędzy wierzchołkami istnieje, to A[i, j] (gdzie i - wiersze, j - kolumny) będzie równa 1. Jeżeli nie istnieje, A[i, j] = 0. Jeżeli krawędź istnieje i ma określoną wagę, w miejsce A[i, j] zamiast 1 można wpisać wartość wagi tej krawędzi.



Rysunek 2: Reprezentacja grafu przy użyciu macierzy sąsiedztwa.

2.2 Lista sąsiedztwa

Kolejnym sposobem reprezentacji grafu może być lista sąsiedztwa przedstawiona za pomocą tablicy n elementowej, w której każdy element jest listą. Numer elementu reprezentuje wierzchołek startowy, natomiast lista w tym elemencie przechowuje numery wierzchołka / wierzchołków końcowych z którym/i początkowy jest połączony krawędzią.



Rysunek 3: Reprezentacja grafu przy użyciu listy sąsiedztwa.

3 Algorytm Dijkstry

3.1 Działanie

Jednym z podstawowych problemów w teorii grafów jest znajdywanie drogi pomiędzy dwoma wybranymi wierzchołkami. Problem ten dobrze rozwiązuje algorytm Dijkstry, który znajduje najoptymalniejszą ścieżkę obliczając przy okazji koszt takiej operacji. Jego działanie polega na utworzeniu dwóch zbiorów Q i S. Zbiór Q reprezentuje wszystkie wierzchołki grafu, natomiast S jest pusty. Algorytm zaczyna działanie od zadnego wierzchołka startowego, gdzie wykonywana jest następująca sekwencja:

Jeżeli Q nie jest pusty:

- Znajdź wierzchołek v o najniższym koszcie dotarcia i przenieś go z Q do S.
- Dla każdej krawędzi k wychodzącej z wierzchołka v:
  - Oznacz wierzchołek znajdujący się na drugim końcu krawędzi k jako u.
  - Jeśli koszt dotarcia do wierzchołka u z wierzchołka v poprzez krawędź k jest mniejszy od aktualnego kosztu dotarcia do wierzchołka u, to:
    - Przypisz kosztowi dotarcia do wierzchołka u koszt dotarcia do wierzchołka v powiększony o wagę krawędzi k.
    - Ustaw wierzchołek v jako poprzednik wierzchołka u.

3.2 Złożoność obliczeniowa

Złożoność obliczeniowa algorytmu zależy od ilości wierzchołków V i krawędzi E. O rzędzie złożoności decyduje sposób implementacji kolejki:

- implementacja poprzez zwykłą tablicę (użyta w tym ćwiczeniu): $O(V^2)$
- implementacja poprzez kopiec: $O(E \cdot log(V))$
- implementacja poprzez kopiec Fibonacciego: $O(E + V \cdot log(V))$

Pierwszy sposób jest efektywny dla gęstszych grafów natomiast drugi dla rzadszych. Trzeci jest praktycznie nieużywany ze względu na mały zysk czasowy w porównaniu do stopnia skomplikowania.

4 Opis i wyniki działania programu

4.1 Założenia

W projekcie zbadano efektywność algorytmu Dijkstry w zależności od metody reprezentacji grafu. Graf zaimplementowano obiektowo reprezentując go za pomocą macierzy sąsiedztwa i listy sąsiedztwa. Algorytm Dijkstry został zaprogramowany przy użyciu zwykłej tablicy, więc jego złożoność wynosi $O(V^2)$. Badania wykonano dla grafów o liczbie wierzchołków: 10, 50, 100, 500, 1000 oraz gęstości: 25%, 50%, 75%, 100% (graf pełny). Dla każdego zestawu parametrów wywołano 100 razy algorytm Dijkstry i zbadano jego efektywność w zależności od tych parametrów i sposobu reprezentacji. Ponadto program wczytuje graf z pliku tekstowego, który również może być w programie wygenerowany. Wyniki badań udokumentowano za pomocą tabel oraz wykresów porównujących czas działania algorytmu z zadanymi parametrami dla macierzy i listy sąsiedztwa.

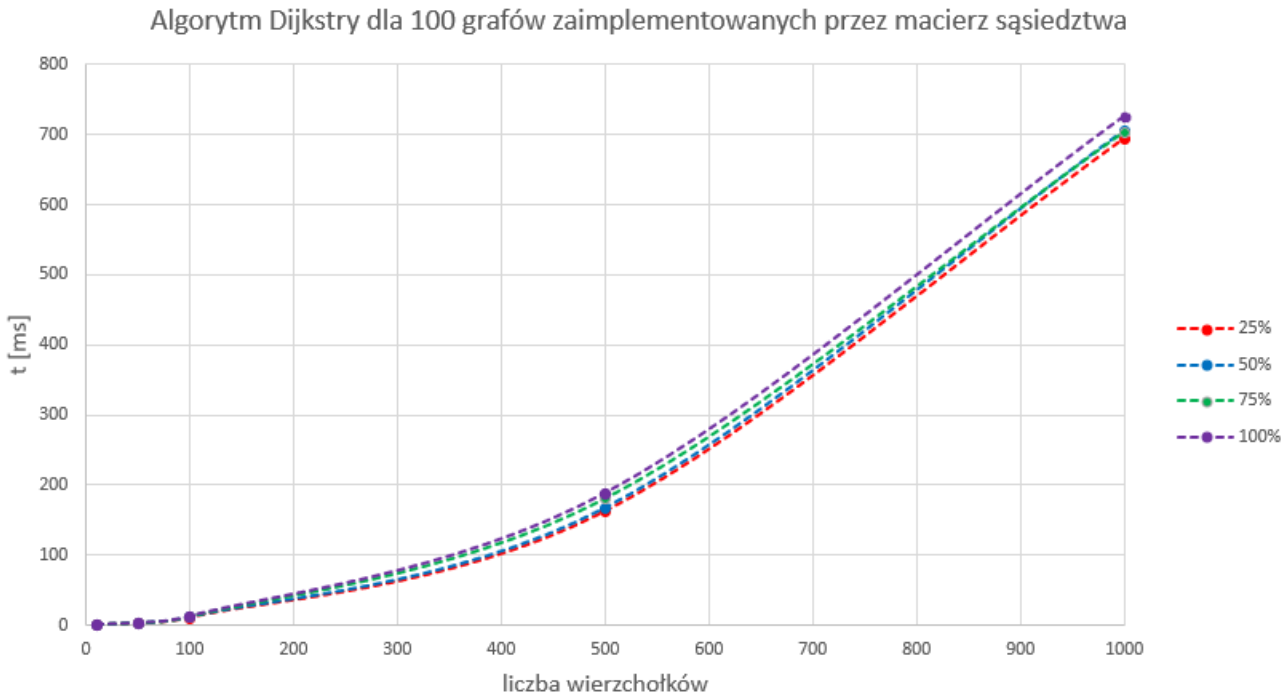
4.2 Tabele i wykresy

W. startowy: 0	Macierz sąsiedztwa [ms]				
l. wierzchołków / gęstość	10	50	100	500	1000
25%	0,1277	2,3419	10,0588	161,811	694,294
50%	0,164	2,9091	10,7303	166,454	705,272
75%	0,195	2,4224	11,0211	179,785	704,009
100%	0,2059	2,9104	12,0053	187,838	726,416

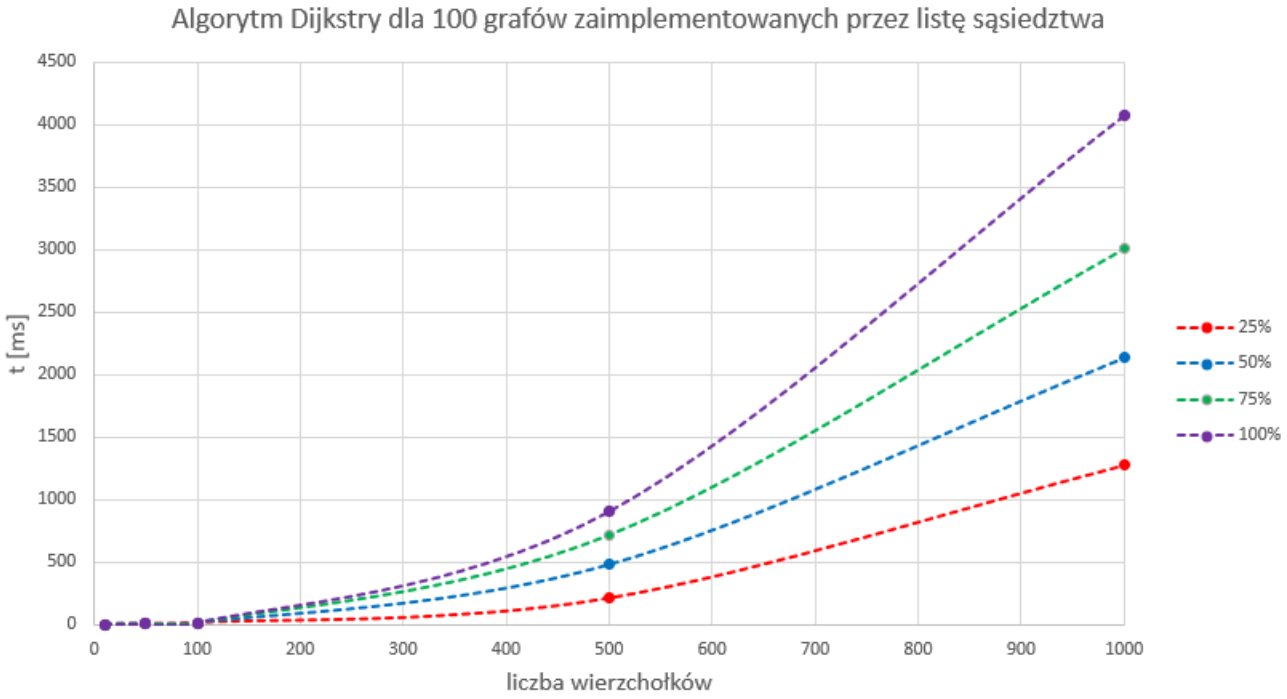
Rysunek 4: Pomiar czasu działania algorytmu Dijkstry dla macierzy sąsiedztwa.

W. startowy: 0	Lista sąsiedztwa [ms]				
l. wierzchołków / gęstość	10	50	100	500	1000
25%	0,1295	2,4883	10,5711	206,105	1271,56
50%	0,1165	2,8979	10,7473	477,089	2137,08
75%	0,1975	2,9627	11,2615	712,977	3005,91
100%	0,1979	3,0969	11,8894	903,085	4071,81

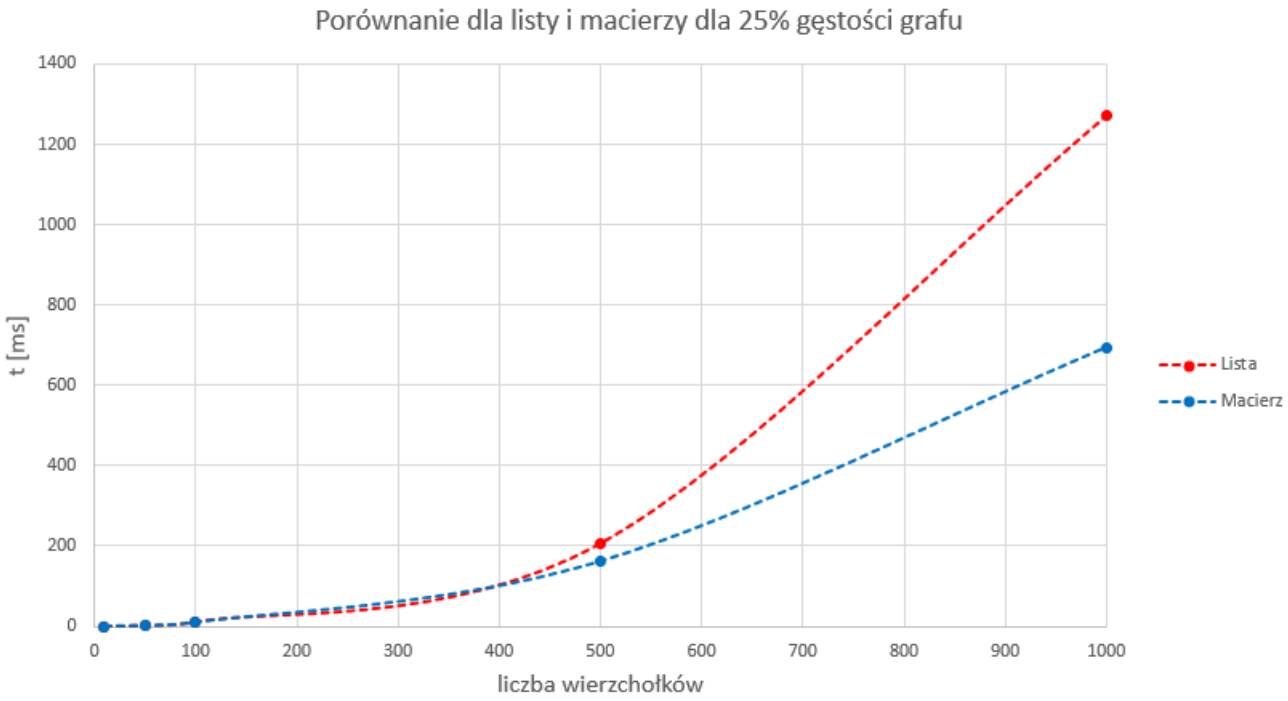
Rysunek 5: Pomiar czasu działania algorytmu Dijkstry dla listy sąsiedztwa.



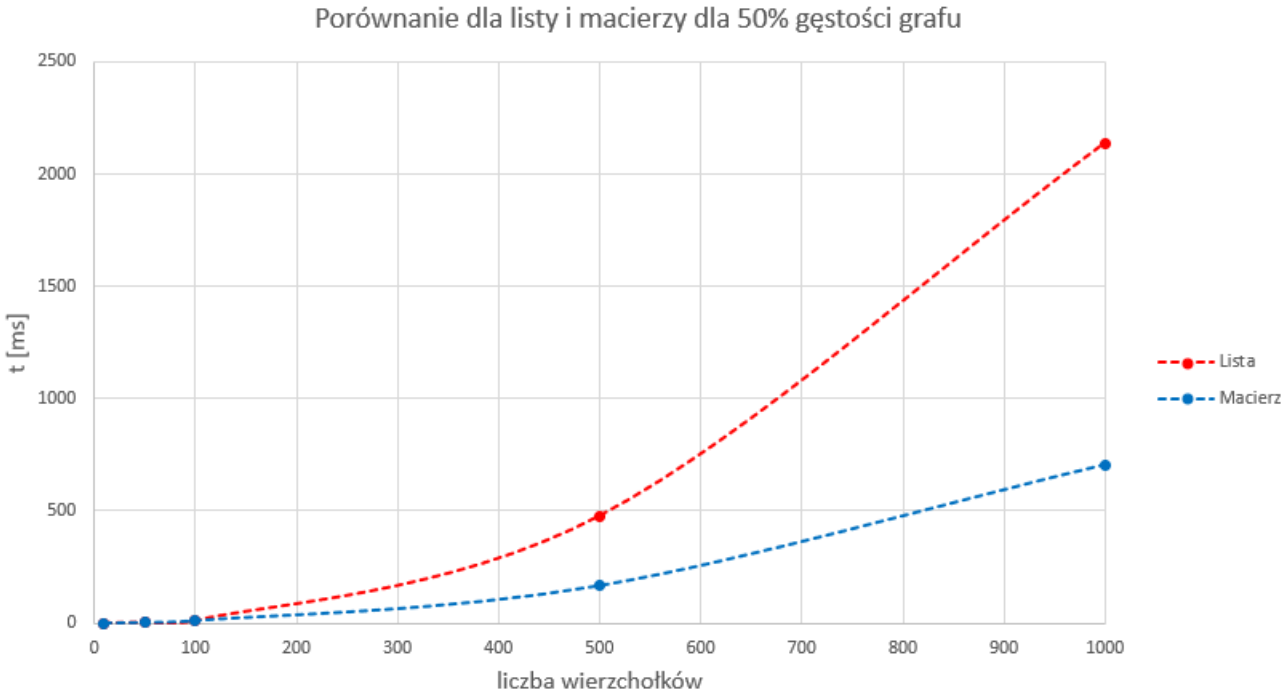
Rysunek 6: Pomiar czasu dla różnych gęstości w zależności od ilości elemenetów dla macierzy sąsiedztwa.



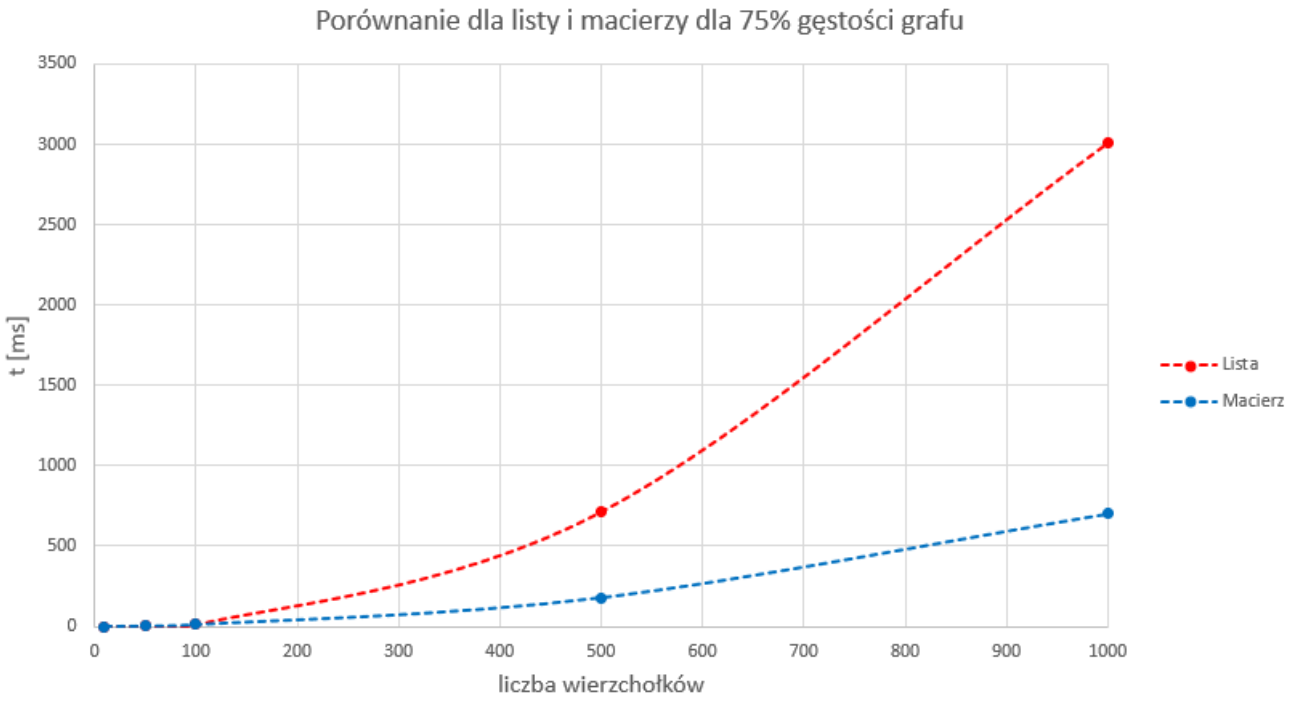
Rysunek 7: Pomiar czasu dla różnych gęstości w zależności od ilości elemenetów dla listy sąsiedztwa.



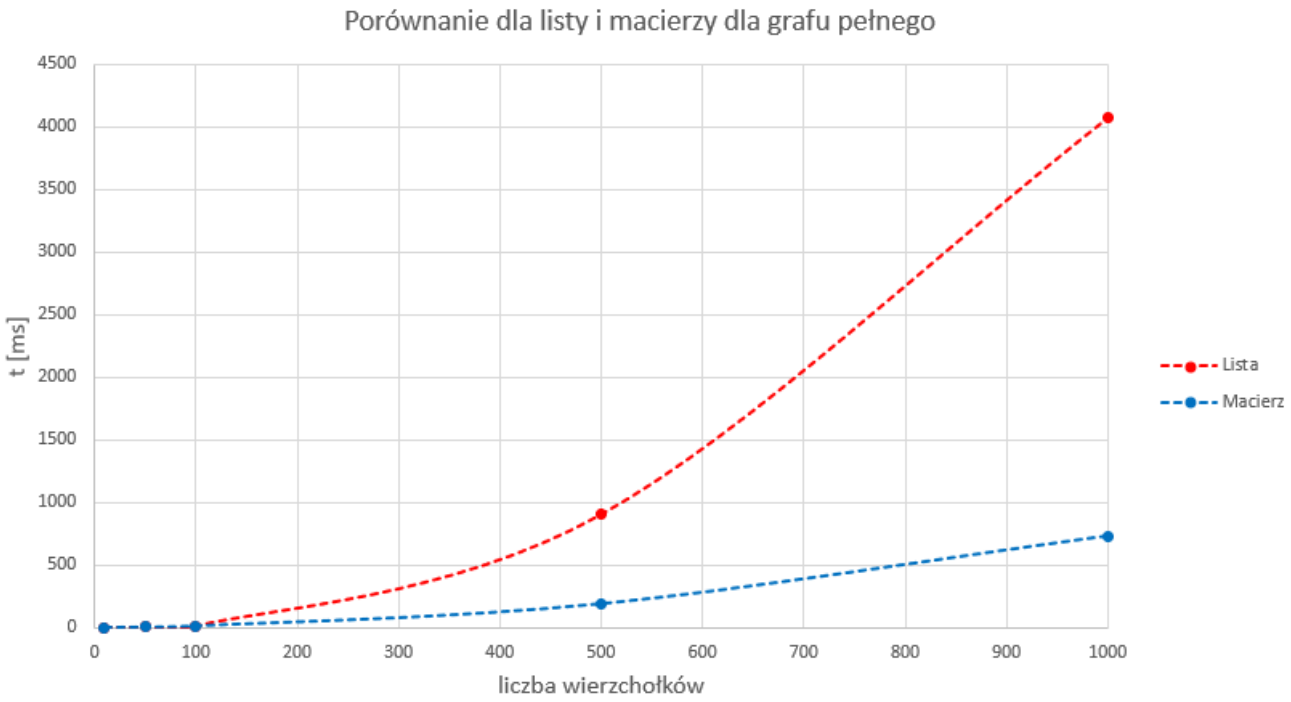
Rysunek 8: Porównanie czasu działania dla macierzy i listy sąsiedztwa przy 25% gęstości grafu..



Rysunek 9: Porównanie czasu działania dla macierzy i listy sąsiedztwa przy 50% gęstości grafu..



Rysunek 10: Porównanie czasu działania dla macierzy i listy sąsiedztwa przy 75% gęstości grafu.



Rysunek 11: Porównanie czasu działania dla macierzy i listy sąsiedztwa przy grafie pełnym

5 Wnioski

- Na podstawie wykresów można zauważyć, że zaimplementowany algorytm Dijkstry ma złożoność kwadratową.
- Dla macierzy sąsiedztwa czas działania algorytmu jest praktycznie taki sam dla różnych gęstości, natomiast dla listy sąsiedztwa widać wyryźne rozbieżności, które są tym większe im więcej jest wierzchołków w grafie.
- Na podstawie porównań wykresów można stwierdzić, że zaimplementowany algorym działa dużo efektywniej dla macierzy sąsiedztwa.
- Wraz ze wzrostem liczby wierzchołków efektywność algorytmu dla macierzy w porównaniu z listą sąsiedztwa jest tym większa im gęstszy jest graf
- Dla grafów o małej ilości wierzchołków (w tym przypadku < 100), algorytm Dijkstry działa porównywalnie efektywnie zarówno dla macierzy i listy sąsiedztwa niezależnie od gęstości. Dla większych grafów efektywność algorytmu dla listy sąsiedztwa spada i jest coraz badziej zależna od gęstości. Może wynikać to z tego, że dla listy sąsiedztwa musimy wykonać więcej operacji aby dostać się do danego elementu grafu.

6 Bibliografia

- `https://pl.wikipedia.org/wiki/Graf_(matematyka)`
- `https://eduinf.waw.pl/inf/alg/001_search/0124.php`
- `http://algorytmy.ency.pl/artykul/algorytm_dijkstry`
- `https://pl.wikipedia.org/wiki/Algorytm_Dijkstry`

]