

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	3
Opis eksperymentu	3
2. Graf nieskierowany i algorytmy minimalnego drzewa rozpinającego	4
Użyte algorytmy i ich złożoności obliczeniowe	4
Opis algorytmów	4
Wyniki	5
Wykresy	7
Wnioski i uwagi	9
3. Graf skierowany i problem najkrótszej ścieżki	10
Użyte algorytmy i ich złożoności obliczeniowe	10
Opis algorytmów	10
Wyniki	11
Wykresy	13
Wnioski	15

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 .xlsb 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 : $\Theta(E * \log V) = \Theta(E * \log E)$
- Algorytm Prim'a
 - Złożoność obliczeniowa : $\Theta(E * \log V)$

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 $V - 1$ 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 cykli w moim projekcie zostało zaimplementowane przy pomocy algorytmu Union-Find. Sortowanie krawędzi, 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

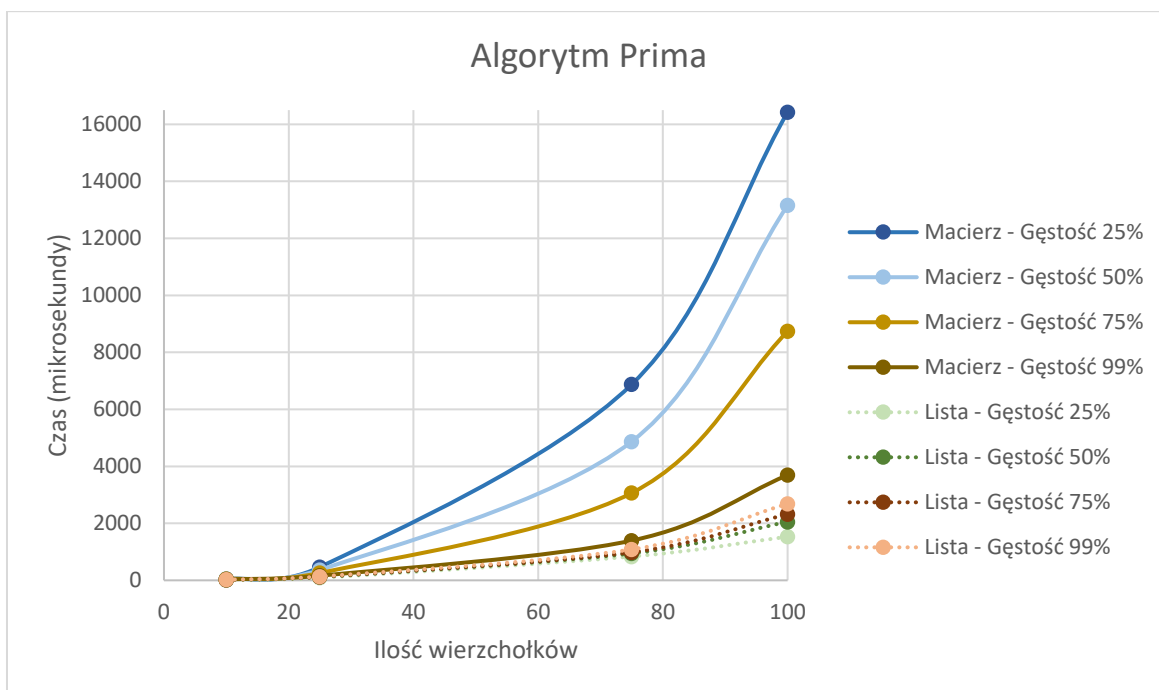
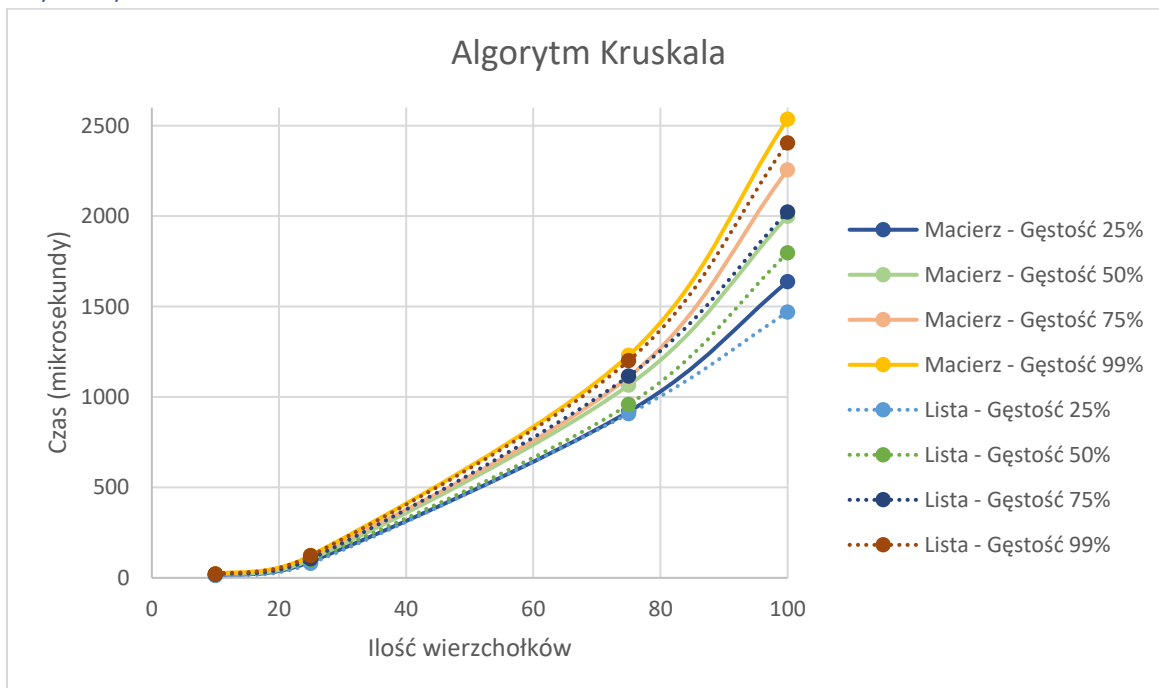
Wyniki

Poniżej znajdują się tabele przedstawiające średnie czasy wykonywania algorytmów MST.

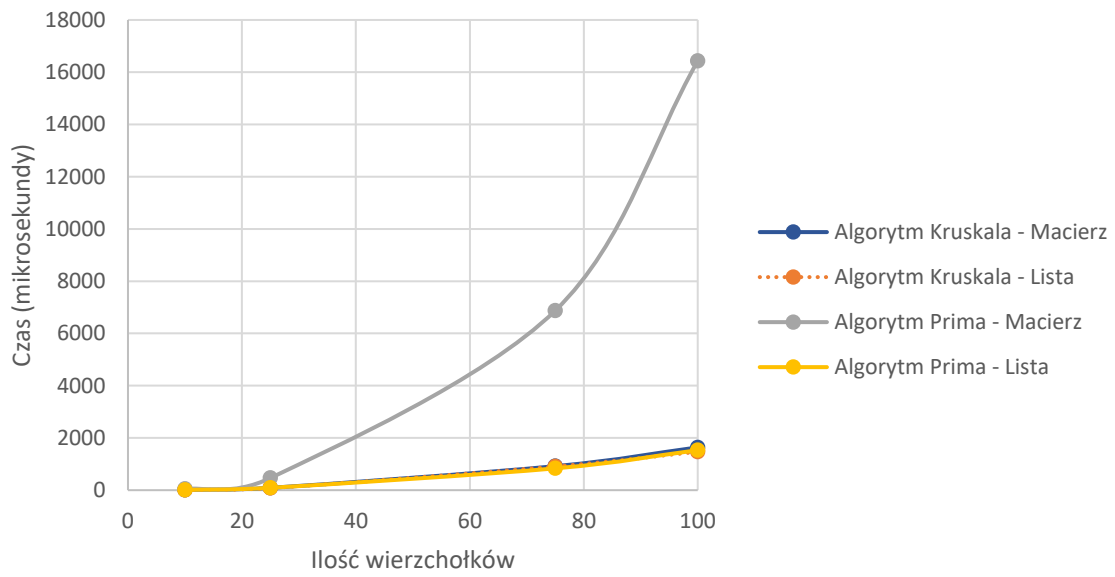
Graf bezkierunkowy							
Algorytm Kruskala							
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
10				25			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
13,275	18,24	19,876	22,211	89,534	101,8	118,022	123,869
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
75				100			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
919,198	1065,371	1110,921	1230,534	1637,489	1998,99	2255,33	2535,864
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			
48,322	53,543	44,415	33,156	466,501	366,122	259,049	162,539
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
75				100			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
6875,341	4871,076	3064,612	1394,539	16427,02	13158,38	8737,792	3692,905
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

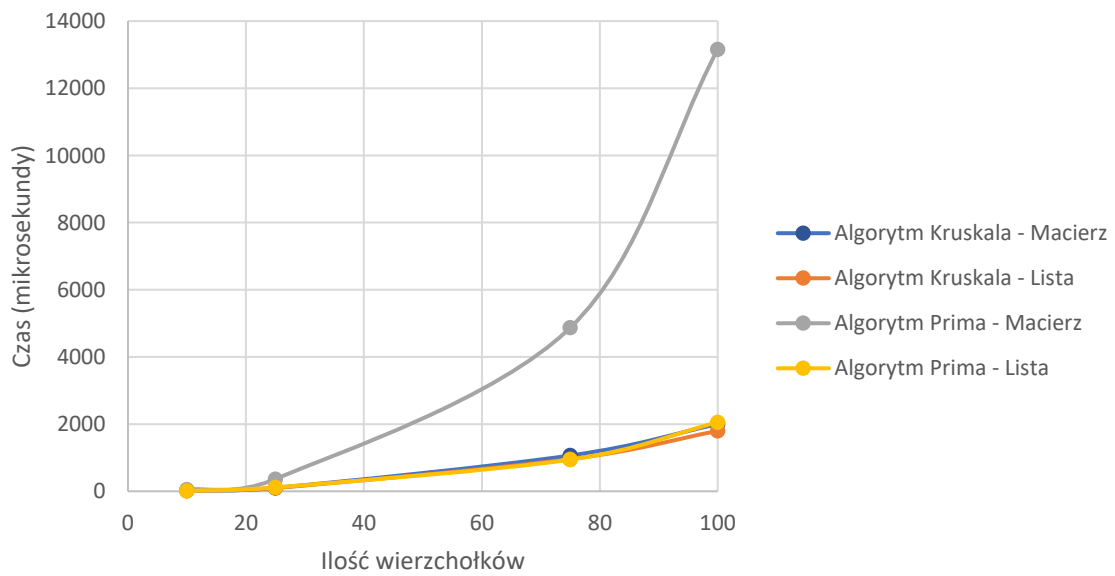
Wykresy

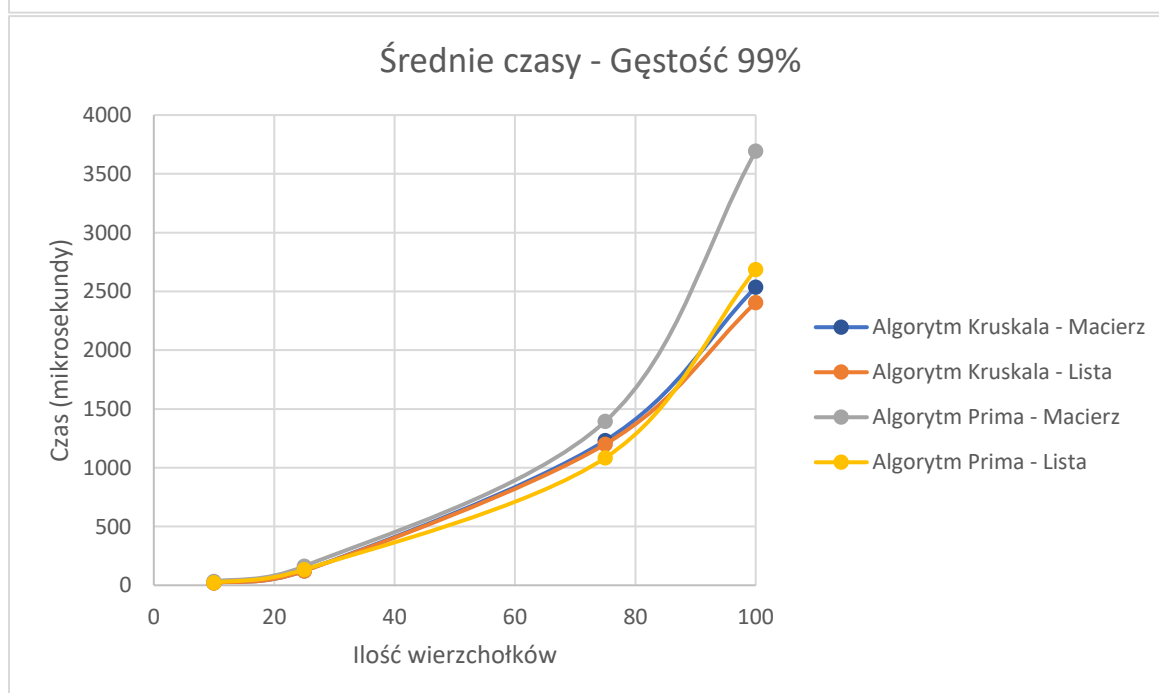
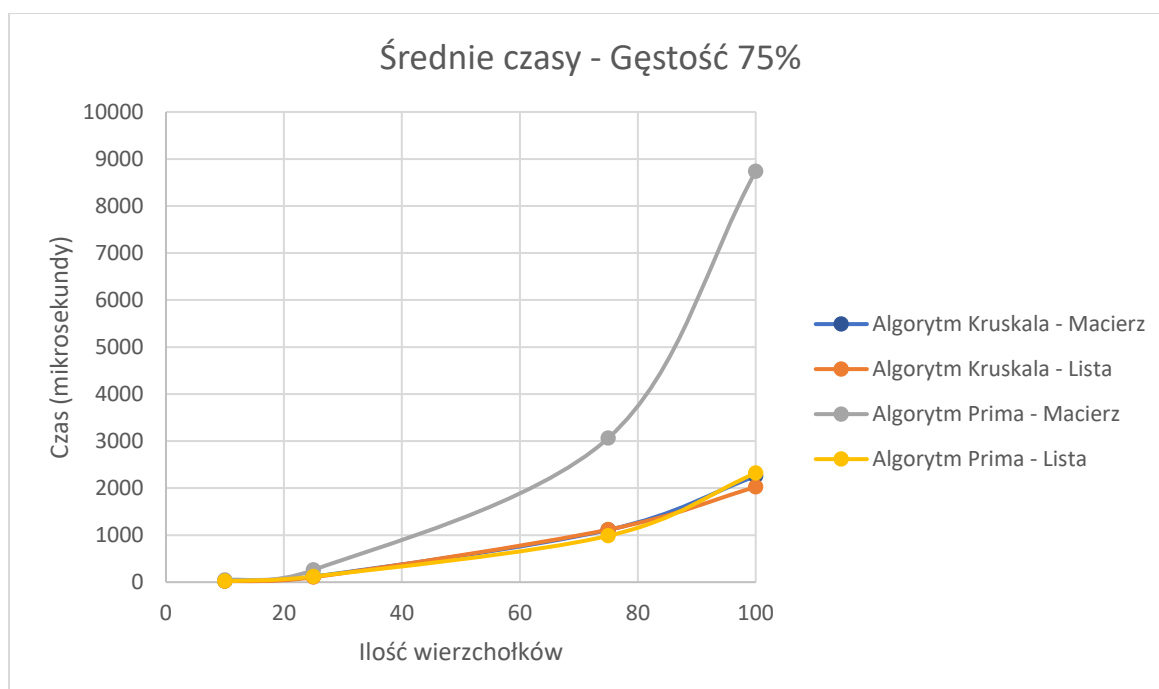


Średnie czasy - Gęstość 25%



Średnie czasy - Gęstość 50%





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. Wyjątkiem jest Algorytm Prima zaimplementowany macierzą – jego czas wykonywania jest największy dla najmniejszej sprawdzonej gęstości. Mogę to przypisać albo do niepoprawnego zaimplementowania tego algorytmu, albo uznać że jest to przypadek specyficzny do architektury dzisiejszych procesorów.

3. Graf skierowany i problem najkrótszej ścieżki

Użyte algorytmy i ich złożoności obliczeniowe

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

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 wierzchołków, 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.

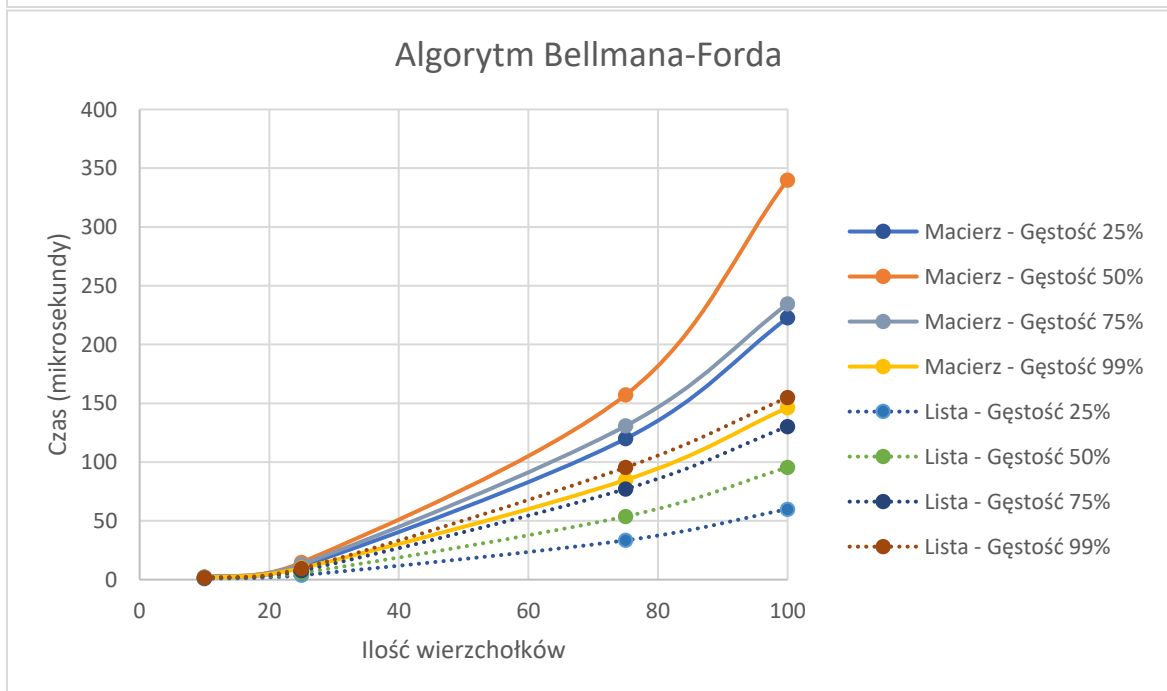
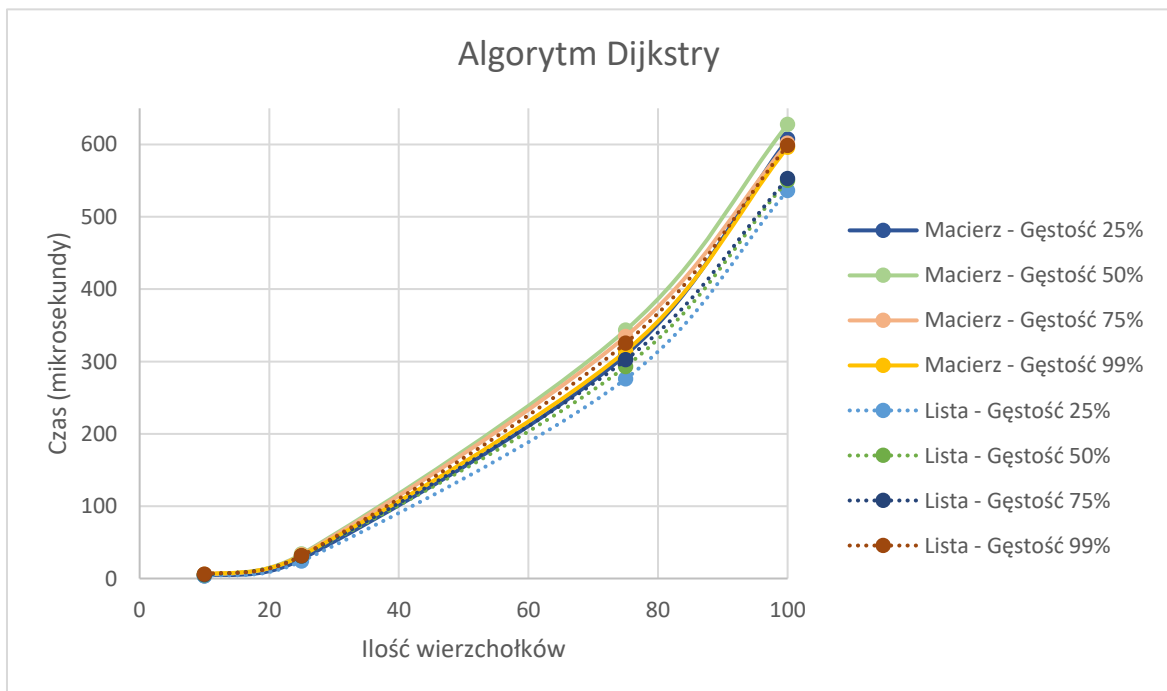
Wyniki

Poniżej znajdują się tabele przedstawiające średnie czasy wykonywania algorytmów SPP.

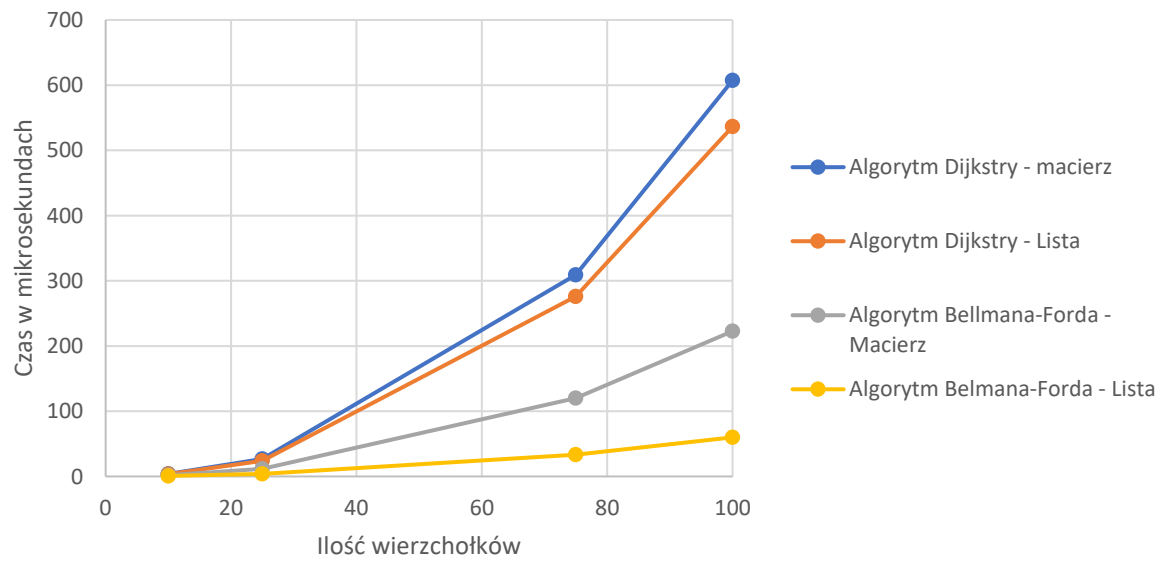
Graf kierunkowy							
Algorytm Dijkstry							
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
10				25			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
3,879	4,537	5,001	5,407	26,925	33,808	32,771	31,759
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
75				100			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
309,119	343,335	334,567	314,168	607,256	627,763	601,746	596,067
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			
1,894	2,208	2,194	1,536	11,504	14,688	13,079	9,817
Macierz							
Ilość wierzchołków				Ilość wierzchołków			
75				100			
Gęstość				Gęstość			
25%	50%	75%	99%	25%	50%	75%	99%
Czas w mikrosekundach				Czas w mikrosekundach			
119,885	157,143	130,76	84,774	222,691	339,925	234,528	146,279
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

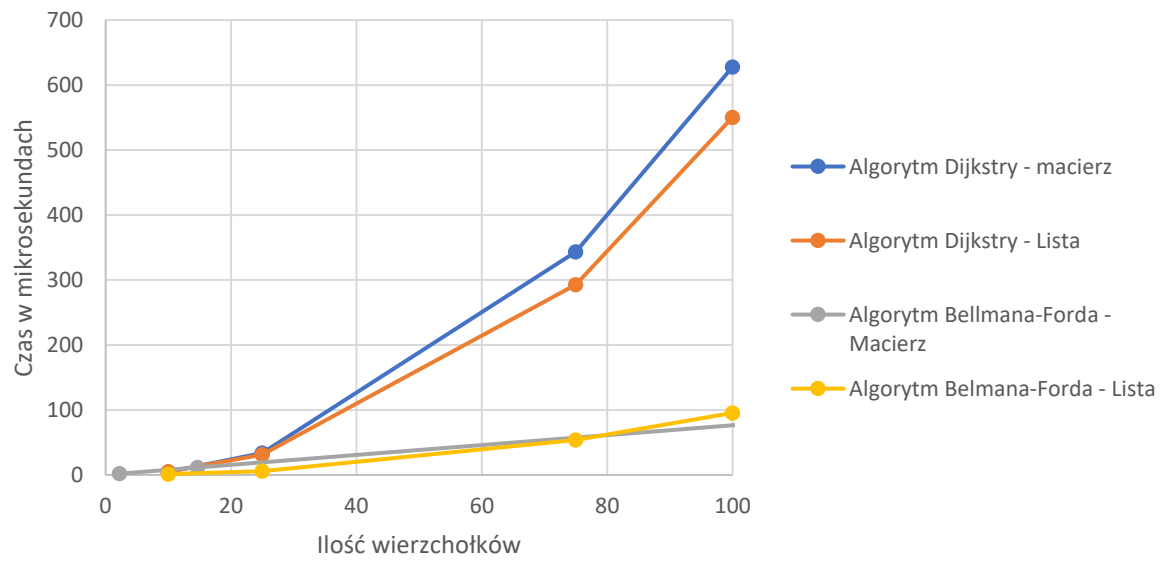
Wykresy

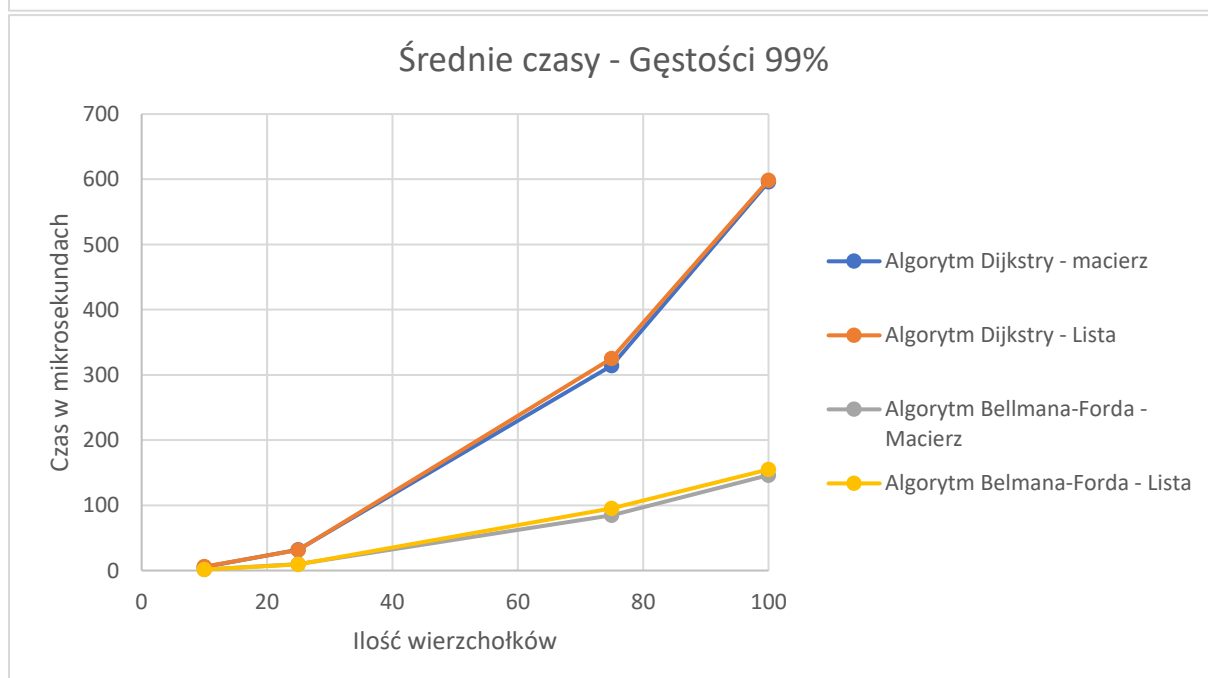
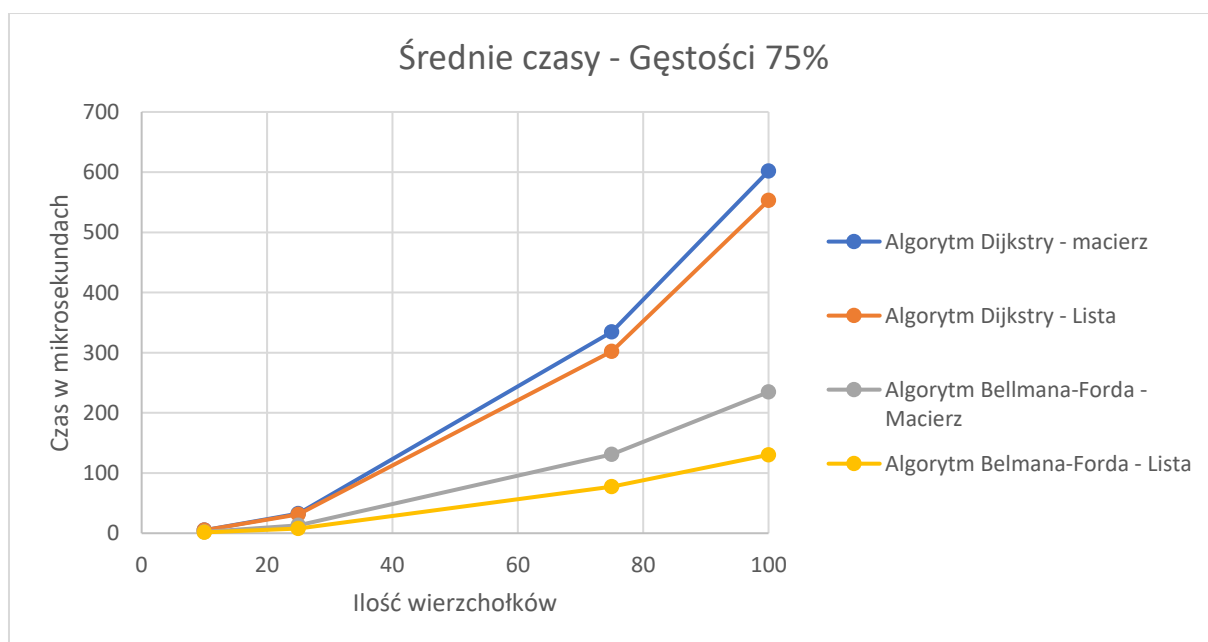


Średnie czasy - Gęstości 25%



Średnie czasy - Gęstości 50%





Wnioski

W wypadku algorytmów SPP zaimplementowanych przeze mnie – ich czas wykonywania rośnie w przewidziany sposób. Jednak z powodu implementacji algorytm Bellmana-Forda wykonuje się w krótszym czasie niż algorytm Dijkstry – pomimo tego że algorytm Dijkstry technicznie ma niższą złożoność obliczeniową.