Zadanie projektowe nr 2

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

Autor sprawozdania: Michał Dziedziak, 263901
Nazwisko i Imię prowadzącego kurs: Mgr. inż. Antoni Sterna
Dzień i godzina zajęć: Wtorek, 15:15 - 16:55 TN

Spis treści

1 Wstęp **3**

1.1 Zarys projektu . . . 3

1.2 Algorytmy . . . 3

1.2.1 Zaimplemtntowane algorytmy . . . 3

1.2.2 Teoretyczne złożoności obliczeniowe . . . 3

2 Plan eksperymentu **3**

2.1 Założenia . . . 3

2.2 Generowanie grafu . . . 3

2.3 Pomiar czasu . . . 4

2.4 Losowanie populacji . . . 4

3 Wyniki pomiarów **5**

3.1 Algorytmy znajdowania minimalnego drzewa rozpinającego . . . 5

3.1.1 Algorytm Prima i macierz sąsiedztwa . . . 5

3.1.2 Algorytm Prima i lista sąsiedztwa . . . 6

3.1.3 Algorytm Kruskala i macierz sąsiedztwa . . . 7

3.1.4 Algorytm Kruskala i lista sąsiedztwa . . . 8

3.2 Algorytmy znajdowania najkrótszej ścieżki w grafie . . . 9

3.2.1 Algorytm Dijkstry i macierz sąsiedztwa . . . 9

3.2.2 Algorytm Dijkstry i lista sąsiedztwa . . . 10

3.2.3 Algorytm Bellmana-Forda i macierz sąsiedztwa . . . 11

3.2.4 Algorytm Bellmana-Forda i lista sąsiedztwa . . . 12

4 Wykresy **13**

4.1 Algorytmy znajdowania minimalnego drzewa rozpinającego . . . 13

4.2 Algorytmy znajdowania najkrótszej ścieżki w grafie . . . 16

5 Wnioski **19**

5.1 Minimalne drzewo rozpinające . . . 19

5.2 Najkrótsza ścieżka w grafie 19

Spis tabel

1 Pomiary algorytmu Prima na macierzy sąsiedztwa 5
2 Pomiary algorytmu Prima na liście sąsiedztwa 6
3 Pomiary algorytmu Kruskala na macierzy sąsiedztwa 7
4 Pomiary algorytmu Kruskala na liście sąsiedztwa 8
5 Pomiary algorytmu Dijkstry na macierzy sąsiedztwa 9
6 Pomiary algorytmu Dijkstry na liście sąsiedztwa 10
7 Pomiary algorytmu Bellmana-Forda na macierzy sąsiedztwa 11
8 Pomiary algorytmu Bellmana-Forda na liście sąsiedztwa 12

Spis rysunków

1 Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania minimalnego drzewa rozpinającego i macierzy sąsiedztwa 13
2 Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania minimalnego drzewa rozpinającego i listy sąsiedztwa 14
3 Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania minimalnego drzewa rozpinającego i gęstości równej 25% 14
4 Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania minimalnego drzewa rozpinającego i gęstości równej 50% 15
5 Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania minimalnego drzewa rozpinającego i gęstości równej 75% 15
6 Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania minimalnego drzewa rozpinającego i gęstości równej 99% 16
7 Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania najkrótszej ścieżki i macierzy sąsiedztwa 16
8 Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania najkrótszej ścieżki i listy sąsiedztwa 17
9 Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania najkrótszej ścieżki i gęstości równej 25% 17
10 Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania najkrótszej ścieżki i gęstości równej 50% 18
11 Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania najkrótszej ścieżki i gęstości równej 75% 18
12 Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania najkrótszej ścieżki i gęstości równej 99% 19

1 Wstęp

1.1 Zarys projektu

W ramach projektu w języku C++ zaimplementowane zostały wybrane algorytmy znajdowania najmniejszego drzewa rozpinającego i najkrótszej ścieżki w grafie. Do reprezentacji grafów użyta została macierz sąsiedztw i lista sąsiedztw. Poniższe sprawozdanie bada zależność czasu od ilości wierzchołków i gęstości grafu podczas wykonywania poszczególnych algorytmów.

1.2 Algorytmy

1.2.1 Zaimplemtntowane algorytmy

- Najmniejsze drzewo rozpinające
  - Algorytm Prima
  - Algorytm Kruskala
- Najkrótsza ścieżka w grafie
  - Algorytm Dijkstry
  - Algorytm Bellmana-Forda

1.2.2 Teoretyczne złożoności obliczeniowe

W tej sekcji przedstawię teoretyczne złożoności obliczeniowe operacji na grafach, celem późniejszego porównania ich z wynikami testów.

Algorytm	Pesymistyczna złożoność	Średnia złożoność	Optymistyczna złożoność
Prima	$O((V+E)logV)$	$O((V+E)logV)$	$O((V+E)logV)$
Kruskal	$O(ElogE)$	$O(ElogE)$	$O(ElogE)$
Dijkstra	$O((E+V)logV)$	$O((E+V)logV)$	$O(E+VlogV)$
Bellman-Ford	$O(V)$	$O(V^2)$	$O(V^2)$

2 Plan eksperymentu

2.1 Założenia

- Macierz sąsiedztwa i niektóre algorytmy wykorzystują umowną wartość maksymalnego int'a do reprezentowania braku krawędzi pomiędzy wierzchołkami lub nieskończonego kosztu dojścia do danego wierzchołka.
- Wagi krawędzi są liczbami całkowitymi.
- Testy wykonywane są dla grafów posiadających od 50 do 500 wierzchołków (z krokiem równym 50). Dla każdej liczby wierzchołków wykonywane są cztery testy dla gęstości grafu równej kolejno 25%, 50%, 75%, 99%. Test dla każdej kombinacji ilości wierzchołków i gęstości grafu wykonywany jest sto razy i ostateczny czas jest średnią tych czasów.

2.2 Generowanie grafu

Dla każdego testu generowany jest nowy graf. Algorytm analizuje każde potencjalne połączenie pomiędzy wierzchołkami i ustawia je lub nie (ma na to 50% szans). Kolejno waga takiego połączenia losowana jest z zakresu od 0 lub -1000 (w zależności od algorytmu) do 1000. Algorytm losuje czy dodaje nowe połączenie do momentu, kiedy wszystkie pozostałe połączenia muszą zostać stworzone w celu zapewnienia wymaganej gęstości (wtedy tworzy pozostałe wierzchołki z losowymi wagami).

2.3 Pomiar czasu

Do pomiaru czasu została napisana osobna klasa "Timer". Korzysta ona z funkcji QueryPerformanceCounter i QueryPerformanceFrequency umożliwiających pomiar czasu z dokładnością do mikro sekund. Klasa ta określa upływ czasu, bazując na dokładnym liczniku.

2.4 Losowanie populacji

Do losowania populacji również została napisana osobna klasa RandomGenerator. Umożliwia ona losowanie pojedynczych liczb lub całych zbiorów. Korzysta z mt19937 - generatora liczb pseudolosowych.

3 Wyniki pomiarów

3.1 Algorytmy znajdowania minimalnego drzewa rozpinającego

3.1.1 Algorytm Prima i macierz sąsiedztwa

Liczba wierzchołków	**Gęstość**	**Czas** $[ms]$
50	50	0.27
50	75	0.33
50	99	0.42
100	25	0.62
100	50	0.98
100	75	1.37
100	99	1.67
150	25	1.33
150	50	2.16
150	75	2.99
150	99	3.72
200	25	2.29
200	50	3.80
200	75	5.41
200	99	7.12
250	25	3.56
250	50	6.07
250	75	9.60
250	99	13.71
300	25	5.39
300	50	9.91
300	75	15.98
300	99	22.02
350	25	7.06
350	50	14.86
350	75	23.82
350	99	31.44
400	25	10.21
400	50	21.17
400	75	32.68
400	99	42.39
450	25	13.91
450	50	28.09
450	75	41.99
450	99	54.18
500	25	18.31
500	50	35.68
500	75	52.57
500	99	67.64

Tabela 1: Pomiary algorytmu Prima na macierzy sąsiedztwa

3.1.2 Algorytm Prima i lista sąsiedztwa

Liczba wierzchołków	**Gęstość**	**Czas** $[ms]$
50	50	0.23
50	75	0.33
50	99	0.40
100	25	0.55
100	50	0.91
100	75	1.26
100	99	1.63
150	25	1.12
150	50	1.91
150	75	2.92
150	99	3.81
200	25	2.01
200	50	3.65
200	75	6.01
200	99	8.21
250	25	3.08
250	50	6.02
250	75	10.47
250	99	15.25
300	25	4.71
300	50	9.94
300	75	17.52
300	99	25.00
350	25	6.52
350	50	15.15
350	75	26.50
350	99	36.39
400	25	9.75
400	50	22.04
400	75	37.04
400	99	50.08
450	25	13.54
450	50	30.50
450	75	48.97
450	99	65.12
500	25	18.12
500	50	38.88
500	75	62.27
500	99	81.71

Tabela 2: Pomiary algorytmu Prima na liście sąsiedztwa

3.1.3 Algorytm Kruskala i macierz sąsiedztwa

Liczba wierzchołków	**Gęstość**	**Czas** $[ms]$
50	50	0.14
50	75	0.19
50	99	0.25
100	25	0.33
100	50	0.51
100	75	0.72
100	99	0.91
150	25	0.81
150	50	1.27
150	75	1.73
150	99	2.21
200	25	1.30
200	50	2.29
200	75	3.46
200	99	4.65
250	25	2.15
250	50	3.86
250	75	5.81
250	99	7.34
300	25	2.77
300	50	6.36
300	75	7.11
300	99	8.54
350	25	3.55
350	50	6.07
350	75	9.50
350	99	13.35
400	25	4.76
400	50	8.79
400	75	13.92
400	99	19.34
450	25	6.30
450	50	12.02
450	75	19.20
450	99	25.97
500	25	8.24
500	50	15.90
500	75	24.72
500	99	32.82

Tabela 3: Pomiary algorytmu Kruskala na macierzy sąsiedztwa

3.1.4 Algorytm Kruskala i lista sąsiedztwa

Liczba wierzchołków	**Gęstość**	**Czas** $[ms]$
50	50	0.35
50	75	0.55
50	99	0.73
100	25	0.77
100	50	1.62
100	75	2.51
100	99	3.40
150	25	1.86
150	50	3.96
150	75	6.25
150	99	8.55
200	25	3.54
200	50	7.77
200	75	12.01
200	99	16.48
250	25	5.79
250	50	12.59
250	75	20.19
250	99	28.44
300	25	8.90
300	50	19.26
300	75	31.81
300	99	47.02
350	25	12.96
350	50	28.93
350	75	51.49
350	99	78.03
400	25	18.97
400	50	42.95
400	75	79.09
400	99	119.90
450	25	24.95
450	50	60.72
450	75	112.45
450	99	165.48
500	25	30.87
500	50	81.31
500	75	151.28
500	99	221.31

Tabela 4: Pomiary algorytmu Kruskala na liście sąsiedztwa

3.2 Algorytmy znajdowania najkrótszej ścieżki w grafie

3.2.1 Algorytm Dijkstry i macierz sąsiedztwa

Liczba wierzchołków	**Gęstość**	**Czas** $[ms]$
50	50	0.04
50	75	0.03
50	99	0.03
100	25	0.10
100	50	0.13
100	75	0.11
100	99	0.09
150	25	0.21
150	50	0.27
150	75	0.23
150	99	0.18
200	25	0.34
200	50	0.44
200	75	0.37
200	99	0.30
250	25	0.51
250	50	0.69
250	75	0.58
250	99	0.46
300	25	0.73
300	50	0.96
300	75	0.80
300	99	0.64
350	25	0.97
350	50	1.29
350	75	1.08
350	99	0.85
400	25	1.25
400	50	1.66
400	75	1.37
400	99	1.09
450	25	1.56
450	50	2.08
450	75	1.71
450	99	1.36
500	25	1.91
500	50	2.55
500	75	2.10
500	99	1.65

Tabela 5: Pomiary algorytmu Dijkstry na macierzy sąsiedztwa

3.2.2 Algorytm Dijkstry i lista sąsiedztwa

Liczba wierzchołków	**Gęstość**	**Czas** $[ms]$
50	50	0.05
50	75	0.02
50	99	0.02
100	25	0.05
100	50	0.05
100	75	0.06
100	99	0.07
150	25	0.08
150	50	0.12
150	75	0.13
150	99	0.24
200	25	0.11
200	50	0.16
200	75	0.29
200	99	0.46
250	25	0.16
250	50	0.28
250	75	0.61
250	99	0.76
300	25	0.31
300	50	0.30
300	75	0.42
300	99	0.61
350	25	0.42
350	50	0.40
350	75	0.72
350	99	1.21
400	25	0.45
400	50	0.61
400	75	1.17
400	99	1.86
450	25	0.72
450	50	0.90
450	75	2.10
450	99	2.80
500	25	0.77
500	50	1.39
500	75	2.66
500	99	3.39

Tabela 6: Pomiary algorytmu Dijkstry na liście sąsiedztwa

3.2.3 Algorytm Bellmana-Forda i macierz sąsiedztwa

Liczba wierzchołków	**Gęstość**	**Czas** $[ms]$
50	50	0.85
50	75	0.88
50	99	0.97
100	25	6.11
100	50	9.95
100	75	9.37
100	99	9.12
150	25	23.99
150	50	36.28
150	75	34.18
150	99	32.47
200	25	59.00
200	50	87.47
200	75	82.29
200	99	77.73
250	25	116.35
250	50	172.03
250	75	161.44
250	99	151.80
300	25	202.37
300	50	298.12
300	75	282.33
300	99	262.81
350	25	322.17
350	50	473.92
350	75	444.97
350	99	417.50
400	25	483.42
400	50	715.49
400	75	665.48
400	99	623.85
450	25	685.95
450	50	1008.25
450	75	946.79
450	99	888.10
500	25	944.15
500	50	1382.63
500	75	1299.42
500	99	1216.66

Tabela 7: Pomiary algorytmu Bellmana-Forda na macierzy sąsiedztwa

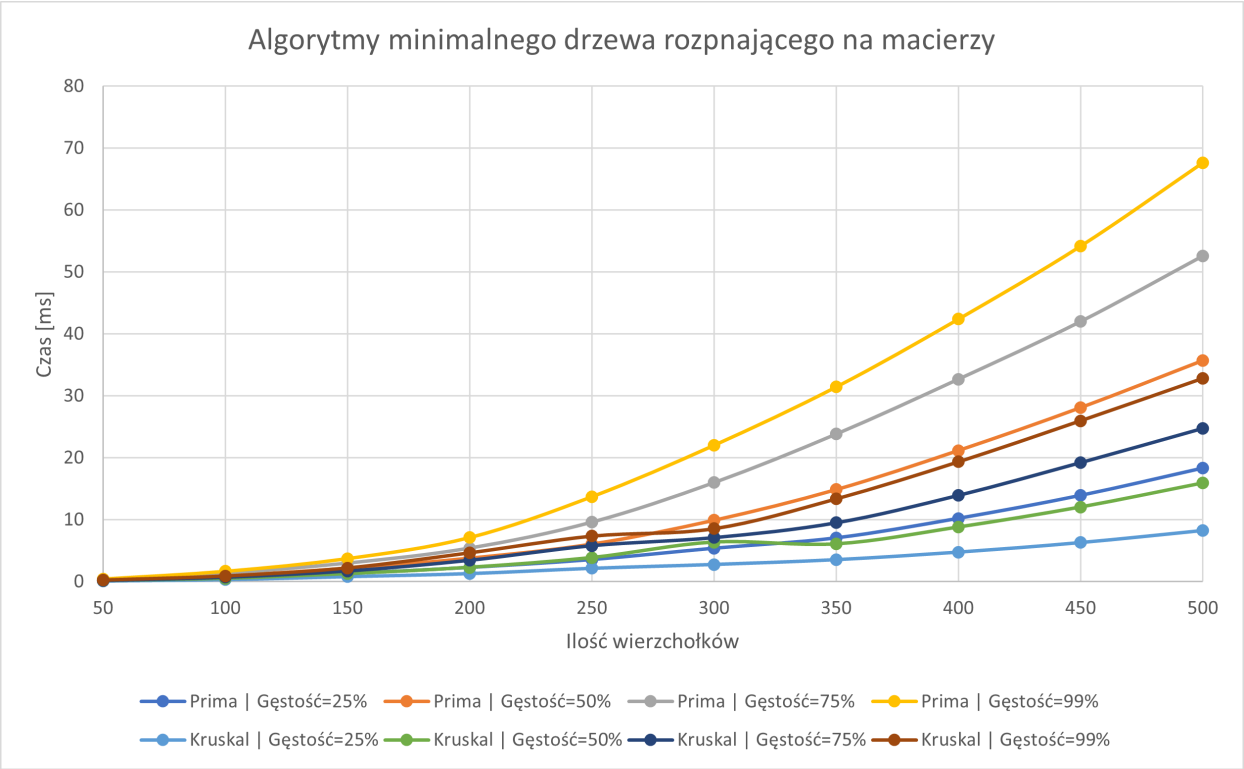
3.2.4 Algorytm Bellmana-Forda i lista sąsiedztwa

Liczba wierzchołków	Gęstość	Czas $[ms]$
50	50	0.24
50	75	0.37
50	99	0.49
100	25	1.00
100	50	2.10
100	75	3.53
100	99	5.27
150	25	3.60
150	50	9.07
150	75	16.39
150	99	23.70
200	25	10.07
200	50	27.35
200	75	44.29
200	99	57.83
250	25	23.90
250	50	57.45
250	75	86.02
250	99	113.73
300	25	47.20
300	50	98.37
300	75	149.38
300	99	198.08
350	25	76.72
350	50	157.04
350	75	251.91
350	99	408.70
400	25	115.07
400	50	228.33
400	75	450.26
400	99	844.20
450	25	164.60
450	50	369.25
450	75	860.86
450	99	1453.53
500	25	225.61
500	50	618.80
500	75	1470.05
500	99	2071.92

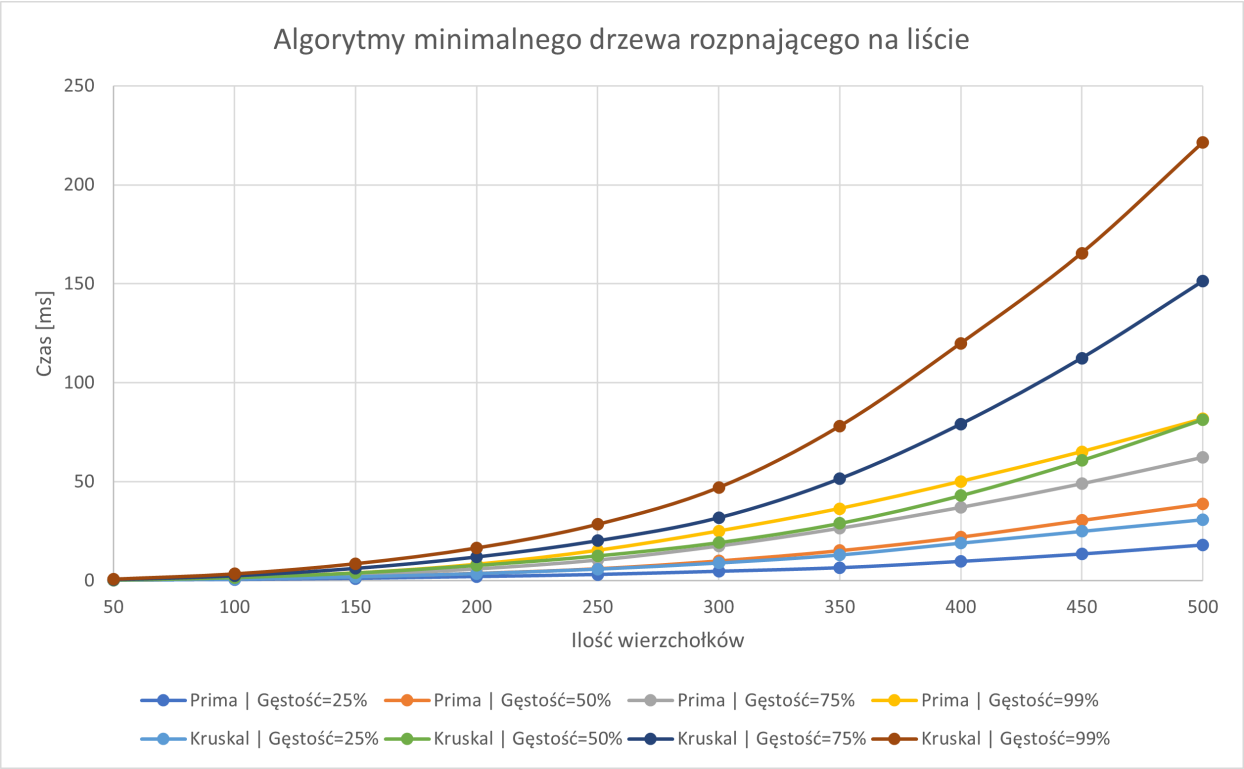
Tabela 8: Pomiary algorytmu Bellmana-Forda na liście sąsiedztwa

4 Wykresy

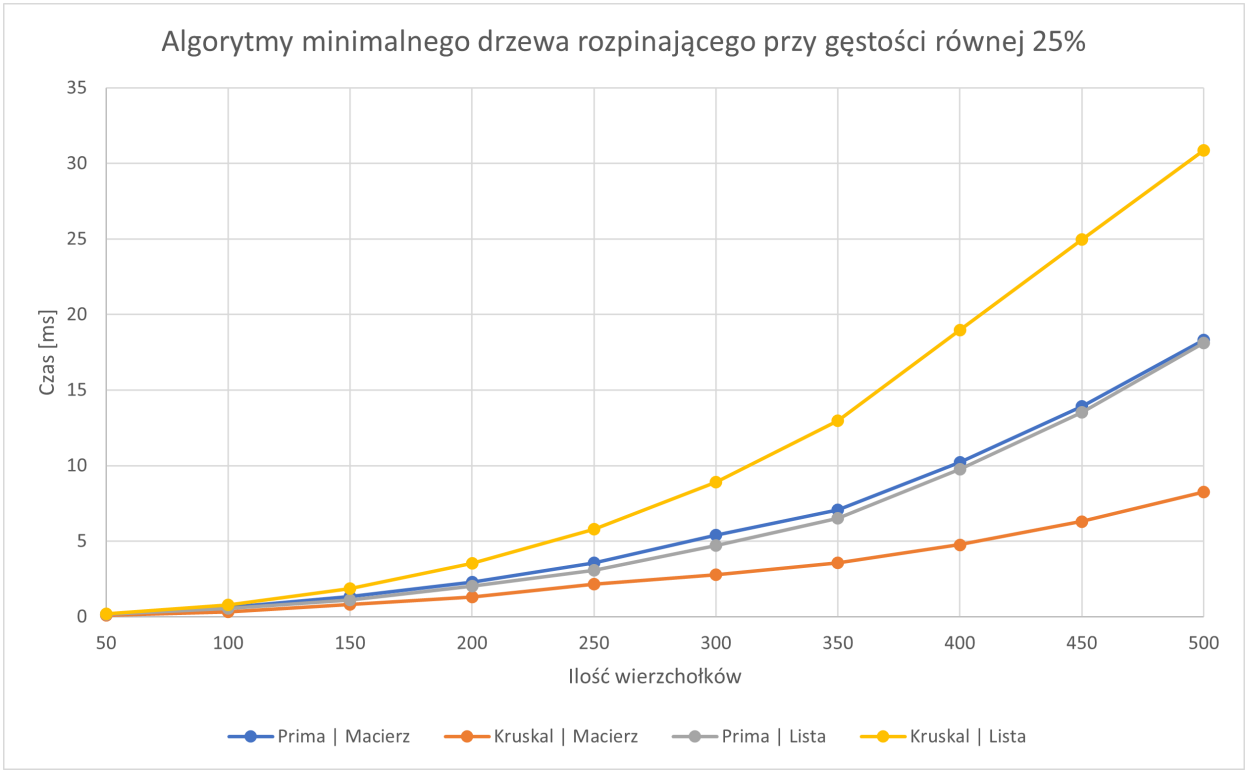
4.1 Algorytmy znajdowania minimalnego drzewa rozpinającego



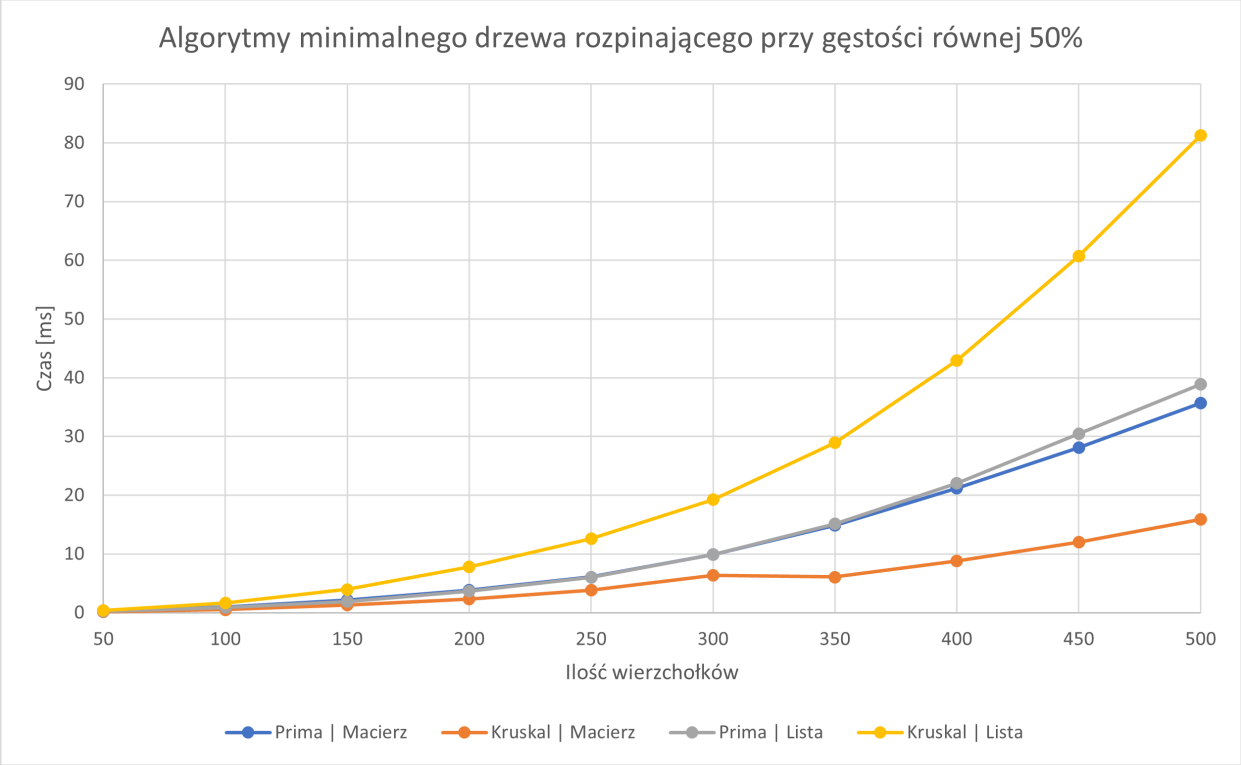
Rysunek 1: Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania minimalnego drzewa rozpinającego i macierzy sąsiedztwa



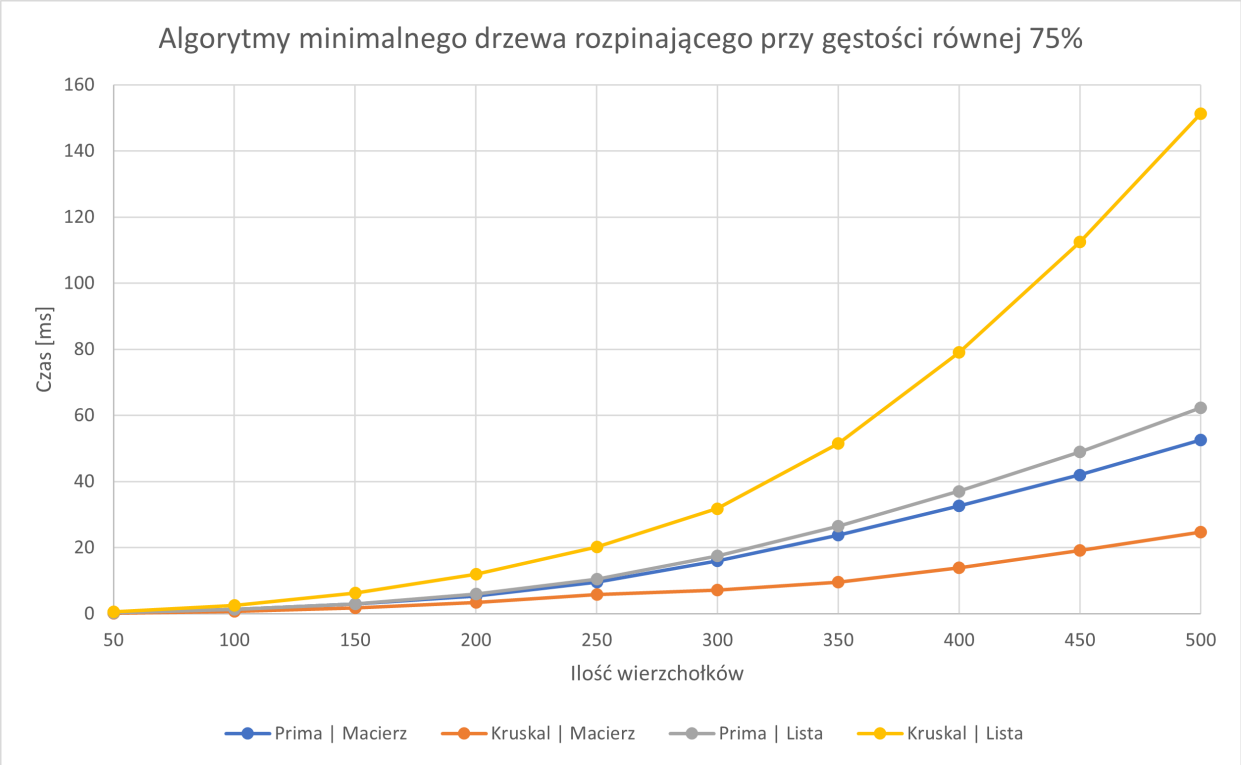
Rysunek 2: Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania minimalnego drzewa rozpinającego i listy sąsiedztwa



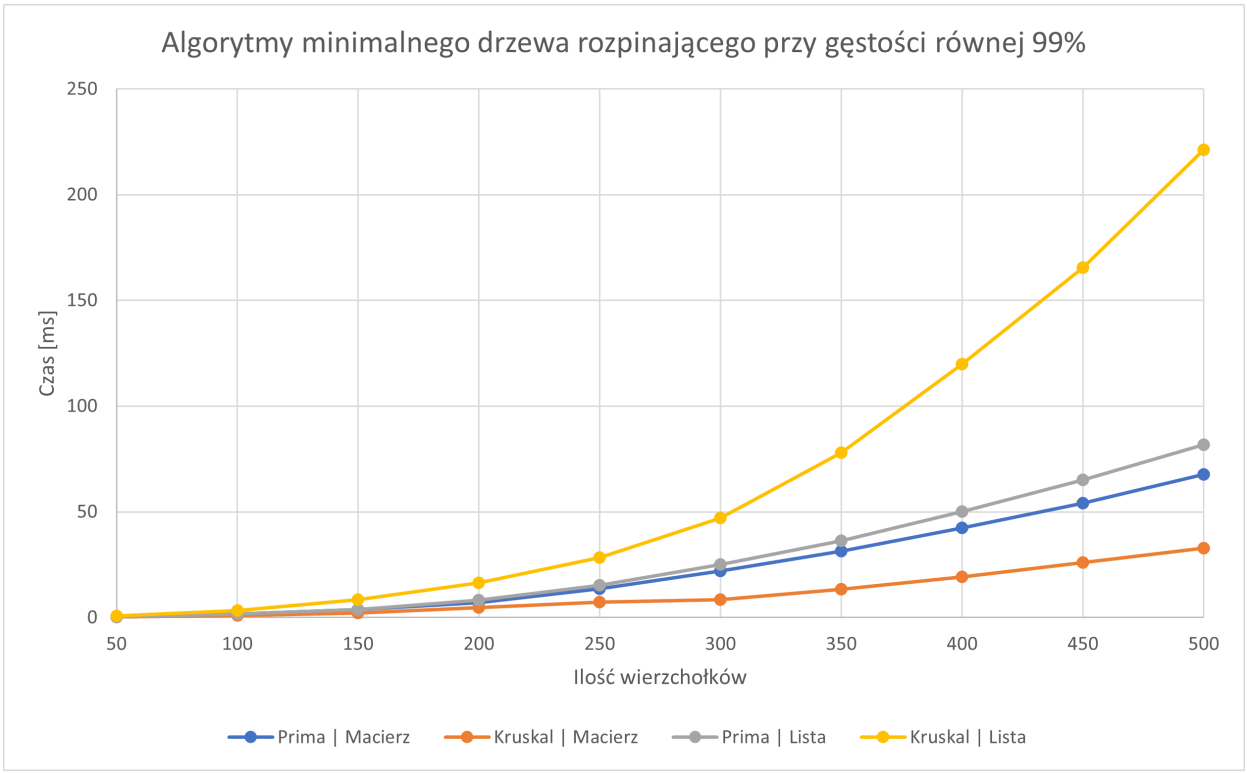
Rysunek 3: Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania minimalnego drzewa rozpinającego i gęstości równej 25%



Rysunek 4: Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania minimalnego drzewa rozpinającego i gęstości równej 50%

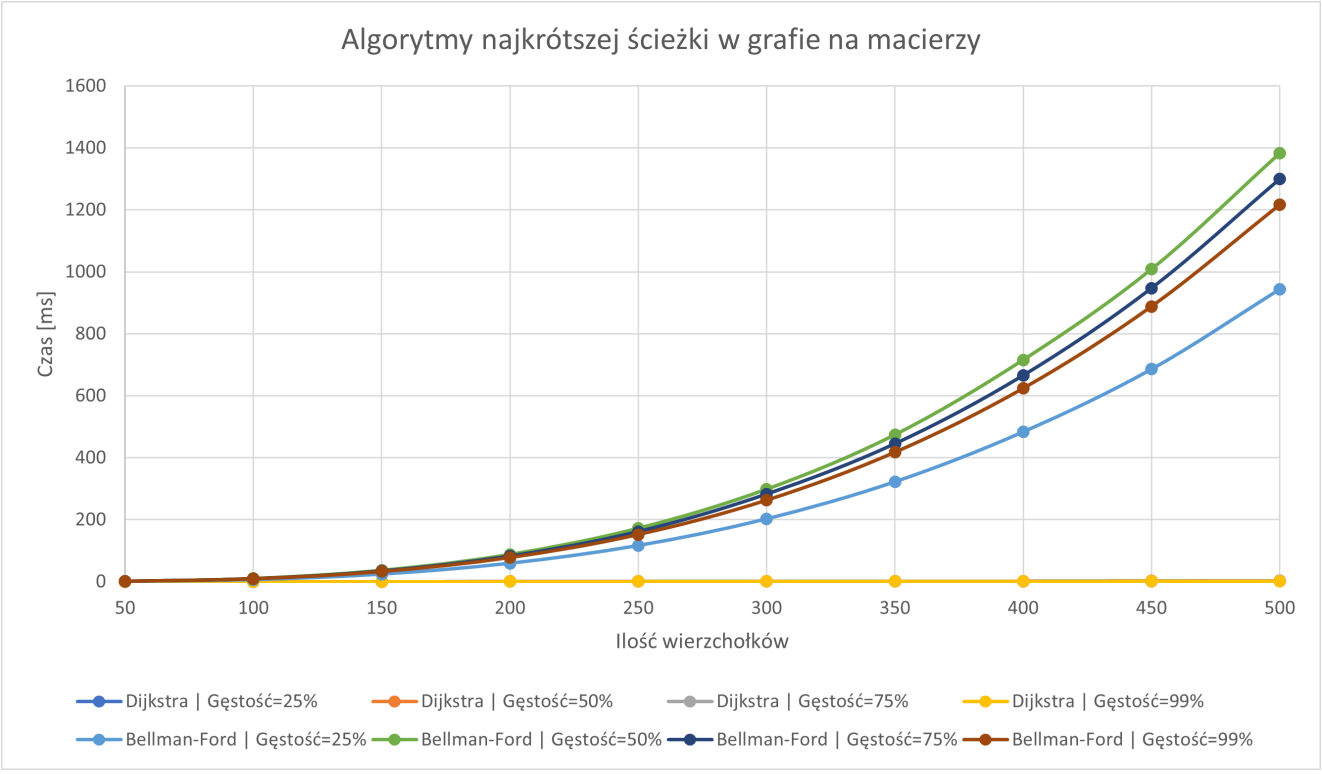


Rysunek 5: Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania minimalnego drzewa rozpinającego i gęstości równej 75%

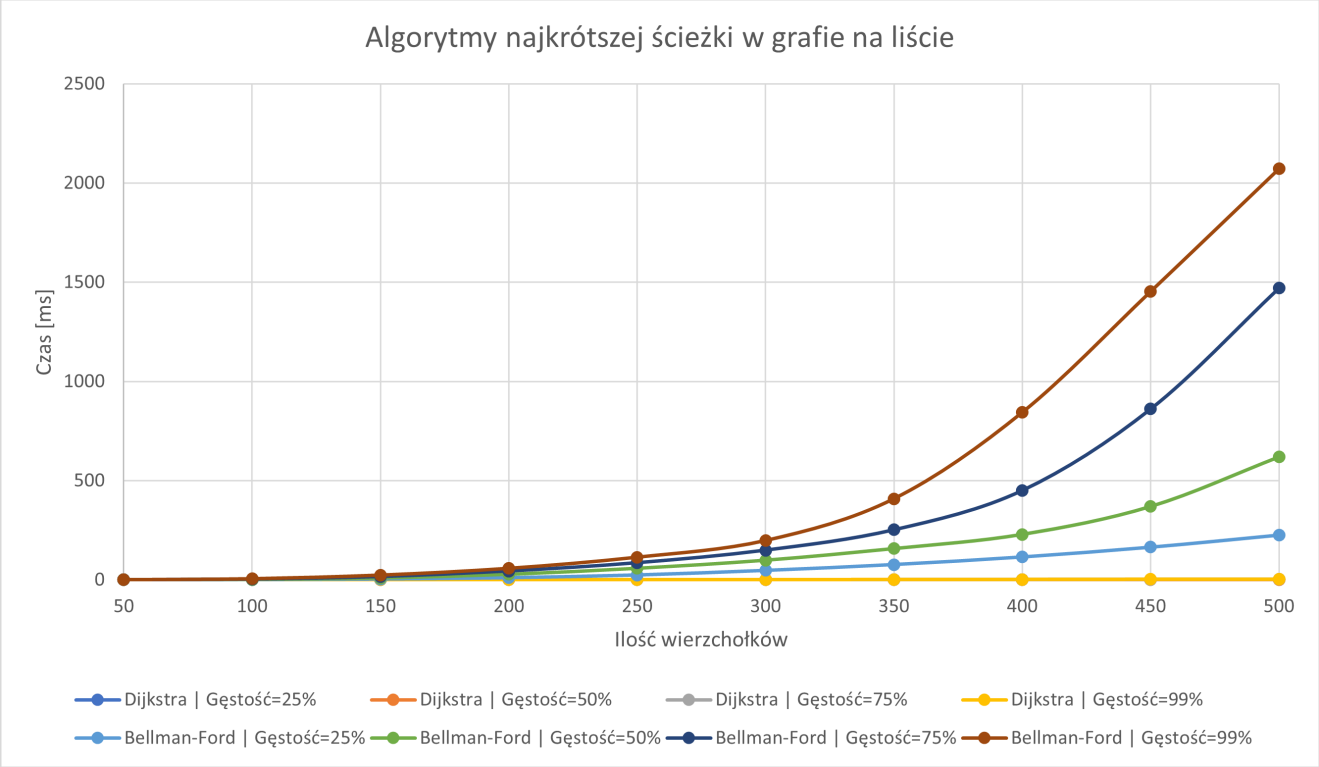


Rysunek 6: Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania minimalnego drzewa rozpinającego i gęstości równej 99%

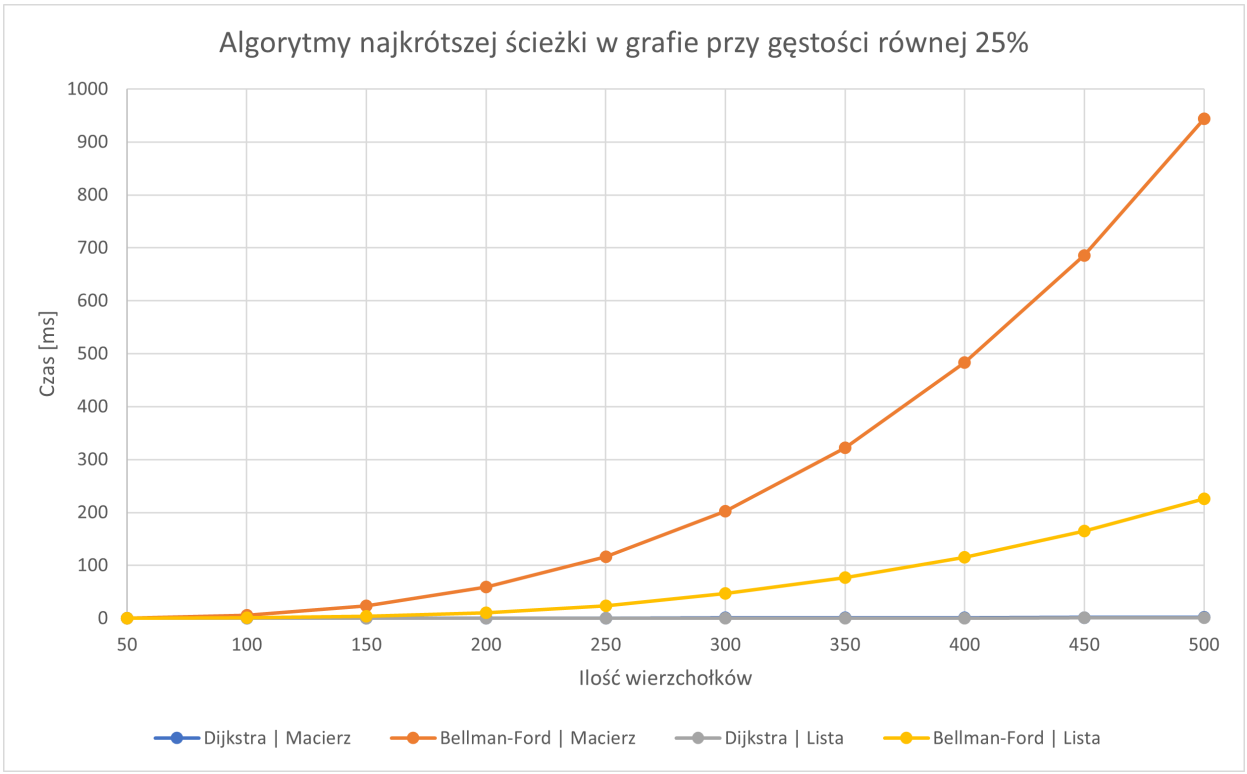
4.2 Algorytmy znajdowania najkrótszej ścieżki w grafie



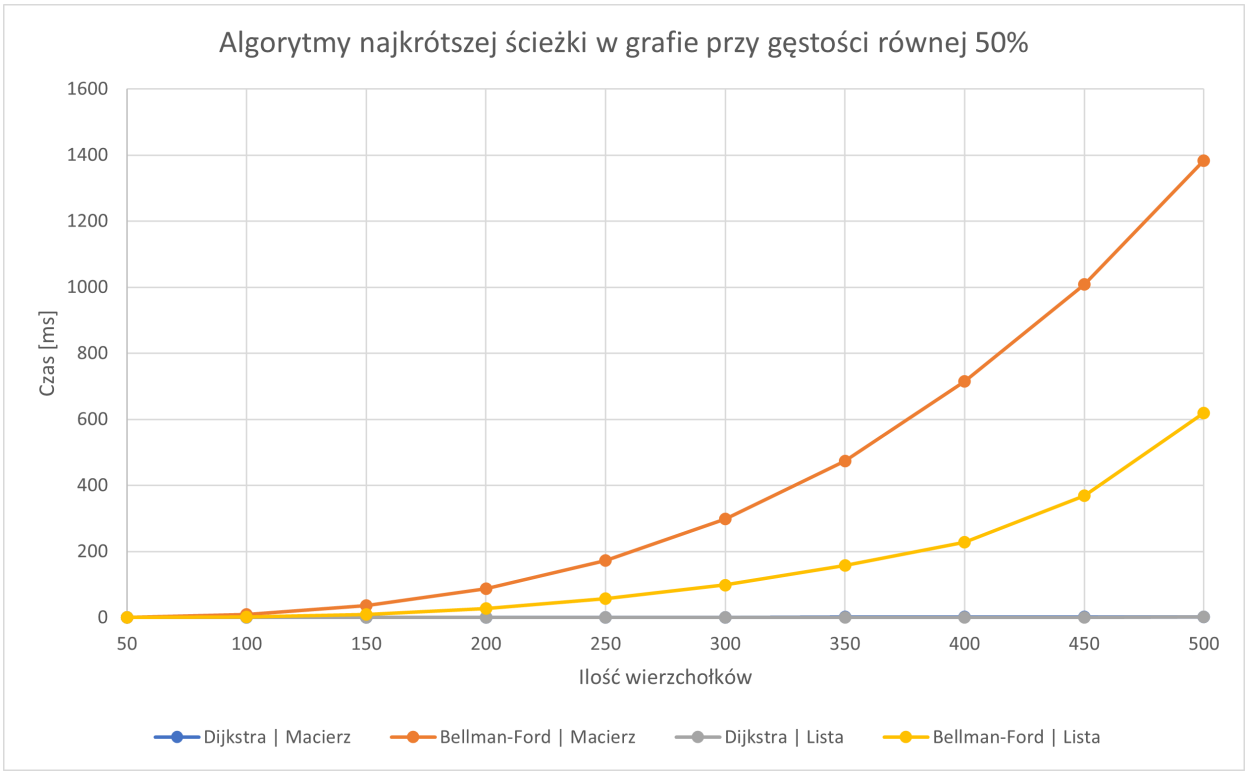
Rysunek 7: Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania najkrótszej ścieżki i macierzy sąsiedztwa



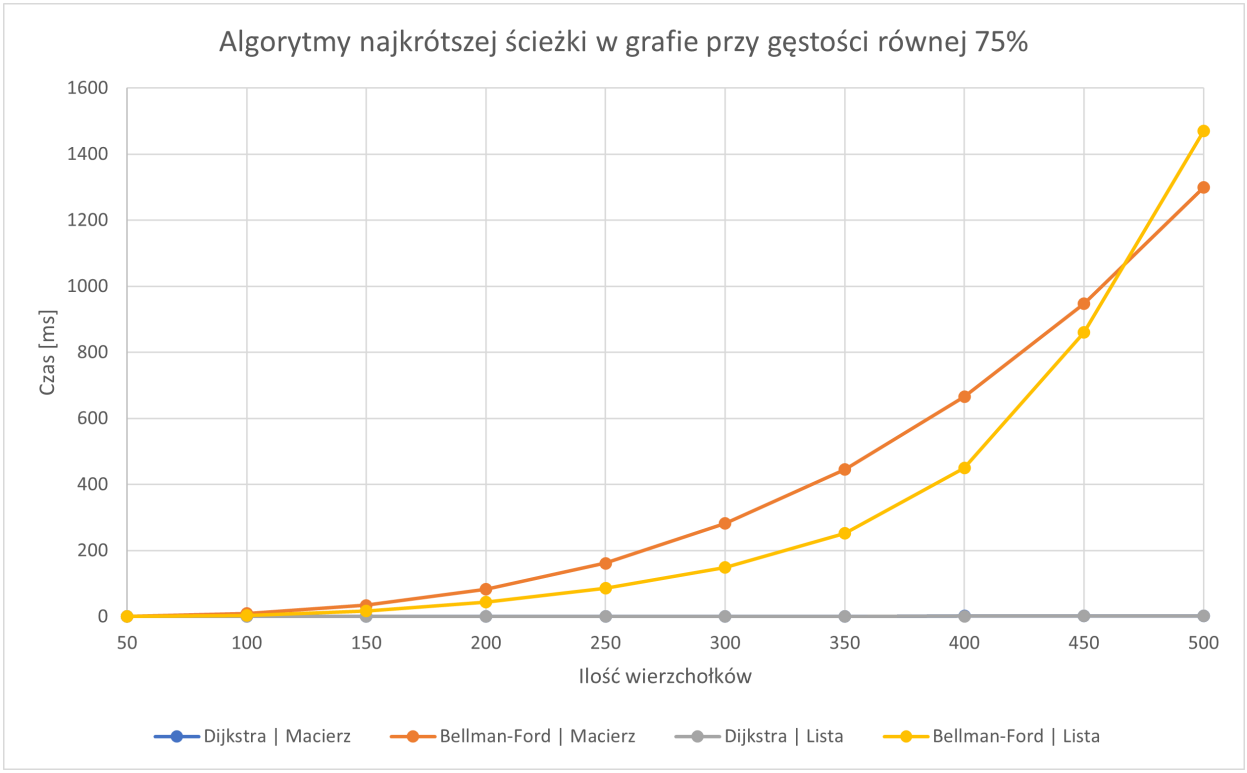
Rysunek 8: Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania najkrótszej ścieżki i listy sąsiedztwa



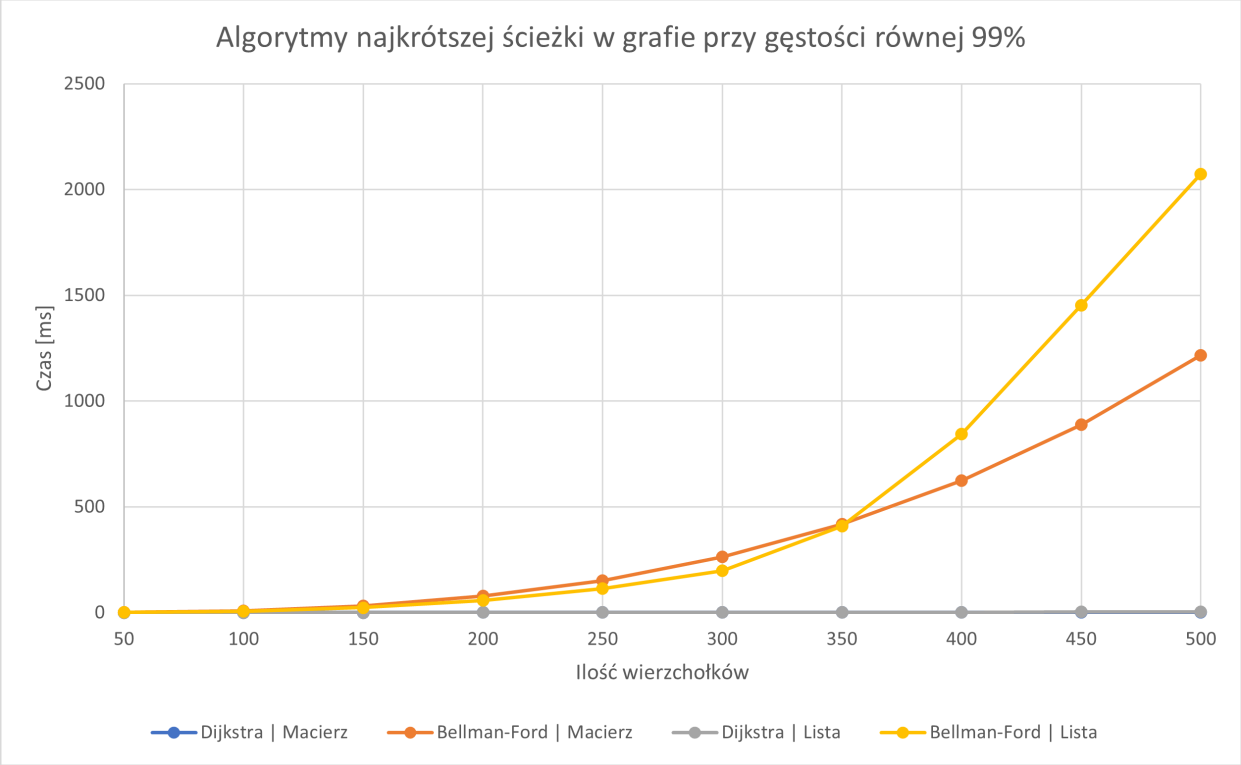
Rysunek 9: Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania najkrótszej ścieżki i gęstości równej 25%



Rysunek 10: Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania najkrótszej ścieżki i gęstości równej 50%



Rysunek 11: Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania najkrótszej ścieżki i gęstości równej 75%



Rysunek 12: Wykres czasu od ilości wierzchołków dla algorytmów wyznaczania najkrótszej ścieżki i gęstości równej 99%

5 Wnioski

5.1 Minimalne drzewo rozpinające

Algorytm Prima i Kruskala służą do znajdowania minimalnego drzewa rozpinającego w grafie. Czas wykonania obu algorytmów wzrasta wraz z liczbą wierzchołków i gęstością grafu.

Algorytm Prima polega na wyborze kolejnych krawędzi o najmniejszej wadze, które nie tworzą cyklu, aż do utworzenia drzewa rozpinającego. Ten algorytm działa efektywniej, gdy graf jest reprezentowany w postaci listy sąsiedztwa, gdzie dla każdego wierzchołka przechowywana jest lista sąsiadujących z nim krawędzi.

Algorytm Kruskala natomiast polega na sortowaniu krawędzi według wagi i stopniowym dodawaniu ich do drzewa rozpinającego, jeżeli nie tworzą cyklu. Ten algorytm działa efektywniej, gdy graf jest reprezentowany w postaci macierzy sąsiedztwa, gdzie dla każdej pary wierzchołków przechowywana jest informacja o istnieniu krawędzi między nimi.

Dla dużych gęstości grafów złożoność czasowa obu algorytmów przypomina złożoność wykładniczą, co oznacza, że czas wykonania znacznie rośnie wraz z liczbą wierzchołków.

Doświadczalnie otrzymane złożoności czasowe algorytmów pokrywają się z tymi teoretycznymi.

5.2 Najkrótsza ścieżka w grafie

Algorytmy Dijkstry i Bellmana-Forda służą do znajdowania najkrótszej ścieżki między dwoma wierzchołkami w grafie. Podobnie jak w przypadku minimalnego drzewa rozpinającego, czas wykonania obu algorytmów wzrasta z liczbą wierzchołków i gęstością grafu.

Algorytm Dijkstry polega na stopniowym wyborze wierzchołków, dla których obliczane są koszty dotarcia do nich.

Algorytm ten działa efektywnie, gdy graf jest reprezentowany w postaci listy sąsiedztwa.

Algorytm Bellmana-Forda natomiast korzysta z relaksacji krawędzi, czyli aktualizacji kosztów dotarcia do wierzchołków w każdej iteracji. Ten algorytm działa efektywnie, gdy graf jest reprezentowany w postaci macierzy sąsiedztwa.

Czas wykonania algorytmu Bellmana-Forda jest znacząco większy niż algorytmu Dijkstry, a dodatkowo rośnie w sposób wykładniczy wraz z liczbą wierzchołków. Dla niższych gęstości grafów, algorytm Bellmana-Forda działa efektywniej na liście sąsiedztwa, natomiast wraz ze wzrostem gęstości, macierz sąsiedztwa staje się bardziej efektywna.

Podobnie jak w przypadku minimalnego drzewa rozpinającego, doświadczalnie otrzymane złożoności czasowe algorytmów pokrywają się z tymi teoretycznymi.