Sprawozdanie z pierwszego zadania projektowego z przedmiotu „Struktury danych i Złożoność obliczeniowa”

Michał Zajdel (263932)

Grupa projektowa: INEK00026P

Kod grupy: K03-37h

Prowadzący: Dr. Inż. Dariusz Banasiak

Spis treści

1. Wstęp

Poniższe opisy struktur danych, oraz eksperymentów wykonanych na tych strukturach danych, zostały napisane na podstawie programu, który został załączony do folderu. Kod źródłowy programu znajduje się w folderze „Kod źródłowy”. Projekt został napisany w IDE CLion od firmy JetBrains. W folderze znajduje się także arkusz kalkulacyjny o nazwie „Pomiary do projektu 2”, w którym znajdują się wszystkie pomiary zrobione na rzecz projektu. W sprawozdaniu przedstawiłem tylko średnie pomiarów czasu, w celu zredukowania miejsca, które zajmują tabele przedstawiające te pomiary czasu. W tabelach przedstawiających średnie pomiary czasu zostały wzięte pod uwagę wszystkie wykonane pomiary dla danej operacji.

Opis eksperymentu

Dla każdego z poniższych algorytmów zmierzono czas zajmujący na poszczególną operację dla następujących wielkości grafów:

- 10 wierzchołków
- 25 wierzchołków
- 75 wierzchołków
- 100 wierzchołków

Oraz poszczególnych gęstości grafów:

- 25%
- 50%
- 75%
- 99%

Przy pomiarze czasu nie zostało wzięte pod uwagę generowanie grafów o poszczególnych ilościach wierzchołków i gęstości. Przy generowaniu krawędzi tych grafów użyto liczb losowych do wypełnienia wag krawędzi, jednak źródłowe i docelowe wierzchołki wygenerowano przy pomocy funkcji która traciła losowość przy większych parametrach gęstości. Liczby losowe wygenerowano przy pomocy funkcji std::random_device oraz std::uniform_int_distribution. Aby wykonać pomiar czasu użyto funkcji QueryPerformanceCounter.

Kod źródłowy projektu, plik wykonywalny, PDF sprawozdania oraz arkusz .xlsx zawierający pomiary można znaleźć na repozytorium na GitHub:

https://github.com/Huntarman/GraphsAndAlgorithms

2. Graf nieskierowany i algorytmy minimalnego drzewa rozpinającego

Użyte algorytmy i ich złożoności obliczeniowe

- Algorytm Kruskal'a
  - Złożoność obliczeniowa : $\Theta(\mathrm{E} * \log V) = \Theta(\mathrm{E} * \log E)$
- Algorytm Prim'a
  - Złożoność obliczeniowa : $\Theta(\mathrm{E} * \log V)$

Zaimplementowanie algorytmu Kruskala czy Prima przy użyciu macierzy może wydłużyć czas trwania algorytmu. W takim przypadku algorytm za każdym razem będzie musiał brać pod uwagę E możliwych połączeń z innymi wierzchołkami.

Opis algorytmów

Algorytm Kruskala, polega na posortowaniu wszystkich krawędzi grafu w zależności od ich wag (od najmniejszej do największej), oraz dodawaniu ich do tablicy przechowującej krawędzie minimalnego drzewa rozpinającego, pomijając krawędzie, które stworzyłyby cykl w MST.

Wykrywanie cyklów w moim projekcie zostało zaimplementowane przy pomocy algorytmu Union-Find. Sortowanie krawędzie, zostało zaimplementowane przy pomocy kolejki priorytetowej, bazującej na kopcu typu min.

Algorytm kończy swoje działanie kiedy w tablicy wierzchołków MST pojawi się V – 1 krawędzi, co będzie oznaczało że wszystkie wierzchołki zostały połączone

Algorytm Prima, polega na przechodzeniu przez kolejne wierzchołki, w kolejności krawędzi o najmniejszych wagach. Zaczynając od losowej/wybranej krawędzi, algorytm dodaje do kolejki priorytetowej krawędzie tego wierzchołka, po czym wyciąga z korzenia kopca krawędź o najmniejszej wadze. Jeśli ta krawędź nie tworzy cyklu w MST, algorytm dodaje ją do MST i przechodzi do wierzchołka docelowego tej krawędzi. Jeśli wierzchołek został już odwiedzony, krawędzi nie są dodawane do kolejki priorytetowej. Jeśli wierzchołek tworzy cykl, korzeń po s-pop'owaniu swojego korzenia, będzie w korzeniu miał kolejną krawędź o najmniejszej wadze.

Algorytm kończy swoje działanie kiedy w tablicy wierzchołków MST pojawi się V – 1 krawędzi, co będzie oznaczało że wszystkie wierzchołki zostały połączone

Wyniki

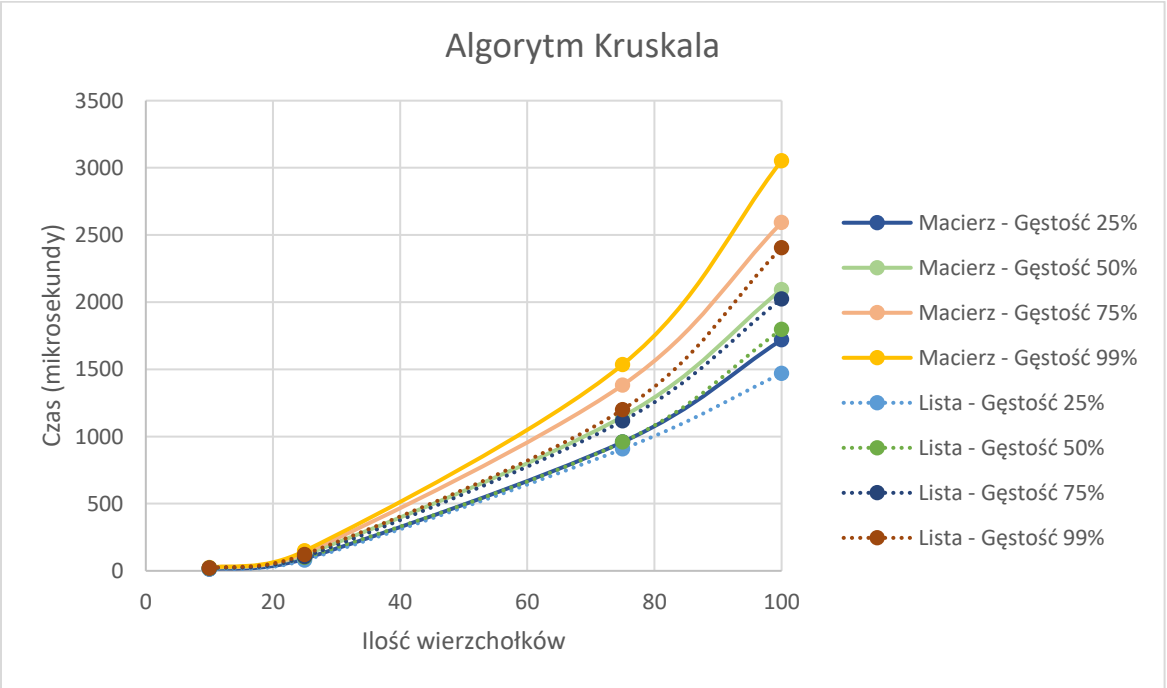
Poniżej znajdują się tabele przedstawiające średnie czasy wykonywania algorytmów MST.

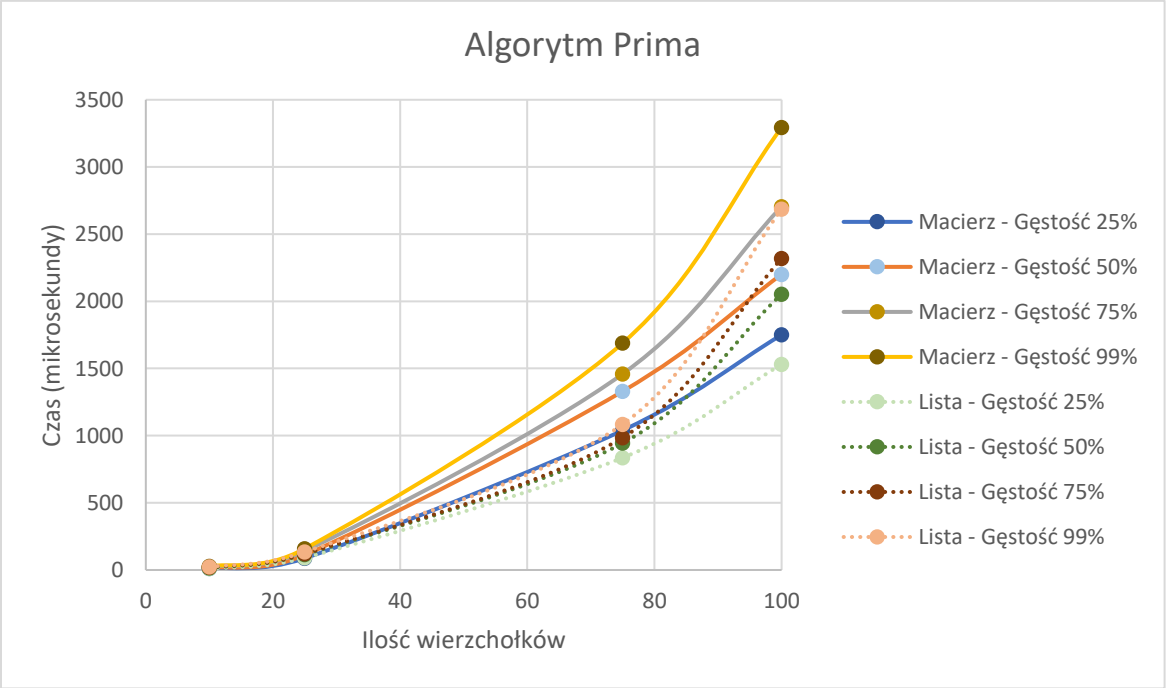
Graf bezkierunkowy							
Algorytm Kruskala							
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
10				25			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
11,949	19,273	23,643	23,791	90,804	120,067	132,938	148,487
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
75				100			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
958,869	1148,296	1383,378	1536,045	1719,426	2094,252	2591,879	3052,488
Lista							
Ilość wierzchołków				Ilość wierzchołków			
10				25			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
13,889	18,348	19,711	20,581	80,134	100,201	105,005	122,434
Lista							
Ilość wierzchołków				Ilość wierzchołków			
75				100			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
907,227	958,563	1116,129	1200,56	1469,292	1796,712	2023,454	2404,753

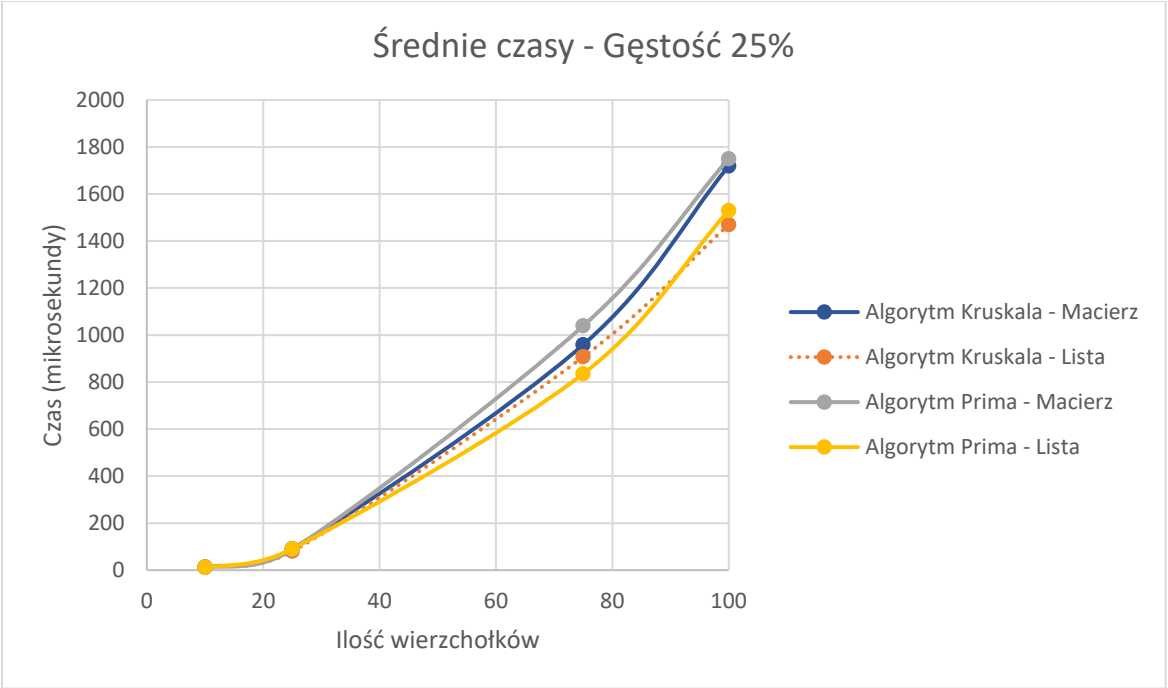
Graf bezkierunkowy

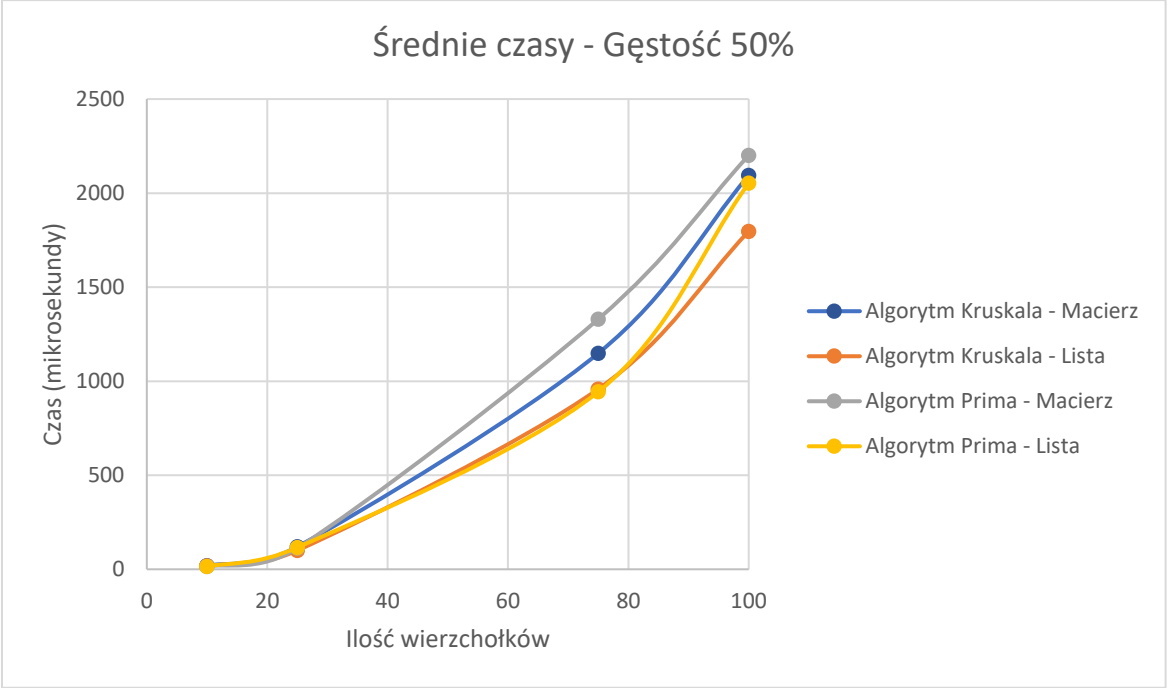
Algorytm Prima							
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
10				25			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
11,394	15,105	22,942	26,978	87,607	112,34	142,421	158,295
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
75				100			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
1039,081	1329,963	1459,495	1688,941	1750,202	2200,744	2704,13	3293,314
Lista							
Ilość wierzchołków				Ilość wierzchołków			
10				25			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
10,785	15,111	19,752	22,791	90,65	115,862	120,373	133,943
Lista							
Ilość wierzchołków				Ilość wierzchołków			
75				100			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
835,491	943,981	985,386	1084,253	1530,004	2053,016	2318,854	2685,728

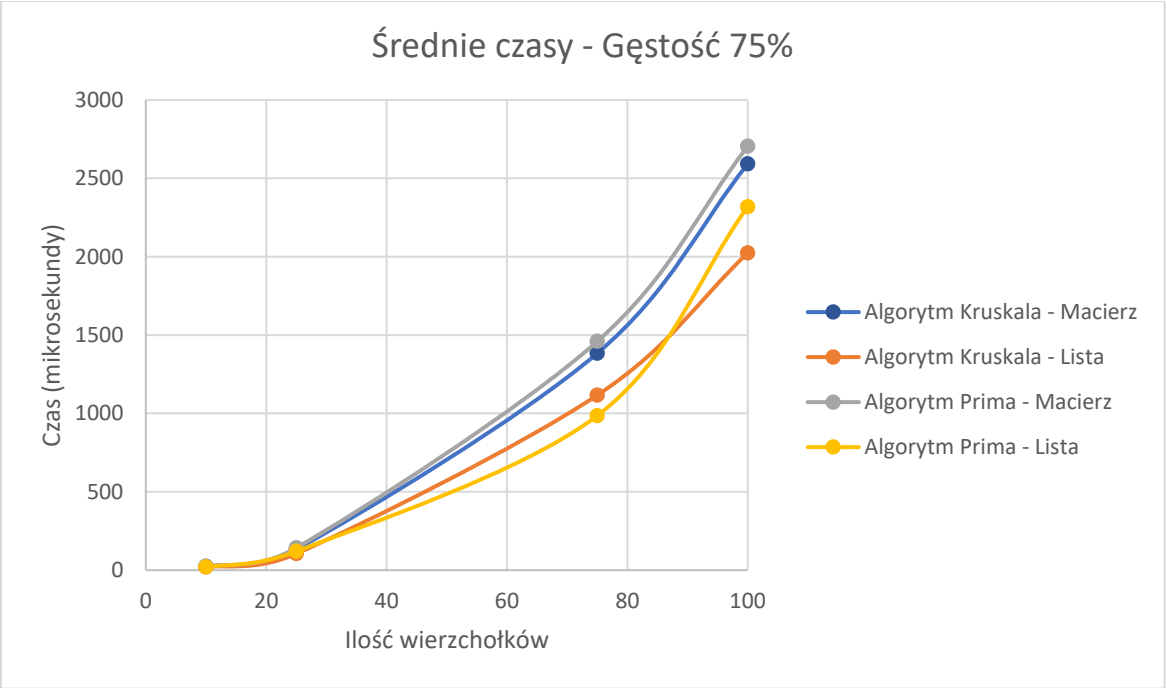
Wykresy

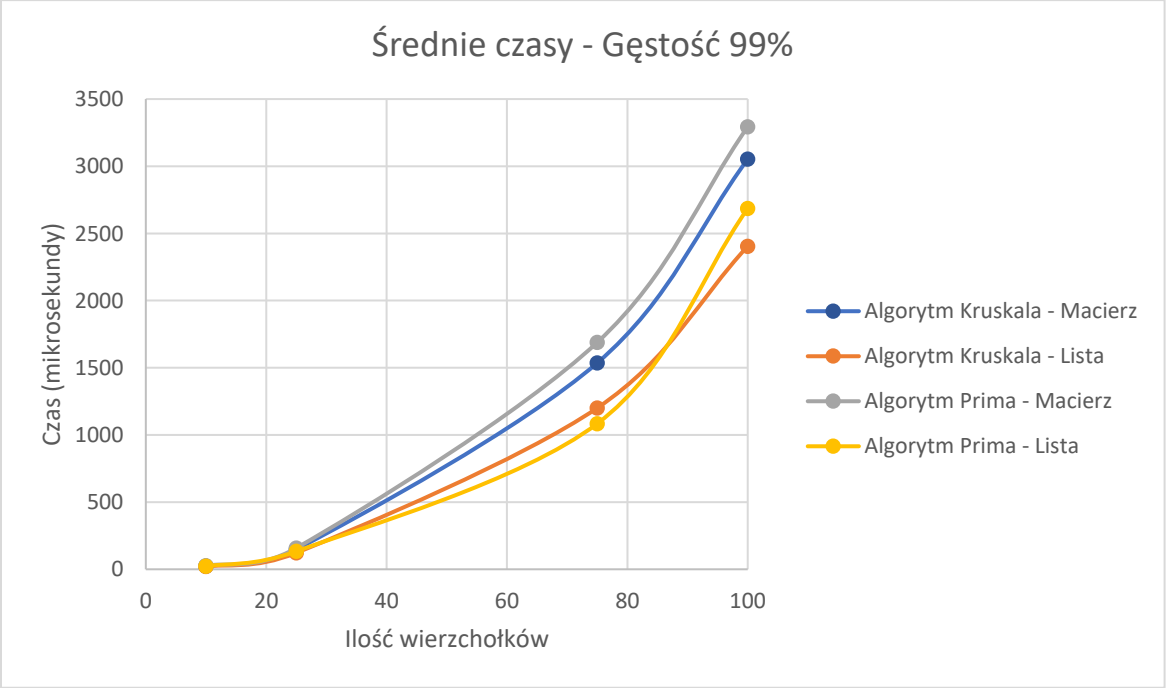

Algorytm Kruskala
Czas (mikrosekundy)
Ilość wierzchołków
0
500
1000
1500
2000
2500
3000
3500
0
20
40
60
80
100
Macierz - Gęstość 25%
Macierz - Gęstość 50%
Macierz - Gęstość 75%
Macierz - Gęstość 99%
Lista - Gęstość 25%
Lista - Gęstość 50%
Lista - Gęstość 75%
Lista - Gęstość 99%


Algorytm Prima
Czas (mikrosekundy)
Ilość wierzchołków
0
500
1000
1500
2000
2500
3000
3500
0
20
40
60
80
100
Macierz - Gęstość 25%
Macierz - Gęstość 50%
Macierz - Gęstość 75%
Macierz - Gęstość 99%
Lista - Gęstość 25%
Lista - Gęstość 50%
Lista - Gęstość 75%
Lista - Gęstość 99%


Średnie czasy - Gęstość 25%
Czas (mikrosekundy)
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0
20
40
60
80
100
Ilość wierzchołków
Algorytm Kruskala - Macierz
Algorytm Kruskala - Lista
Algorytm Prima - Macierz
Algorytm Prima - Lista


Średnie czasy - Gęstość 50%
Czas (mikrosekundy)
2500
2000
1500
1000
500
0
0
20
40
60
80
100
Ilość wierzchołków
Algorytm Kruskala - Macierz
Algorytm Kruskala - Lista
Algorytm Prima - Macierz
Algorytm Prima - Lista





Wnioski i uwagi

Algorytmy wykonują się w poprawny sposób, a w większości przypadków czas ich wykonywania rośnie zgodnie z założoną złożonością – im więcej jest wierzchołków i krawędzi tym dłuższy czas wykonywania. Czas wykonania obu algorytmów w obu implementacjach jest zbliżony w przypadku niskiej ilości/gęstości krawędzi, jednak im bardziej ta rosnie tym dłużej będzie wykonywał się algorytm bazujący na macierzy incydencji.

3. Graf skierowany i problem najkrótszej ścieżki

Użyte algorytmy i ich złożoności obliczeniowe

- Algorytm Dijkstry
  - Złożoność obliczeniowa : $\Theta(\mathrm{E} + \mathrm{V}\log V)$
- Algorytm Bellmana-Forda
  - Złożoność obliczeniowa : $\Theta(\mathrm{V} * \mathrm{E})$

Zaimplementowanie tych algorytmów przy użyciu macierzy może wydłużyć czas trwania algorytmu. W takim przypadku algorytm za każdym razem będzie musiał brać pod uwagę V – 1 możliwych połączeń z innymi wierzchołkami.

Opis algorytmów

Algorytm Dijkstry, polega na aktualizowaniu najkrótszych aktualnych ścieżek nieodwiedzonych wierzchołków. Początkowo droga wszystkich wierzchołków poza startowym wynosi nieskończoność, w programie jednak użyta została maksymalna wartość 32 bitowej liczby naturalnej. Algorytm aktualizuje ścieżki i drogi wierzchołków, do których wychodzą krawędzie, Po czym w kolejnej iteracji powtarza to działanie, biorąc pod uwagę wierzchołek o najmniejszej aktualnej drodze, jeśli nie jest odwiedzony.

Algorytm kończy działanie kiedy zostały odwiedzone wszystkie wierzchołki.

Algorytm Bellmana-Forda, wykonuje V-1 iteracji, kolejno zmieniając drogi wierzchołów, zaczynając od pierwszej krawędzi wierzchołka o indeksie 0 w każdej z nich. Algorytm w każdej iteracji przechodzi przez wszystkie krawędzie wszystkich wierzchołków. Jeśli dana krawędź zmniejszyłaby całkowitą drogę docelowego wierzchołka, następuje zmiana i krawędzie które wychodzą z tego wierzchołka prawdopodobnie także będą zmieniały drogę. Algorytm po V – 1 operacjach zwróci najkrótszą drogę do każdego wierzchołka.

Algorytm kończy działanie kiedy wykonane zostało V – 1 operacji, lub w trakcie całej operacji nie zaszła żadna zmiana – co oznacza że minimalna ścieżka została już znaleziona.

Wyniki

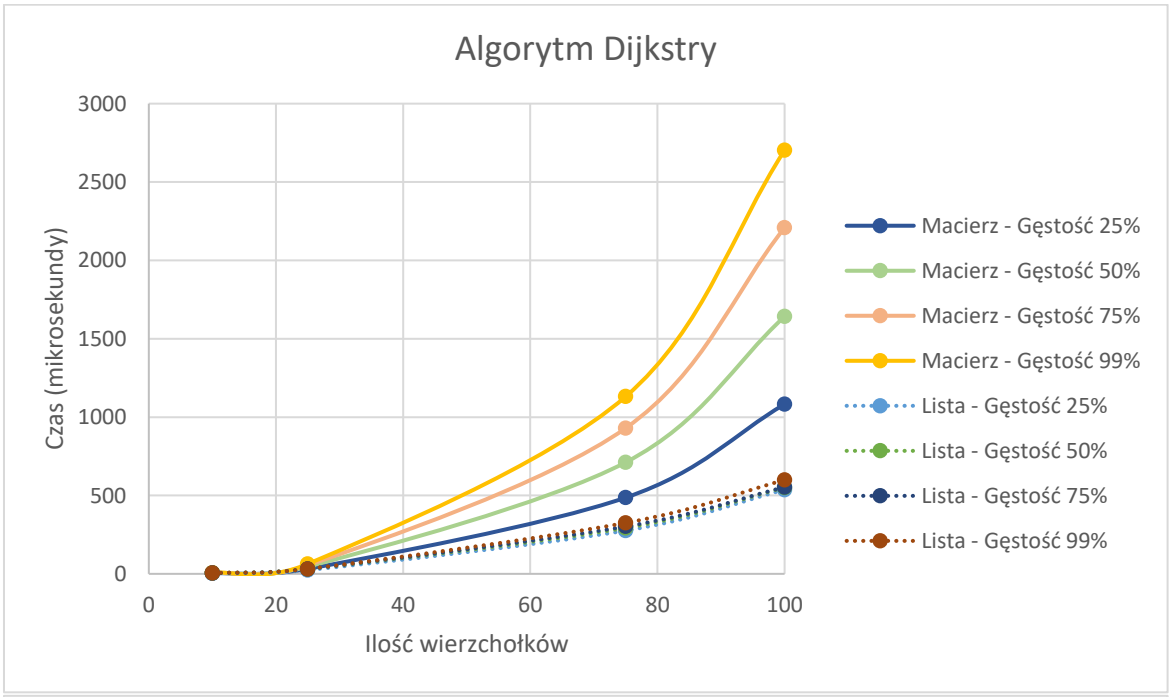
Poniżej znajdują się tabele przedstawiające średnie czasy wykonywania algorytmów SPP.

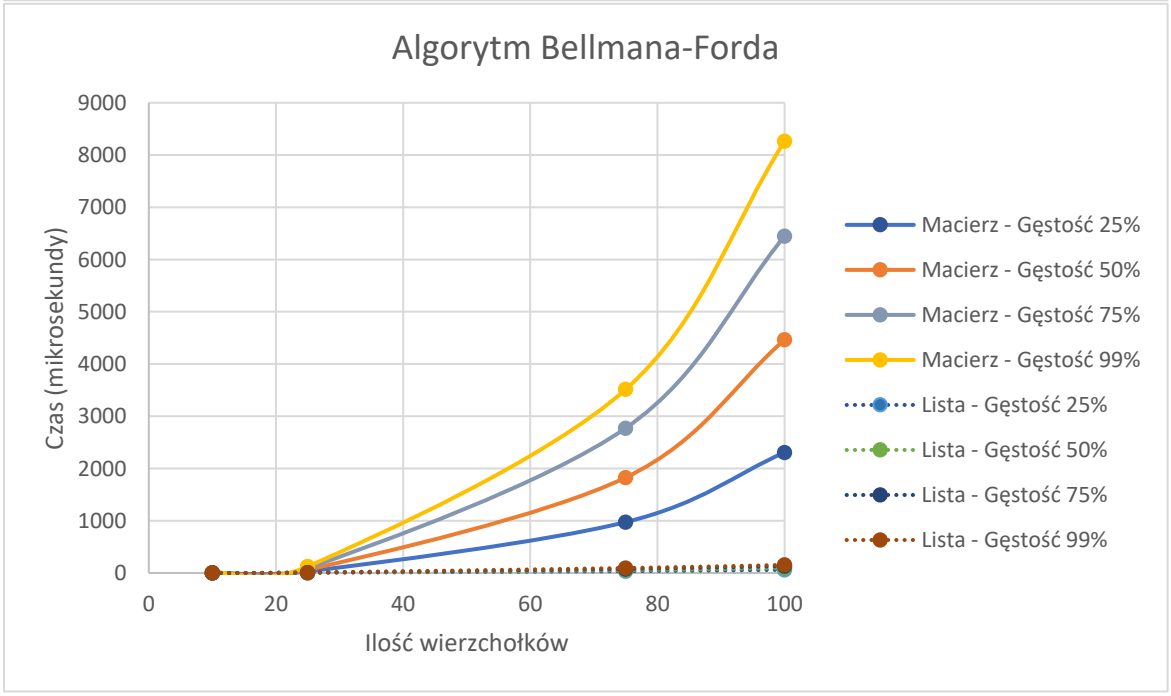
Graf kierunkowy							
Algorytm Dijkstry							
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
10				25			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
3,526	5,104	6,3	7,659	31,951	45,903	54,637	63,431
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
75				100			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
486,567	711,59	929,785	1131,3	1082,466	1641,792	2208,442	2702,348
Lista							
Ilość wierzchołków				Ilość wierzchołków			
10				25			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
3,494	4,975	5	5,979	24,025	31,122	31,157	31,447
Lista							
Ilość wierzchołków				Ilość wierzchołków			
75				100			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
275,923	292,953	302,277	325,342	536,324	550,319	552,915	598,607

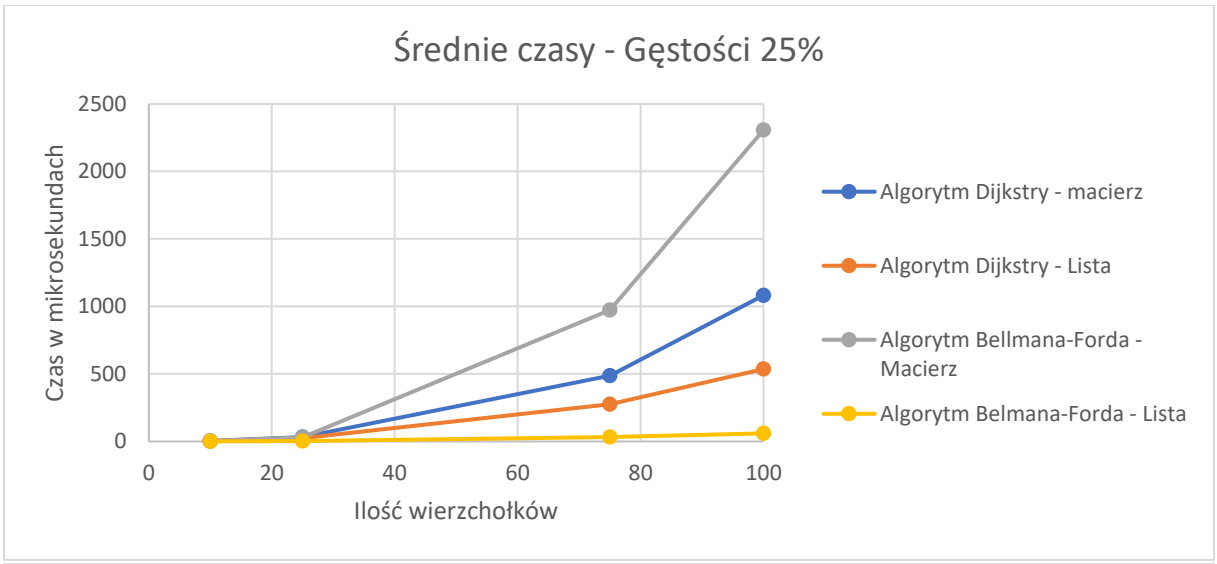
Graf kierunkowy

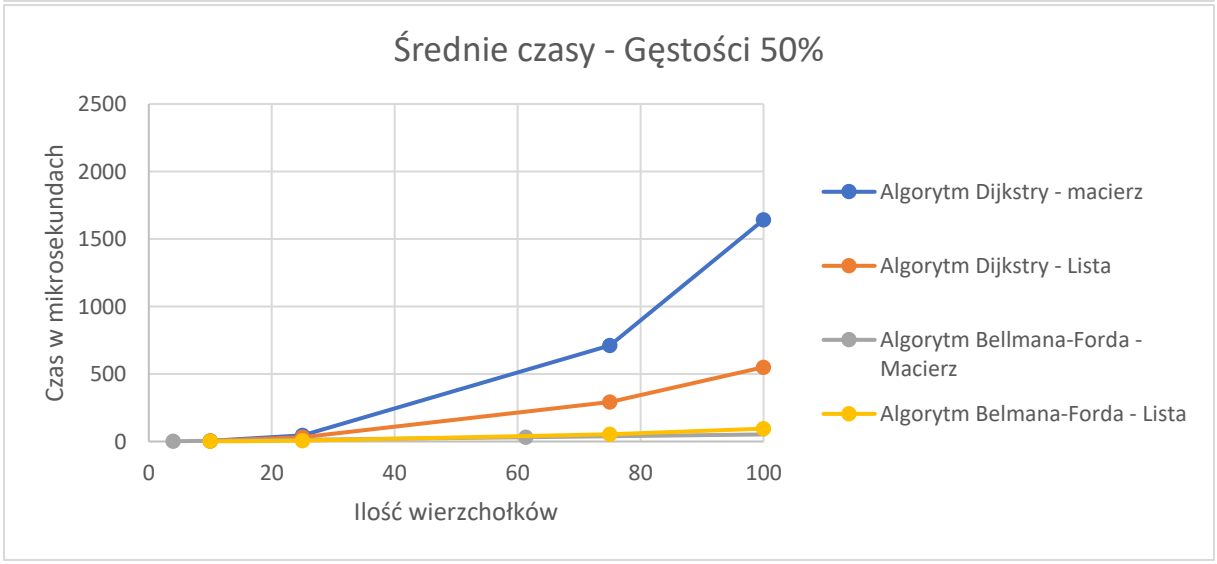
Algorytm Bellmana-Forda							
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
10				25			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
2,122	3,979	5,845	7,658	31,185	61,356	94,245	124,587
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
75				100			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
973,341	1827,388	2772,599	3514,318	2306,775	4464,349	6446,121	8263,222
Lista							
Ilość wierzchołków				Ilość wierzchołków			
10				25			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
0,801	1,246	1,393	1,53	3,789	5,772	7,679	9,494
Lista							
Ilość wierzchołków				Ilość wierzchołków			
75				100			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
33,437	53,777	77,141	95,432	59,906	95,46	130,286	155,004

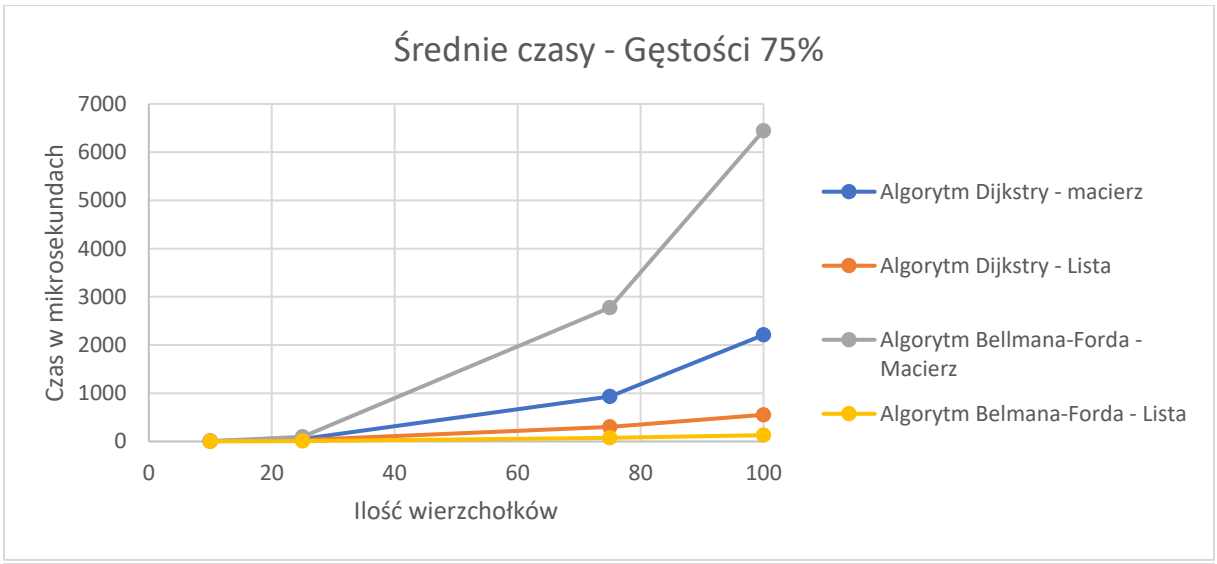
Wykresy

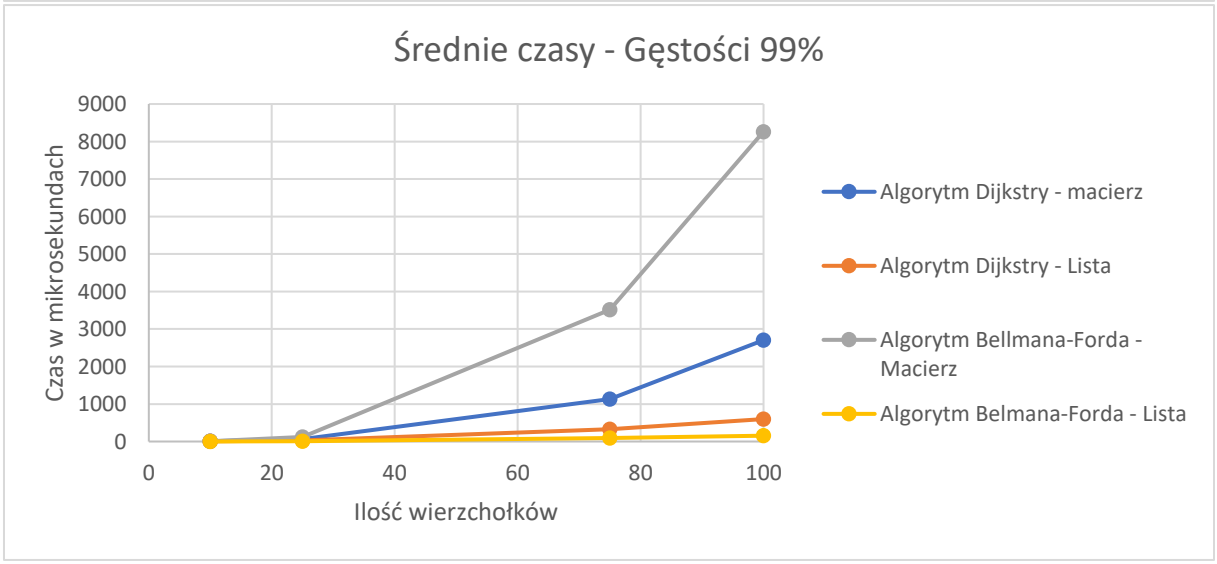

Algorytm Dijkstry
Czas (mikrosekundy)
3000
2500
2000
1500
1000
500
0
0
20
40
60
80
100
Ilość wierzchołków
Macierz - Gęstość 25%
Macierz - Gęstość 50%
Macierz - Gęstość 75%
Macierz - Gęstość 99%
Lista - Gęstość 25%
Lista - Gęstość 50%
Lista - Gęstość 75%
Lista - Gęstość 99%


Algorytm Bellmana-Forda
Czas (mikrosekundy)
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
Ilość wierzchołków
Macierz - Gęstość 25%
Macierz - Gęstość 50%
Macierz - Gęstość 75%
Macierz - Gęstość 99%
Lista - Gęstość 25%
Lista - Gęstość 50%
Lista - Gęstość 75%
Lista - Gęstość 99%


Średnie czasy - Gęstości 25%
Czas w mikrosekundach
2500
2000
1500
1000
500
0
0
20
40
60
80
100
Ilość wierzchołków
Algorytm Dijkstry - macierz
Algorytm Dijkstry - Lista
Algorytm Bellmana-Forda - Macierz
Algorytm Belmana-Forda - Lista


Średnie czasy - Gęstości 50%
Czas w mikrosekundach
2500
2000
1500
1000
500
0
0
20
40
60
80
100
Ilość wierzchołków
Algorytm Dijkstry - macierz
Algorytm Dijkstry - Lista
Algorytm Bellmana-Forda - Macierz
Algorytm Belmana-Forda - Lista





Wnioski

W wypadku algorytmów SPP zaimplementowanych przeze mnie – ich czas wykonywania rośnie w przewidziany sposób. Algorytm Bellmana-Forda wykonuje się w krótszym czasie niż algorytm Dijkstry - algorytm Dijkstry ma niższą złożoność obliczeniową. W przypadku problemu SPP wyraźnie widać, że jeden algorytm wykonuje się o wiele szybciej niż drugi.