Politechnika Wrocławska

Mikroprojekt 3
Projektowanie i analiza algorytmów

Wydział Informatyki i Telekomunikacji
Informatyczne Systemy Automatyki

Aleksander Żołnowski, 272536

Spis treści

1 Wstęp

Zadanie polegało na zaimplementowaniu algorytmu Dijkstry i przebadaniu jego efektywności w zależności od implementacji grafu. W projekcie zaimplementowano reprezentacje grafu za pomocą listy sąsiedztwa oraz macierzy sąsiedztwa. Algorytm Dijkstry jest powszechnie stosowany do znajdowania najkrótszych ścieżek w grafach ważonych bez ujemnych wag.

Należało zaimplementować rozwiązania dla przypadków:

- Znalezienie najkrótszej ścieżki od wybranego wierzchołka do wszystkich pozostałych wierzchołków
- Znalezienie najkrótszej ścieżki pomiędzy dwoma wybranymi wierzchołkami

2 Implementacja

2.1 Lista sąsiedztwa

Lista sąsiedztwa (ang. Adjacency List) jest to jeden ze sposobów reprezentacji grafu, gdzie każdy wierzchołek ma przypisaną listę sąsiadujących z nim wierzchołków oraz wagę krawędzi łączącej te wierzchołki. Do zaimplementowania algorytmu Dijkstry użyto kolejki priorytetowej w postaci kopca. W tej wersji algorytmu, lista sąsiedztwa jest wykorzystywana do iteracji przez sąsiadujące wierzchołki.

2.2 Macierz sąsiedztwa

Macierz sąsiedztwa (ang. Adjacency Matrix) to kolejny ze sposobów reprezentacji grafu. W tej implementacji graf jest przechowywany w wektorze wektorów jako macierz V x V, gdzie V to liczba wierzchołków. Elementy [i] [j] macierzy zawierają wagę macierzy łączącą wierzchołki, a wartość 0 oznacza brak krawędzi. Algorytm Dijkstry wykorzystuje macierz sąsiedztwa do sprawdzania wag krawędzi między wierzchołkami. Do zaimplementowania tego algorytmu użyto kolejki priorytetowej na kopcu.

3 Badania

W celu wyznaczenia wydajności algorytmu Dijkstry dla różnych reprezentacji grafu (lista sąsiedztwa i macierz sąsiedztwa) oraz różnych gęstości, badania przeprowadzono dla grafów o pięciu różnych liczbach wierzchołków tj. 10, 50, 100, 500 i 1000. Dla każdej liczby wierzchołków analizowano cztery gęstości grafu: 25%, 50%, 75% oraz pełne grafy (100%). Dla każdego z tych zestawów generowano 100 losowych instancji grafów i mierzono średni czas wykonania algorytmu.

Gęstość grafu wyznaczono za pomocą wzoru $d = \frac{2m}{n(n-1)}$, gdzie m – liczba krawędzi, n liczba wierzchołków grafu.

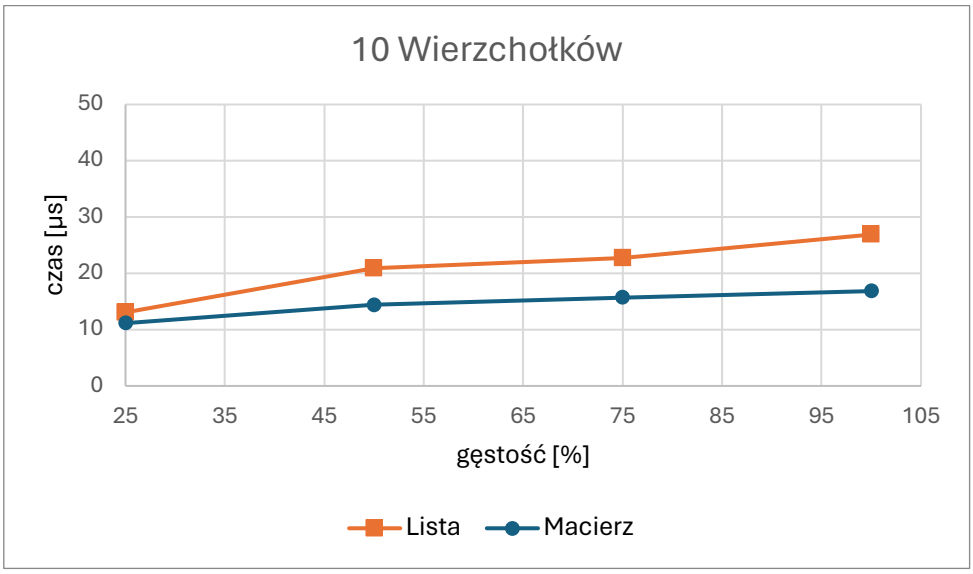
3.1 Wierzchołki

Badania pokazujące wpływ gęstości grafu na algorytm Dijkstry dla stałych wielkości grafów.

3.1.1 10 Wierzchołków

gęstość [%]	Lista [µs]	Macierz [µs]
25	13,07	11,15
50	20,86	14,38
75	22,68	15,71
100	26,9	16,85

Tabela 1 Wyniki dla grafu o 10 wierzchołkach

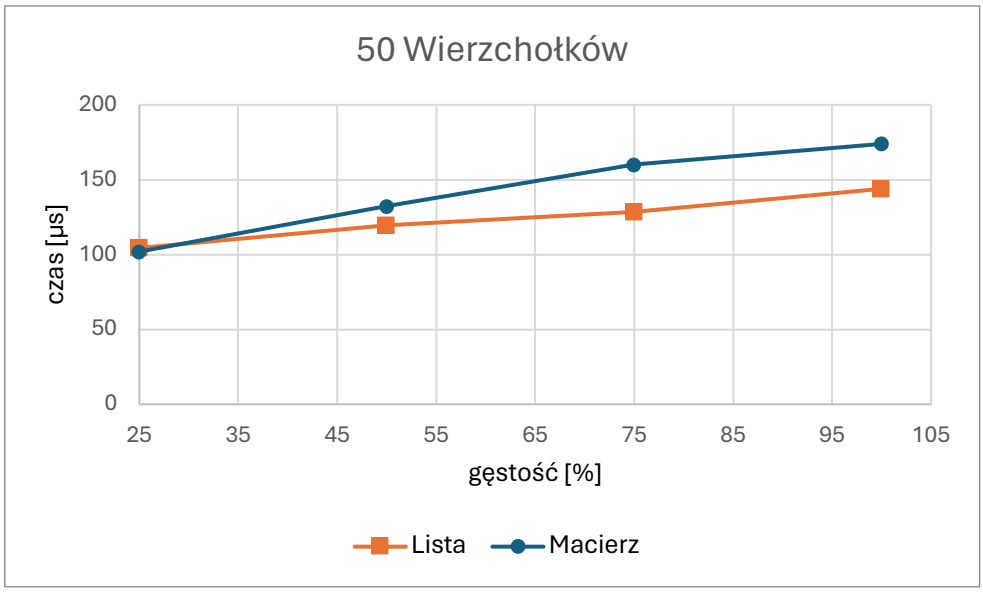


Rysunek 1 Charakterystyka algorytmu Dijkstry dla grafu o 10 wierzchołkach

3.1.2 50 Wierzchołków

gęstość [%]	Lista [µs]	Macierz [µs]
25	104,45	101,94
50	119,6	132,22
75	128,5	160,01
100	143,93	173,92

Tabela 2 Wyniki dla grafu o 50 wierzchołkach

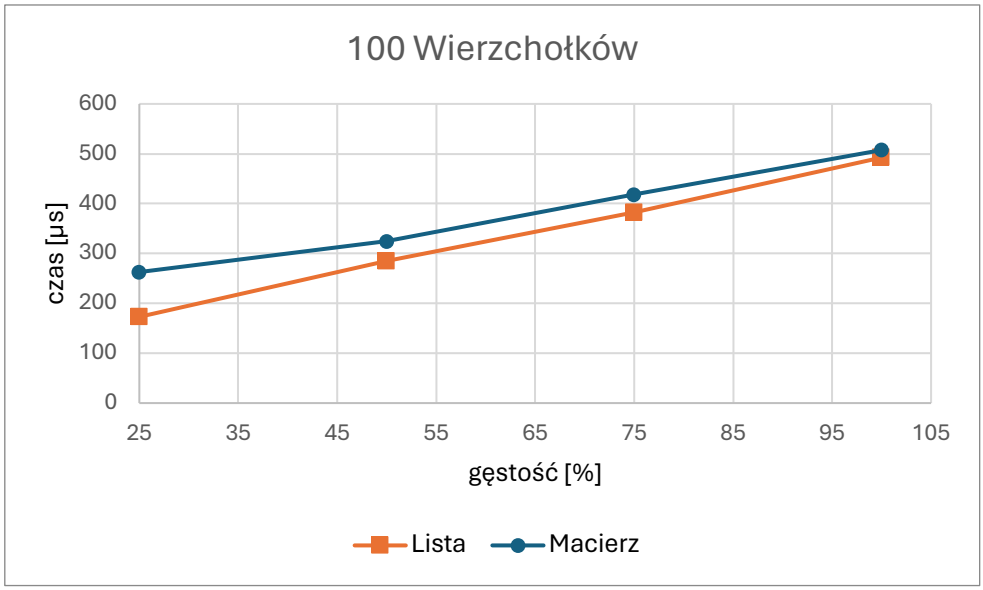


Rysunek 2 Charakterystyka algorytmu Dijkstry dla grafu o 50 wierzchołkach

3.1.3 100 Wierzchołków

gęstość [%]	Lista [µs]	Macierz [µs]
25	172,95	262,75
50	285,01	324,66
75	381,93	418,33
100	492,41	507,43

Tabela 3 Wyniki dla grafu o 100 wierzchołkach

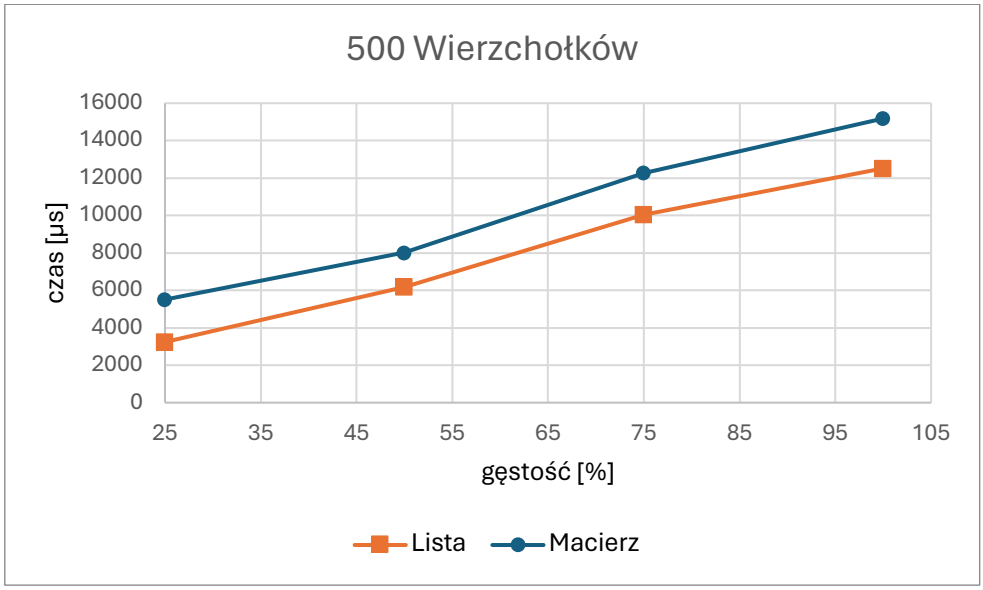


Rysunek 3 Charakterystyka algorytmu Dijkstry dla grafu o 100 wierzchołkach

3.1.4 500 Wierzchołków

gęstość [%]	Lista [µs]	Macierz [µs]
25	3241,6	5509,44
50	6171,76	8023,82
75	10738,1	13277,1
100	12519,1	15183,1

Tabela 4 Wyniki dla grafu o 500 wierzchołkach

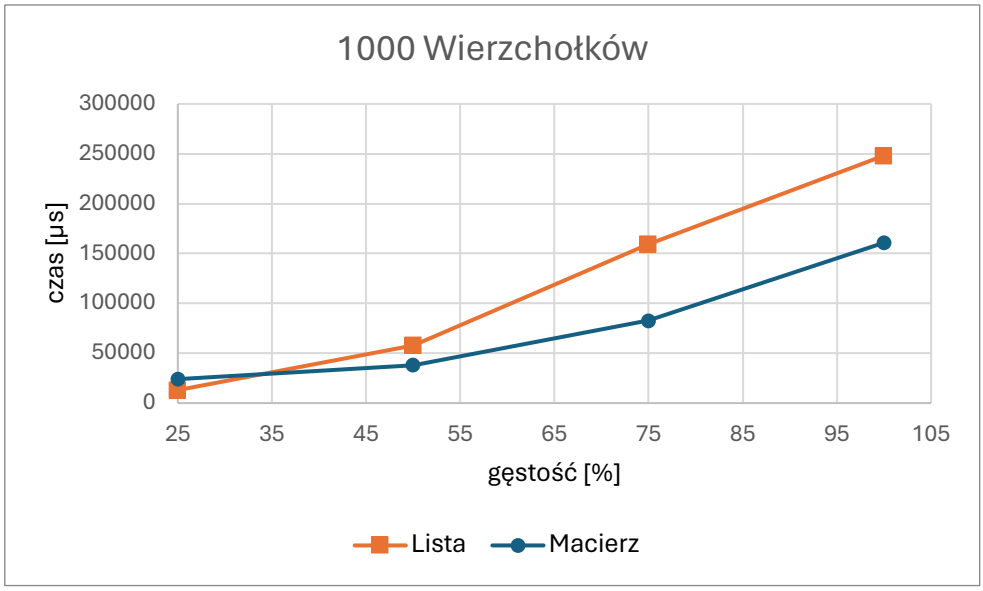


Rysunek 4 Charakterystyka algorytmu Dijkstry dla grafu o 500 wierzchołkach

3.1.5 1000 Wierzchołków

gęstość [%]	Lista [µs]	Macierz [µs]
25	13023,6	24007,6
50	57603,1	38138,3
75	159030	82663,1
100	248153	160793

Tabela 5 Wyniki dla grafu o 1000 wierzchołkach



Rysunek 5 Charakterystyka algorytmu Dijkstry dla grafu o 1000 wierzchołkach

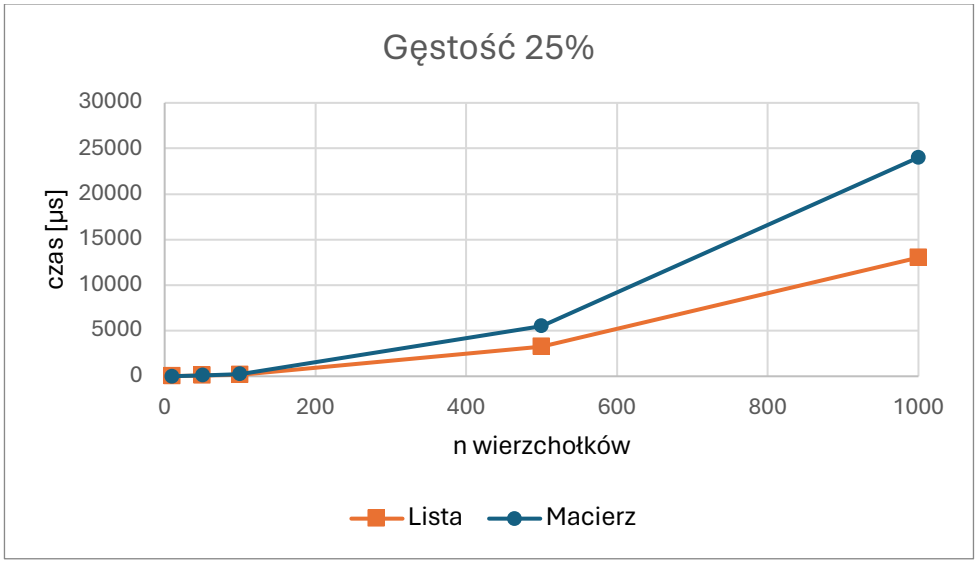
3.2 Gęstość

Badania pokazujące wpływ liczby wierzchołków na wydajność algorytmu Dijkstry dla stałych gęstości.

3.2.1 Gęstość 25%

n	Lista [µs]	Macierz [µs]
10	13,07	11,15
50	104,45	101,94
100	172,95	262,75
500	3241,6	5509,44
1000	13023,6	24007,6

Tabela 6 Wyniki dla grafów o gęstości 25%

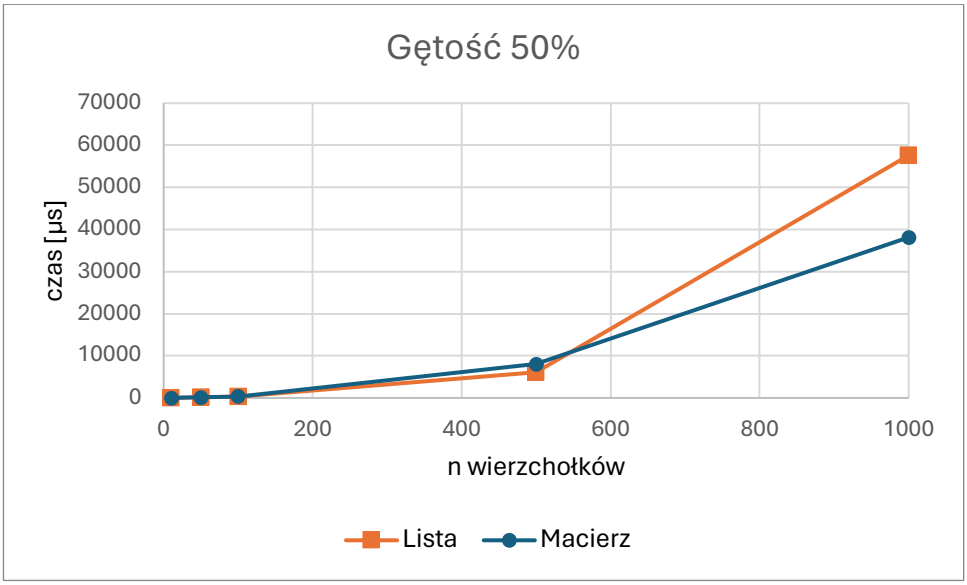


Rysunek 6 Charakterystyka algorytmu Dijkstry dla grafów o gęstości 25%

3.2.2 Gęstość 50%

n	Lista [µs]	Macierz [µs]
10	20,86	14,38
50	119,6	132,22
100	285,01	324,66
500	6171,76	8023,82
1000	57603,1	38138,3

Tabela 7 Wyniki dla grafów o gęstości 50%

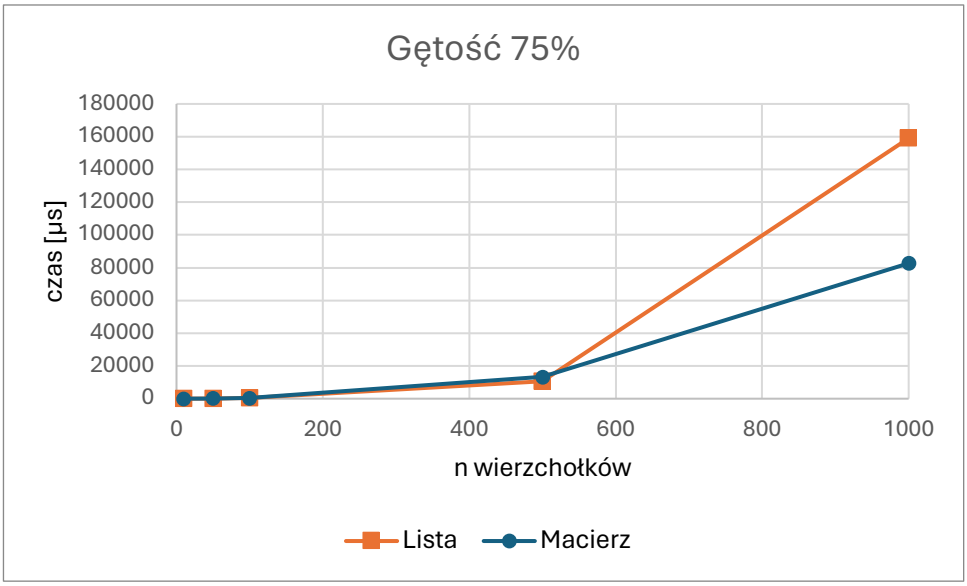


Rysunek 7 Charakterystyka algorytmu Dijkstry dla grafów o gęstości 50%

3.2.3 Gęstość 75%

n	Lista [µs]	Macierz [µs]
10	22,68	15,71
50	128,5	160,01
100	381,93	418,33
500	10738,1	13277,1
1000	159030	82663,1

Tabela 8 Wyniki dla grafów o gęstości 75%

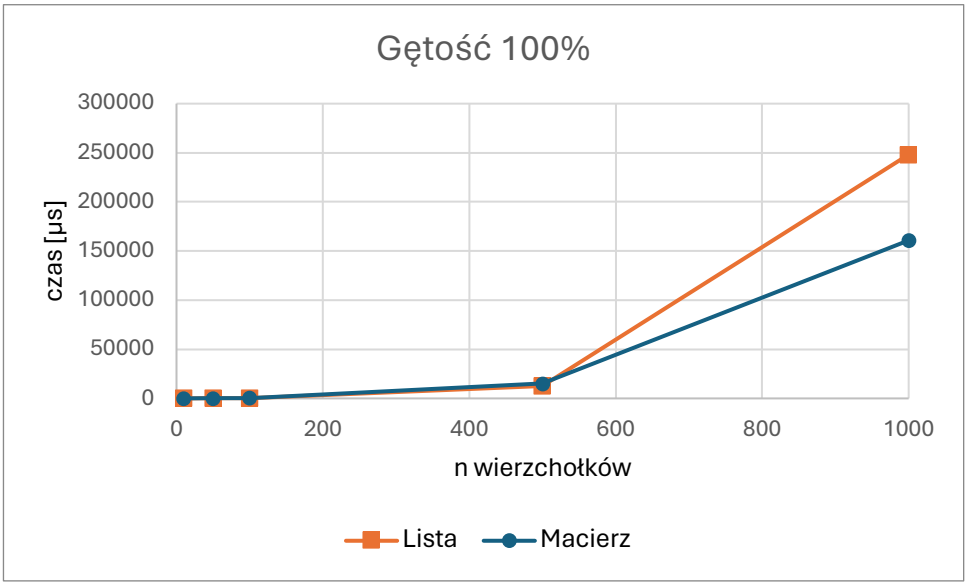


Rysunek 8 Charakterystyka algorytmu Dijkstry dla grafów o gęstości 75%

3.2.4 Gęstość 100%

n	Lista [µs]	Macierz [µs]
10	26,9	16,85
50	143,93	173,92
100	492,41	507,43
500	12519,1	15183,1
1000	248153	160793

Tabela 9 Wyniki dla grafów o gęstości 100%



Rysunek 9 Charakterystyka algorytmu Dijkstry dla grafów o gęstości 100%

4 Wnioski

- Złożoność obliczeniowa algorytmu Dijkstry wynosi O(E*logV), gdzie E to liczba krawędzi a V liczba wierzchołków.
- Porównując wyniki dla różnych reprezentacji grafu i różnych gęstości, można zauważyć, że lista sąsiedztwa często osiąga lepsze czasy wykonania dla mniejszych grafów o mniejszej gęstości, podczas gdy dla większych grafów macierz sąsiedztwa może być bardziej wydajna.
- Wydajność algorytmu Dijkstry zależy nie tylko od reprezentacji grafu, ale także od jego rozmiaru i gęstości.
- Implementacja algorytmu Dijkstry wymagała użycia kolejki priorytetowej na kopcu. Sama implementacja nie jest skomplikowana.
- Implementacja reprezentacji grafu jest mniej skomplikowana dla macierzy sąsiedztwa, która wykorzystuje wektory, niż listy sąsiedztwa bazującej na listach wiązanych.

5 Bibliografia

- Data Structures and Algorithms in C++, 2nd Edition Michael T. Goodrich, Roberto Tamassia, David M. Mount
- Cormen T., Leiserson C.E., Rivest R.L., Stein C., Wprowadzenie do algorytmów, WNT