Wrocław, 03.03.18

Prowadzący: dr inż. Zbigniew Buchalski
Filip Mazur 226018

Struktury Danych i Złożoność Obliczeniowa

Sprawozdanie - Projekt nr 2

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

1. Wprowadzenie

W projekcie należało zaimplementować wybrane algorytmy grafowe, oraz dokonać pomiaru czasu ich wykonania.

Problemy, które rozpatrzono to:

1) Wyznaczanie Minimalnego Drzewa Rozpinającego – MST
2) Wyznaczanie najkrótszej ścieżki w grafie

Oba algorytmy miały zostać wykonane dla postaci macierzowej(macierz sąsiedztwa) oraz listowej(lista sąsiedztwa). W moim projekcie do zbadania problemów wykorzystałem algorytmy Prima – dla MST, oraz Dijkstry – dla najkrótszej ścieżki.

Reprezentacje Grafu

Graf w postaci Listy Sąsiedztwa

Jest to tablica jednowymiarowa zawierająca n innych list, gdzie n to liczba wierzchołków. Indeksy tablicy są wierzchołkami początkowymi, natomiast w listach znajdują się wszystkie wierzchołki sąsiadujące z danym. Kolejność sąsiadów nie ma znaczenia.

Graf w postaci Macierzy Sąsiedztwa

Jest to tablica dwuwymiarowa n x n gdzie n jest ilością wierzchołków w grafie. Macierz sąsiedztwa wypełnia się następująco:

0 – Gdy pomiędzy wierzchołkami nie ma krawędzi

1 (lub waga krawędzi) – Gdy pomiędzy wierzchołkami rozpięta jest krawędź.

Wykorzystane Algorytmy:

Algorytm Prima używany jest do znajdowania Minimalnego Drzewa rozpinającego w grafie. Na początku, algorytm dodaje do zbioru reprezentującego drzewo krawędź o najmniejszej wadze, łączącą wierzchołek początkowy z dowolnym wierzchołkiem. W każdym kolejnym kroku procedura dodaje do zbioru najlżejszą krawędź wśród krawędzi łączących wierzchołki już odwiedzone z nieodwiedzonymi.
Złożoność obliczeniowa Algorytmu Prima to: O(E * log V), gdzie E to ilość krawędzi, a V ilość wierzchołków.

Algorytm Dijkstry w spójnym grafie ważonym ma za zadanie znaleźć najkrótsze ścieżki od wierzchołka wybranego do wszystkich pozostałych wyliczając również koszty dojścia czyli sumy wag krawędzi w danej ścieżce. Złożoność algorytmu Dijkstry to: O(E * log V), gdzie E to ilość krawędzi, a V ilość wierzchołków. Algorytm Dijkstry nie powinien być stosowany w przypadku gdy mamy ujemne wagi krawędzi.

Pomiar Czasu

Do pomiaru czasu wykorzystano funkcję:
QueryPerformanceCounter(_otut LARGE_INTEGER *lpPerformanceCount);
Pomiar czasu przeprowadzono w mikrosekundach(ms) po czym zamieniono je na nanosekundy(ns).

Pomiary

Pomiary wykonano dla grafów 50, 100, 150 i 200 elementowych o gęstościach 25, 50, 75 oraz 100 procent
Uśrednione wyniki wszystkich pomiarów przedstawiają poniższe tabele oraz wykresy.

2. Część eksperymentalna

2.1. Wyniki Testów

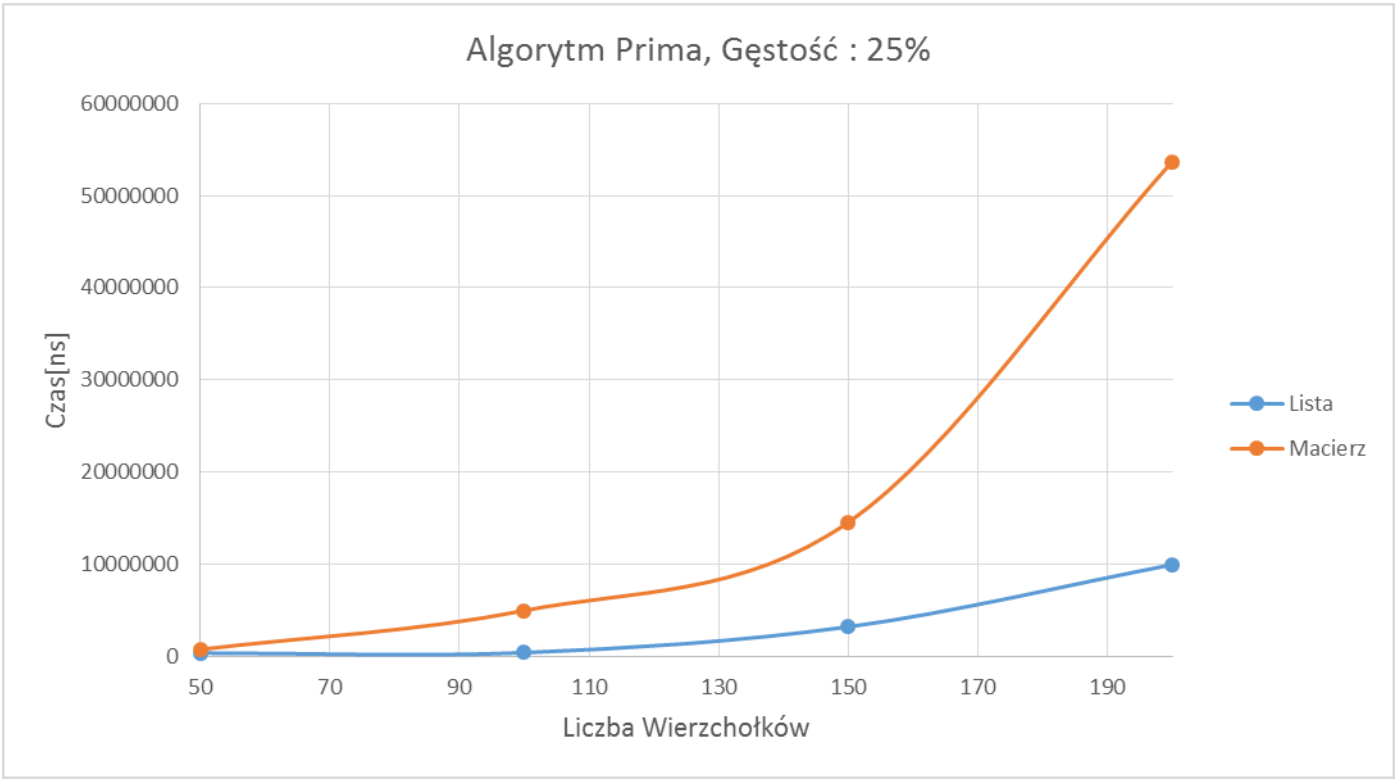
2.1.1 Algorytm Prima

Liczba Wierzchołków	Gęstość	Lista Czas[ns]
50	25	360439
	50	787463
	75	1568925
	99	1541591
100	25	401824
	50	932543
	75	2749946
	99	3747156
150	25	3209255
	50	3555508
	75	4122559
	99	6792534
200	25	9953470
	50	16065763
	75	10989656
	99	14927177

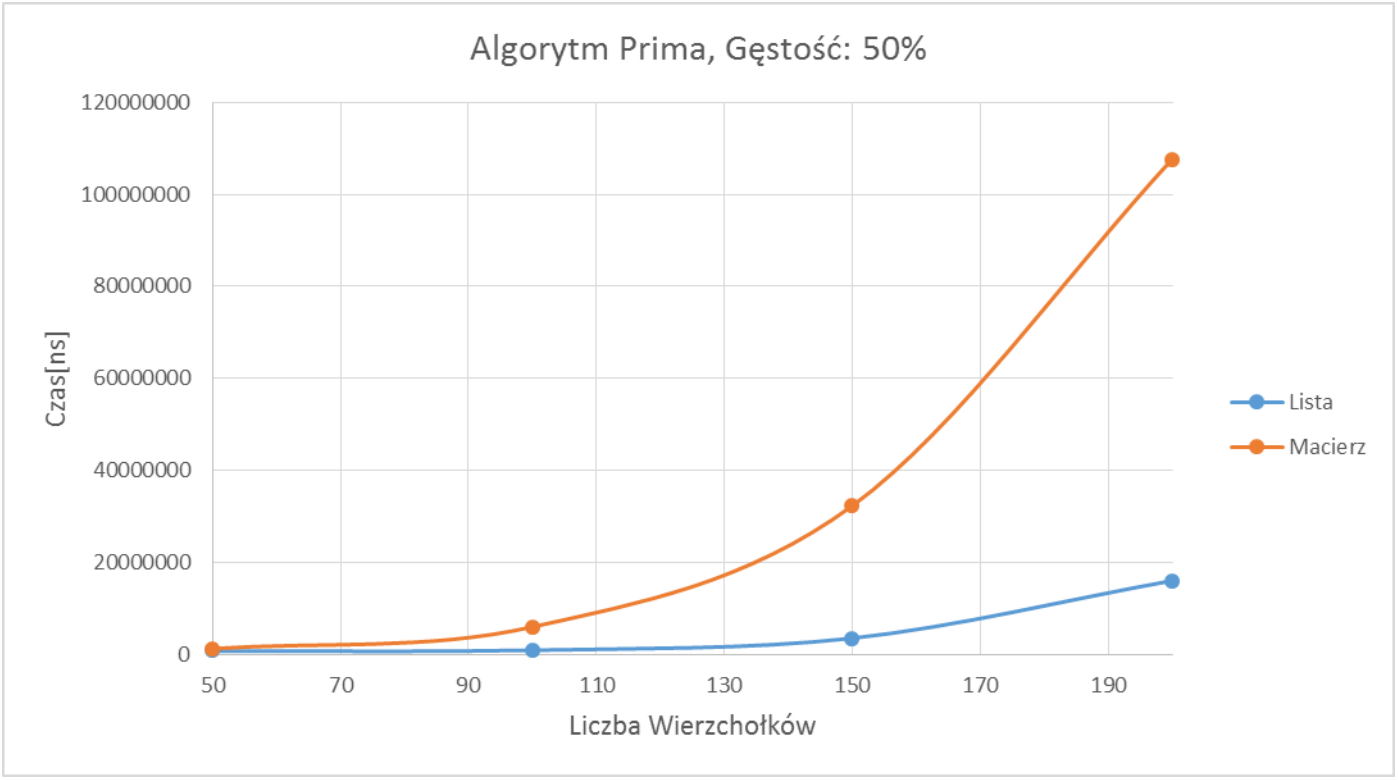
Tabela 1. Uśrednione wyniki dla poszczególnych ilości wierzchołków i gęstości na liście sąsiedztwa

Liczba Wierzchołków	Gęstość	Macierz Czas[ns]
50	25	758550
	50	1259848
	75	1463007
	99	2481380
100	25	4965496
	50	5999183
	75	6787388
	99	7478617
150	25	14551176
	50	32332536
	75	44203668
	99	59786253
200	25	53700866
	50	107684981
	75	161890611
	99	180503413

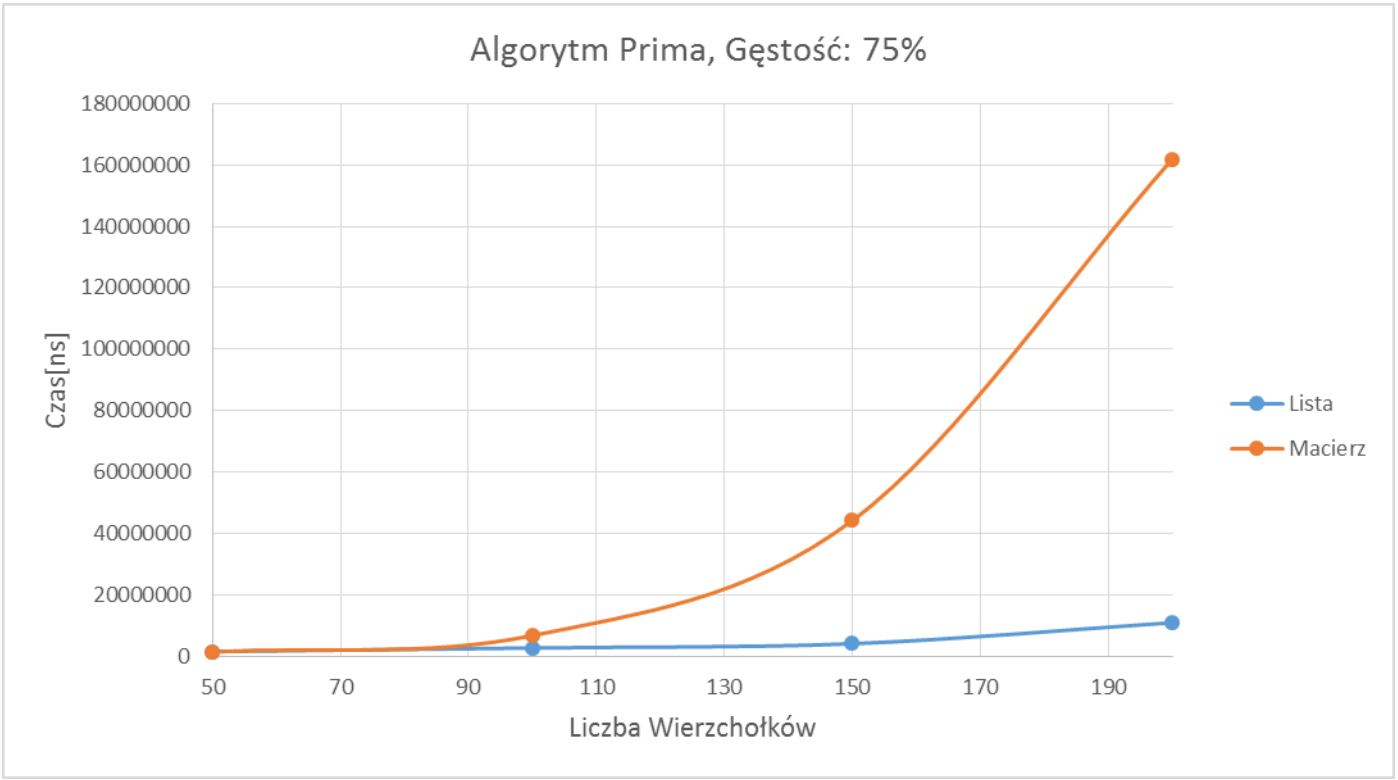
Tabela 2. Uśrednione wyniki dla poszczególnych ilości wierzchołków i gęstości w macierzy sąsiedztwa



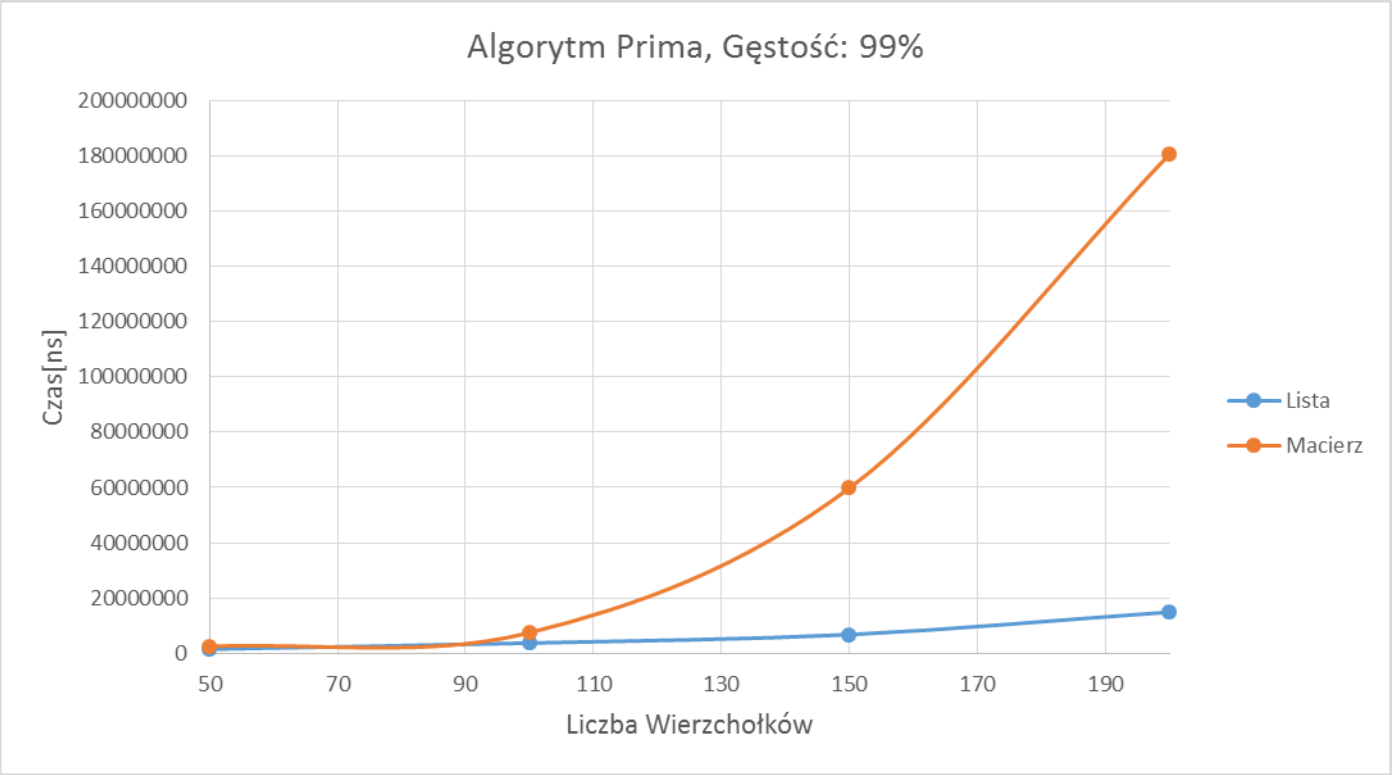
Wykres 1. Zależność czasu potrzebnego na przeprowadzenie algorytmu od ilości wierzchołków dla 25% gęstości



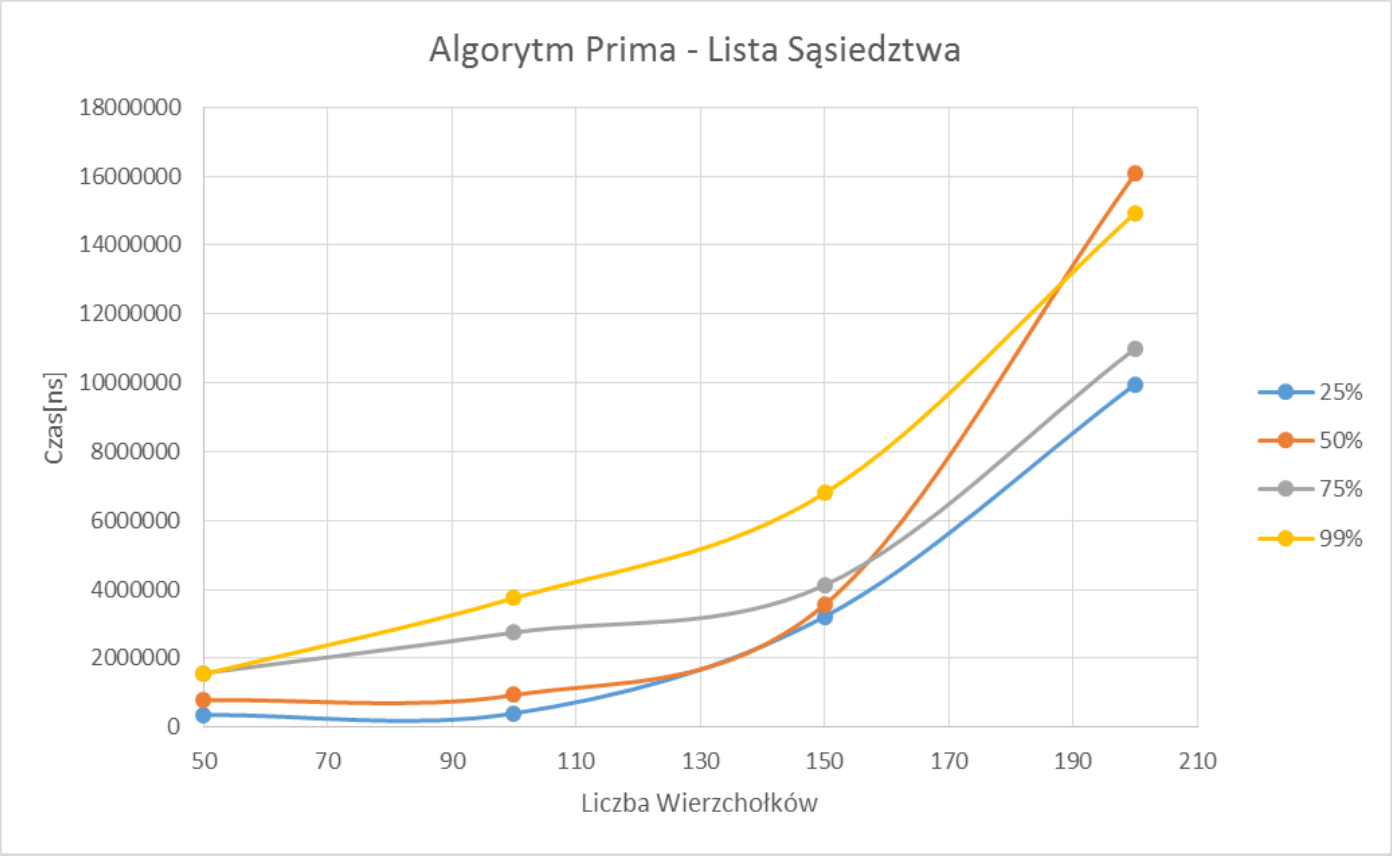
Wykres 2. Zależność czasu potrzebnego na przeprowadzenie algorytmu od ilości wierzchołków dla 50% gęstości



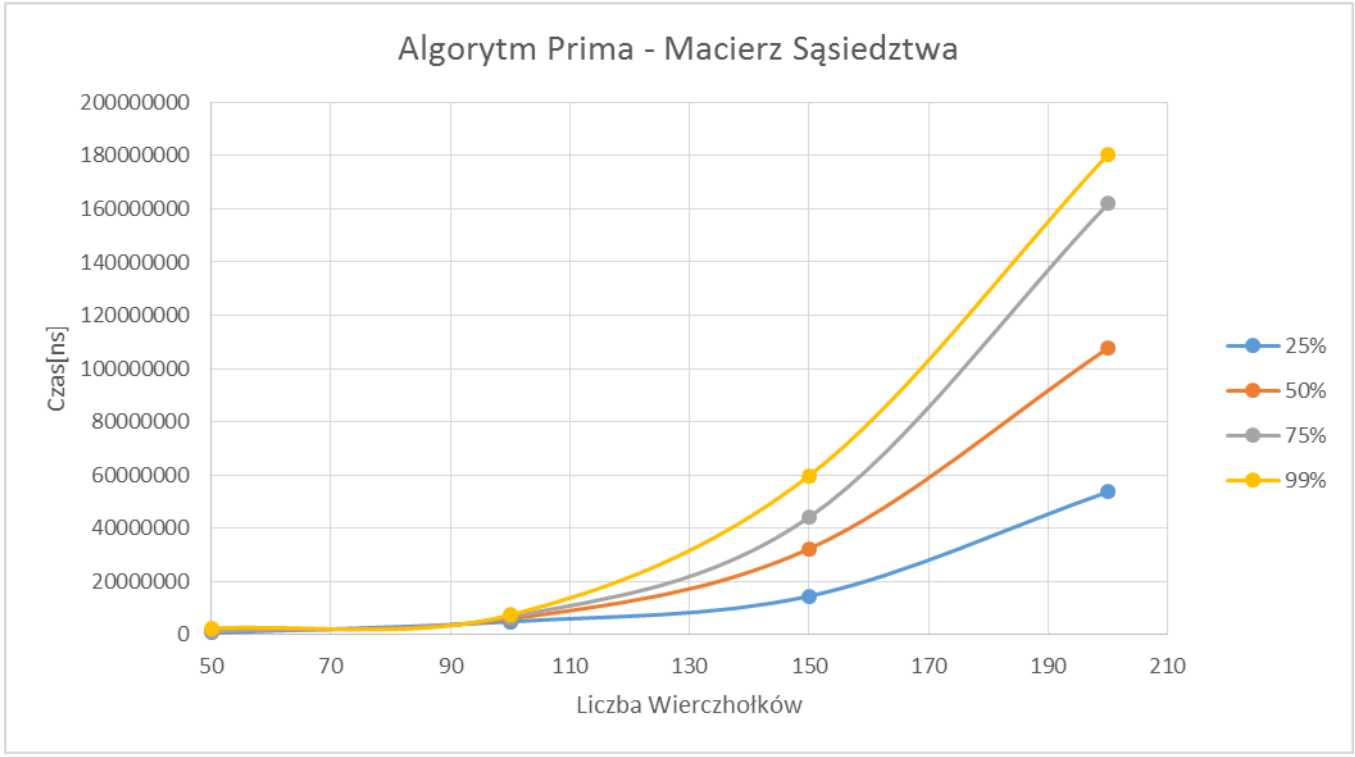
Wykres 3. Zależność czasu potrzebnego na przeprowadzenie algorytmu od ilości wierzchołków dla 75% gęstości



Wykres 4. Zależność czasu potrzebnego na przeprowadzenie algorytmu od ilości wierzchołków dla 99% gęstości



Wykres 5. Zestawienie zależności(dla poszczególnych gęstości grafów) czasów potrzebnych na przeprowadzenie algorytmu od ilości wierzchołków dla Listy Sąsiedztwa



Wykres 6. Zestawienie zależności(dla poszczególnych gęstości grafów) czasów potrzebnych na przeprowadzenie algorytmu od ilości wierzchołków dla Macierzy Sąsiedztwa

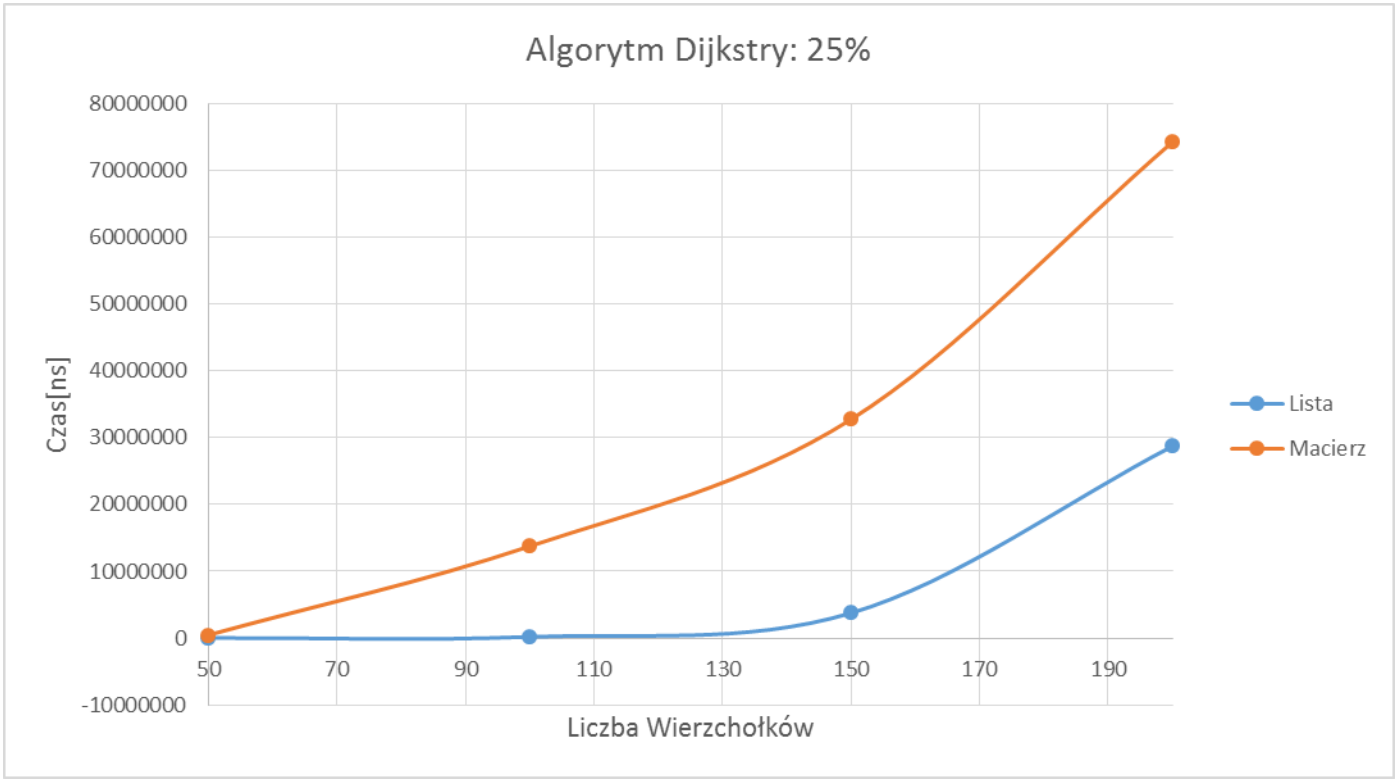
2.1.2. Algorytm Dijkstry

Liczba Wierzchołków	Gęstość	Lista Czas[ns]
50	25	31975
	50	387430
	75	54510
	100	88932
100	25	199637
	50	2088570
	75	383772
	99	657203
150	25	3759270
	50	4047310
	75	1168305
	99	2283985
200	25	28713480
	50	16705230
	75	2310269
	99	3810265

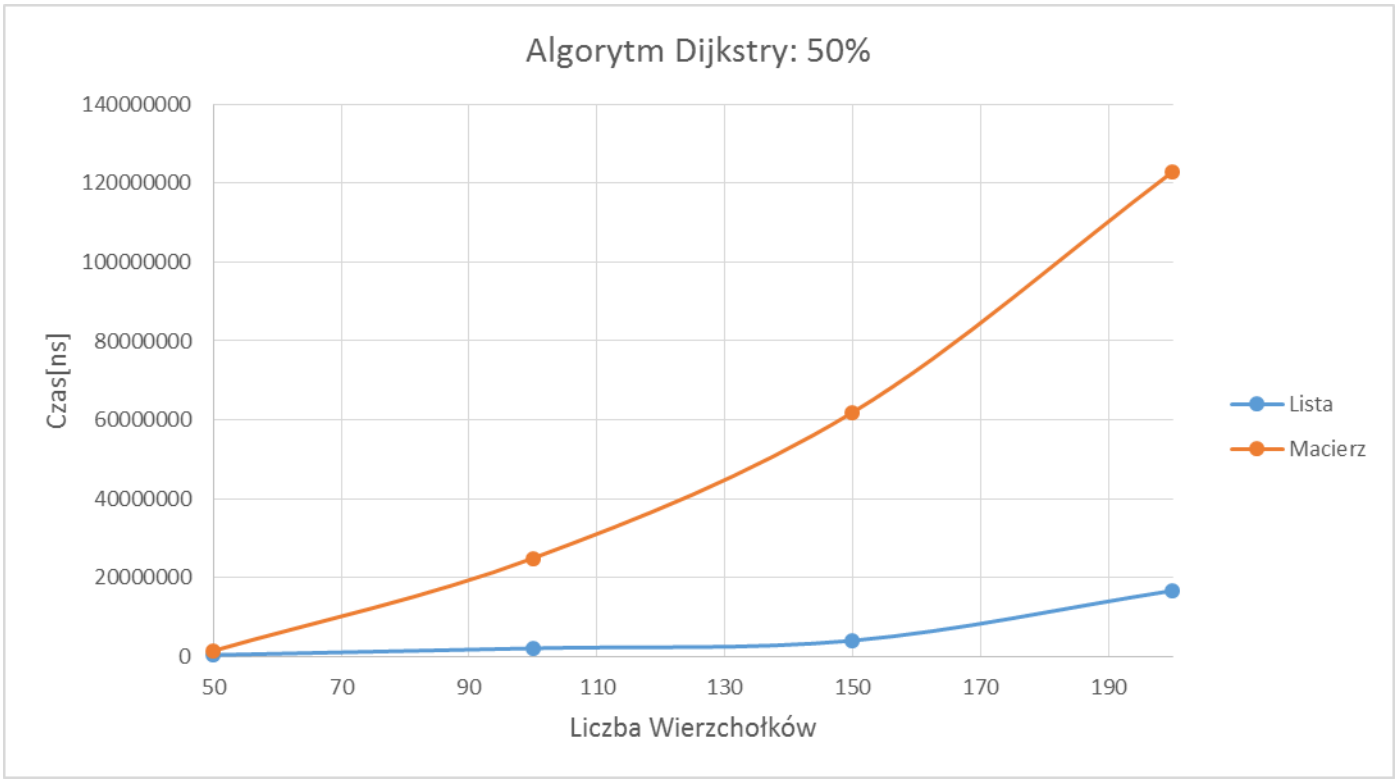
Tabela 3 Uśrednione wyniki dla poszczególnych ilości wierzchołków i gęstości na liście sąsiedztwa

Liczba Wierzchołków	Gęstość	Macierz Czas[ns]
50	25	479831
	50	1432682
	75	5621146
	99	4871614
100	25	13758936
	50	24927563
	75	45673520
	99	64310002
150	25	32742661
	50	61937219
	75	304042700
	99	130817903
200	25	74238209
	50	122873642
	75	682194329
	99	288481267

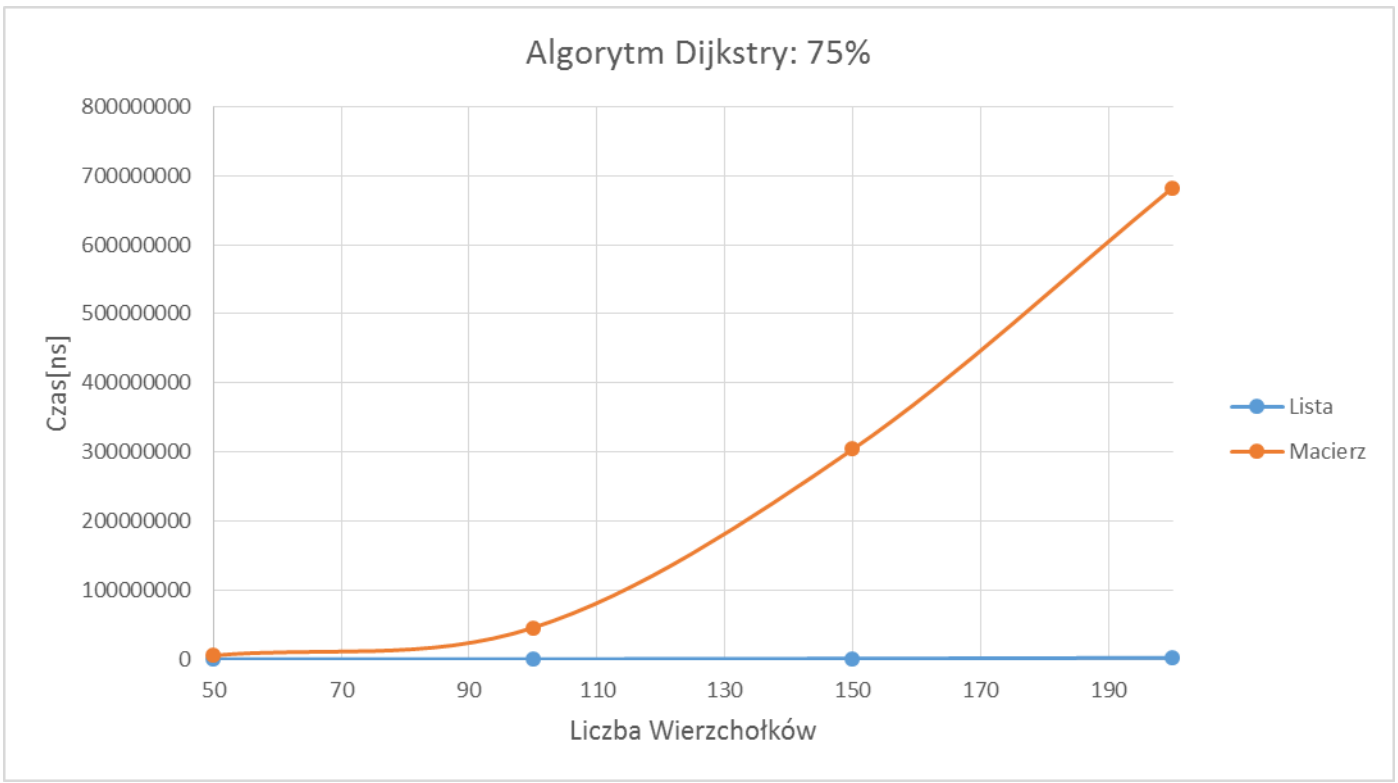
Tabela 4. Uśrednione wyniki dla poszczególnych ilości wierzchołków i gęstości w macierzy sąsiedztwa



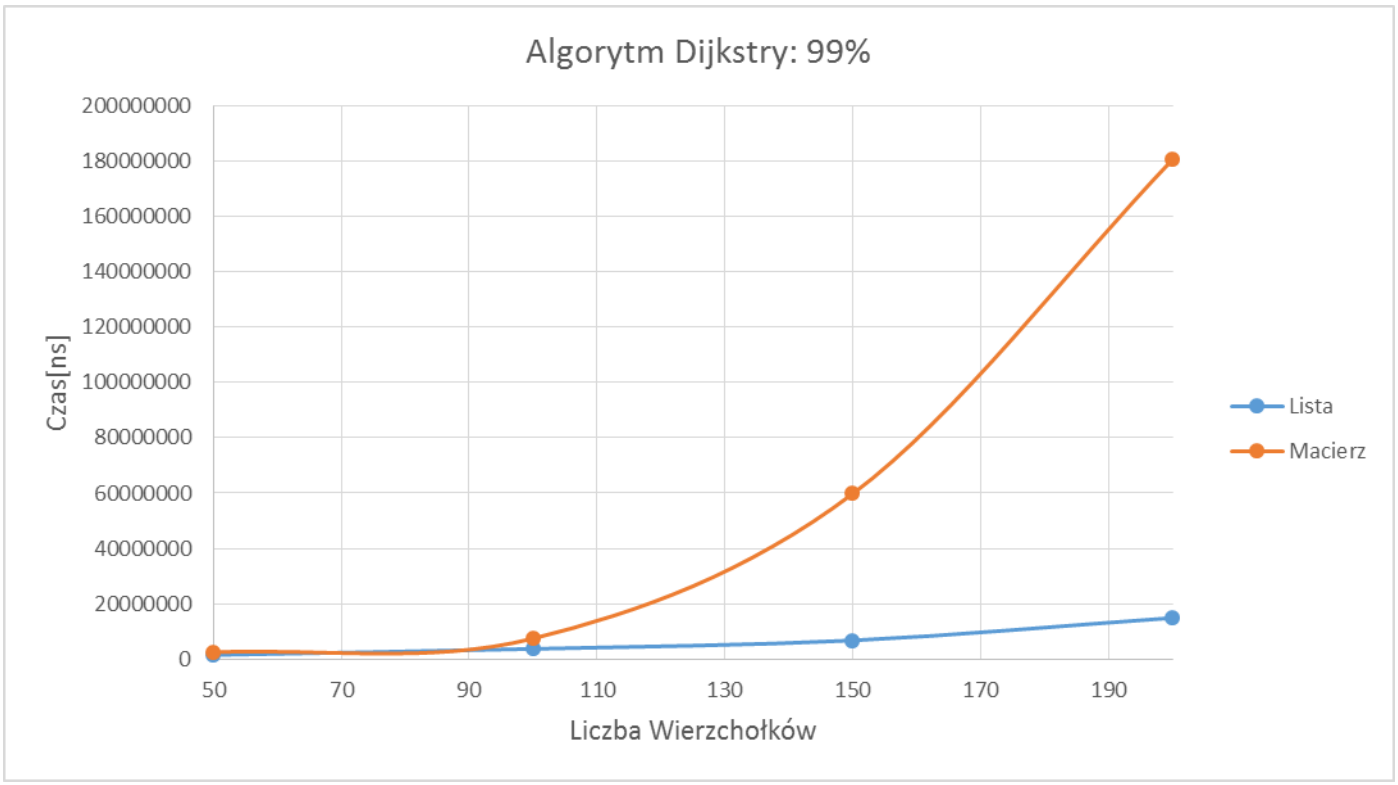
Wykres 7.



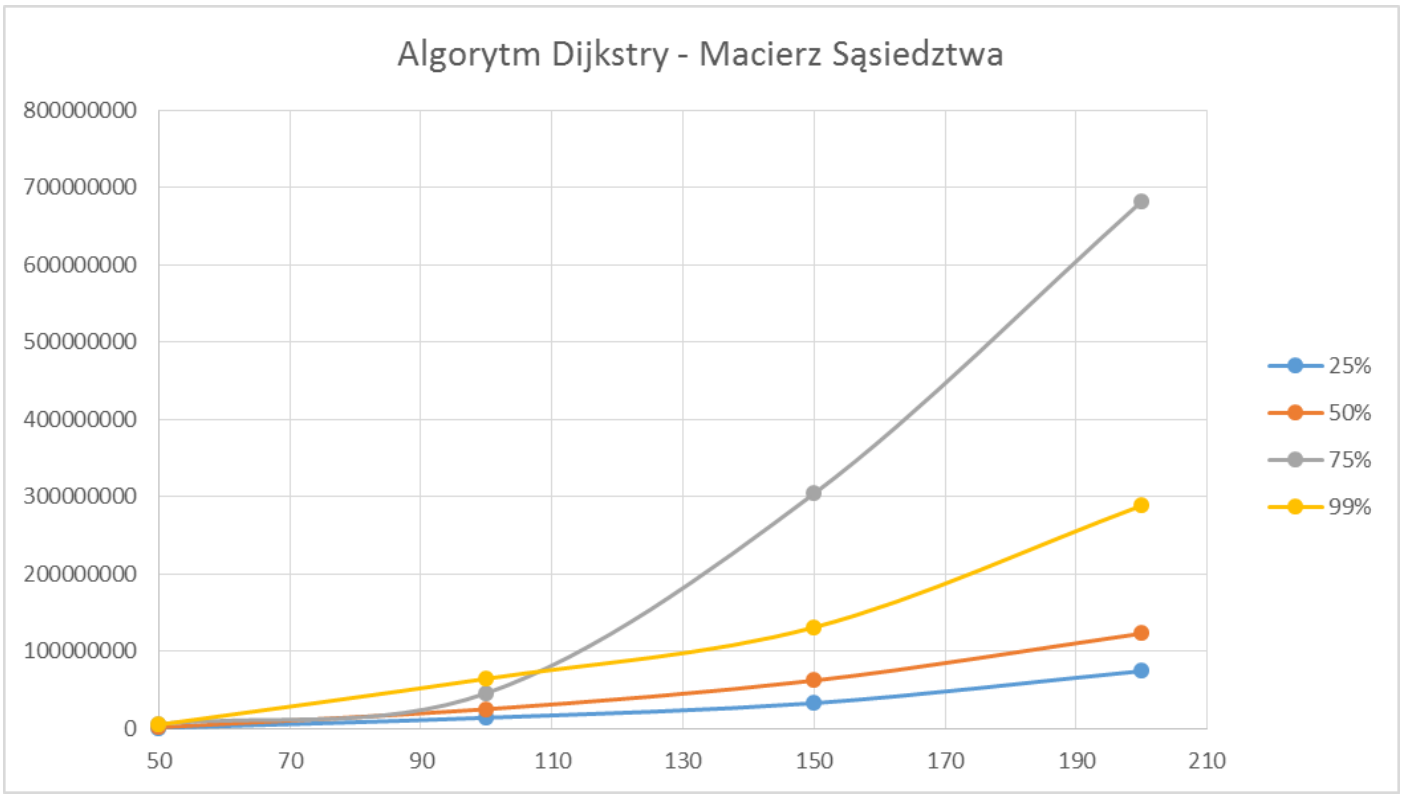
Wykres 8.



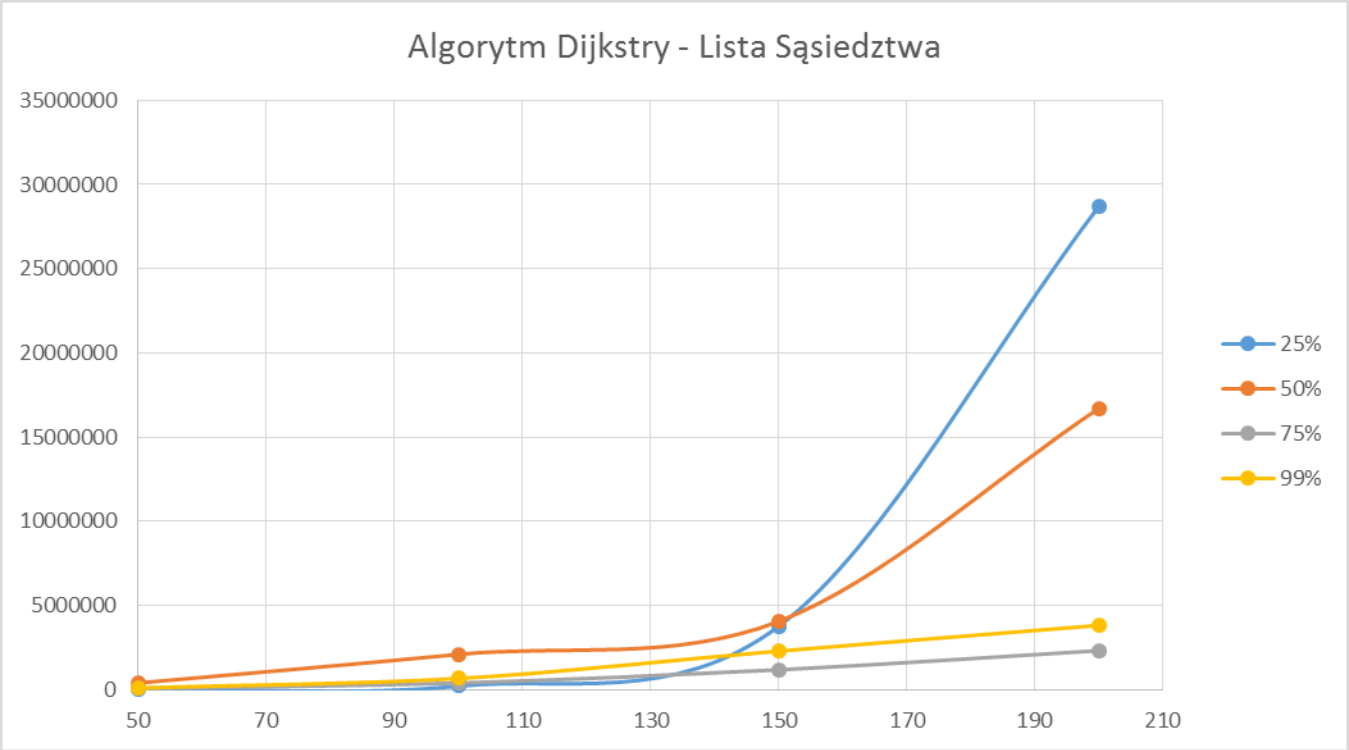
Wykres 9.



Wykres 10.



Wykres 11.



Wykres 12

3. Wnioski

Ponieważ do przeprowadzenia algorytmów zostały użyte tzw. metody naiwne złożoność obu algorytmów zmieniła się na O(V^2), gdzie V jest ilością wierzchołków. Wykresy zamieszczone wcześniej tylko potwierdzają, że metoda naiwna stosowana w jakimkolwiek algorytmie grafowym sprawia, że jego efektywność maleje. Zaobserwować można jak bardzo wzrasta ilość czasu potrzebna na wykonanie się algorytmu w zależności od ilości wierzchołków