Struktury danych i złożoność obliczeniowa – Projekt

Sprawozdanie

Uladzimir Kaviaka(257276)

Zadanie projektowe 2	
Badanie efektywności algorytmów grafowych w zależności od rozmiaru instancji oraz sposobu reprezentacji grafu w pamięci komputera	
Prowadzący	Mgr. Inż. Antoni Sterna
Termin zajęć	TN, poniedziałek, 11:15
Termin oddania:	25.05.2021

Spis treści

Wstęp

Projekt polega na napisaniu programu w języku bez maszyny wirtualnej i zmierzenie czasu wykonania dla algorytmów:

- Wyznaczanie minimalnego drzewa rozpinającego
    1. Algorytm Prima
    2. Algorytm Kruskala
- Wyznaczanie najkrótszej ścieżki w grafie
    1. Algorytm Dijkstry
    2. Algorytm Bellmana – Forda

Do przechowywania grafów w pamięci komputera realizowane są dwie metody:

- Macierz sąsiedztwa
- Lista sąsiadów

Przy realizowaniu projektu uwzględniono następujące założenia:

- Waga krawędzi jest liczbą całkowitą
- Wszystkie struktury alokowane dynamicznie i zajmują jak najmniej miejsca.
- Program napisany w języku C++ bez użycia biblioteki STL
- Do generowanie liczb użyto maszyny losującej mt19937 (Mersenne twister, Matsumoto i Nishimura 1998)
- Pomiaru czasu dokonano za pomocą funkcji *QueryPerformanceCounter*
- Wyniki w tabelach uśrednione dla 50 pomiarów dla każdego algorytmu

Złożoność obliczeniowa

Definicja

Złożoność obliczeniowa jest definiowana jako określanie ilości zasobów potrzebnych do rozwiązania problemów obliczeniowych. Wyrażana za pomocą funkcji parametru, od której zależy jej wartość.

W zależności od rozważanego zasobu można wyróżnić dwa typy złożoności

1. Złożoność czasowa – wyrażana za pomocą zwykłych jednostek czasu, np. [ms]. Miarą złożoności czasowej jest liczba wykonanych operacji podstawowych (podstawienie, porównanie itd.) w zależności od rozmiaru danych wejściowych.
2. Złożoność pamięciowa – jest miarą ilości wykorzystanej pamięci. Wyraża się za pomocą funkcji użytych zasobów od rozmiaru danych wejściowych. Również możliwe obliczanie rozmiaru potrzebnych zasobów wyrażonych w bitach/bajtach.

Typy złożoności

- Złożoność optymistyczna – określa najkrótszy czas wykonania algorytmu dla najbardziej uporządkowanego zbioru danych wejściowych.
- Złożoność średnia – określa czas wykonania algorytmu dla losowego zbioru danych.
- Złożoność pesymistyczna – określa najdłuższy czas wykonanie algorytmu dla najbardziej nieuporządkowanego zbioru danych wejściowych.

Złożoności badanych algorytmów

Złożoność algorytmu Kruskala

$O(E * \log(V))$

Złożoność algorytmu Prima

$O(E * \log(V))$

Złożoność algorytmu Dijkstry

$O(E * \log(V))$

Złożoność algorytmu Bellmana – Forda

$O(V * E)$

Gdzie E – liczba krawędzi grafu, V – liczba wierzchołków grafu

Plan projektu

W projekcie zostały zaimplementowane wymienione wyżej algorytmy. Zgodnie z założeniami w projekcie zostały zaimplementowane dwa sposoby reprezentacji grafu w pamięci komputera: macierz sąsiedztwa oraz lista sąsiadów.

Dodatkowo napisana funkcja do wczytywania grafu, z pliku który ma odpowiednią strukturę:

W pierwszej linii są kolejno zapisane parametry: liczba krawędzi, liczba wierzchołków, wierzchołek początkowy, wierzchołek końcowy. W następujących linijkach są dane dotyczące poszczególnych krawędzi: wierzchołek startowy, wierzchołek końcowy, waga krawędzi łączącej te wierzchołki.

Pomiar czasów wykonania algorytmów

Dla pomiaru czasów przyjęto następujące liczby wierzchołków: 500, 800, 1000, 1500, 2000. Zgodnie z wymaganiami projektowymi pomiary czasów zrobiono na następujących gęstościach grafu: 25%, 50%, 75%, 99%.

Gęstość grafu – stosunek liczby krawędzi grafu do maksymalnej liczby krawędzi. Maksymalna liczba krawędzi wynosi $\frac{V*(V-1)}{2}$, taka wartość jest osiągana w przypadku, gdy mamy graf pełny, czyli każdy wierzchołek jest bezpośrednio połączony z pozostałymi.

Następnie przed rozpoczęciem generowania grafu sprawdzamy jaki algorytm jest wybrany po to, żeby ustawić, czy graf ma być skierowany albo nie. Po czym był generowany graf oraz wykonywany pomiar czasu wybranego algorytmu.

Dodatkowo była napisana funkcja generowania randomowego grafu do tego, aby przeprowadzać pomiary czasu, ta funkcja przyjmuje trzy parametry: liczba wierzchołków, czy graf jest skierowany oraz liczba krawędzi, która jest potrzebne do tego, aby ustawić właściwą gęstość grafu.

Wyniki pomiarów

Wyszukiwanie minimalnego drzewa rozpinającego – Algorytm Prima

Macierz sąsiedztwa

gęstość	25%	50%	75%	99%
Liczba wierzch.	ms	ms	ms	ms
500	51	53	59	65
800	162	173	189	206
1000	260	297	335	357
1500	685	788	868	926
2000	1351	1578	1682	1886

Lista sąsiadów

gęstość	25%	50%	75%	99%
Liczba wierzch.	ms	ms	ms	ms
500	74	92	136	190
800	231	367	554	794
1000	436	726	1085	1554
1500	1377	2361	3601	5522
2000	2031	4523	5601	7238

Wyszukiwanie minimalnego drzewa rozpinającego – Algorytm Kruskala

Metoda tworzenia posortowanej listy krawędzi nie jest brana pod uwagę przy pomiarach czasu.

Macierz sąsiedztwa

gęstość	25%	50%	75%	99%
Liczba wierzch.	ms	ms	ms	ms
500	0,31	0,40	0,54	0,66
800	0,76	1,19	1,63	2,1
1000	1,13	1,98	2,72	3,81
1500	2,91	5,83	11,15	18,29
2000	6,16	10,92	18,23	29,51

Lista sąsiadów

gęstość	25%	50%	75%	99%
Liczba wierzch.	ms	ms	ms	ms
500	0,21	0,37	0,51	0,62
800	0,67	1,14	1,62	2,23
1000	1,01	1,94	2,61	3,63
1500	2,67	5,12	11,07	16,51
2000	5,89	11,21	17,56	29,23

Wyszukiwanie najkrótszej ścieżki w grafie – Algorytm Dijkstry

Macierz sąsiedztwa

gęstość	25%	50%	75%	99%
Liczba wierzch.	ms	ms	ms	ms
500	35,42	34,86	34,429	33,98
800	109,37	100,99	99,05	98,68
1000	182,29	176,03	173,21	173,27
1500	464,65	465,79	463,67	458,32
2000	933,42	933,17	937,02	926,09

Lista sąsiadów

gęstość	25%	50%	75%	99%
Liczba wierzch.	ms	ms	ms	ms
500	59,39	55,80	62,74	69,10
800	157,89	173,44	201,19	229,16
1000	262,62	303,39	361,03	415,99
1500	834,92	1191,55	1540,77	1877,53
2000	1975,54	2963,18	3953,08	4918,18

Wyszukiwanie najkrótszej ścieżki w grafie – Algorytm Bellmana – Forda

Macierz sąsiedztwa

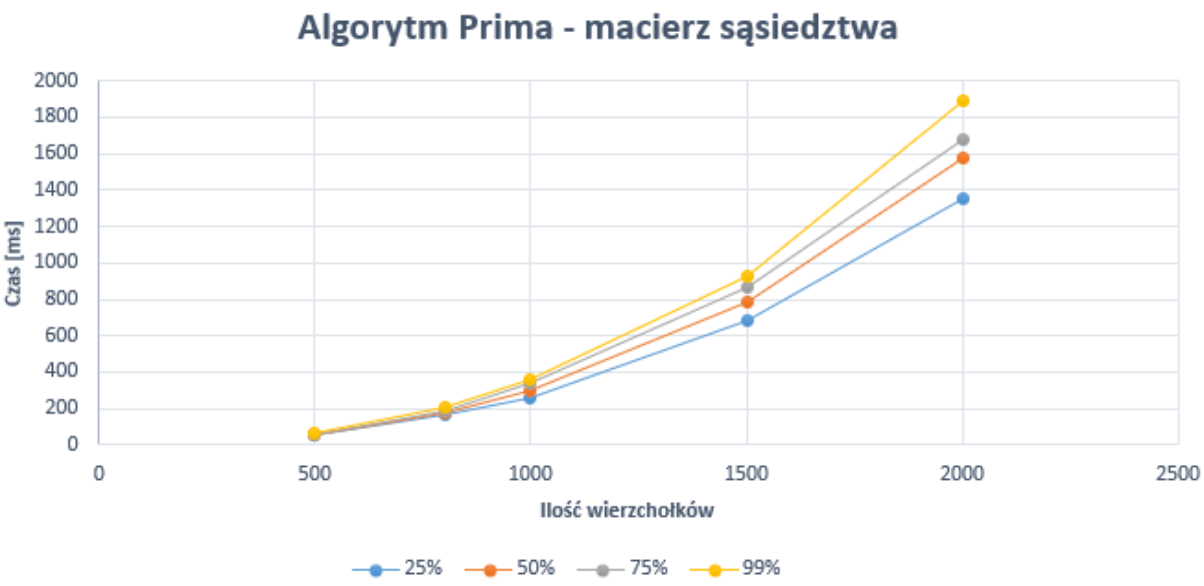
gęstość	25%	50%	75%	99%
Liczba wierzch.	ms	ms	ms	ms
500	33,34	65,89	99,89	128,75
800	142,37	284,57	462,48	600,01
1000	290,94	569,73	975,62	1392,12
1500	1709,46	3585,14	5461,93	7348,11
2000	7352,24	14525,51	22150,91	28167,33

Lista sąsiadów

gęstość	25%	50%	75%	99%
Liczba wierzch.	ms	ms	ms	ms
500	338,62	695,12	1040,22	1388,93
800	2207,11	4522,32	6783,11	8921,22
1000	11567,41	14112,23	18541,91	20410,98
1500	22567,12	24921,86	29132,56	33392,56
2000	36512,33	40211,98	43956,76	48512,67

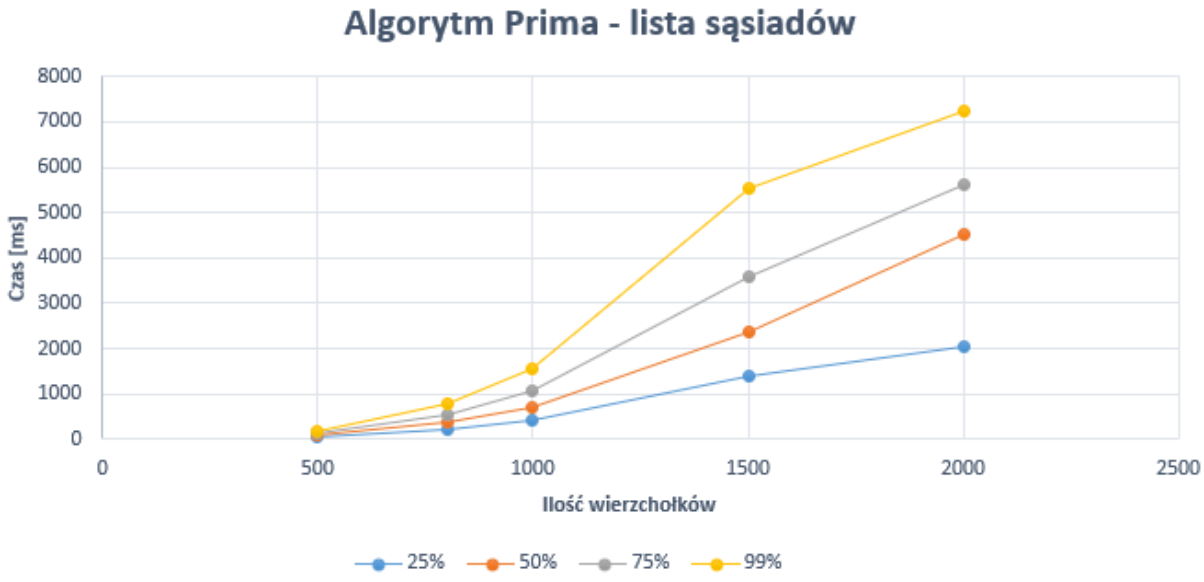
Wykresy

Algorytm Prima – Macierz sąsiedztwa



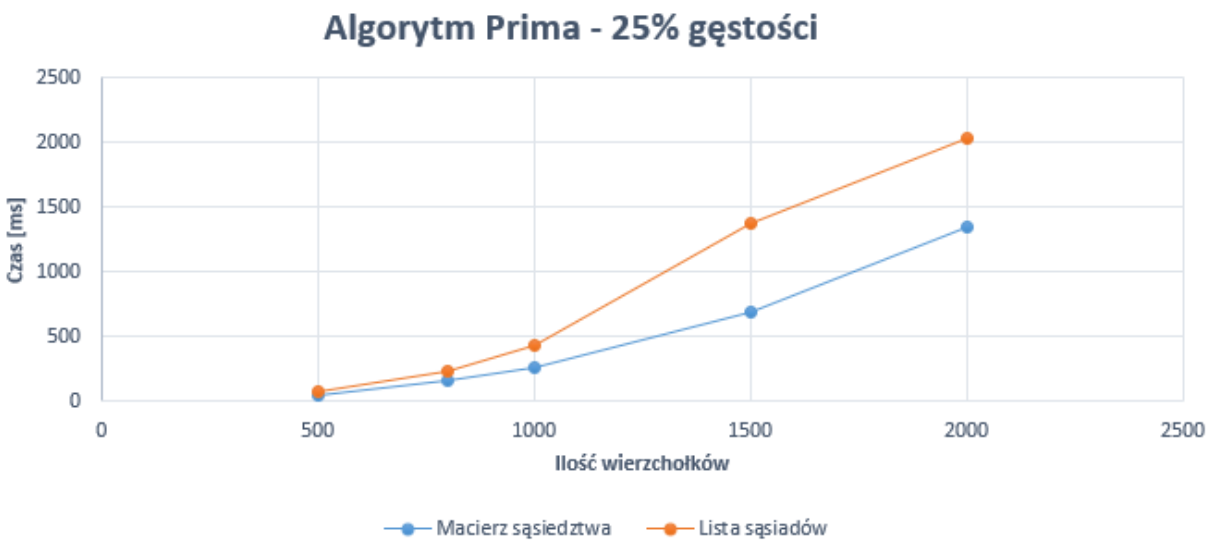
Rysunek 1. Wyniki pomiarów macierz sąsiedztwa

Algorytm Prima – Lista sąsiadów



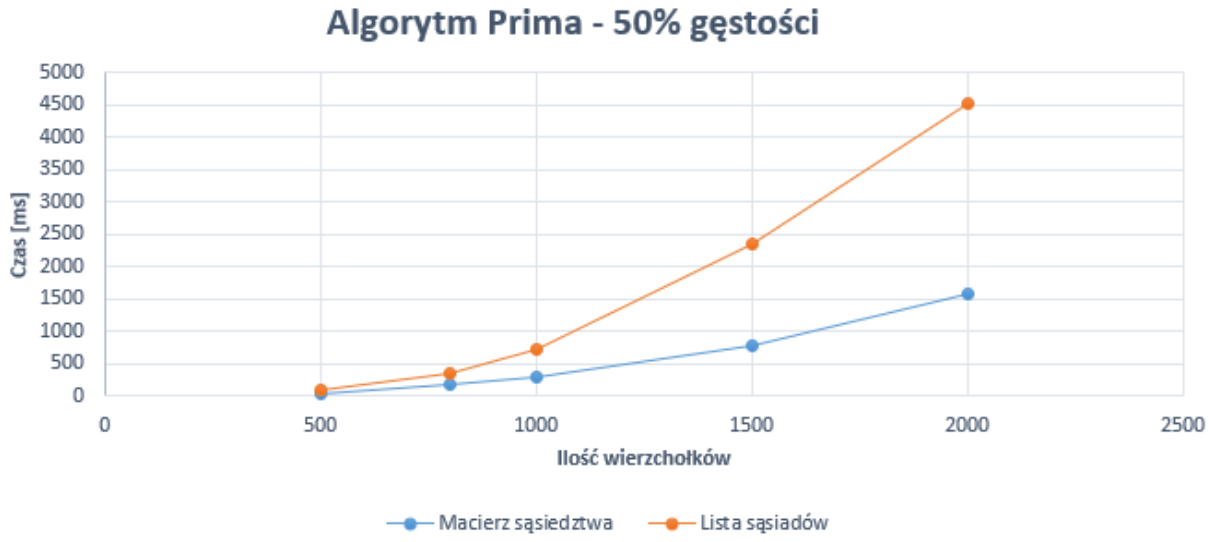
Rysunek 2. Wyniki pomiarów lista sąsiadów

Graf o gęstości 25%



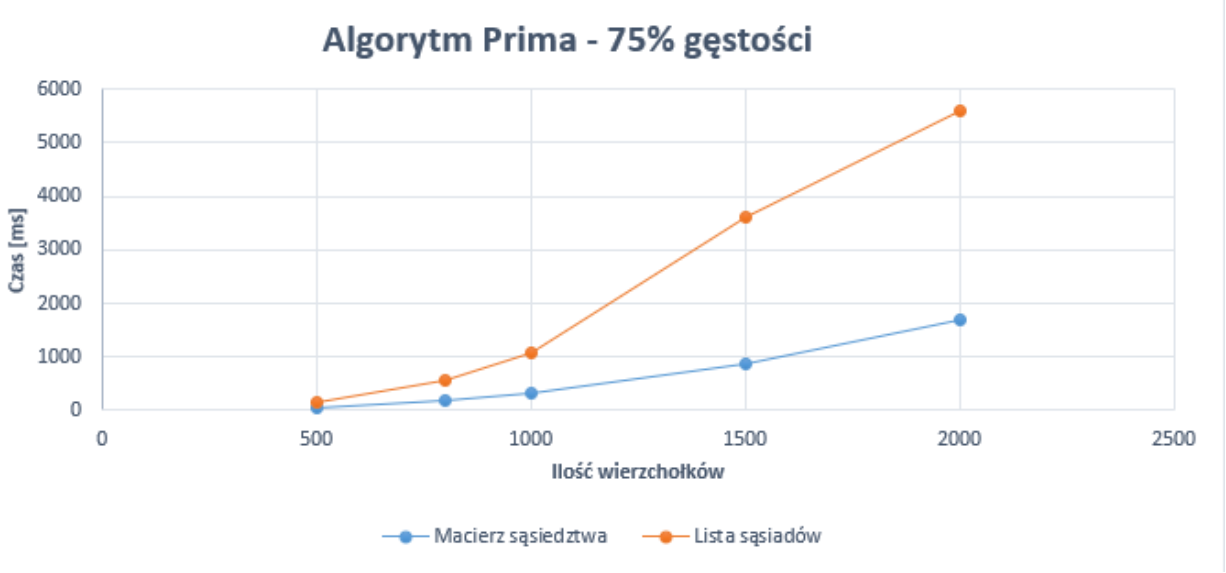
Rysunek 3. Gęstość grafu 25%

Graf o gęstości 50%



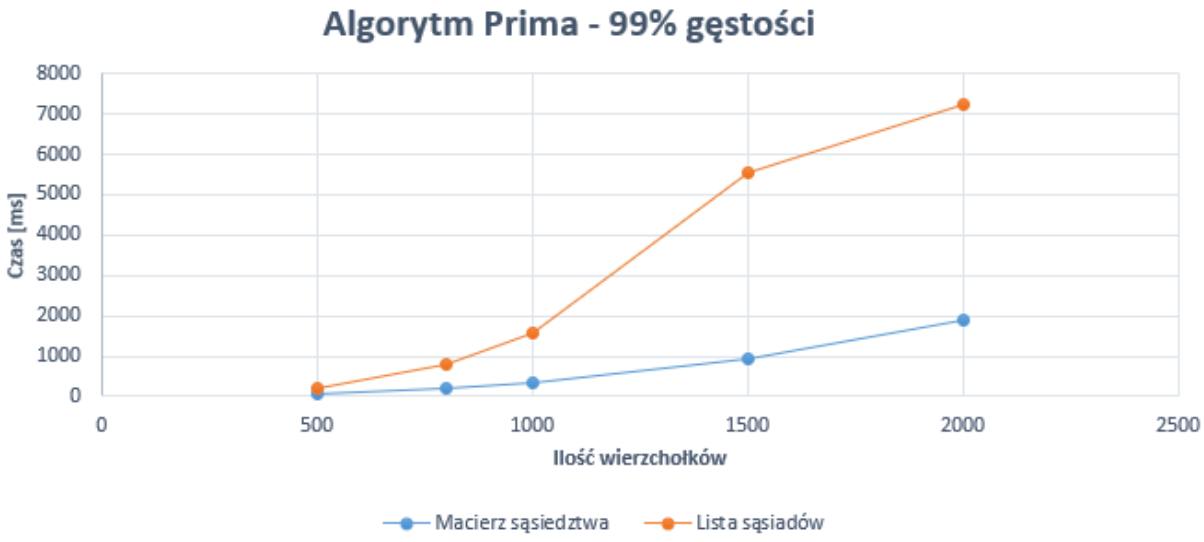
Rysunek 4. Gęstość grafu 50%

Graf o gęstość 75%



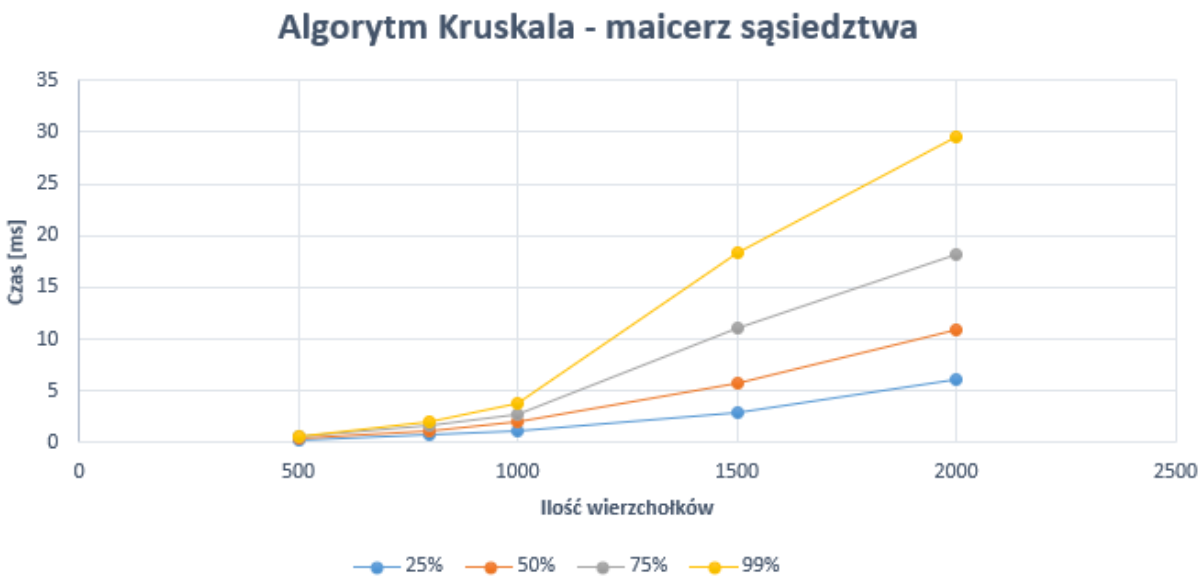
Rysunek 5. Gęstość grafu 75%

Graf o gęstości 99%



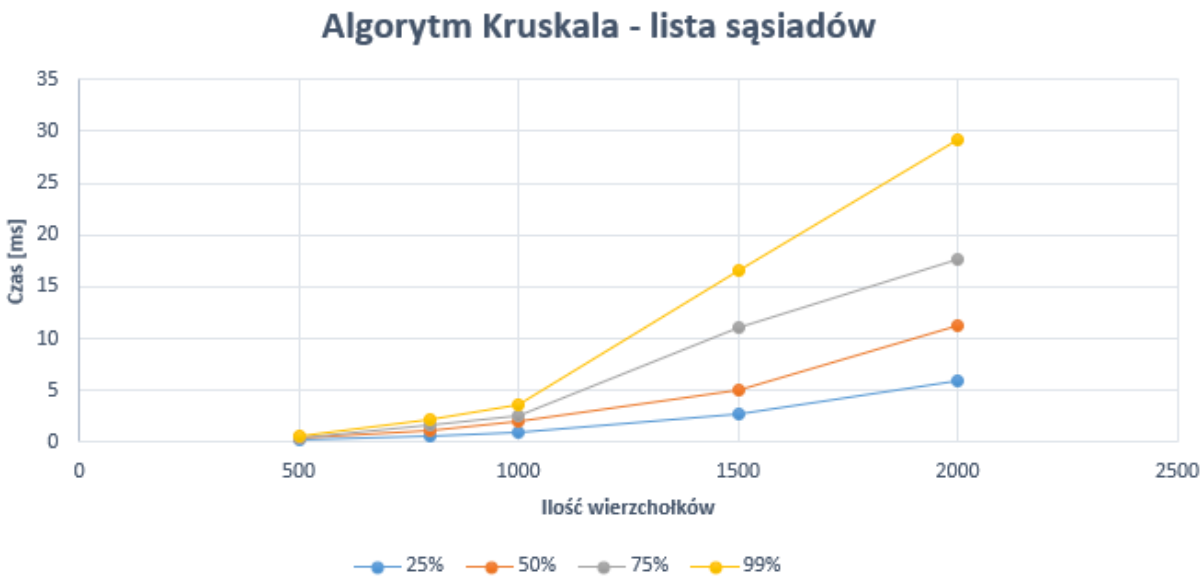
Rysunek 6. Gęstość grafu 99%

Algorytm Kruskala – Macierz sąsiedztwa



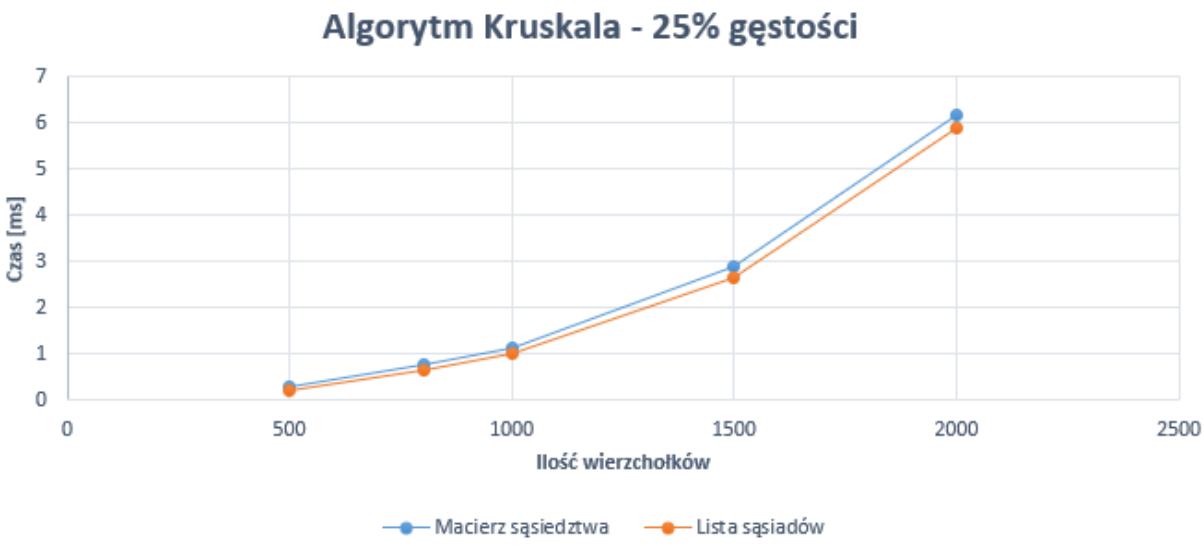
Rysunek 7. Wyniki pomiarów macierz sąsiedztwa

Algorytm Kruskala – Lista sąsiadów



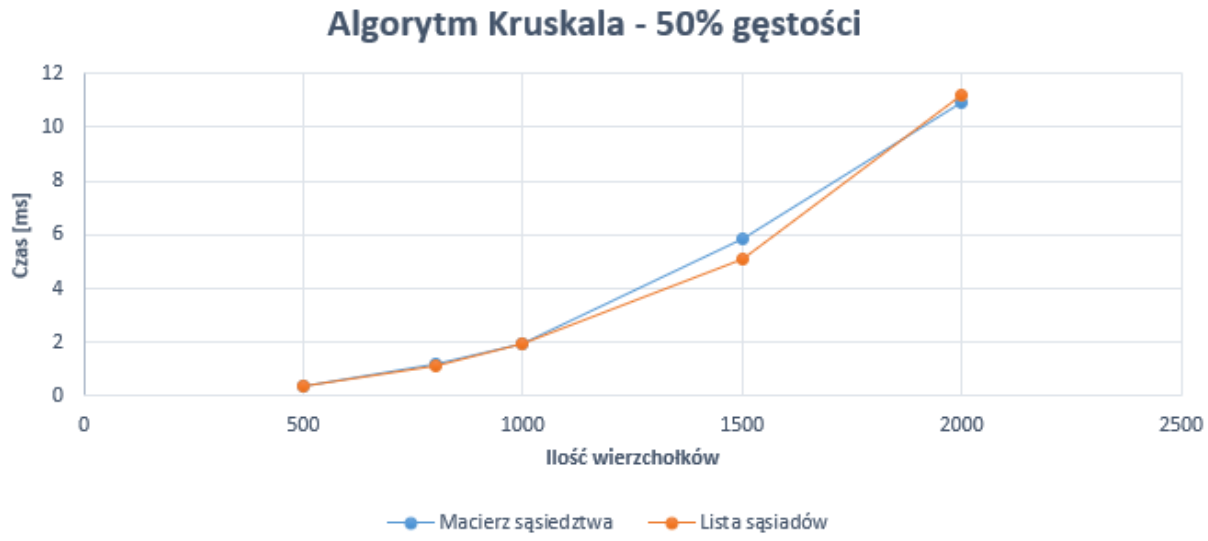
Rysunek 8. Wyniki pomiarów lista sąsiadów

Graf o gęstości 25%



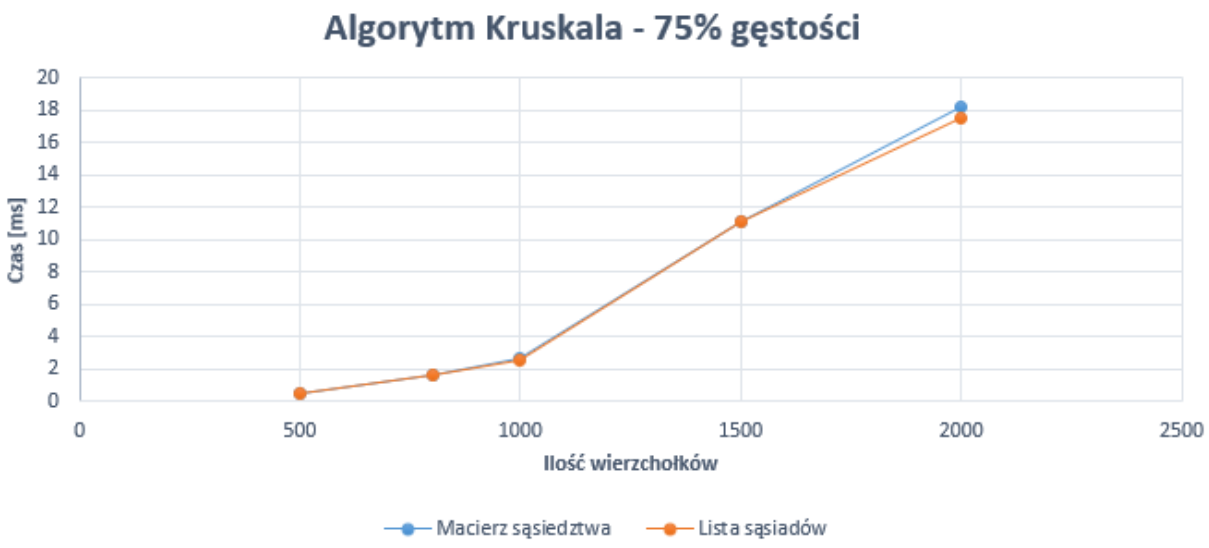
Rysunek 9. Gęstość grafu 25%

Graf o gęstości 50%



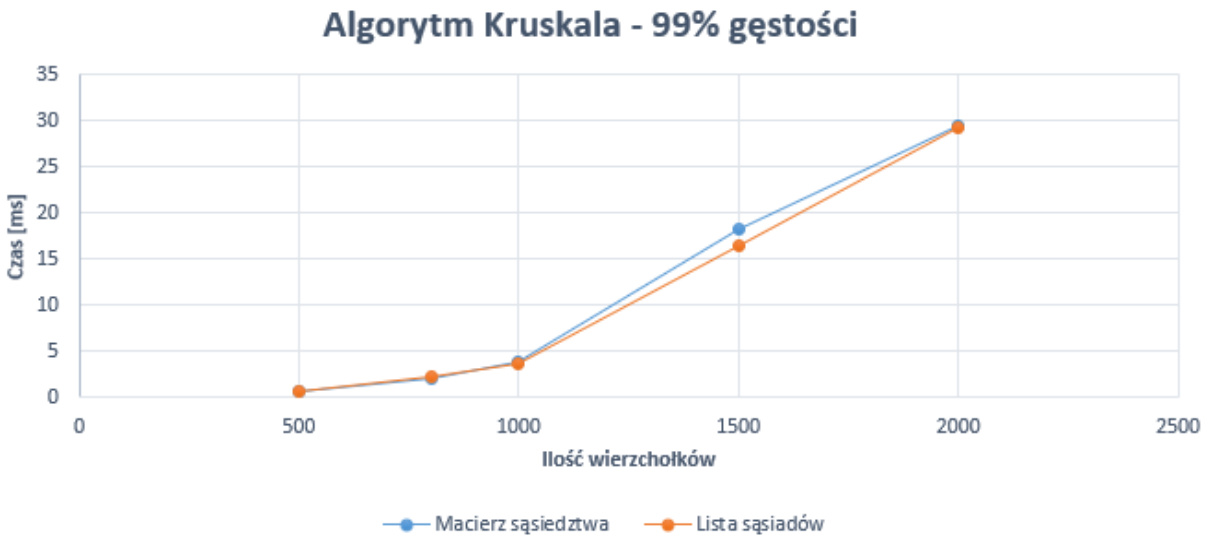
Rysunek 10. Gęstość grafu 50%

Graf o gęstości 75%



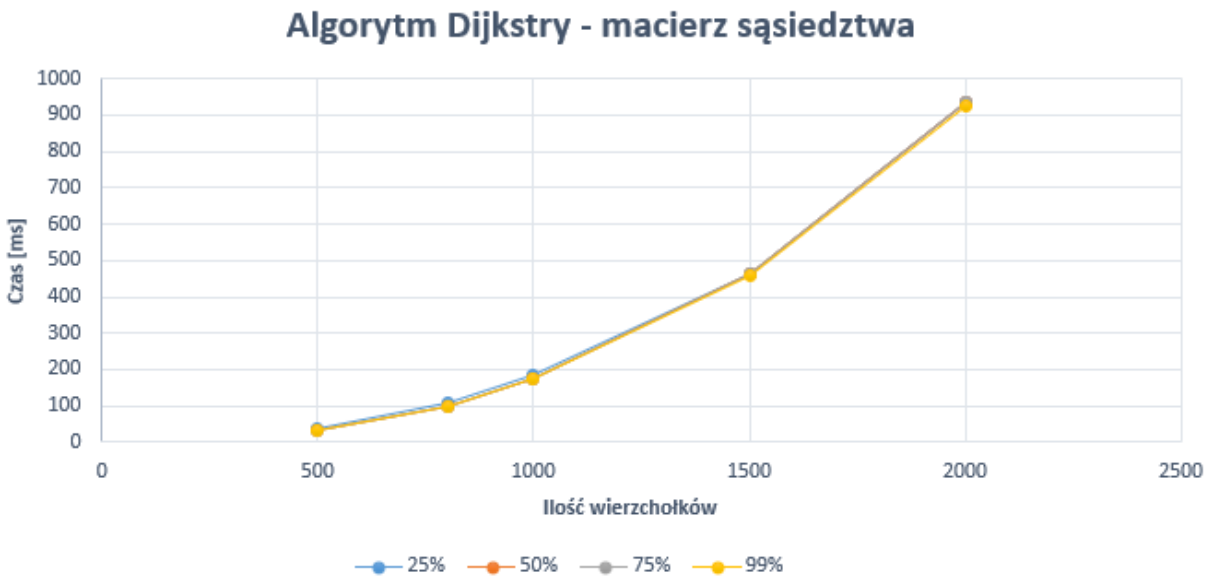
Rysunek 11. Gęstość grafu 75%

Graf o gęstości 99%



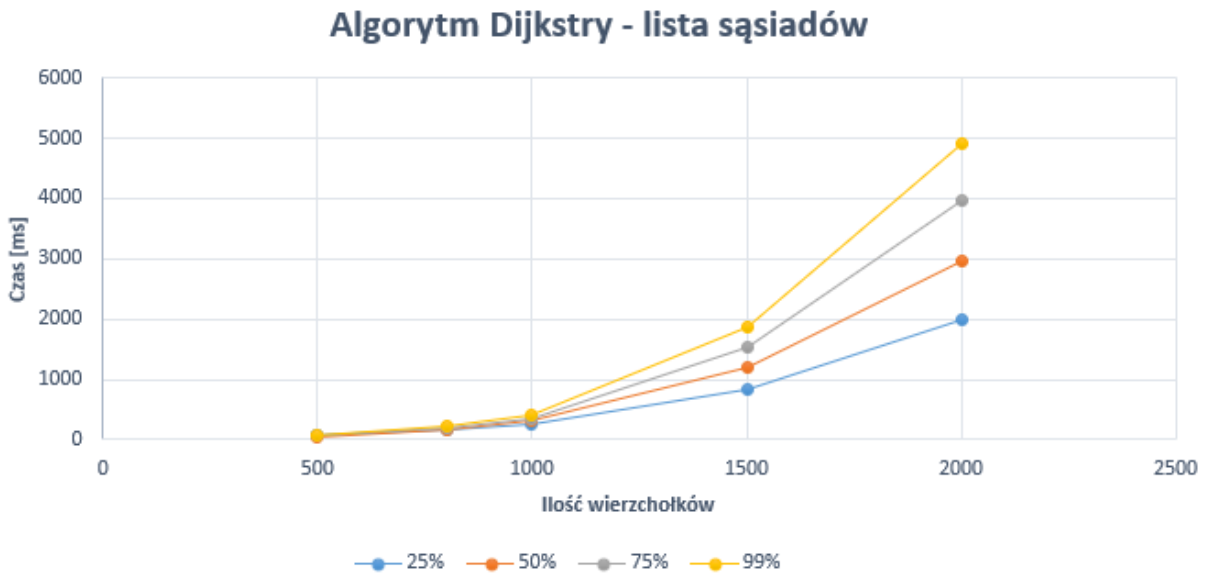
Rysunek 12. Gęstość grafu 99%

Algorytm Dijkstry – Macierz sąsiedztwa



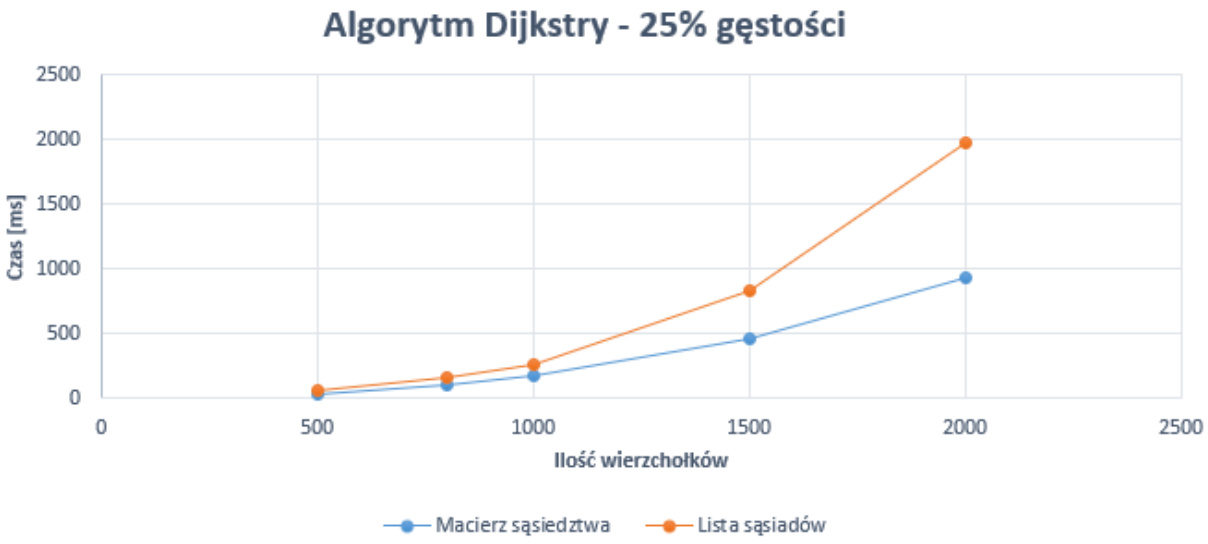
Rysunek 13. Wyniki pomiarów macierz sąsiedztwa

Algorytm Dijkstry – Lista sąsiadów



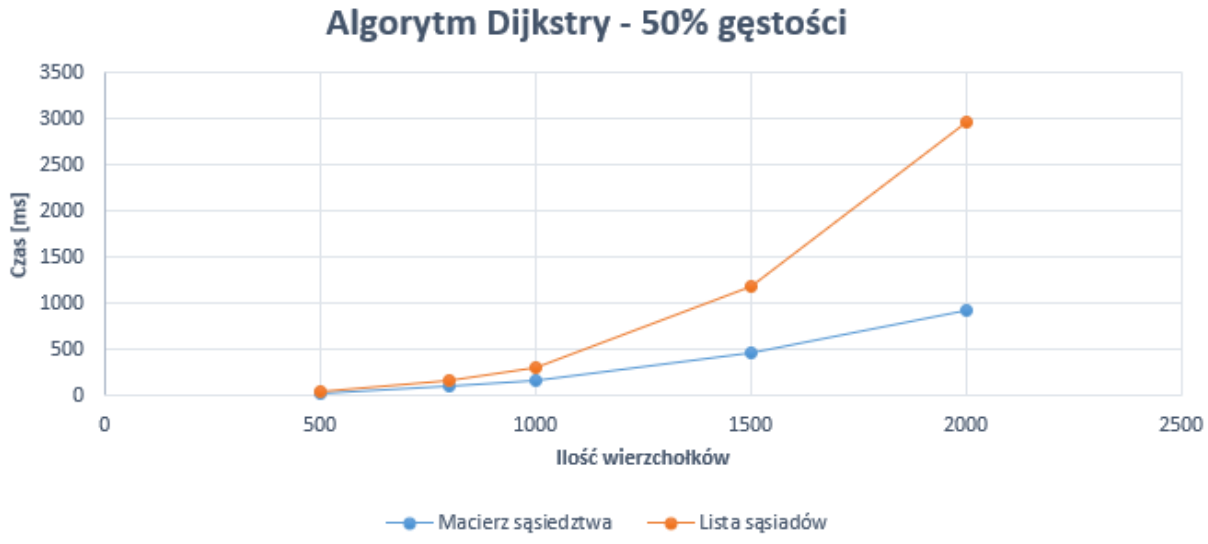
Rysunek 14. Wyniki pomiarów lista sąsiadów

Graf o gęstości 25%



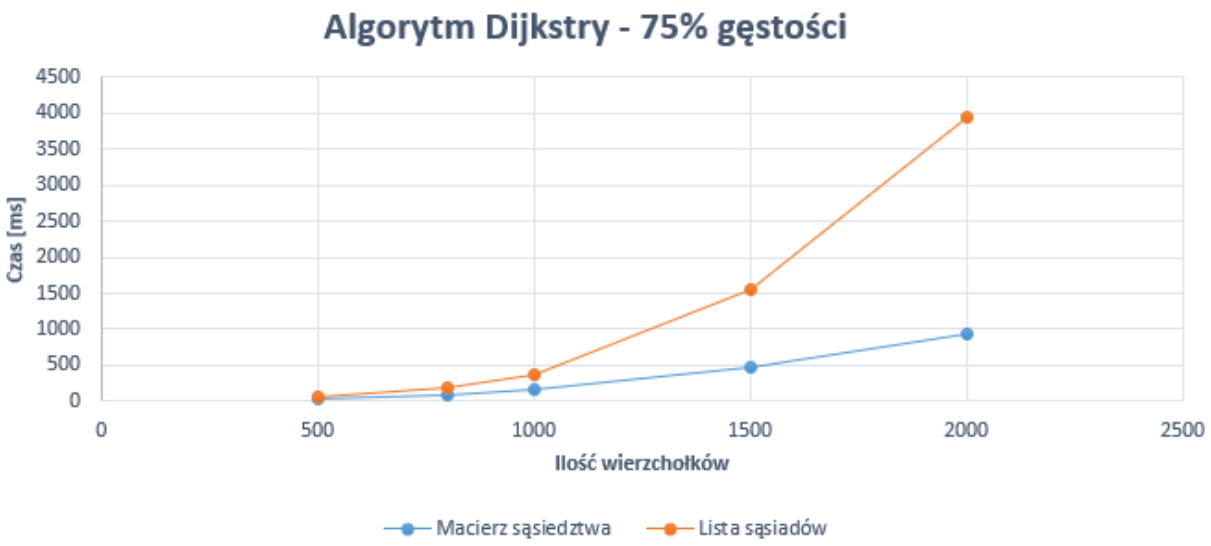
Rysunek 15. Gęstość grafu 25%

Graf o gęstości 50%



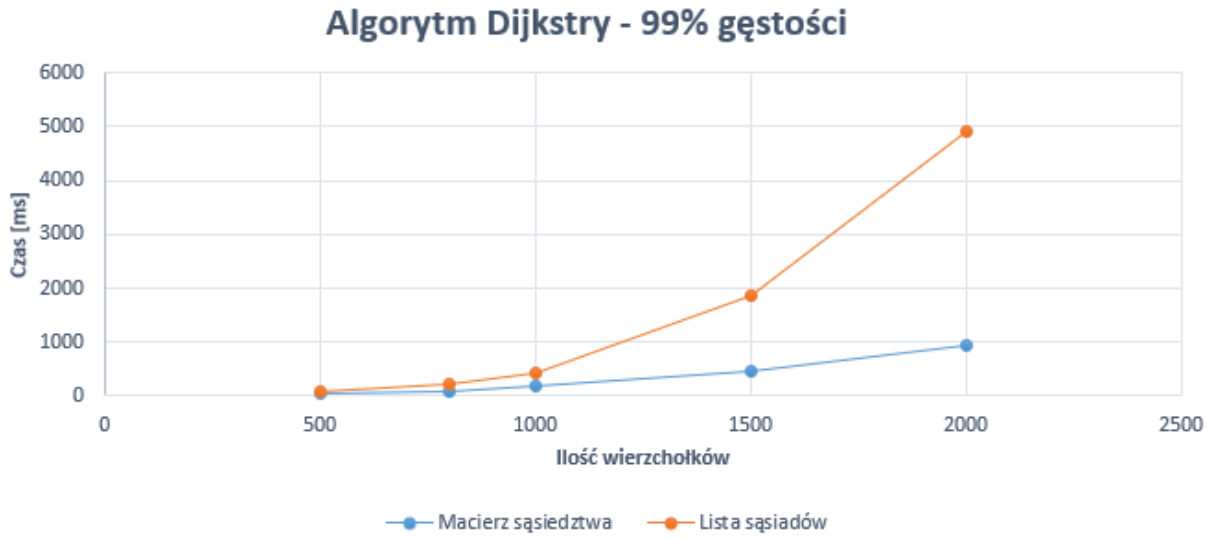
Rysunek 16. Gęstość grafu 50%

Graf o gęstości 75%



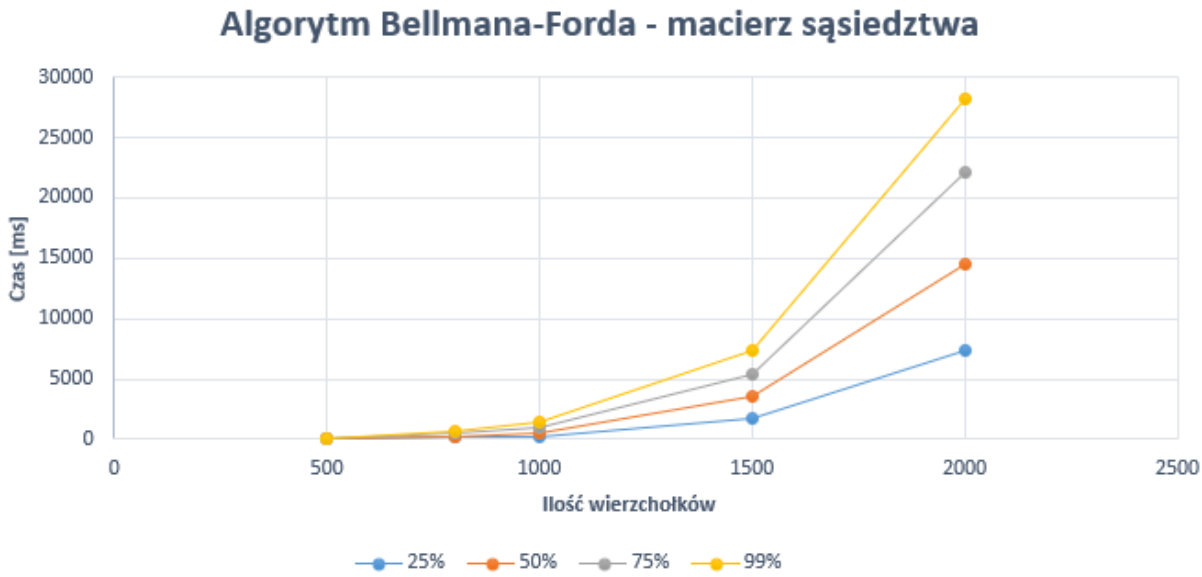
Rysunek 17. Gęstość grafu 75%

Graf o gęstości 99%



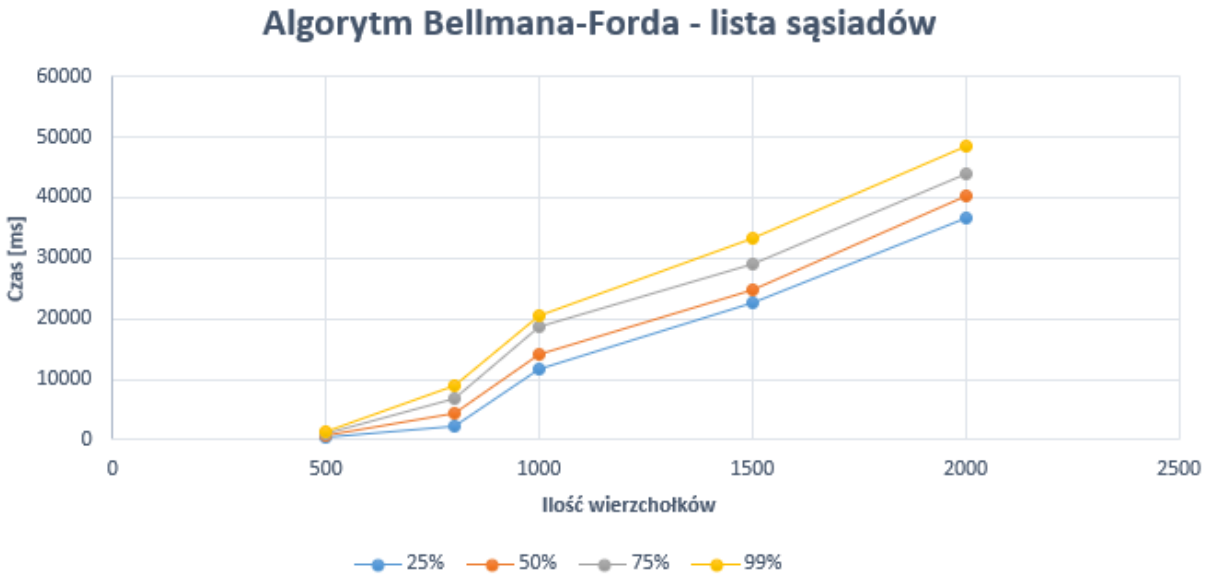
Rysunek 18. Gęstość grafu 99%

Algorytm Bellmana – Forda – Macierz sąsiedztwa



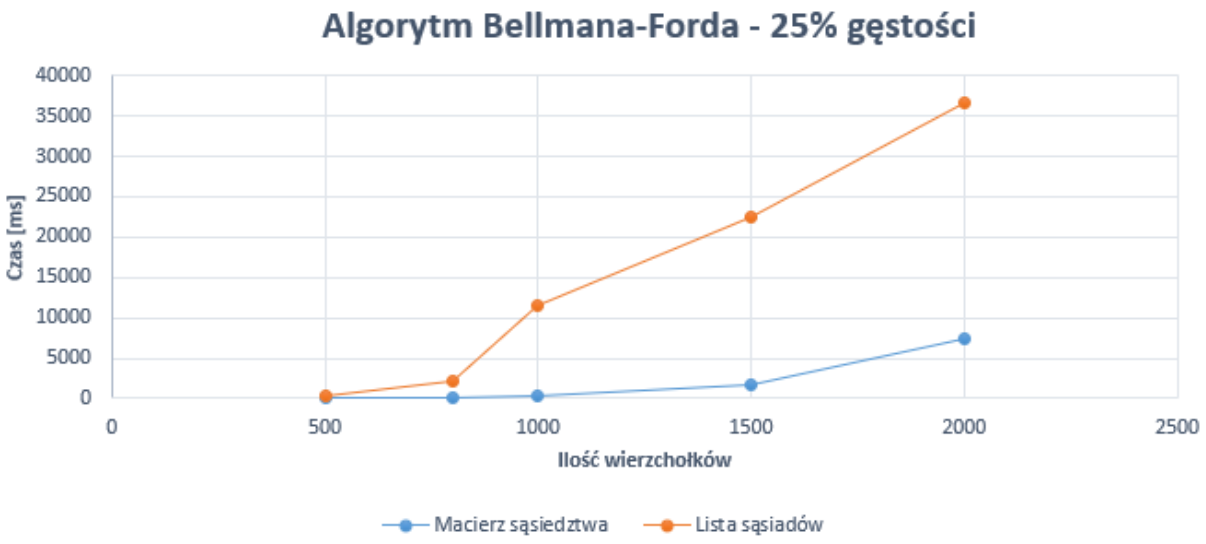
Rysunek 19. Wyniki pomiarów macierz sąsiedztwa

Algorytm Bellmana – Forda – Lista sąsiadów



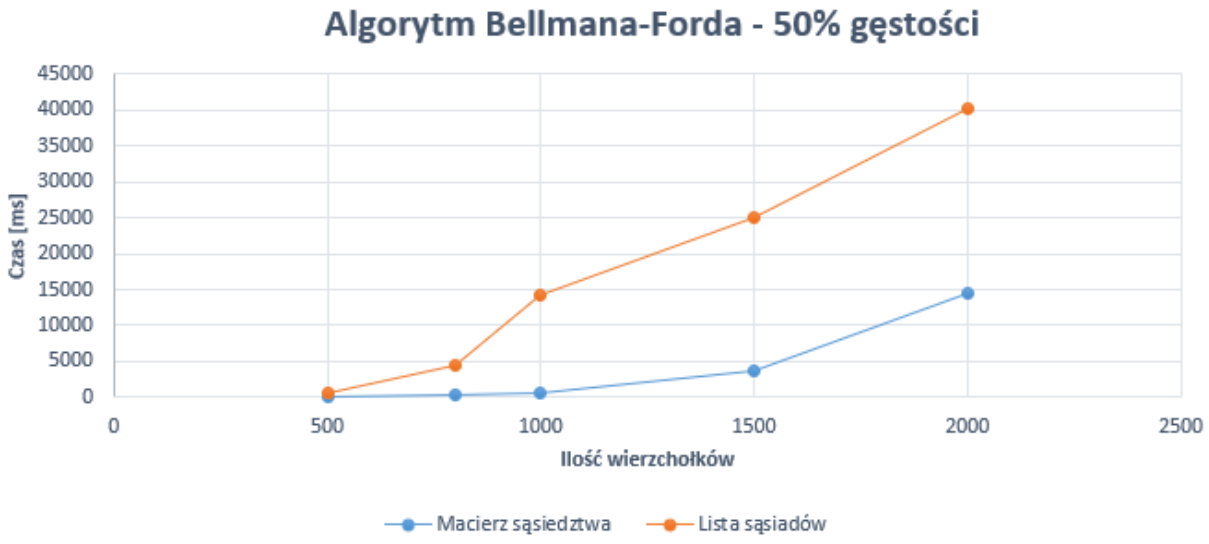
Rysunek 20. Wyniki pomiarów lista sąsiadów

Graf o gęstości 25%



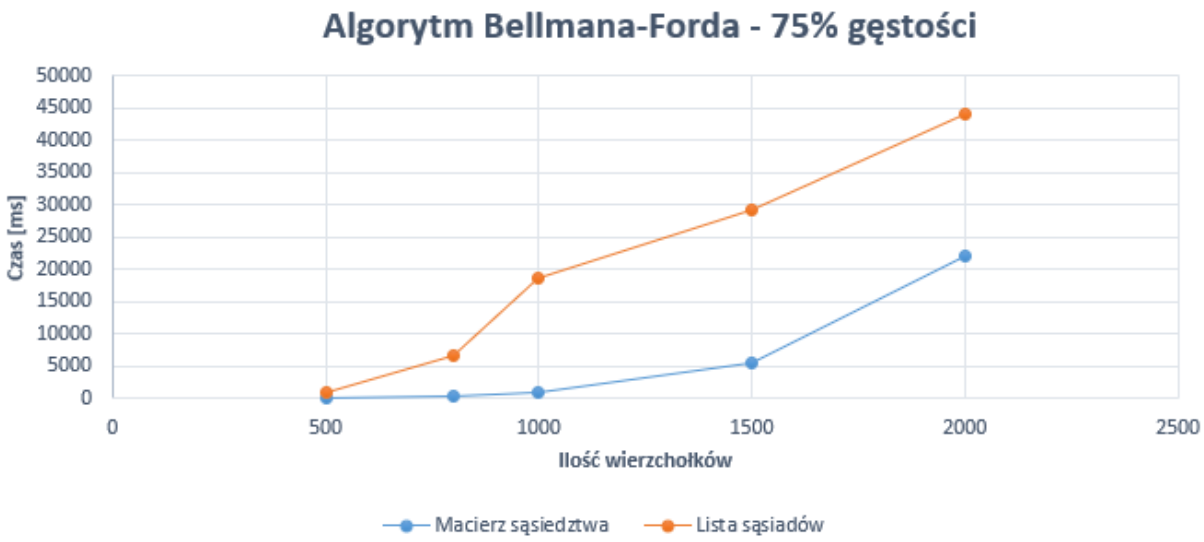
Rysunek 21. Gęstość grafu 25%

Graf o gęstości 50%



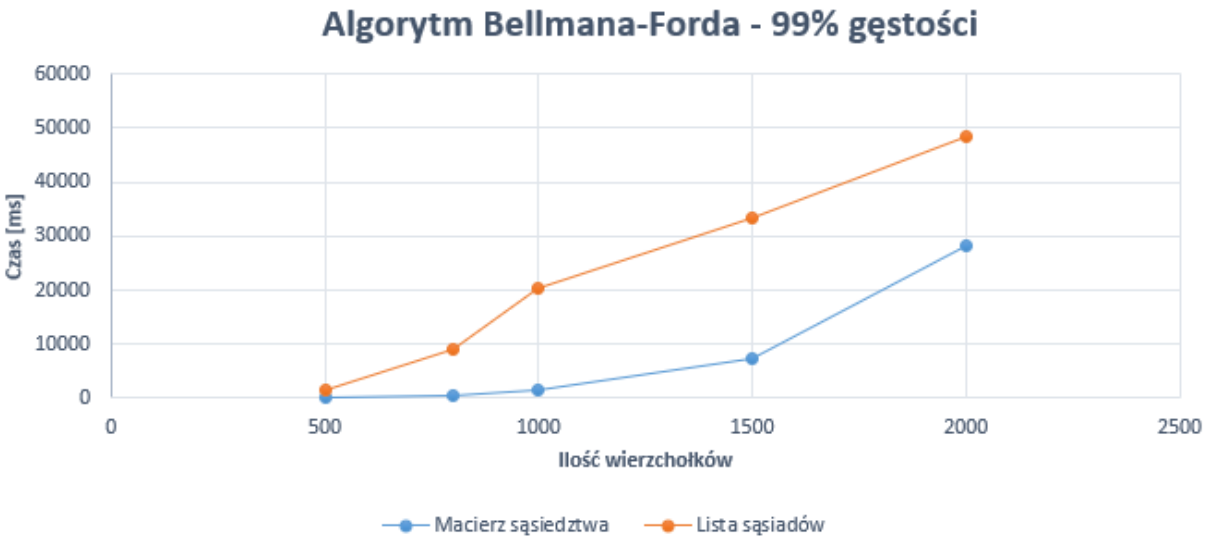
Rysunek 22. Gęstość grafu 50%

Graf o gęstości 75%



Rysunek 23. Gęstość grafu 75%

Graf o gęstości 99%



Rysunek 24. Gęstość grafu 99%

Wnioski

We wszystkich algorytmach reprezentacja grafu w postaci macierzy sąsiedztwa daje o wiele lepsze rezultaty niż lista sąsiadów. Wynika najprawdopodobniej z tego faktu, że do znalezienia wartości wagi krawędzi, w przypadku macierzy sąsiedztwa, znając wierzchołek początkowy i końcowy szukanej krawędzi, wystarczy po prostu wziąć wartość z tablicy dwuwymiarowej o odpowiednich indeksach. Natomiast w przypadku listy sąsiadów musimy kolejno przejść w najgorszym przypadku przez wszystkich sąsiadów co jednoznacznie potrzebuje więcej czasu niż macierz sąsiedztwa. Również zauważyłem, że Algorytm Bellmana – Forda potrzebował o kilka razy więcej czasu niż algorytm Dijkstry, moim zdaniem spowodowano to tym, że w tym algorytmie musimy przy każdej iteracji sprawdzać zawsze nasze wszystkie krawędzi za co musimy zapłacić większym czasem. Zauważyłem również, że dla algorytmu Dijkstry czas wykonania algorytmu nie różni się dla określonej liczby wierzchołków, czyli mając określoną ilość wierzchołków niezależnie od tego jaką mamy gęstość czas jest prawie taki sam. (rys. 13).

Materiały

[1] https://pl.wikipedia.org/wiki/Algorytm_Prima

[2] https://pl.wikipedia.org/wiki/Algorytm_Dijkstry

[3] https://pl.wikipedia.org/wiki/Algorytm_Bellmana-Forda

[4] https://pl.wikipedia.org/wiki/Algorytm_Kruskala

[5] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford *Wprowadzenie do algorytmów*.