STRUKTURY DANYCH I ZŁOŻONOŚĆ OBLICZENIOWA
projekt

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

AUTOR:

PATRYCJA LANGKAFEL, NR 252744

ZADANIE PROJEKTOWE 2

TERMIN: PONIEDZIAŁEK 9.15, PARZYSTY

PROWADZĄCY: mgr ANTONI STERNA

1. WSTĘP

Zadaniem projektowym było napisanie programu i zmierzenie czasu wykonania algorytmów grafowych dla grafów w postaci macierzy sąsiedztwa i listy sąsiadów.

Algorytmy do przygotowania:

- Wyznaczanie minimalnego drzewa rozpinającego (algorytm Prima i Kruskala)
- Wyznaczanie najkrótszej ścieżki w grafie (algorytm Dijkstry i Bellmana-Forda).

1.1. ZAŁOŻENIA IMPLEMENTACJI GRAFÓW

Przy implementacji grafów przyjęto następujące założenia:

- Program napisany został w języku C++, bez użycia gotowych bibliotek.
- Wszystkie struktury danych są alokowane dynamiczne.
- Wagami krawędzi są liczby całkowite (int) większe od zera.

Przy implementacji grafów w postaci macierzy sąsiedztwa przyjęto założenia:

- Macierz jest tablicą dwuwymiarową alokowaną dynamicznie.
- Jeśli krawędź nie łączy się z danym wierzchołkiem element macierzy ma wartość równą zero (0).

Przy implementacji grafów w postaci listy sąsiedztwa przyjęto założenia:

- Struktura krawędzi zawierają wartość wagi krawędzi łączącej oraz wierzchołek startowy i wierzchołek końcowy.

1.2. GENEROWANIE GRAFÓW

Funkcja generująca grafy (skierowane i nieskierowane) jako argumenty ma gęstość grafu i liczbę wierzchołków i dążymy do jak najbardziej losowego rozłożenia krawędzi.

2. TEORIA-ZŁOŻONOŚĆ OBLICZENIOWA

2.1. WSTĘP TEORETYCZNY

Złożoność jest wyrażana jako funkcja parametru, od którego zależna jest jej wartość. Można wyróżnić dwa podstawowe typy złożoności obliczeniowej:

- Czasowa złożoność obliczeniowa - czas potrzebny by dany algorytm został wykonany.
- Pamięciowa złożoność obliczeniowa - ilość pamięci wykorzystywanej w celu realizacji algorytmu lub przechowania określonej liczby danych.

Dla wszystkich algorytmów, w których złożoności są zależne od zbioru danych, można wyróżnić trzy typy:

- Złożoność optymistyczna (czas wykonania algorytmu dla najkorzystniejszego zbioru danych).

• Złożoność średnia (średnia uzyskiwana z dokonanych pomiarów).

• Złożoność pesymistyczna (czas wykonania algorytmu dla najmniej korzystnego zbioru danych).

Wykorzystujemy dwie zmienne zależne od grafu:

• E - liczba krawędzi grafu

• V - liczba wierzchołków grafu

2.2. ZŁOŻONOŚĆ OBLICZENIOWA – PRIM

O rzędzie złożoności decyduje implementacja kolejki priorytetowej:

• dla implementacji poprzez zwykłą tablicę złożoność wynosi $O(E \cdot V)$.

• dla implementacji kolejki poprzez kopiec, złożoność wynosi $O(E \cdot \log(V))$.

2.3. ZŁOŻONOŚĆ OBLICZENIOWA – KRUSKAL

Dla każdego grafu złożoność obliczeniowa algorytmu Kruskala jest równa

$O(E \cdot \log(V))$.

2.4. ZŁOŻONOŚĆ OBLICZENIOWA – DIJSKTRA

O rzędzie złożoności decyduje implementacja kolejki priorytetowej:

• dla implementacji poprzez zwykłą tablicę złożoność wynosi O(V^2).

• dla implementacji kolejki poprzez kopiec, złożoność wynosi $O(E \cdot \log(V))$.

2.5. ZŁOŻONOŚĆ OBLICZENIOWA – BELLMAN FORD

Dla każdego grafu złożoność obliczeniowa algorytmu Bellmana-Forda jest równa

$O(E \cdot V)$.

3. PRZEBIEG EKSPERYMENTU

1. Testy zostały wykonane automatycznie.
2. Wyniki są zapisane w pliku *wyniki.txt*.
3. Badania były przeprowadzone dla liczb wierzchołków: 50, 100, 150, 300, 400 i gęstości podanych przez prowadzącego: 0.25, 0.5, 0.75, 0.99.
4. Czasy wykonania algorytmów uzyskałam, dzięki bibliotece <chrono> (wyniki w nanosekundach). W tabelach czasy są przedstawione w mikrosekundach.
5. Przy testach wykorzystałam uśrednioną wartość z 100 wyników dla odpowiednich algorytmów i reprezentacji.

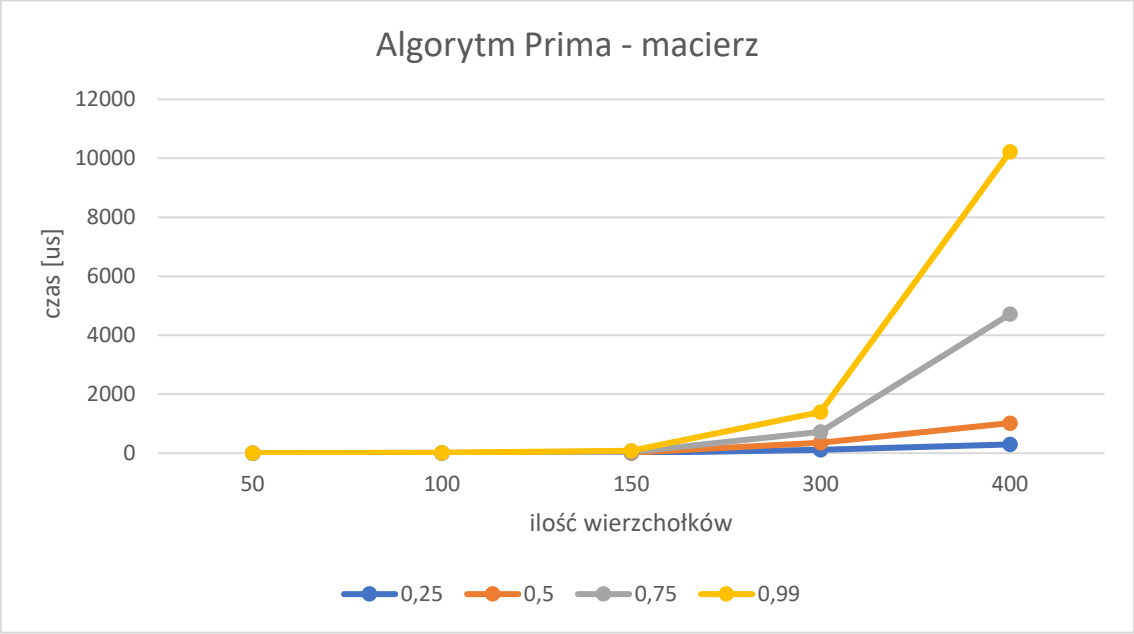
4. WYNIKI BADAŃ

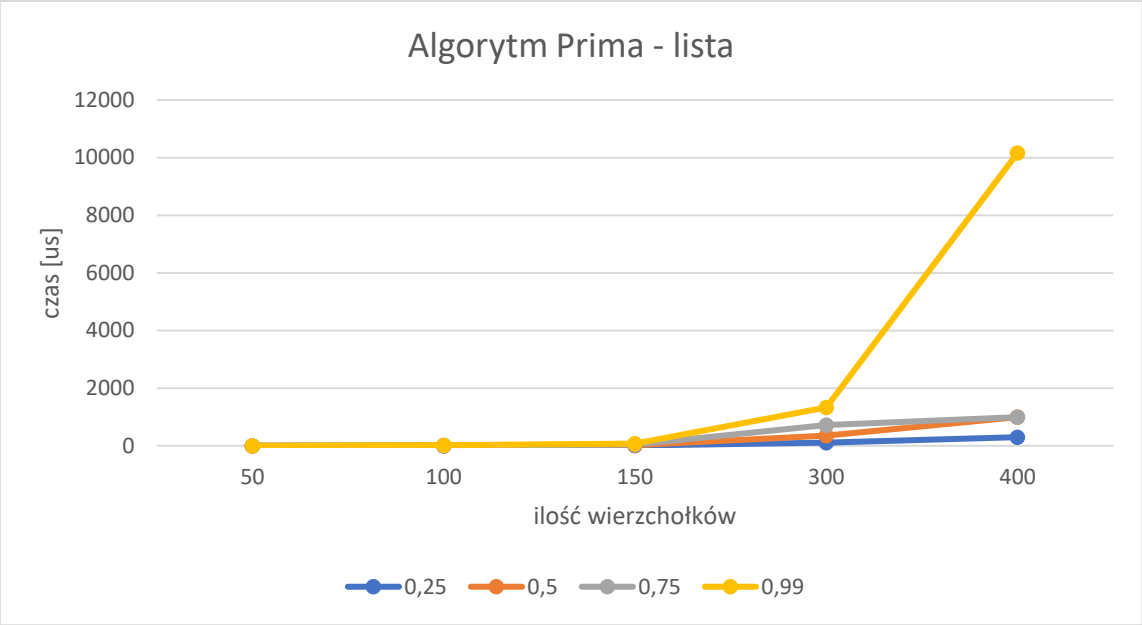
4.1. ALGORYTM PRIMA

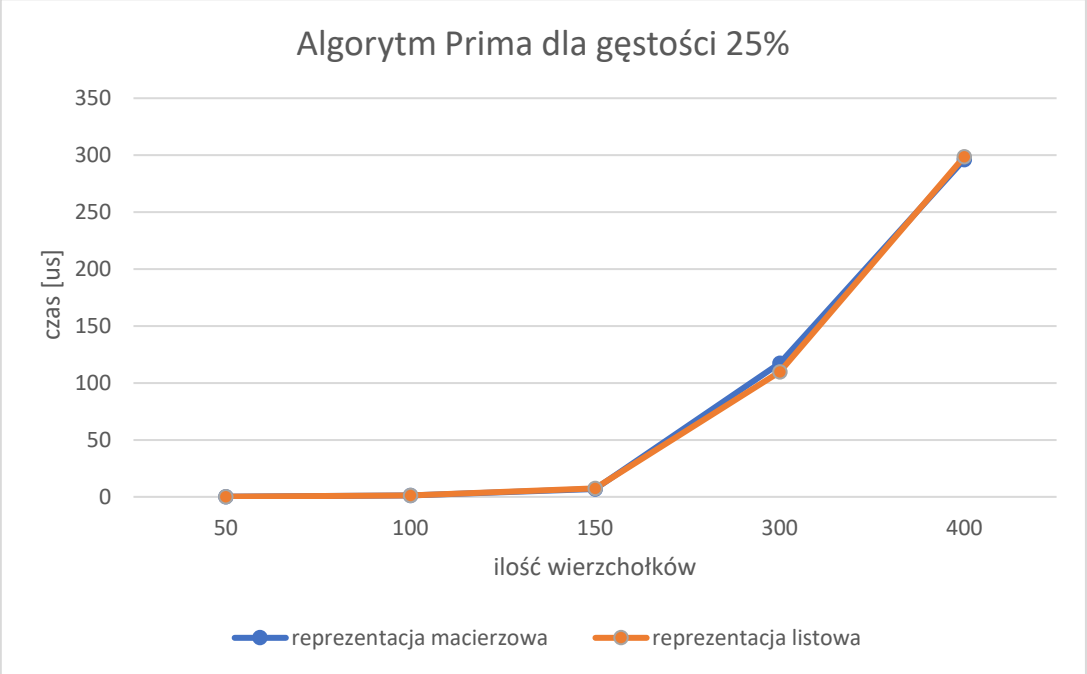
4.1.1. Tabela wyników.

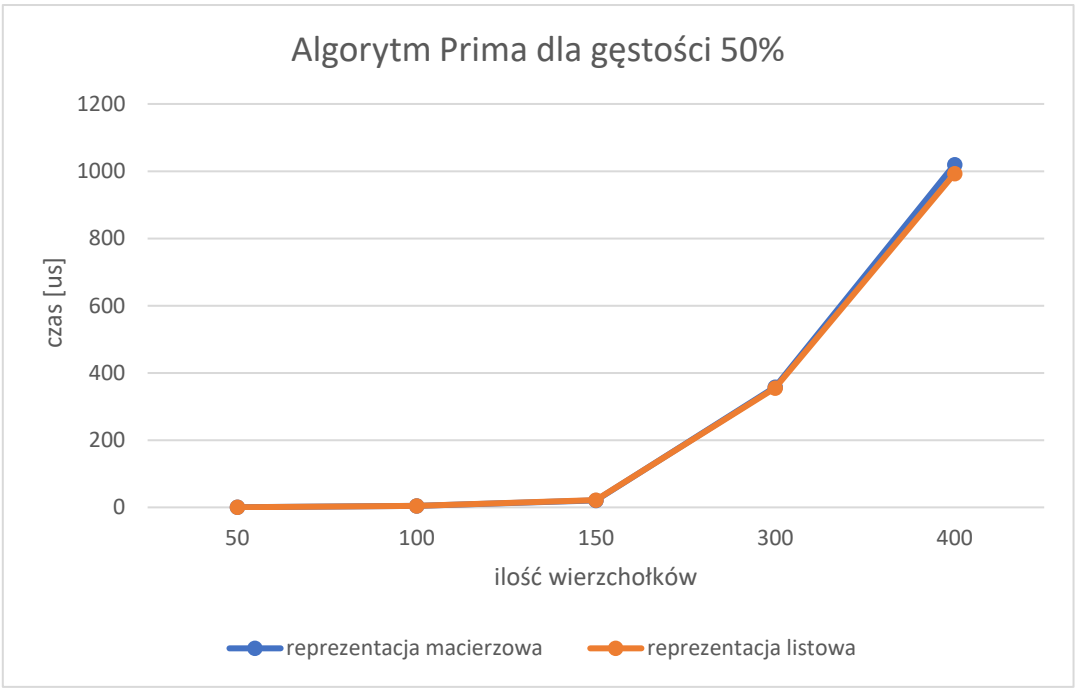
Liczba wierzchołków	Gęstość grafu	Rep. Macierzowa [us]	Rep. Listowa[us]
50	0,25	0,16	0,16
50	0,5	0,47	0,31
50	0,75	0,67	0,83
50	0,99	1,13	1,12
100	0,25	1,30	1,35
100	0,5	4,12	4,59
100	0,75	10,08	10,03
100	0,99	15,57	15,17
150	0,25	6,92	7,33
150	0,5	20,92	21,21
150	0,75	42,19	42,26
150	0,99	78,18	76,77
300	0,25	117,37	109,87
300	0,5	357,52	354,53
300	0,75	722,98	715,98
300	0,99	1398,73	1335,76
400	0,25	296,16	298,69
400	0,5	1019,89	992,44
400	0,75	4712,87	992,44
400	0,99	10222,71	10161,56

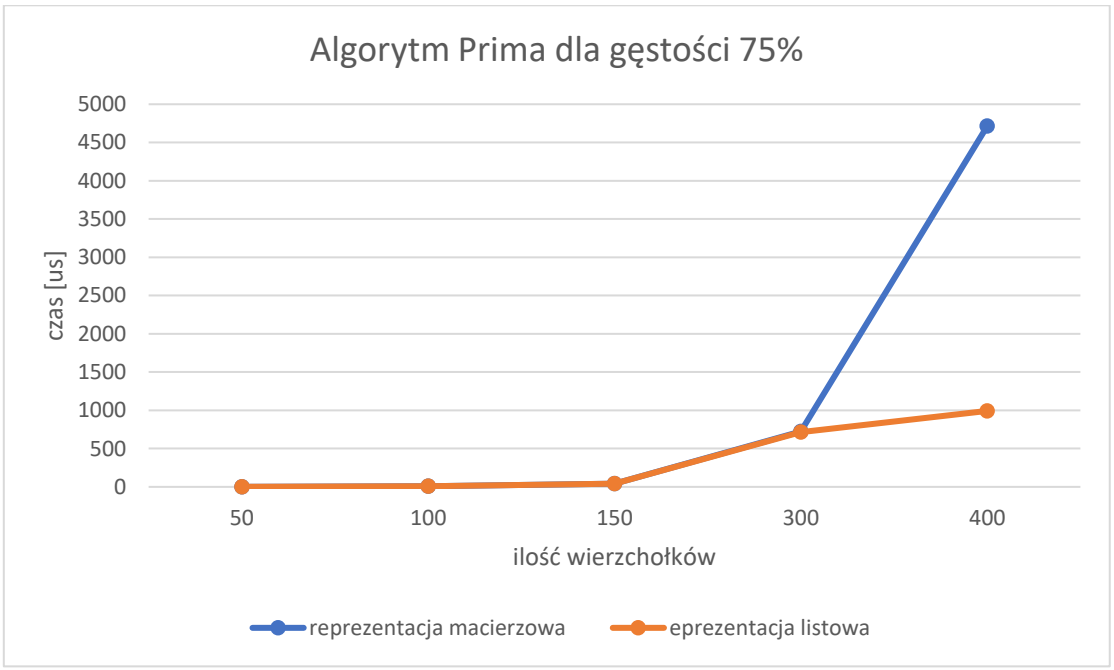
4.1.2. Wykresy.

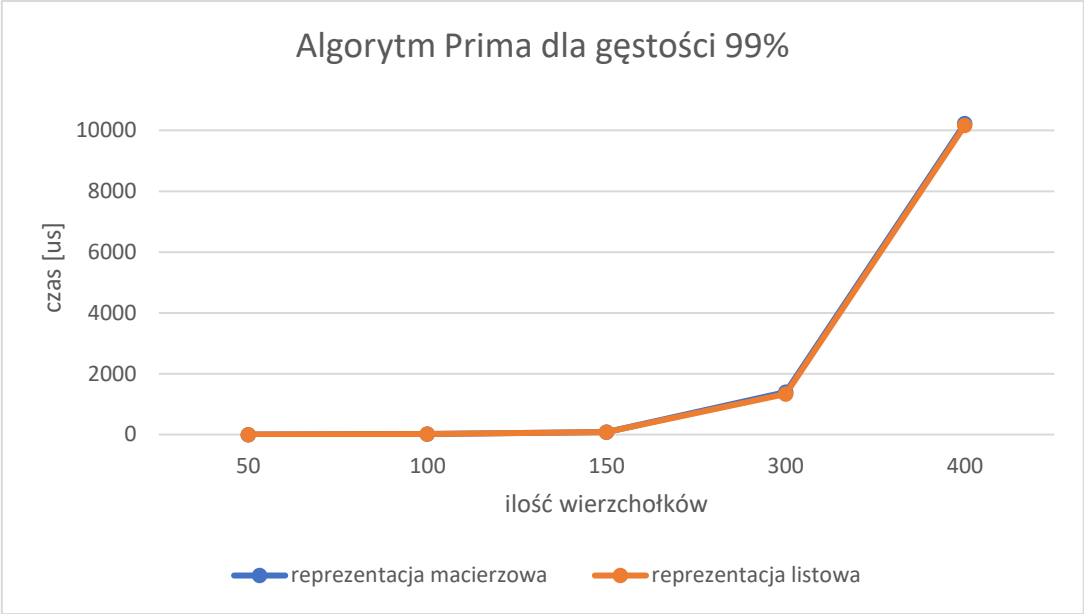



Algorytm Prima - lista
12000
10000
8000
6000
4000
2000
0
czas [us]
50
100
150
300
400
ilość wierzchołków
0,25
0,5
0,75
0,99


Algorytm Prima dla gęstości 25%
350
300
250
200
150
100
50
0
czas [us]
50
100
150
300
400
ilość wierzchołków
reprezentacja macierzowa
reprezentacja listowa


Algorytm Prima dla gęstości 50%
1200
1000
800
600
400
200
0
czas [us]
50
100
150
300
400
ilość wierzchołków
reprezentacja macierzowa
reprezentacja listowa


Algorytm Prima dla gęstości 75%
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
czas [us]
50
100
150
300
400
ilość wierzchołków
reprezentacja macierzowa
eprezentacja listowa



4.1.3. Podsumowanie.

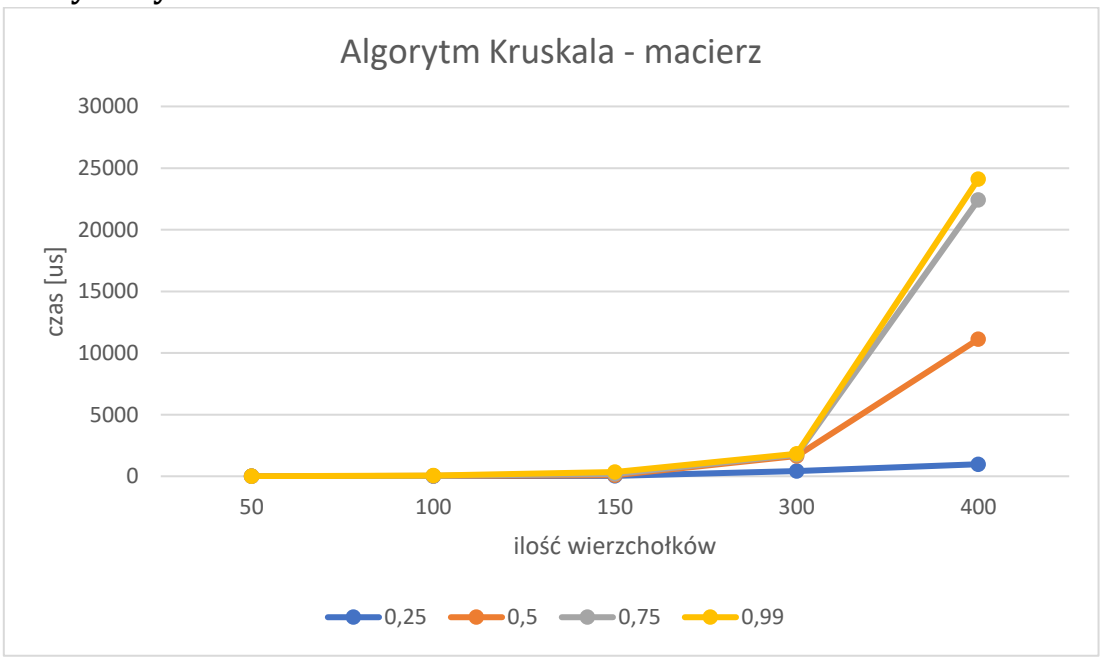
Dla mojej implementacji bardziej optymalna jest reprezentacja listowa, chociaż czasy wykonania dla odpowiednich gęstości są podobne. Różnicę widać przy gęstości 75%.

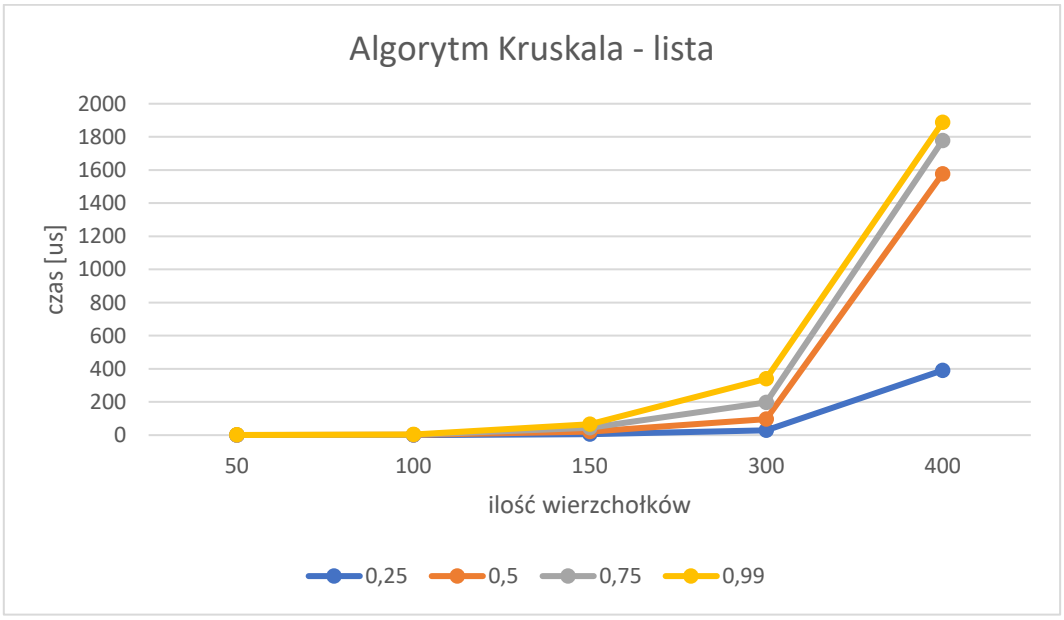
4.2. ALGORYTM KRUSKALA

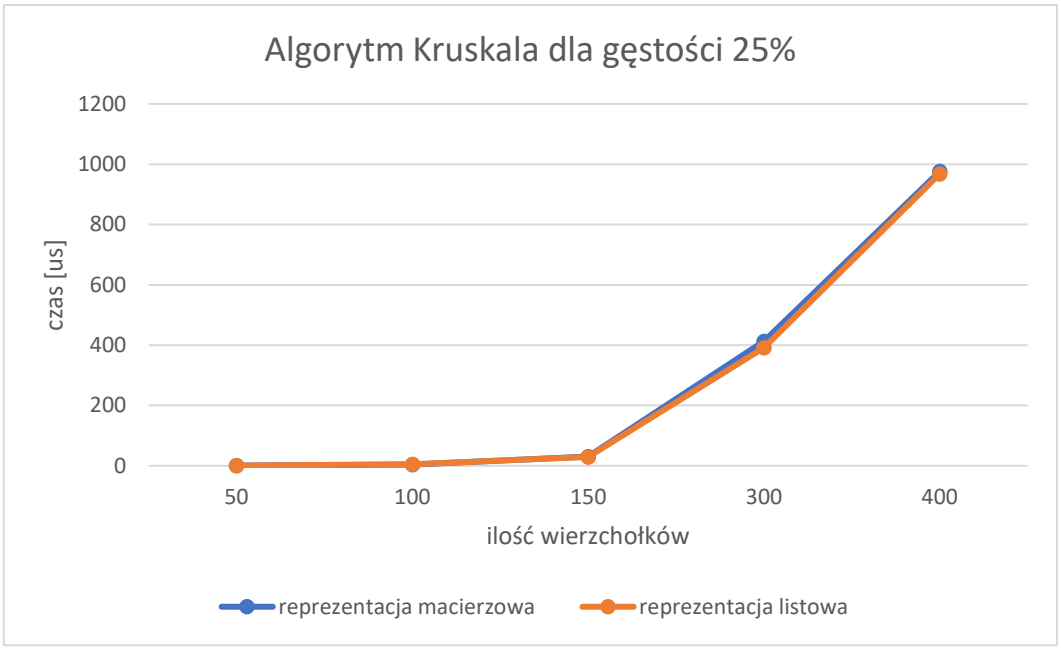
4.2.1. Tabela wyników.

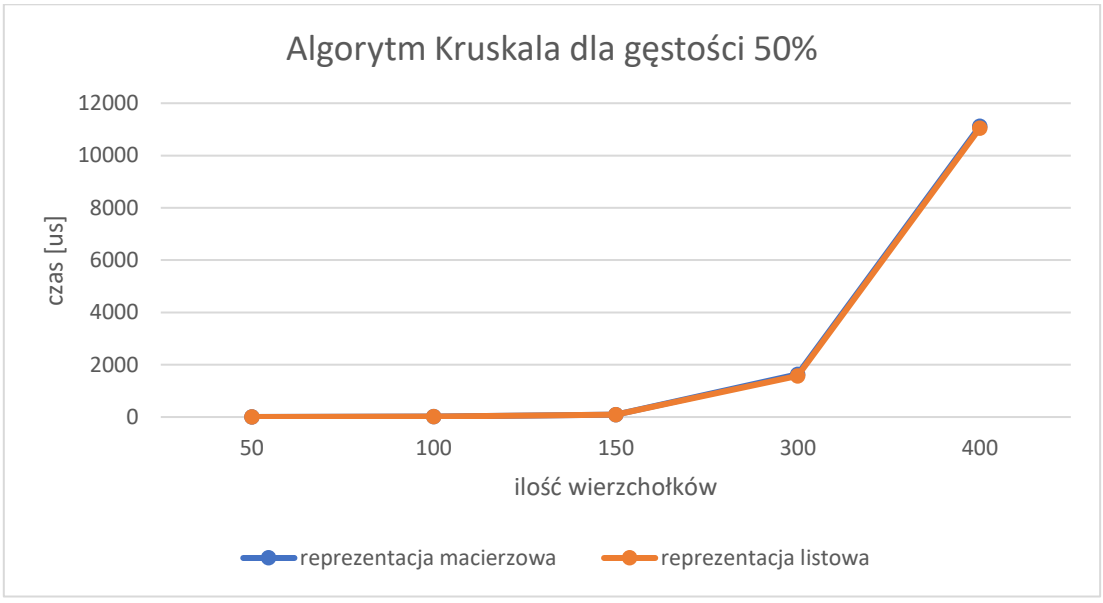
Liczba wierzchołków	Gęstość grafu	Rep. Macierzowa [us]	Rep. Listowa[us]
50	0,25	0,61	0,42
50	0,5	1,54	1,05
50	0,75	3,22	1,73
50	0,99	3,88	3,92
100	0,25	4,30	5,47
100	0,5	21,29	21,38
100	0,75	45,75	44,48
100	0,99	71,99	66,12
150	0,25	30,43	29,75
150	0,5	93,15	97,23
150	0,75	201,44	196,78
150	0,99	350,22	340,16
300	0,25	413,00	391,07
300	0,5	1634,27	1577,16
300	0,75	1734,27	1777,16
300	0,99	1834,27	1887,16
400	0,25	975,98	967,95
400	0,5	11116,96	11046,52
400	0,75	22416,94	24047,78
400	0,99	24116,96	26046,52

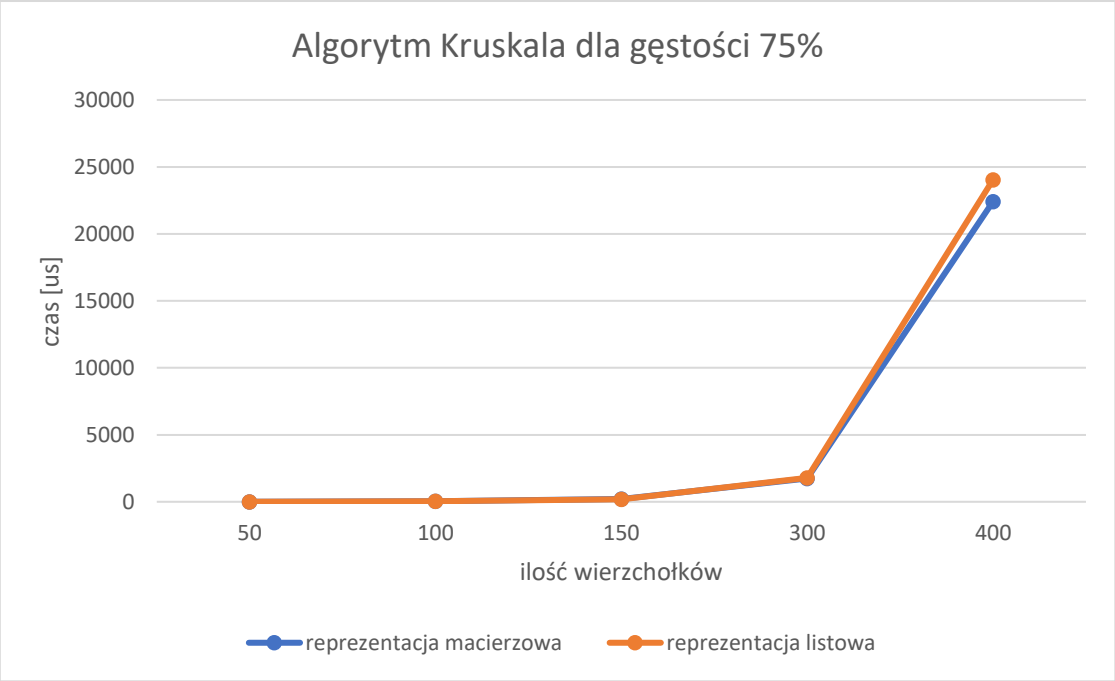
4.2..2. Wykresy.

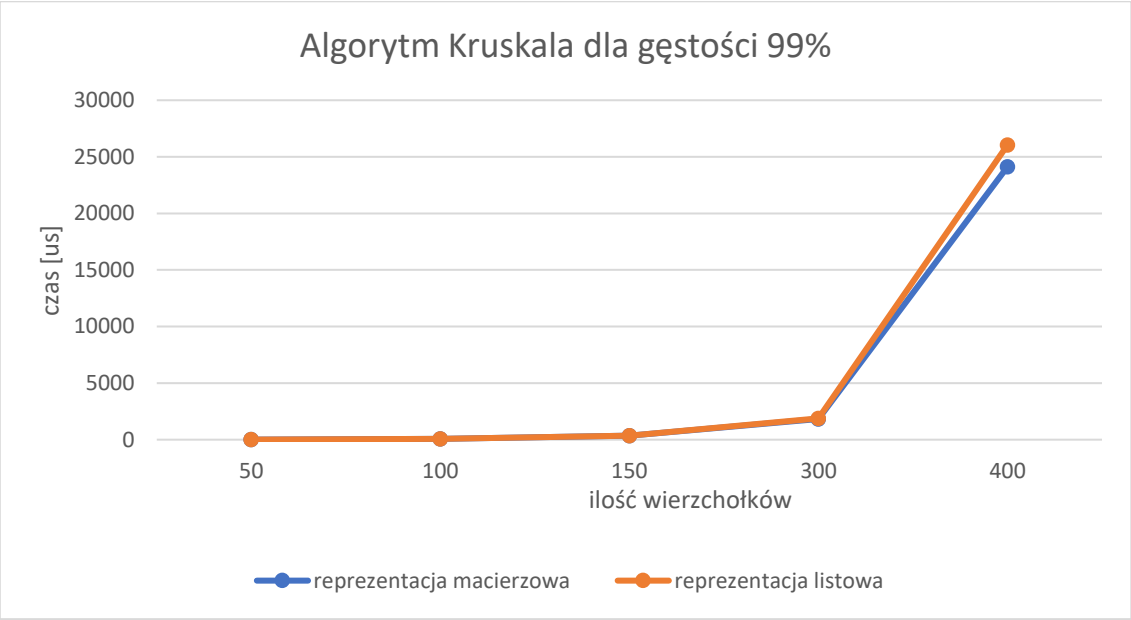

Algorytm Kruskala - macierz
30000
25000
20000
15000
10000
5000
0
czas [us]
50
100
150
300
400
ilość wierzchołków
0,25
0,5
0,75
0,99


Algorytm Kruskala - lista
2000
1800
1600
1400
1200
1000
800
600
400
200
0
czas [us]
50
100
150
300
400
ilość wierzchołków
0,25
0,5
0,75
0,99


Algorytm Kruskala dla gęstości 25%
1200
1000
800
600
400
200
0
czas [us]
50
100
150
300
400
ilość wierzchołków
reprezentacja macierzowa
reprezentacja listowa


Algorytm Kruskala dla gęstości 50%
12000
10000
8000
6000
4000
2000
0
czas [us]
50
100
150
300
400
ilość wierzchołków
reprezentacja macierzowa
reprezentacja listowa





4.2.3. Podsumowanie.

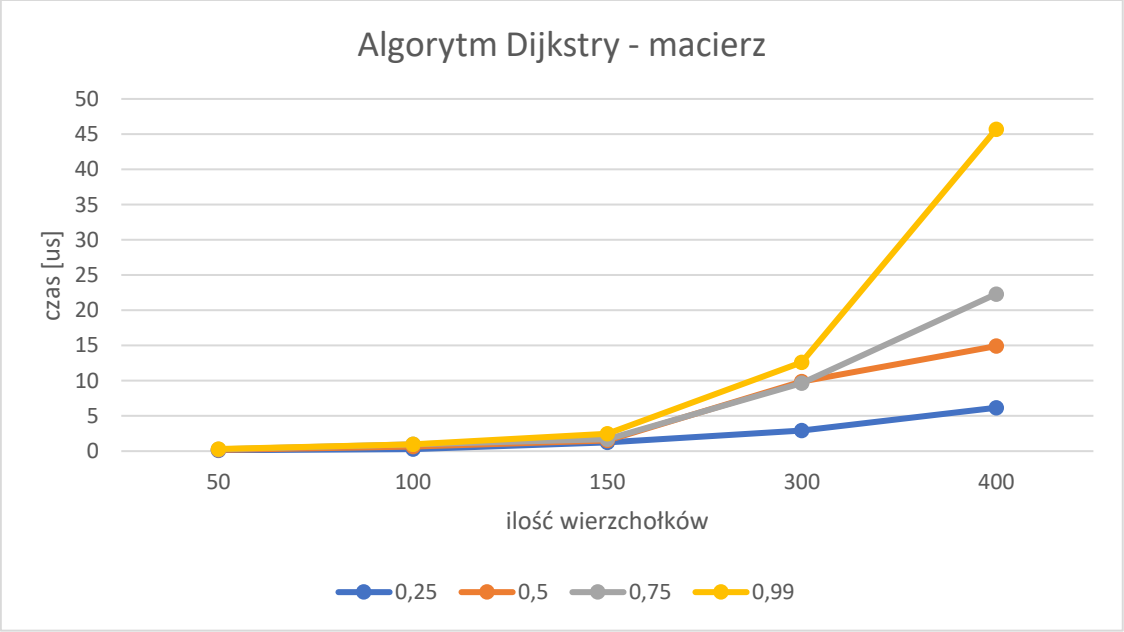
Minimalnie bardziej optymalna jest reprezentacja macierzowa, chociaż czasy wykonywania algorytmu dla obu reprezentacji są podobne przy zwiększającej się gęstości grafu.

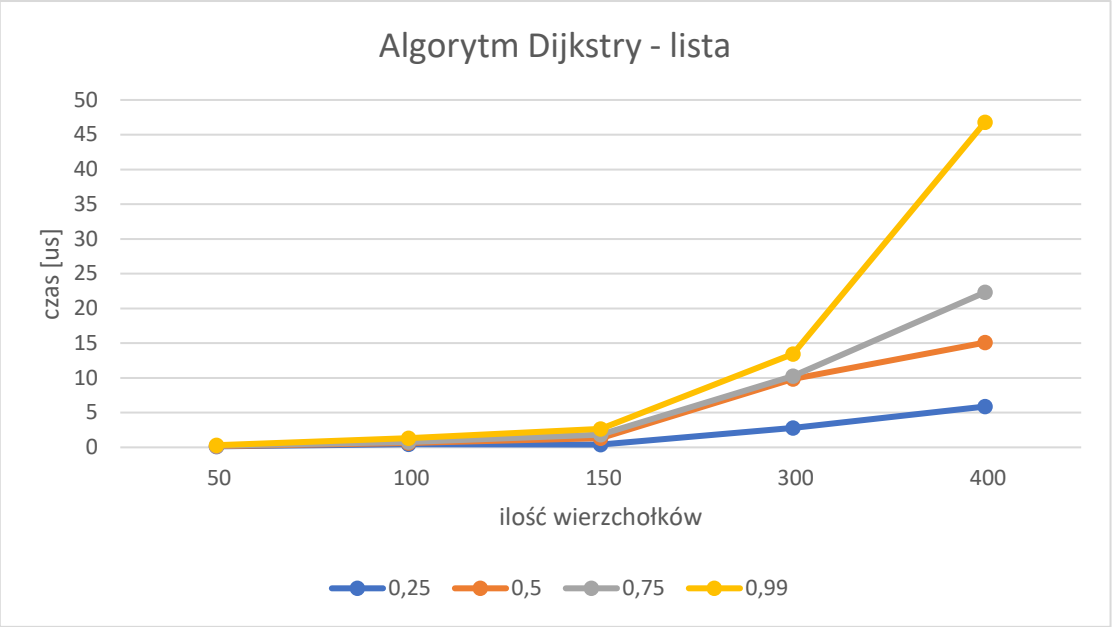
4.3. ALGORYTM DIJSKTRY

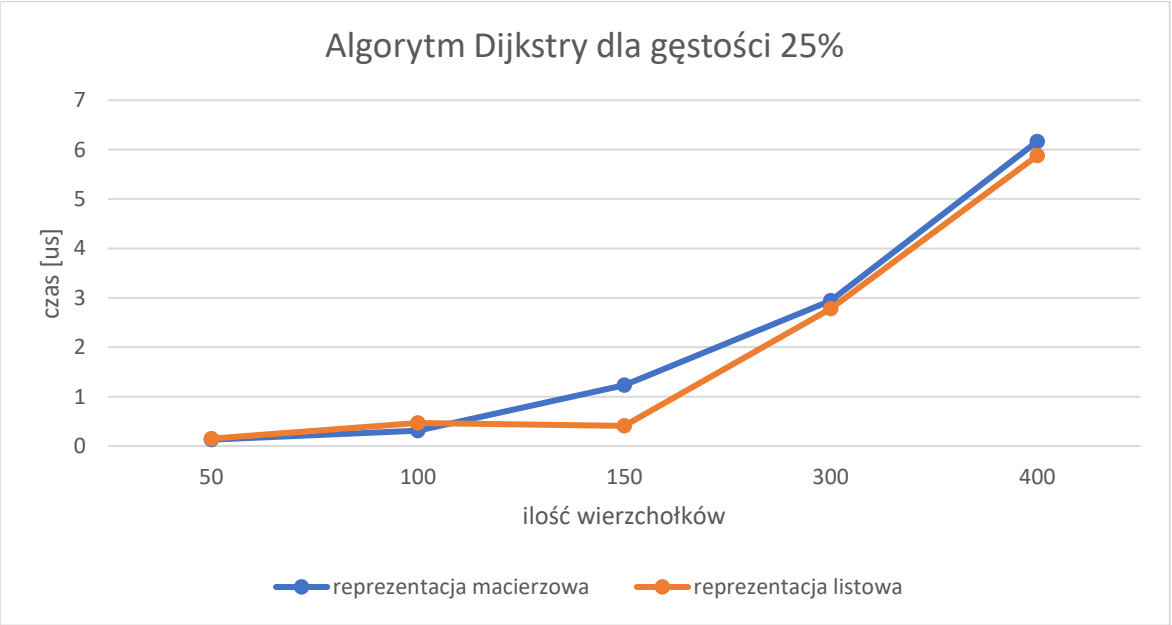
4.3.1. Tabela wykresów.

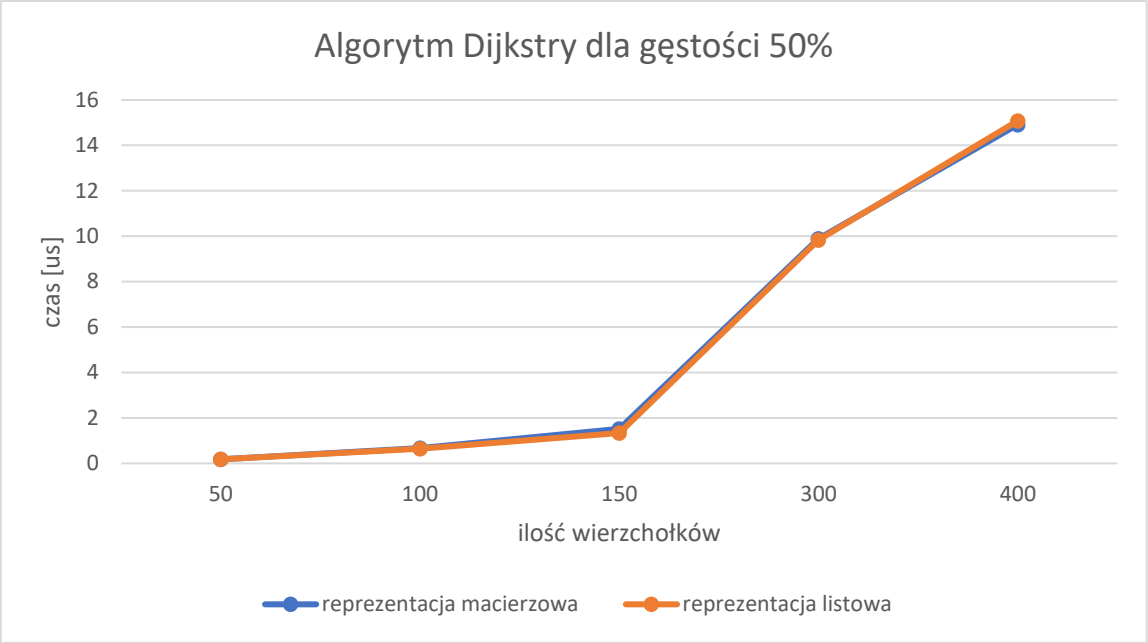
Liczba wierzchołków	Gęstość grafu	Rep. Macierzowa [us]	Rep. Listowa[us]
50	0,25	0,13	0,15
50	0,5	0,18	0,18
50	0,75	0,26	0,24
50	0,99	0,31	0,31
100	0,25	0,31	0,47
100	0,5	0,67	0,65
100	0,75	0,99	0,75
100	0,99	0,99	1,34
150	0,25	1,23	0,41
150	0,5	1,52	1,33
150	0,75	1,69	1,85
150	0,99	2,45	2,66
300	0,25	2,94	2,78
300	0,5	9,90	9,83
300	0,75	9,63	10,26
300	0,99	12,62	13,43
400	0,25	6,16	5,88
400	0,5	14,91	15,08
400	0,75	22,31	22,31
400	0,99	45,70	46,81

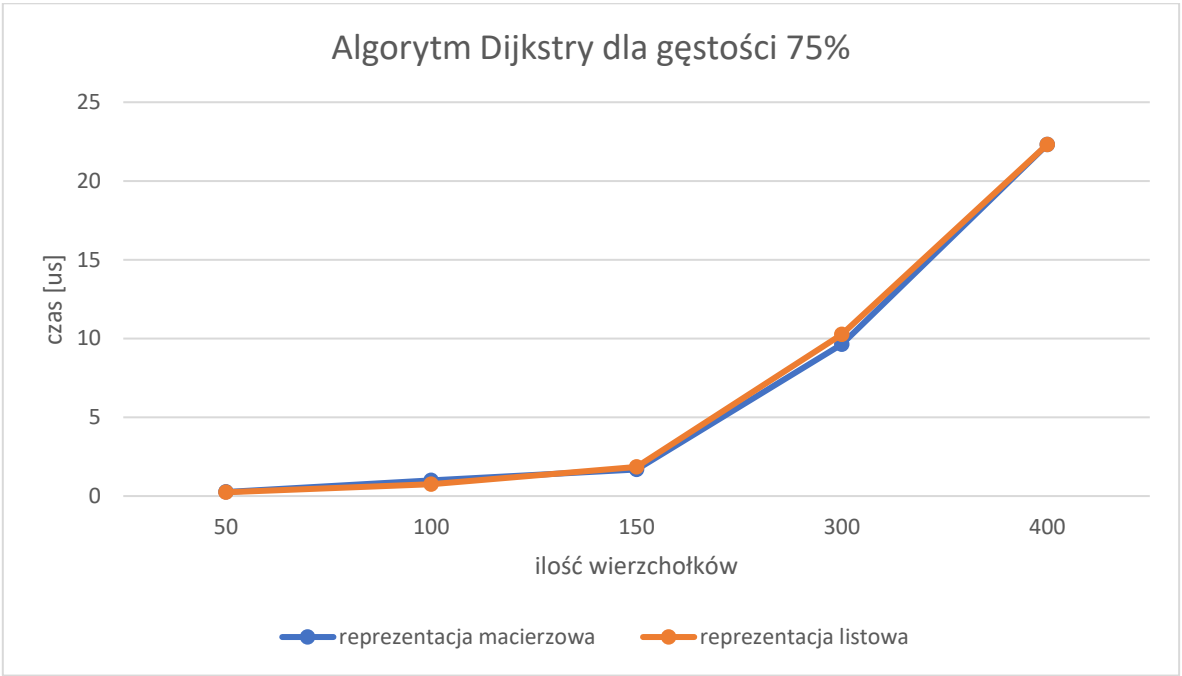
4.3.2. Wykresy.

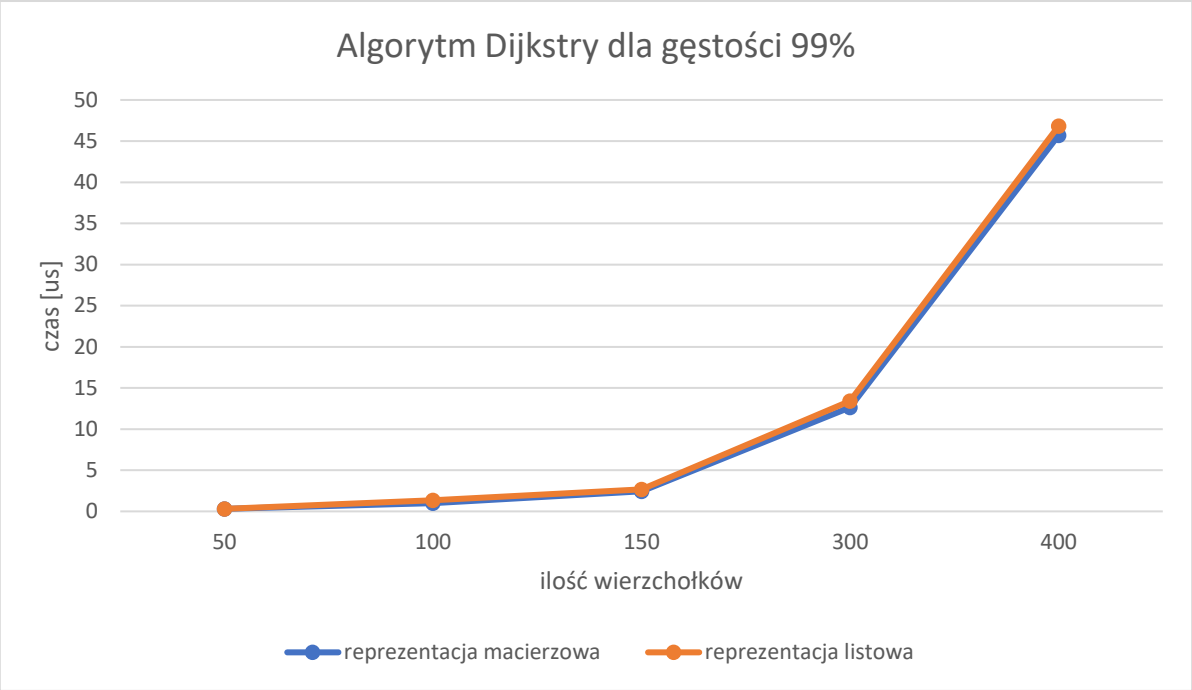



Algorytm Dijkstry - lista
50
45
40
35
30
25
20
15
10
5
0
czas [us]
50
100
150
300
400
ilość wierzchołków
0,25
0,5
0,75
0,99


Algorytm Dijkstry dla gęstości 25%
7
6
5
4
3
2
1
0
czas [us]
50
100
150
300
400
ilość wierzchołków
reprezentacja macierzowa
reprezentacja listowa


Algorytm Dijkstry dla gęstości 50%
16
14
12
10
8
6
4
2
0
czas [us]
50
100
150
300
400
ilość wierzchołków
reprezentacja macierzowa
reprezentacja listowa


Algorytm Dijkstry dla gęstości 75%
25
20
15
10
5
0
czas [us]
50
100
150
300
400
ilość wierzchołków
reprezentacja macierzowa
reprezentacja listowa



4.3.3. Podsumowanie.

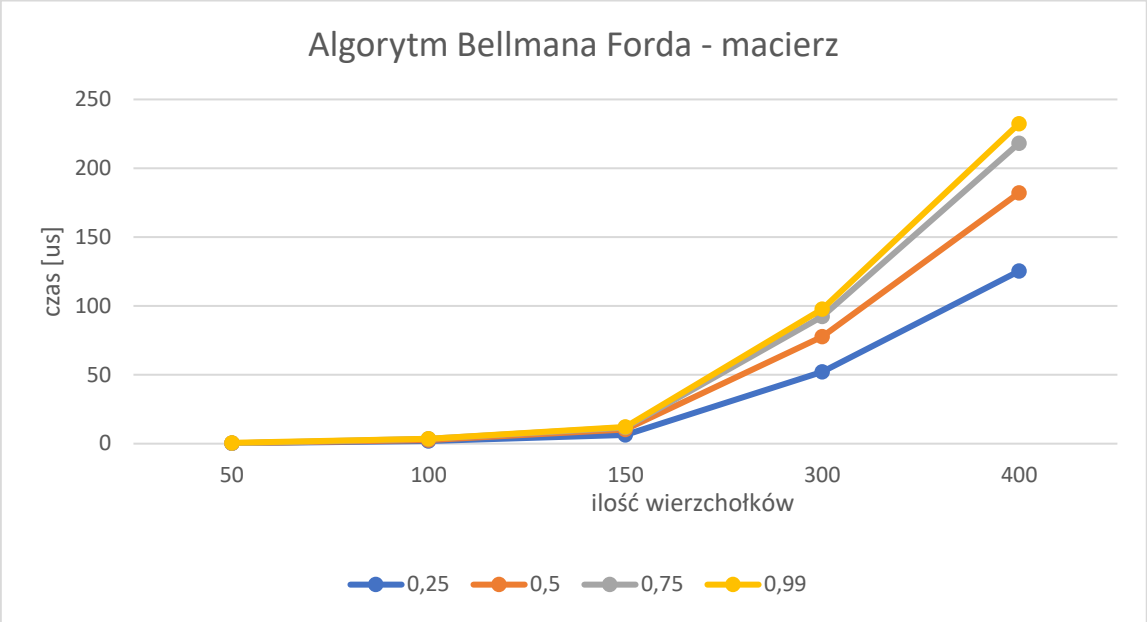
Podobnie jak przy poprzednim algorytmie, dla mojej implementacji bardziej optymalna jest reprezentacja macierzowa. Ale dla gęstości 25% wyniki są rozbieżne dla przyjętego wniosku.

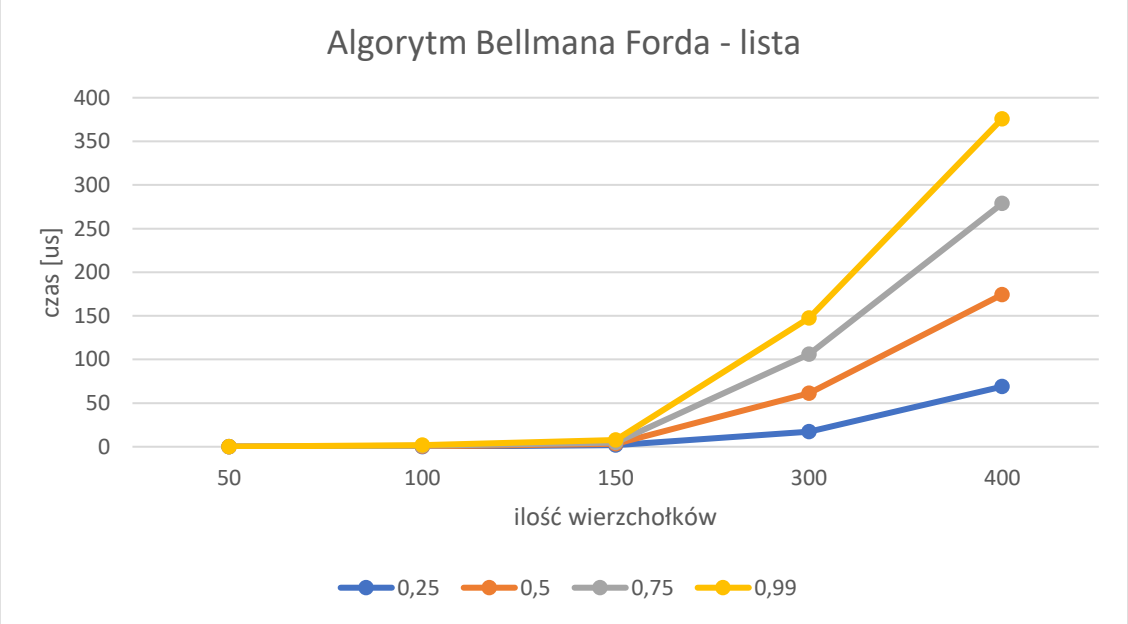
4.4. ALGORYTM BELLMANA-FORDA

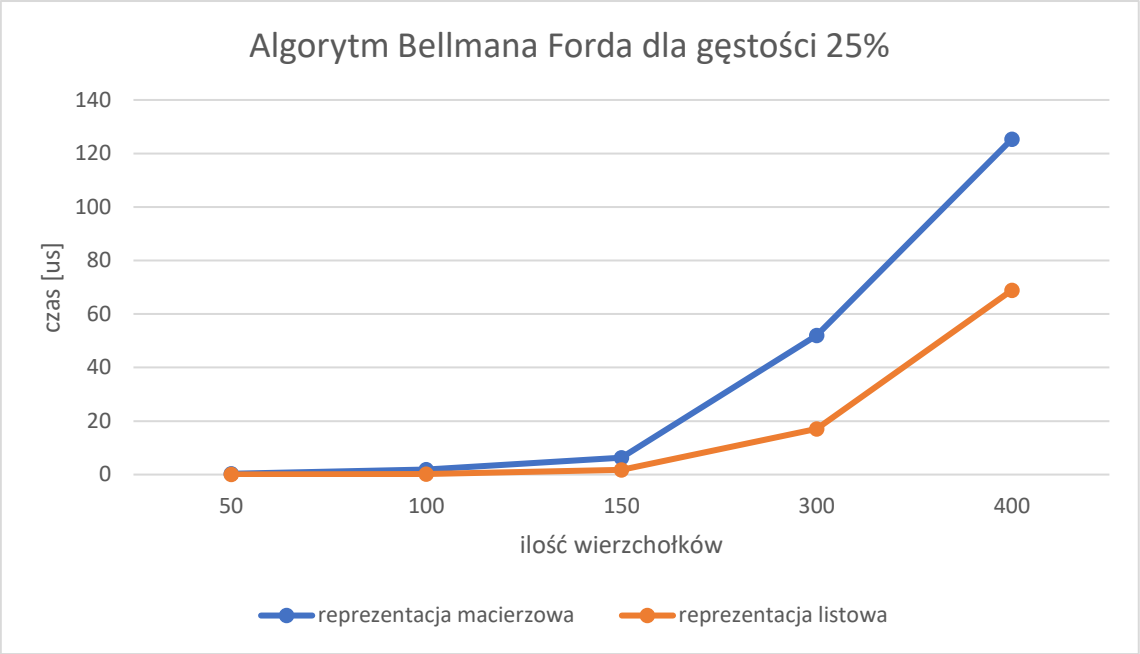
4.4.1. Tabela wyników.

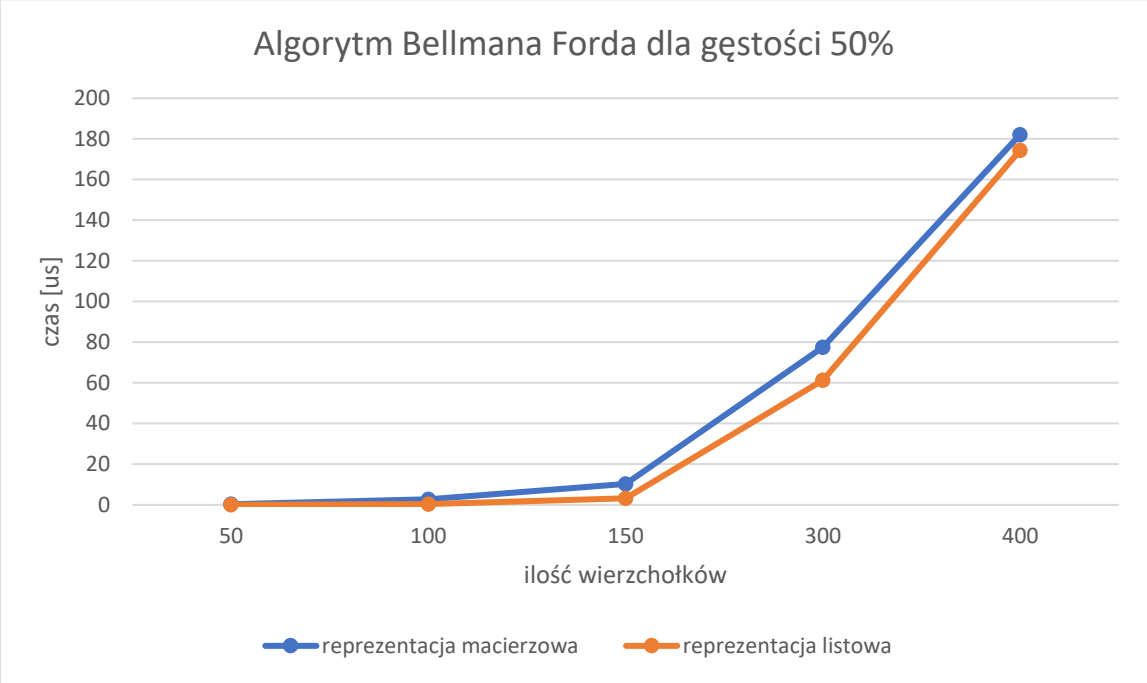
Liczba wierzchołków	Gęstość grafu	Rep. Macierzowa [us]	Rep. Listowa[us]
50	0,25	0,31	0,05
50	0,5	0,31	0,06
50	0,75	0,22	0,16
50	0,99	0,63	0,16
100	0,25	1,91	0,16
100	0,5	2,70	0,41
100	0,75	3,56	1,26
100	0,99	3,61	1,90
150	0,25	6,37	1,72
150	0,5	10,26	3,24
150	0,75	11,67	4,74
150	0,99	12,20	7,62
300	0,25	52,05	17,10
300	0,5	77,61	61,33
300	0,75	92,37	106,08
300	0,99	97,74	147,36
400	0,25	125,31	68,88
400	0,5	182,11	174,41
400	0,75	218,19	278,68
400	0,99	232,44	375,87

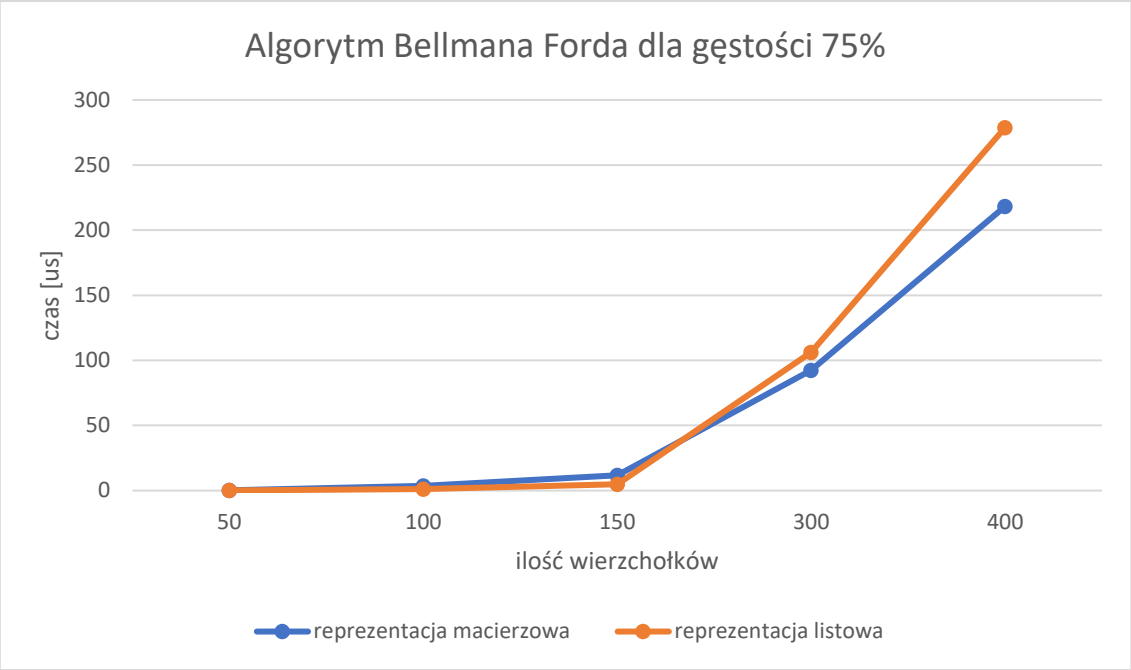
4.4.2. Wykresy.

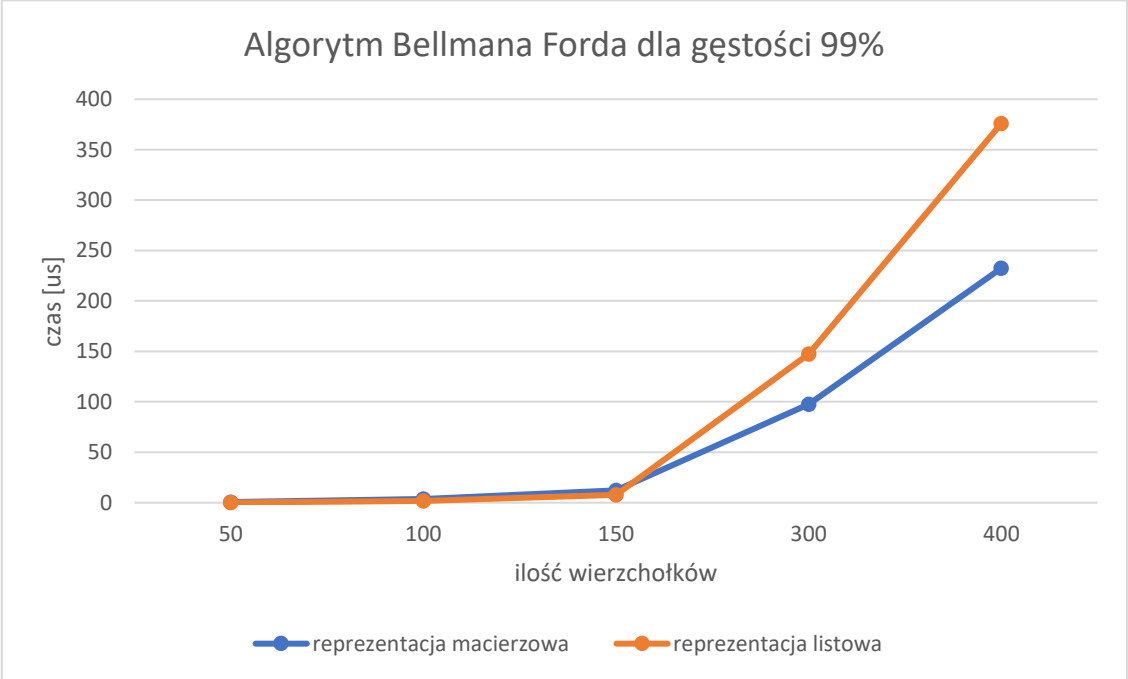

Algorytm Bellmana Forda - macierz
250
200
150
100
50
0
czas [us]
50
100
150
300
400
ilość wierzchołków
0,25
0,5
0,75
0,99


Algorytm Bellmana Forda - lista
400
350
300
250
200
150
100
50
0
czas [us]
50
100
150
300
400
ilość wierzchołków
0,25
0,5
0,75
0,99


Algorytm Bellmana Forda dla gęstości 25%
140
120
100
80
60
40
20
0
czas [us]
50
100
150
300
400
ilość wierzchołków
reprezentacja macierzowa
reprezentacja listowa


Algorytm Bellmana Forda dla gęstości 50%
200
180
160
140
120
100
80
60
40
20
0
czas [us]
50
100
150
300
400
ilość wierzchołków
reprezentacja macierzowa
reprezentacja listowa





4.4.3. Podsumowanie.

Dla tego algorytmu bardziej optymalna jest reprezentacja listowa. Dla reprezentacji w formie listy czas wykonywania wolniej wzrastają wraz ze zwiększeniem gęstości grafu.

5. PODSUMOWANIE

Dla algorytmu Prima minimalnie gorsze wyniki uzyskiwała reprezentacja macierzowa. W algorytmie wyszukiwania MST Kruskala dla obu struktur pomiary dawały podobne wyniki, a z czasem macierz osiągała lepsze rezultaty, szczególnie przy większych gęstościach. W przypadku wyszukiwania MST, lepsze wyniki miał algorytm Kruskala.

Dla algorytmu Djikstry zależność czasu wykonania algorytmu wzrasta kwadratowo, wraz ze zwiększaniem liczby wierzchołków oraz jest bardziej optymalny niż algorytm Bellmana-Forda. Algorytm Bellmana-Forda działa o rzędy wielkości szybciej dla listy sąsiedztwa.