STRUKTURY DANYCH I ZŁOŻONOŚĆ OBLICZENIOWA – PROJEKT

Temat: Badanie efektywności algorytmów grafowych w zależności od rozmiaru instancji oraz sposobu reprezentacji grafu w pamięci komputera.

Autor: Michał Tomaszewicz, 235568

Prowadzący projekt: Dr inż. Dariusz Banasiak

1.Wstęp

Celem projektu było zaimplementowanie algorytmów dla dwóch podanych problemów – znalezienie minimalnego drzewa spinającego (MST) oraz najkrótszej ścieżki w grafie. Do problemu MST wykorzystałem algorytm Prima, a do znajdowania najkrótszej ścieżki algorytmu Dijkstry. Każdy algorytm działa dla obydwu reprezentacji grafu w pamięci komputera – listy sąsiedztwa (następników) oraz macierzy incydencji.

Teoretyczne złożoności obliczeniowe zaimplementowanych algorytmów: (|V| - ilość wierzchołków, |E| - ilość krawędzi)

Algorytm Prima:

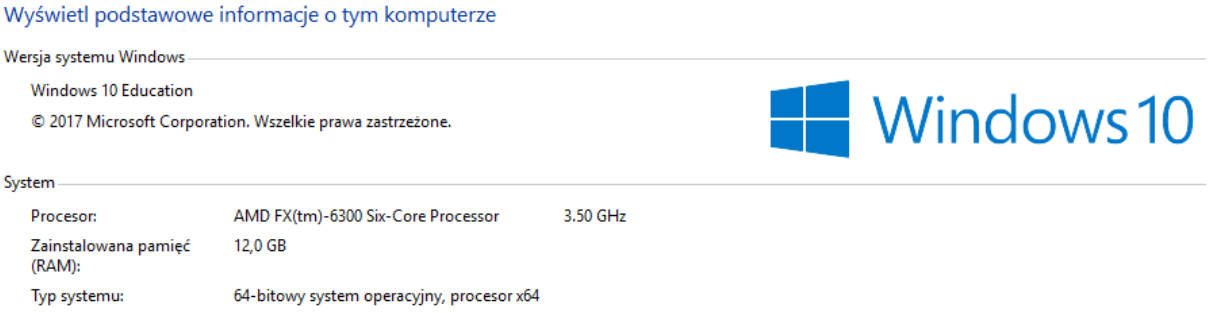
- dla macierzy incydencji $O(|V|^2)$
- dla listy sąsiadów $O(|E|*\log|V|)$

Algorytm Dijkstry:

- dla macierzy incydencji $O(|V|^2)$
- dla listy sąsiadów $O(|E|*\log|V|)$

2. Plan eksperymentu

Efektywność oceniana byłą na podstawie pomiaów czasu w jakim algorytm zwróci wynik. Pomiary wykonywane były w zależności od rozmiaru grafu oraz jego gęstości. Wybrałem pięć reprezentatywnych ilości wierzchołków: 20 40, 60 80 i 100 i dla każdego z nich wykonałem pomiary dla następujących gęstości: 25%, 50% 75% oraz 99%. Grafy generowane były za pomocą funkcji rand() – losowo wypełniałem macierz incydencji, tworząc krawędź, tak aby powstał graf spójny z silną składową spójności. Spójność zapewnił mi algorytm przeszukiwania grafu wszerz, który sprawdzał, czy z każdego wierzchołka można dostać się do pozostałych wierzchołków. Grafy były generowane aż do momentu spełnienia warunku spójności. W grafach nie było pętli, ale dopuszczalne są wielkokrotne krawędzie. Pomiarów dokonywałem za pomocą zaproponowanej przez Prowadzącego funkcji QueryPerformanceCounter. Czas generowania grafu nie był uwzględniany w pomiarze. Do przeprowadzenia eksperymentu używałem komputera PC z procesorem AMD FX – 6300 3,50 GHz i 12 GB pamięci RAM.



3.Tabele z wynikami pomiarów – czasu uśrednione

Algorytm Prima – lista sąsiedztwa (czas[ms])

wierzchołki	gęstość			
	25%	50%	75%	99%
20	0,07	0,10	0,10	0,13
40	0,30	0,43	0,49	0,55
60	0,74	1,04	1,20	1,44
80	1,45	1,99	2,54	3,04
100	2,50	3,45	4,38	5,48

Algorytm prima – macierz incydencji (czas[ms])

wierzchołki	gęstość			
	25%	50%	75%	99%
20	0,28	0,64	0,95	1,25
40	2,97	6,31	9,64	12,69
60	12,52	29,84	45,98	60,85
80	41,17	86,51	131,76	174,36
100	95,18	196,45	301,57	403,73

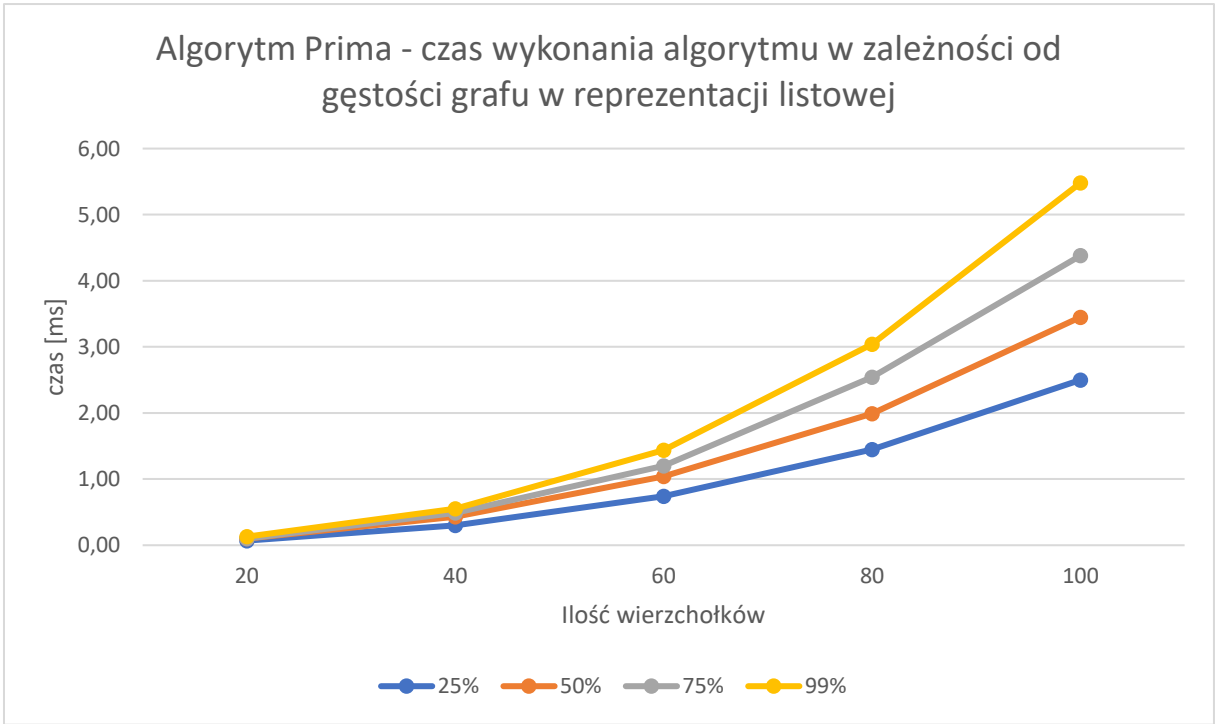
Algorytm Dijkstry – lista sąsiedztwa (czas[µs])

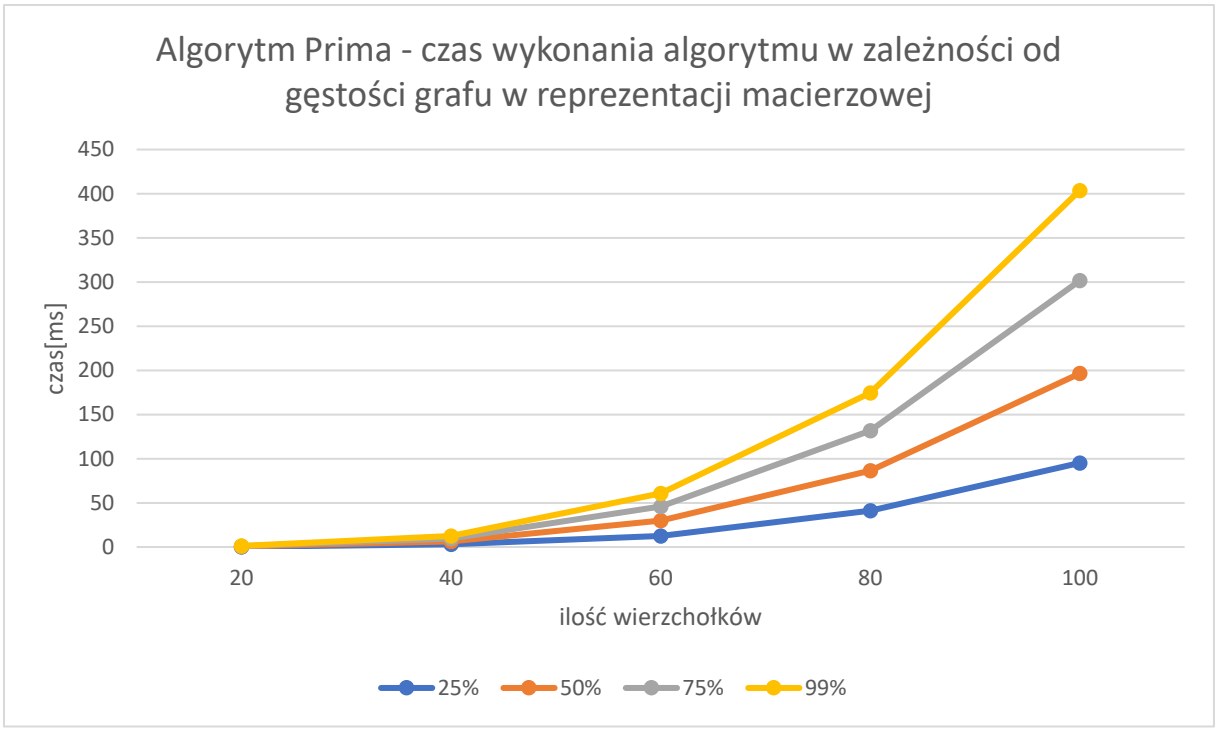
wierzchołki	gęstość			
	25%	50%	75%	99%
20	4,84	6,71	8,15	11,28
40	12,81	17,02	20,68	24,97
60	22,6	32,71	38,76	50,9
80	35,85	52,53	66,35	76,24
100	54,18	76,85	97,26	116,01

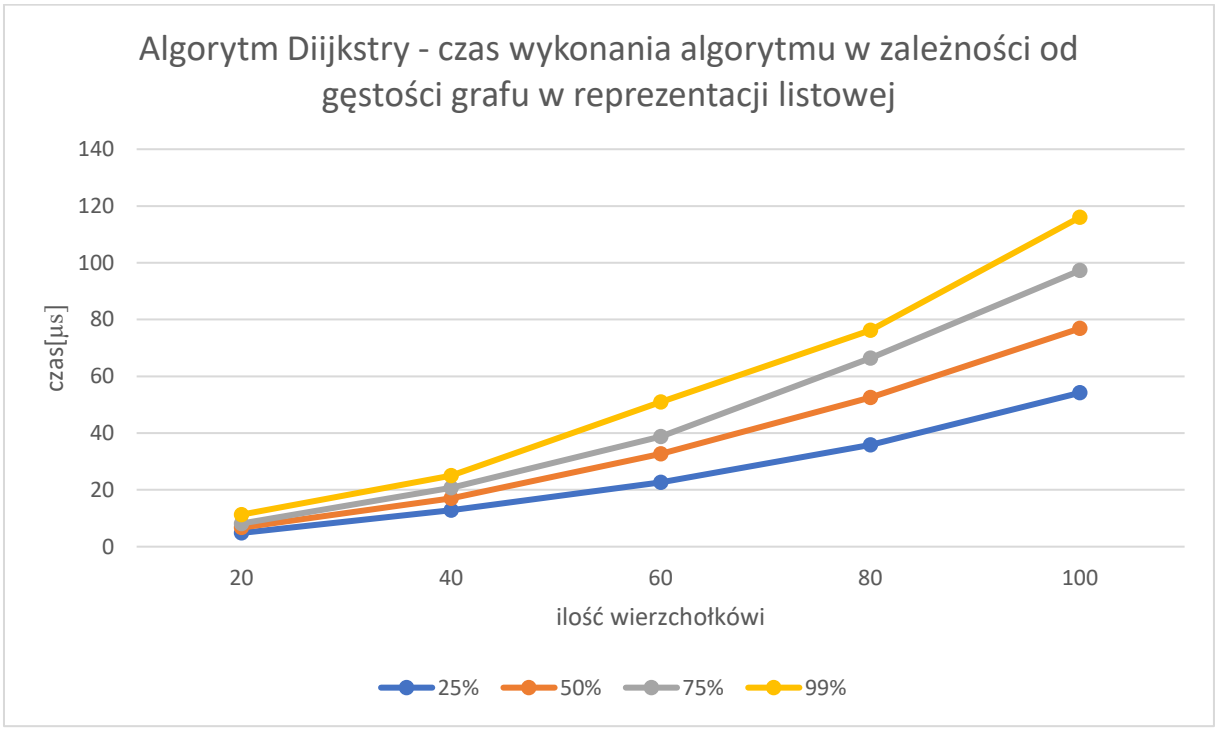
Algorytm Dijkstry – macierz incydencji (czas[µs])

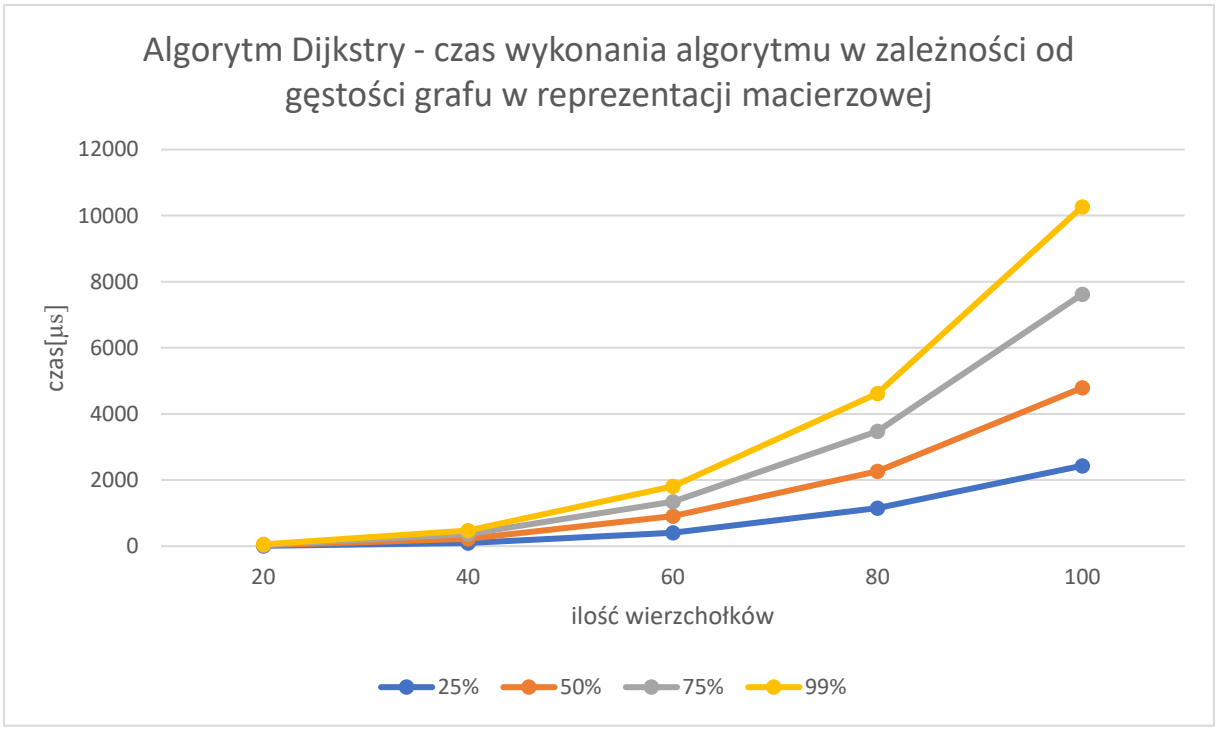
wierzchołki	gęstość			
	25%	50%	75%	99%
20	12,14	27,19	43,26	55,57
40	101,44	224,16	364,05	479,46
60	407,73	913,51	1345,28	1815,62
80	1149,54	2269,47	3475,23	4621,7
100	2435,24	4791,21	7624,79	10268,32

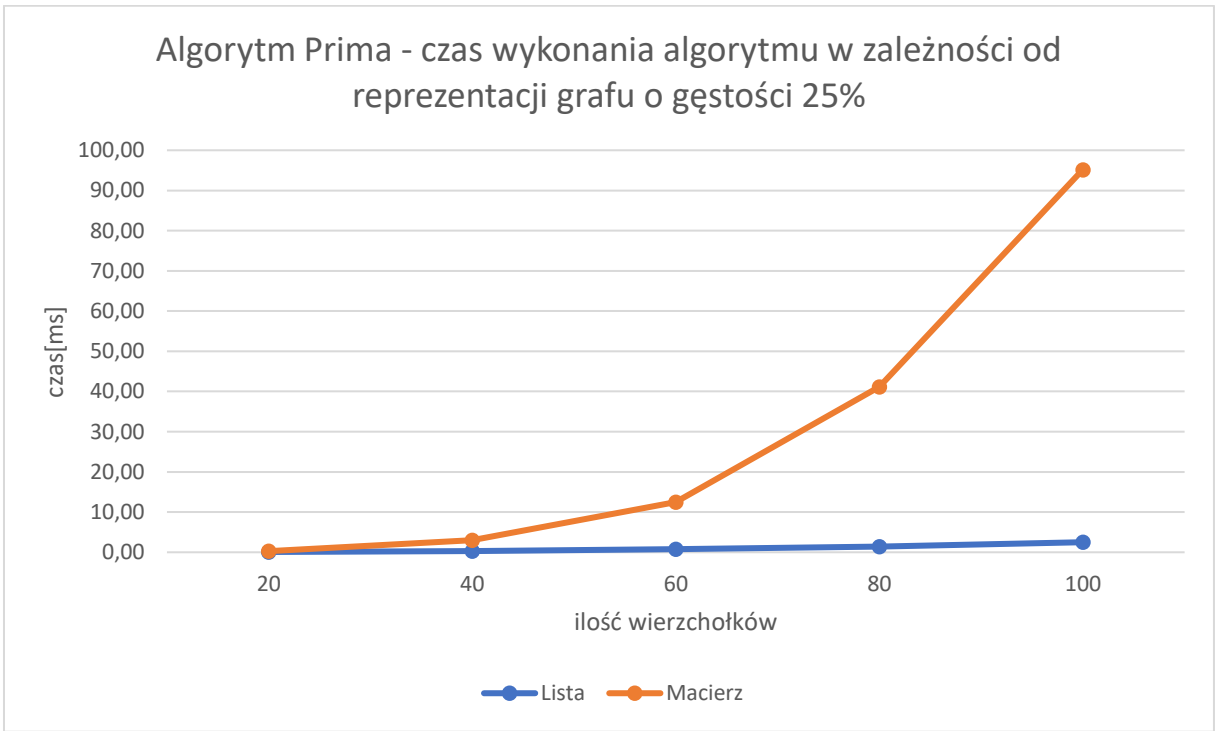
4. Wykresy

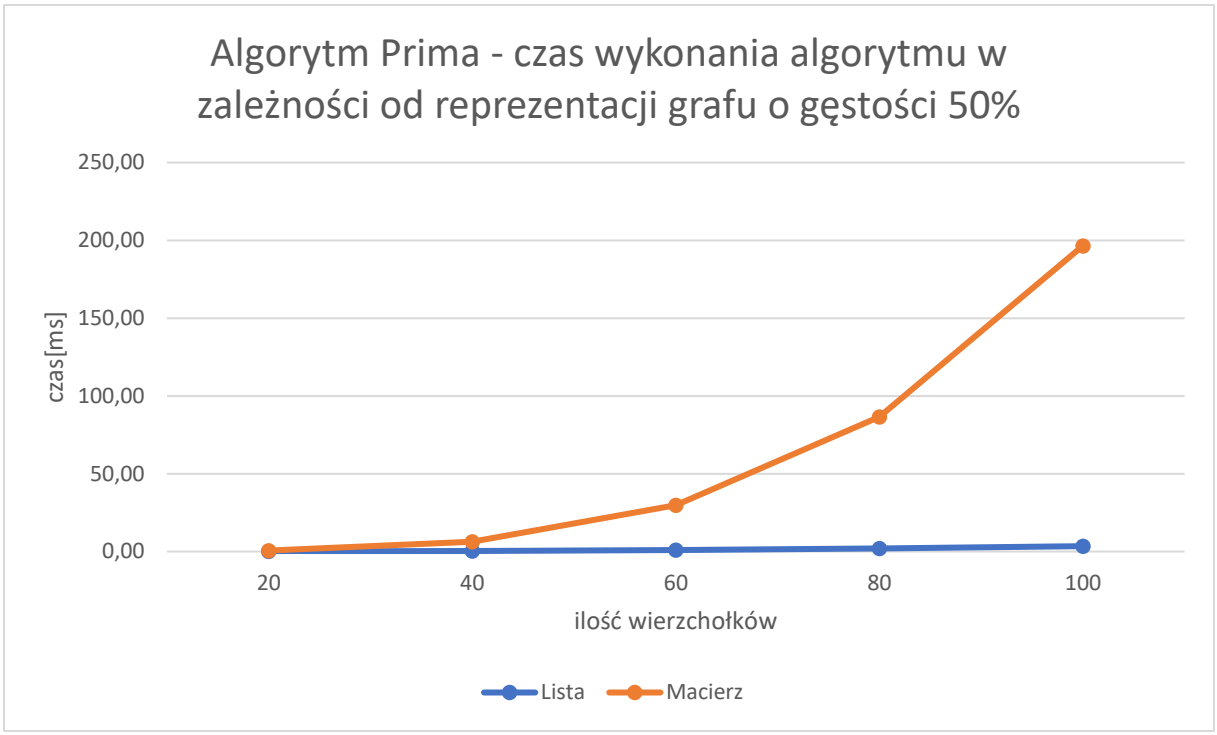


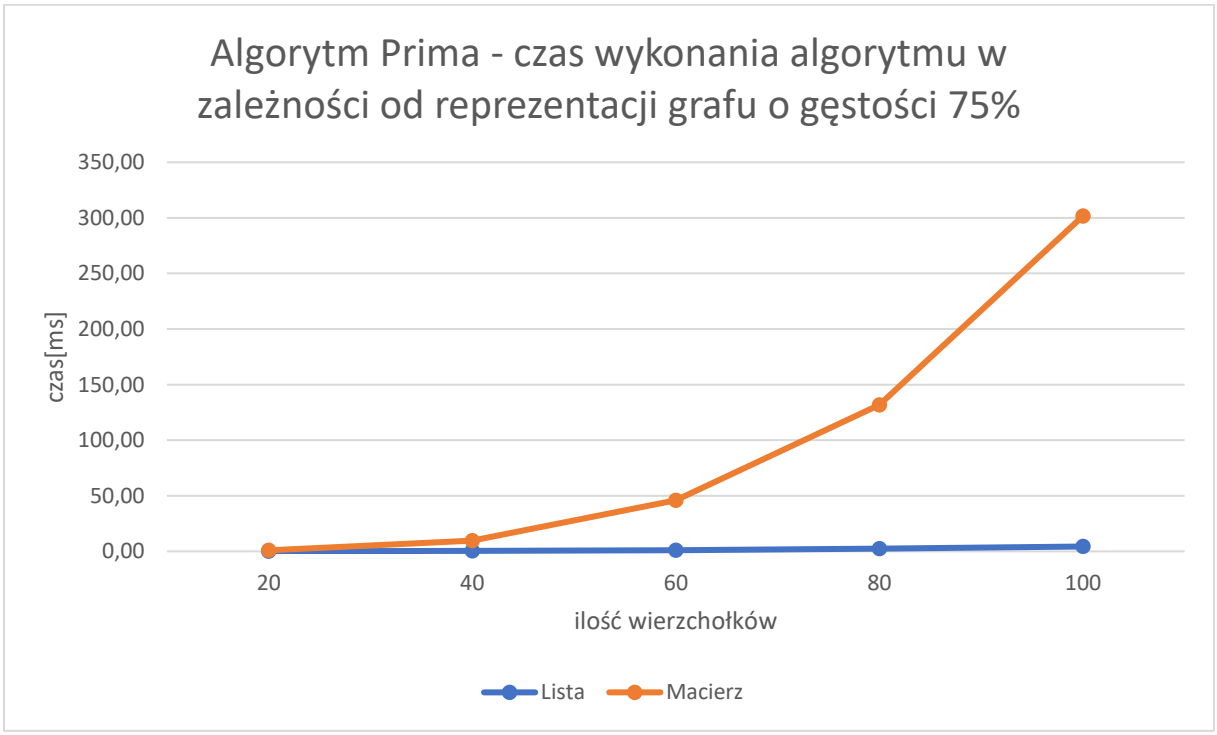

Algorytm Prima - czas wykonania algorytmu w zależności od gęstości grafu w reprezentacji macierzowej
czas[ms]
450
400
350
300
250
200
150
100
50
0
20
40
60
80
100
ilość wierzchołków
25%
50%
75%
99%

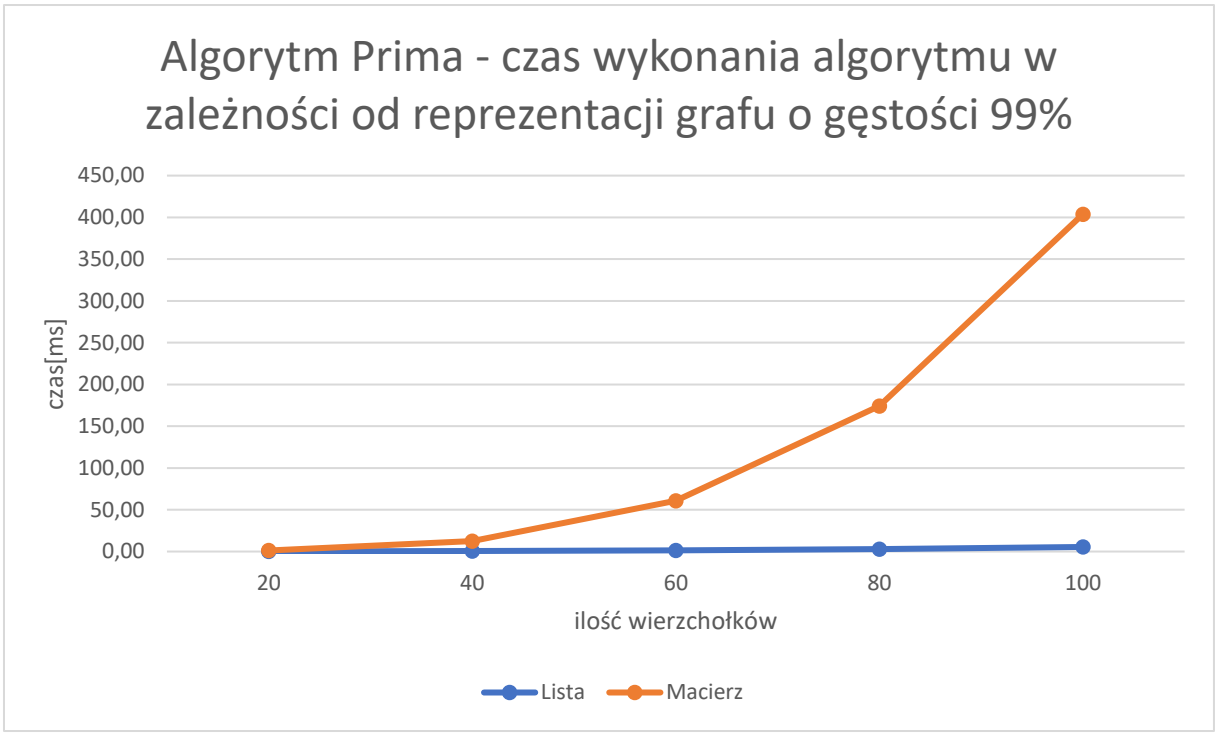

Algorytm Diijkstry - czas wykonania algorytmu w zależności od gęstości grafu w reprezentacji listowej
czas[µs]
140
120
100
80
60
40
20
0
20
40
60
80
100
ilość wierzchołkówi
25%
50%
75%
99%

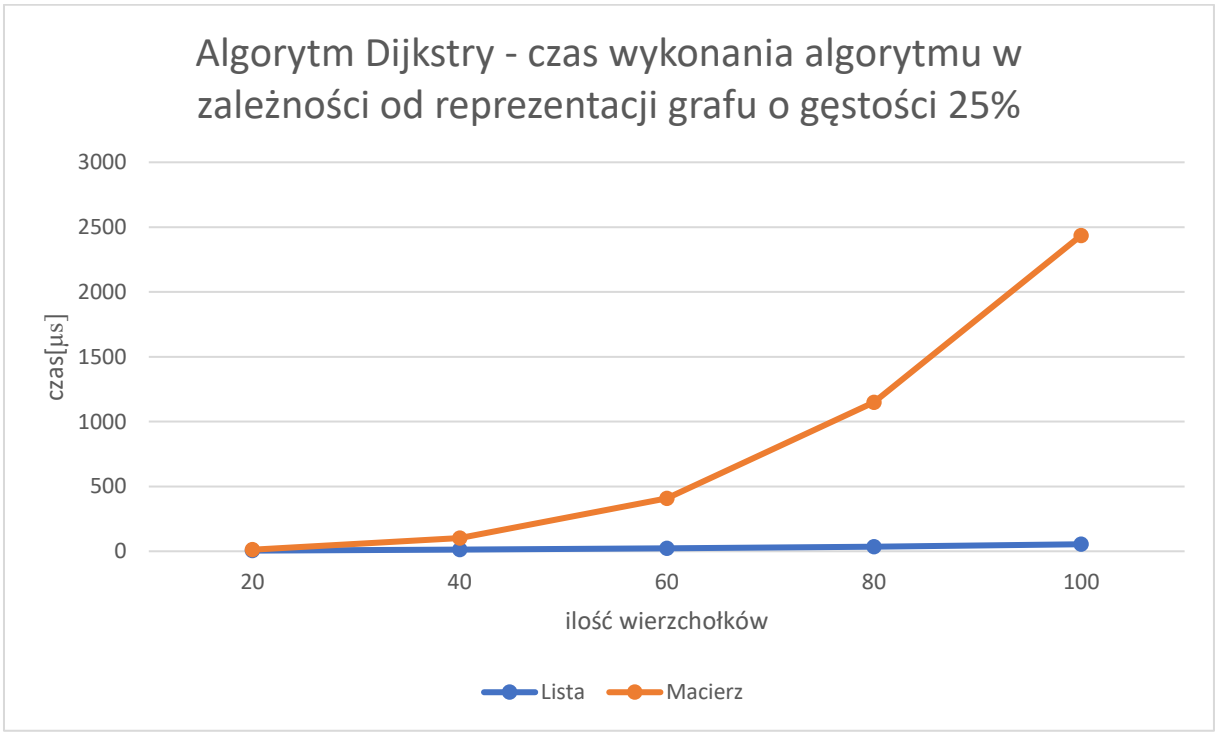

Algorytm Dijkstry - czas wykonania algorytmu w zależności od gęstości grafu w reprezentacji macierzowej
12000
10000
8000
6000
4000
2000
0
czas[µs]
20
40
60
80
100
ilość wierzchołków
25%
50%
75%
99%

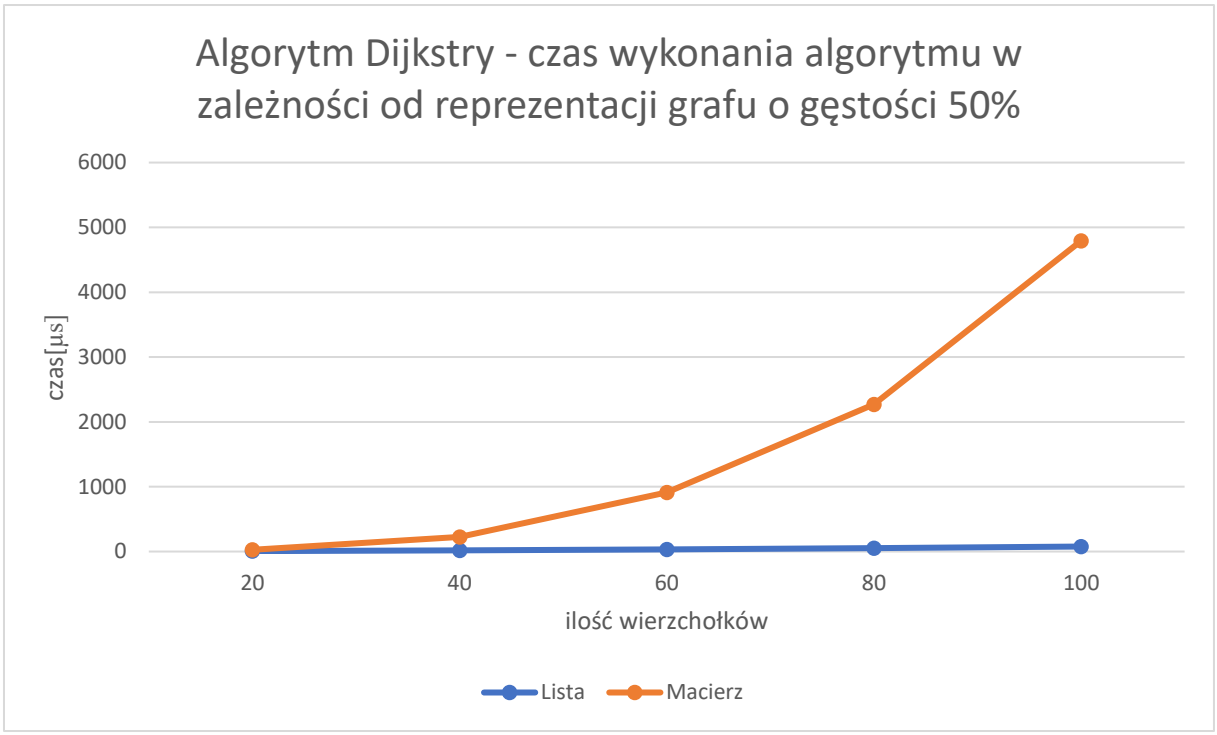

Algorytm Prima - czas wykonania algorytmu w zależności od reprezentacji grafu o gęstości 25%
100,00
90,00
80,00
70,00
60,00
50,00
40,00
30,00
20,00
10,00
0,00
czas[ms]
20
40
60
80
100
ilość wierzchołków
Lista
Macierz

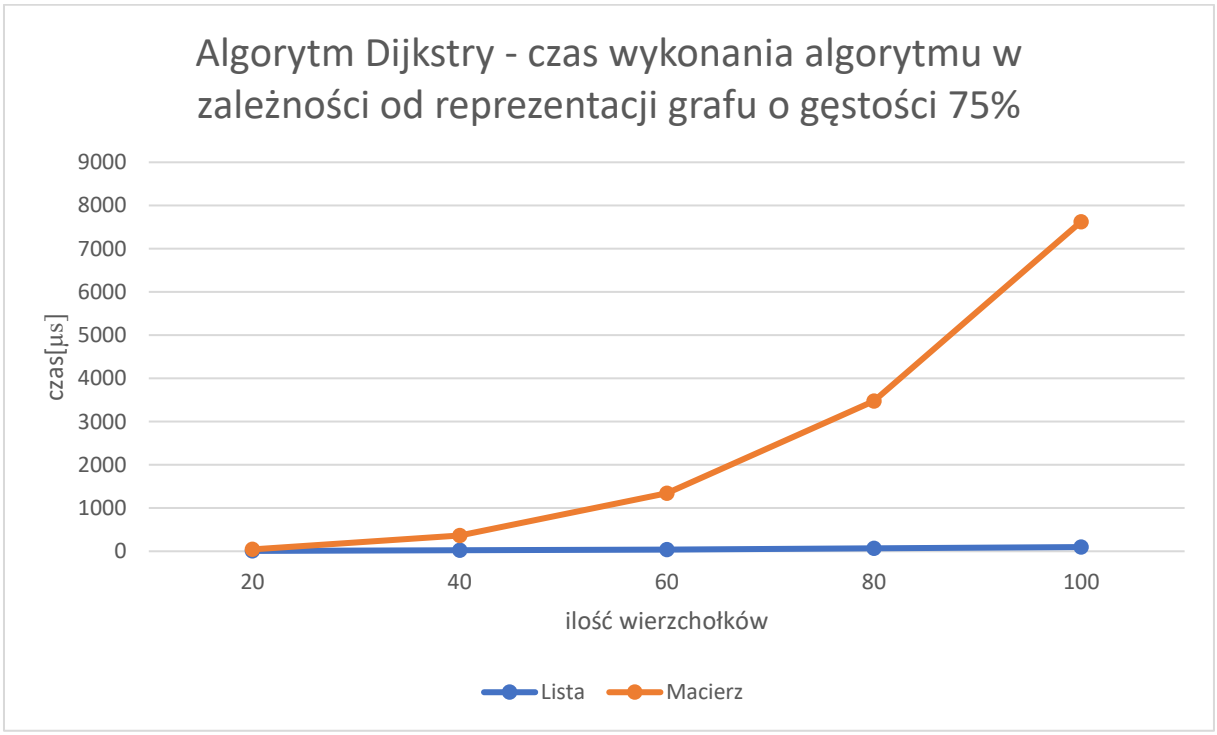

Algorytm Prima - czas wykonania algorytmu w zależności od reprezentacji grafu o gęstości 50%
250,00
200,00
150,00
100,00
50,00
0,00
czas[ms]
20
40
60
80
100
ilość wierzchołków
Lista
Macierz

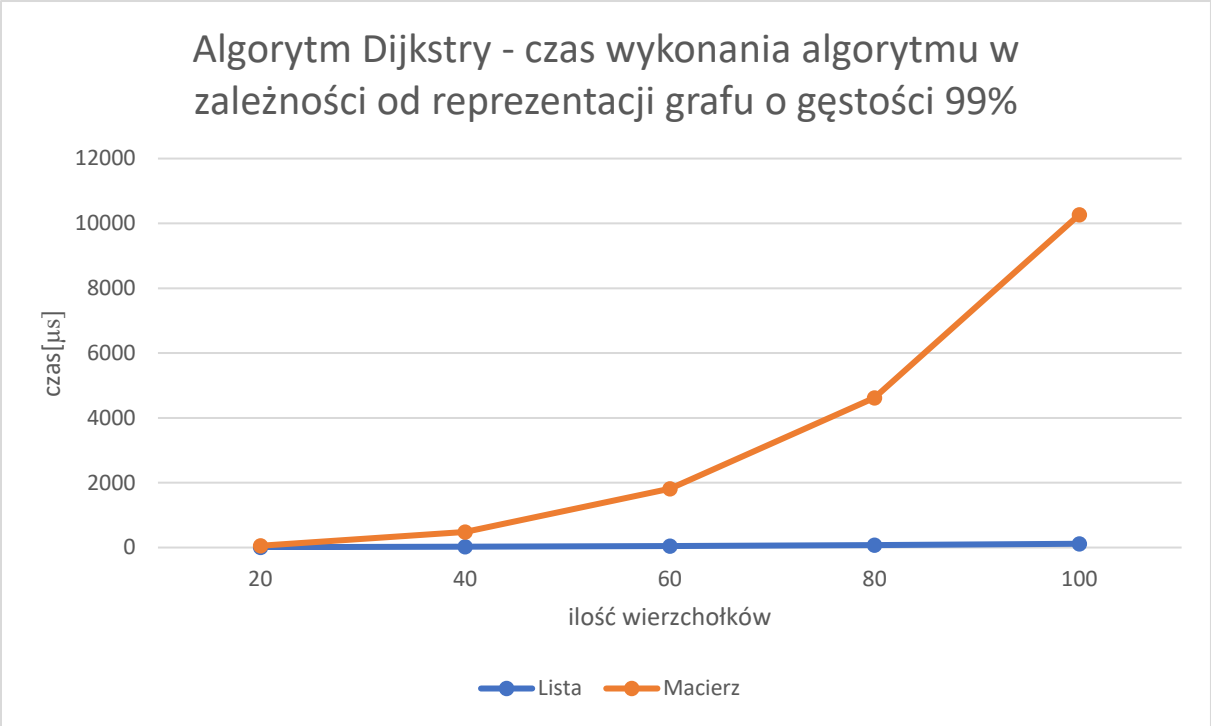

Algorytm Prima - czas wykonania algorytmu w zależności od reprezentacji grafu o gęstości 75%
350,00
300,00
250,00
200,00
150,00
100,00
50,00
0,00
czas[ms]
20
40
60
80
100
ilość wierzchołków
Lista
Macierz


Algorytm Prima - czas wykonania algorytmu w zależności od reprezentacji grafu o gęstości 99%
450,00
400,00
350,00
300,00
250,00
200,00
150,00
100,00
50,00
0,00
czas[ms]
20
40
60
80
100
ilość wierzchołków
Lista
Macierz


Algorytm Dijkstry - czas wykonania algorytmu w zależności od reprezentacji grafu o gęstości 25%
3000
2500
2000
1500
1000
500
0
czas[µs]
20
40
60
80
100
ilość wierzchołków
Lista
Macierz


Algorytm Dijkstry - czas wykonania algorytmu w zależności od reprezentacji grafu o gęstości 50%
6000
5000
4000
3000
2000
1000
0
czas[µs]
20
40
60
80
100
ilość wierzchołków
Lista
Macierz


Algorytm Dijkstry - czas wykonania algorytmu w zależności od reprezentacji grafu o gęstości 75%
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
czas[µs]
20
40
60
80
100
ilość wierzchołków
Lista
Macierz



5. Wnioski

Podsumowując udało mi się zaimplementować po jednym algorytmie grafowym dla każdego problemu. Algorytmy działające na liście sąsiedztwa działają znacznie szybciej w porównaniu do macierzy incydencji. W algorytmie najkrótszej ścieżki użyłem kolejki priorytetowej, stąd czas wykonania algorytmu jest zauważalnie krótszy niż dla algorytmu minimalnego drzewa spinającego.