Kacper Sarzyński 248856

Struktury Danych i Złożoność Obliczeniowa – Projekt

Zadanie Projektowe 2

Termin: Czwartek 11.15 – 13.00 TN

Prowadzący: Dr. Inż. Jarosław Mierzwa

1. Wstęp

   Celem tego zadania była implementacja dwóch reprezentacji grafu: w postaci Macierzy sąsiedztwa oraz listy sąsiadów. Następnie za zadanie mięliśmy zaimplementować algorytmy znalezienia MST (algorytm Prima oraz Kruskala) i algorytmy wyznaczenia najkrótszej drogi z podanego wierzchołka do każdego innego (algorytm Djkstry i algorytm Forda-Bellmana).
   Teoretyczne złożoności znajdują się w poniższej tabeli

Algorytm:	Złożoność
Algorytm Prima	O(E * log(V))
Algorytm Kruskala	O(E * log(V))
Algorytm Djkstry	O(E + V * log(V))
Algorytm Forda_Bellmana	O(E * V)

2. Plan Eksperymentu

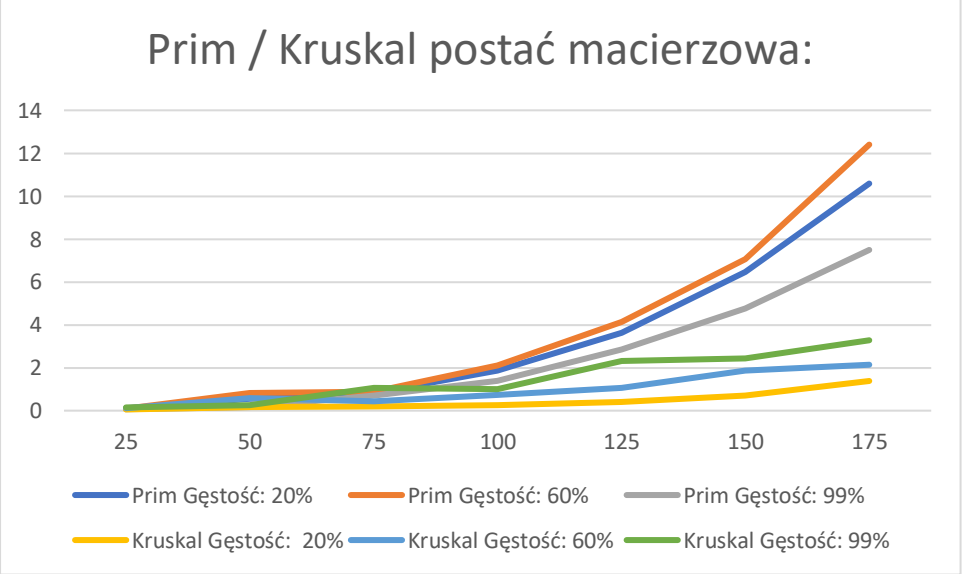
   Pomiary przeprowadzane były na następujących wielkościach grafów:
   25, 50, 75, 100, 125, 150, 175 wierzchołków, dla gęstości równych 20%, 60%, 99%
   Pomiary powstały przy użyciu *queryperformancecounter.* Zakres wartości kosztów krawędzi był duży, od 1 do wielokrotnośći liczby wierzchołków.

   Grafy generowane były losowo tworzone były grafy bez krawędzi o podanej ilości wierzchołków, następnie dodawana była wymagana ilość krawędzi, dla testów zapewniona była spójność grafów. W programie znajduję się funkcja pokazująca przykładowy test, z zapisem danych do pliku txt, w czasie pomiarów używana była podobna, jednak wykonująca więcej testów, i uśredniająca wynik.

3. Wyniki: (czasy podane są w ms)

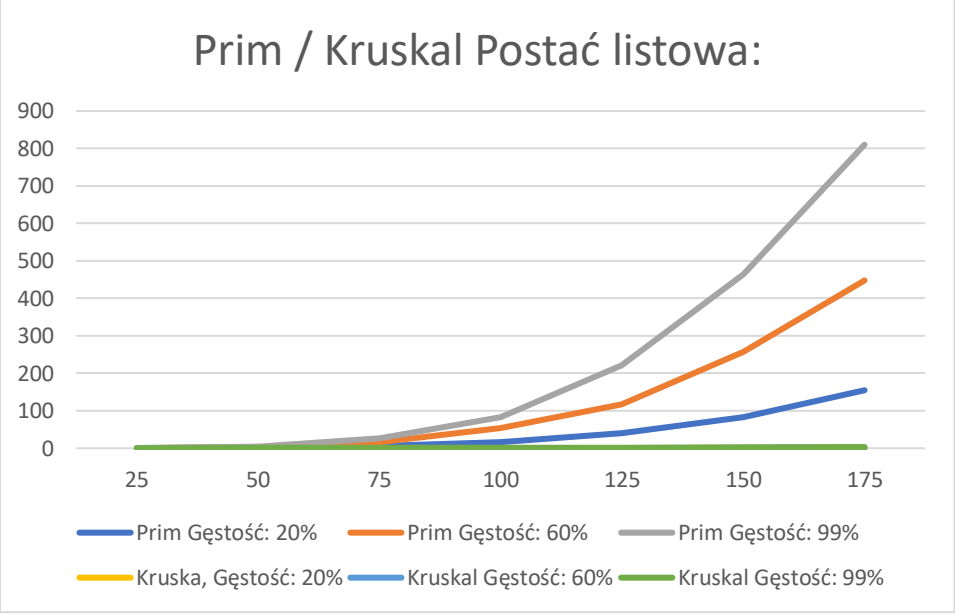
a) Prim / Kruskal postać macierzowa:

Prim Gęstość: 20%	0,0956	0,5529	0,8292	1,8696	3,639	6,4582	10,5865
Prim Gęstość: 60%	0,0947	0,8136	0,8726	2,119	4,1395	7,0721	12,402
Prim Gęstość: 99%	0,0675	0,1793	0,7183	1,4022	2,8533	4,7672	7,4923
Kruskal Gęstość: 20%	0,056	0,1758	0,1944	0,2696	0,4192	0,712	1,3822
Kruskal Gęstość: 60%	0,1148	0,5913	0,4313	0,7214	1,0498	1,874	2,1401
Kruskal Gęstość: 99%	0,1498	0,2433	1,0543	1,0016	2,3015	2,4293	3,2802
wierzchołki	25	50	75	100	125	150	175



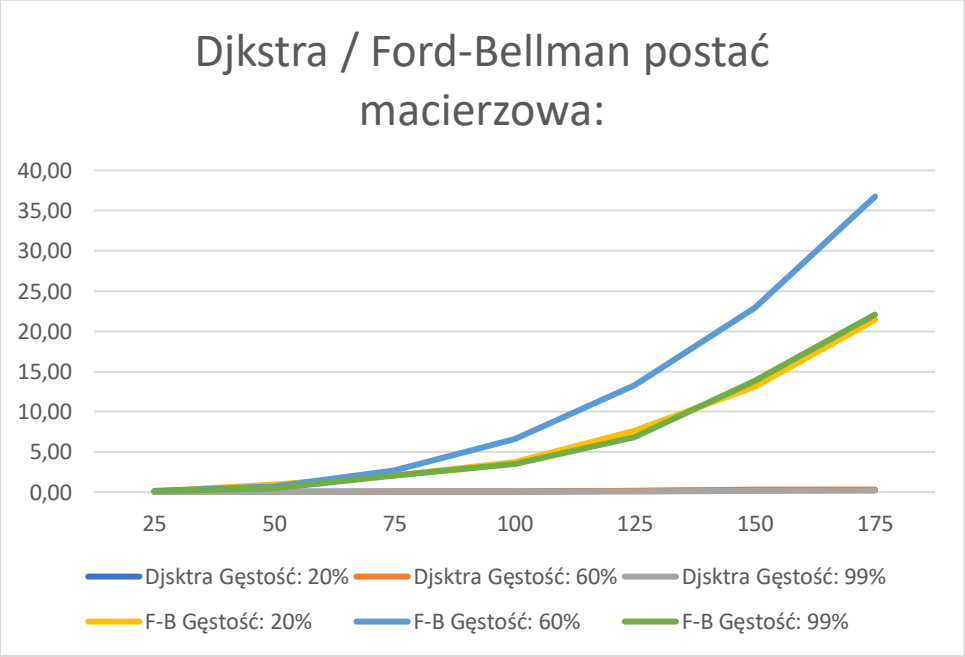
b) Prim / Kruskal Postać listowa:

Prim Gęstość: 20%	0,1777	2,3415	6,4533	16,4515	40,2797	82,2044	154,778
Prim Gęstość: 60%	0,4951	3,0468	16,3855	53,1243	117,46	257,476	447,81
Prim Gęstość: 99%	0,7395	4,9802	26,8058	83,6008	221,129	464,679	810,182
Kruska, Gęstość: 20%	0,0443	0,1356	0,2015	0,2799	0,4514	0,5664	0,7488
Kruskal Gęstość: 60%	0,0898	0,1533	0,3598	0,6437	1,0663	1,526	1,983
Kruskal Gęstość: 99%	0,1347	0,23	0,5952	0,9589	1,7156	2,4023	3,2265
wierzchołki	25	50	75	100	125	150	175



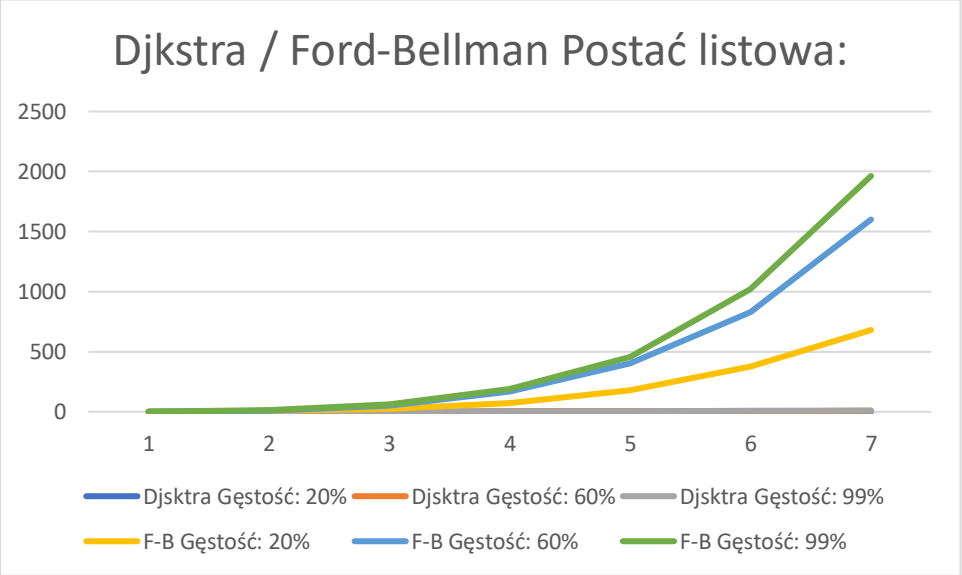
c) Djkstra / Ford-Bellman postać macierzowa:

Djsktra Gęstość: 20%	0,0136	0,0481	0,0488	0,0778	0,1183	0,2492	0,2577
Djsktra Gęstość: 60%	0,0193	0,0293	0,0618	0,1061	0,163	0,2811	0,3288
Djsktra Gęstość: 99%	0,017	0,0206	0,0713	0,0747	0,1142	0,1655	0,2082
F-B Gęstość: 20%	0,0929	0,943	2,1028	3,7041	7,6604	13,1555	21,4296
F-B Gęstość: 60%	0,1586	0,7282	2,7151	6,572	13,3044	22,9382	36,7531
F-B Gęstość: 99%	0,1296	0,456	2,0638	3,515	6,8687	13,8204	22,0809
wierzchołki	25	50	75	100	125	150	175



d) Djkstra / Ford-Bellman Postać listowa:

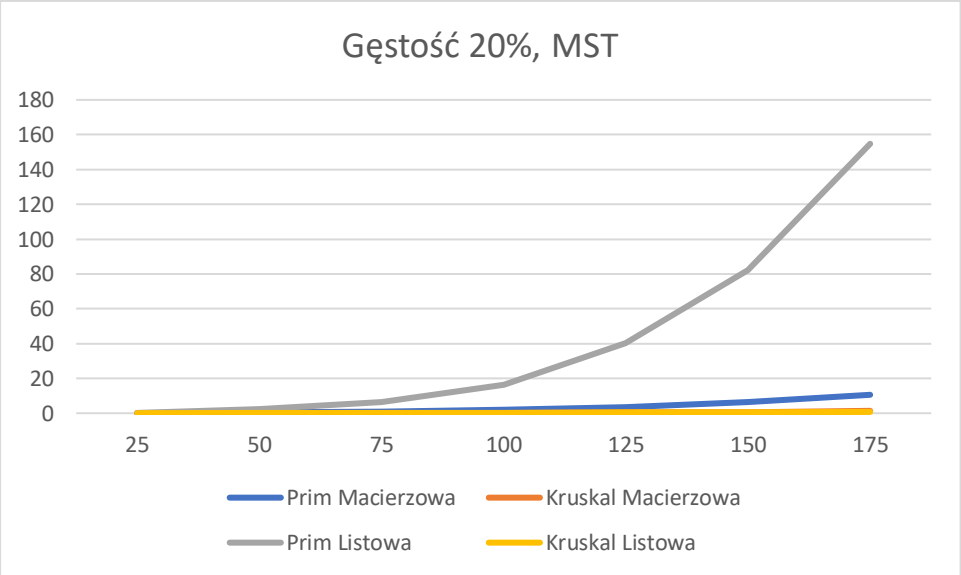
Djsktra Gęstość: 20%	0,0403	0,2405	0,3298	0,7696	1,4252	2,6151	3,9218
Djsktra Gęstość: 60%	0,2444	0,2026	0,745	1,8952	3,205	5,3782	9,2076
Djsktra Gęstość: 99%	0,0776	0,2291	0,835	1,8666	3,7687	7,1076	10,618
F-B Gęstość: 20%	0,5279	10,5911	23,3002	70,204	180,88	374,084	680,776
F-B Gęstość: 60%	1,5386	10,376	52,8382	167,305	400,791	829,805	1601,1
F-B Gęstość: 99%	1,4258	13,0023	59,6089	189,55	456,911	1020,86	1963,32
wierzchołki	25	50	75	100	125	150	175



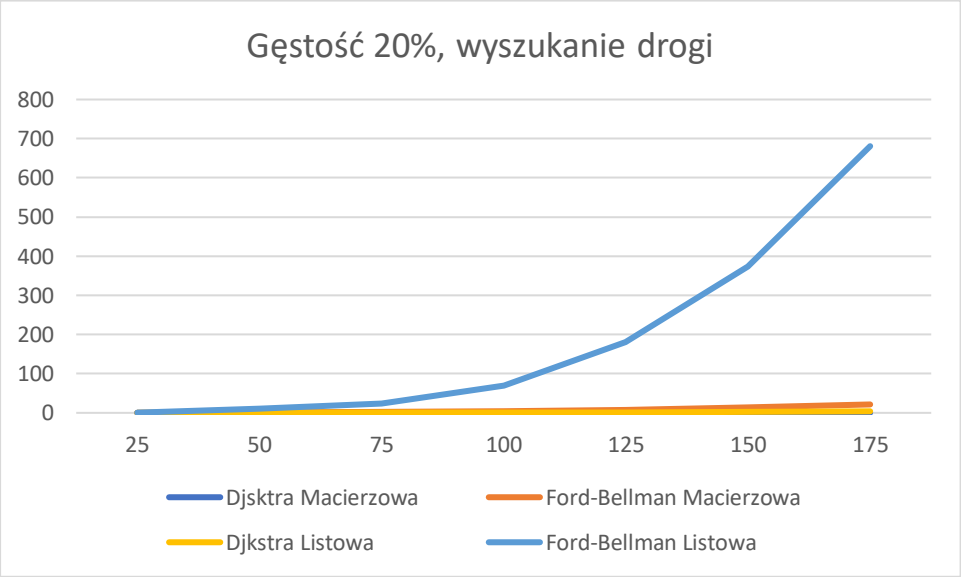
Zestawienie algorytmów tego samego typu na obu reprezentacjach dla danej gęstości:

e) Gęstość 20%

Prim Macierzowa	0,0956	0,5529	0,8292	1,8696	3,639	6,4582	10,586
Kruskal Macierzowa	0,056	0,1758	0,1944	0,2696	0,4192	0,712	1,3822
Prim Listowa	0,1777	2,3415	6,4533	16,4515	40,2797	82,2044	154,77
Kruskal Listowa	0,0443	0,1356	0,2015	0,2799	0,4514	0,5664	0,7488
Wierzchołki	25	50	75	100	125	150	175

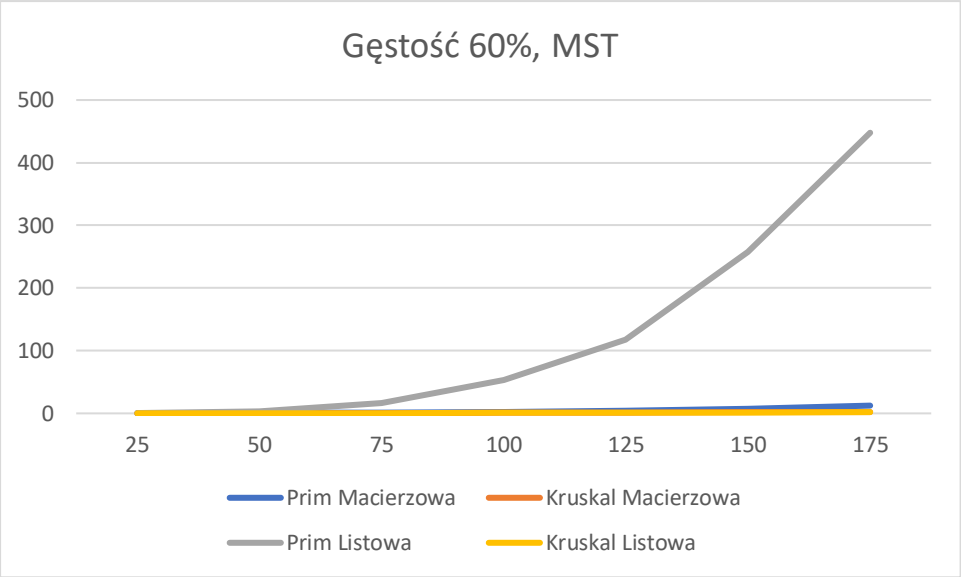


Djsktra Macierzowa	0,0136	0,0481	0,0488	0,0778	0,1183	0,2492	0,2577
Ford-Bellman Macierzowa	0,0929	0,943	2,1028	3,7041	7,6604	13,1555	21,429
Djkstra Listowa	0,0403	0,2405	0,3298	0,7696	1,4252	2,6151	3,9218
Ford-Bellman Listowa	0,5279	10,5911	23,3002	70,204	180,88	374,084	680,77
Wierzchołki	25	50	75	100	125	150	175

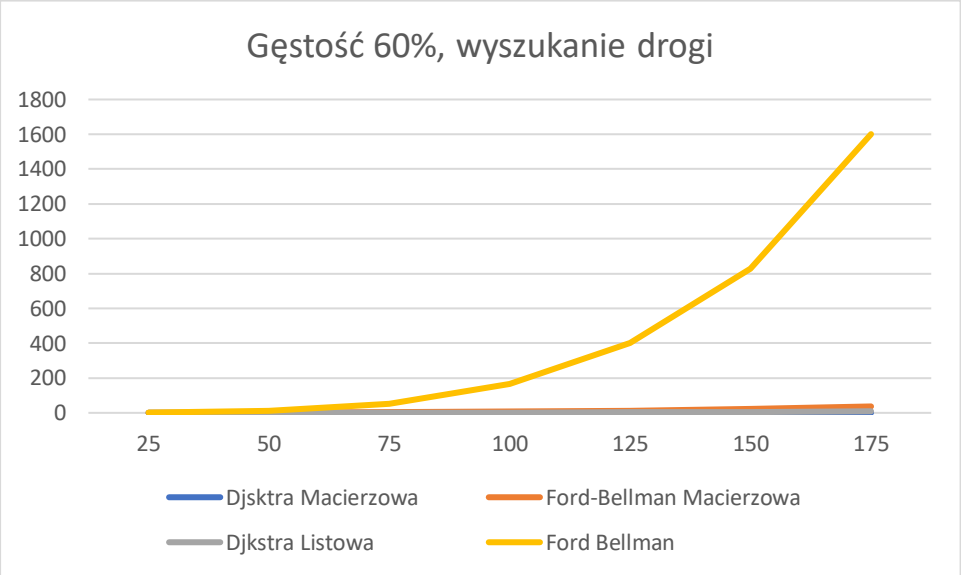


f) Gęstość 60%

Prim Macierzowa	0,0947	0,8136	0,8726	2,119	4,1395	7,0721	12,402
Kruskal Macierzowa	0,1148	0,5913	0,4313	0,7214	1,0498	1,874	2,1401
Prim Listowa	0,4951	3,0468	16,3855	53,1243	117,46	257,476	447,81
Kruskal Listowa	0,0898	0,1533	0,3598	0,6437	1,0663	1,526	1,983
Wierzchołki	25	50	75	100	125	150	175

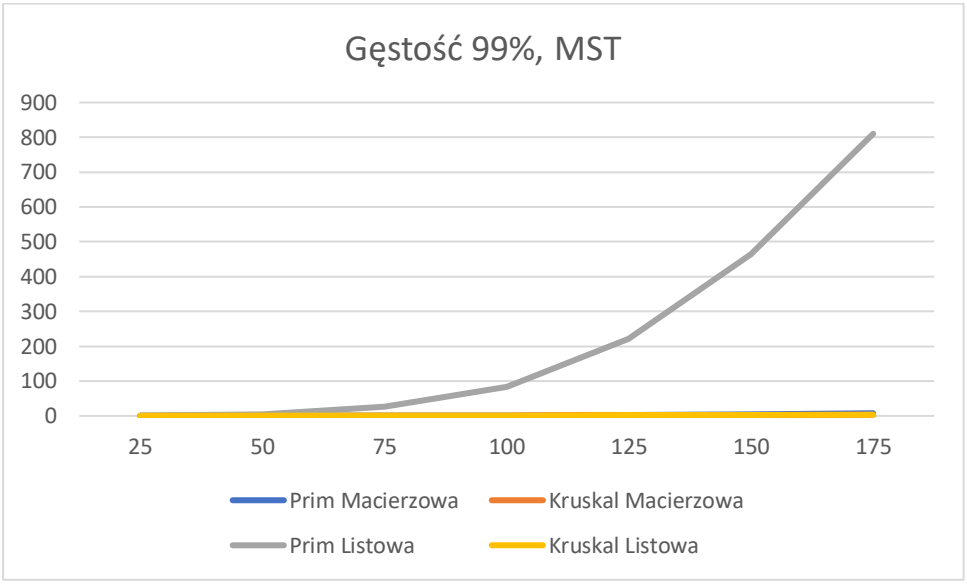


Djsktra Macierzowa	0,0193	0,0293	0,0618	0,1061	0,163	0,2811	0,328
Ford-Bellman Macierzowa	0,1586	0,7282	2,7151	6,572	13,3044	22,9382	36,75
Djkstra Listowa	0,2444	0,2026	0,745	1,8952	3,205	5,3782	9,207
Ford Bellman Listowa	1,5386	10,376	52,8382	167,305	400,791	829,805	1601
Wierzchołki	25	50	75	100	125	150	175

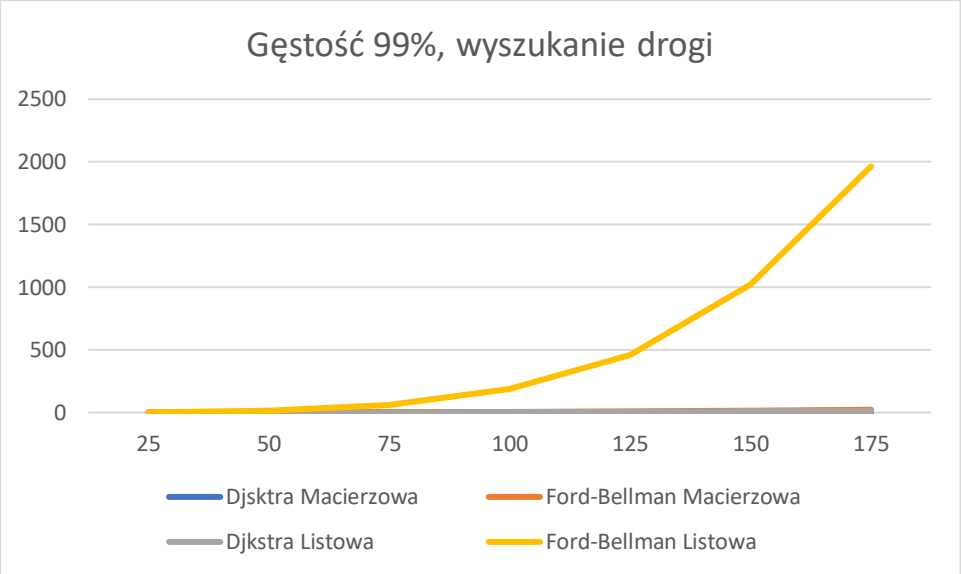


g) Gęstość 99%

Prim Macierzowa	0,067	0,1793	0,7183	1,4022	2,8533	4,7672	7,492
Kruskal Macierzowa	0,149	0,2433	1,0543	1,0016	2,3015	2,4293	3,280
Prim Listowa	0,739	4,9802	26,8058	83,6008	221,129	464,679	810,1
Kruskal Listowa	0,134	0,23	0,5952	0,9589	1,7156	2,4023	3,2265
Wierzchołki	25	50	75	100	125	150	175



Djsktra Macierzowa	0,017	0,0206	0,0713	0,0747	0,1142	0,1655	0,208
Ford-Bellman Macierzowa	0,1296	0,456	2,0638	3,515	6,8687	13,8204	22,08
Djkstra Listowa	0,0776	0,2291	0,835	1,8666	3,7687	7,1076	10,61
Ford-Bellman Listowa	1,4258	13,0023	59,6089	189,55	456,911	1020,86	1963
Wierzchołki	25	50	75	100	125	150	175



4. Wnioski:
   W algorytmie wyznaczającym MST oba algorytmy mają taką samą teoretyczną złożoność, z pomiarów wynika, że algorytm Kruskala jest szybszy, znaczna różnica może co prawda wynikać z nienajlepszej implementacji, ponadto sprawniej działa macierzowa reprezentacja.
   W algorytmie wyszukiwania najkrótszych ścieżek, lepiej prezentuje się implementacja algorytmu Djkstry, przy każdej gęstości najgorzej wypada algorytm Forda-Bellmana dla implementacji listowej, co może sugerować już złożoność.
   Porównując czasy działania algorytmów dla obu reprezentacji widać, że lepiej wypada macierzowa, co może częściowo wynikach z implementacji ale wiele operacji takich jak np. wyszukanie danej krawędzi jest szybsze. Reprezentacja macierzowa była według mnie łatwiejsza oraz bardziej intuicyjna do stworzenia, chociaż trzeba w niej przechowywać również puste krawędzie.
   W przeprowadzonych w późniejszym etapie testów wynika, że różnice w czasie działania algorytmów na tej samej reprezentacji większe są dla implementacji listowej.