Struktury danych i złożoność obliczeniowa – **projekt nr 2**

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

Autor: Filip Przygoński – 248892
Grupa zajęciowa: dr inż. Dariusz Banasiak - wtorek 15:15 TN

Wstęp

Projekt miał na celu zbadanie efektywności algorytmów grafowych wyznaczających minimalne drzewo rozpinające (MST) oraz wyznaczających najkrótszą ścieżkę w grafie (SPF). Należało zbadać algorytmy na dwóch sposobach reprezentacji grafu w komputerze, na macierzy incydencji oraz na listach sąsiedztwa (następników).

Krótki opis badanych algorytmów:

MST – Algorytm Prima – dzieli graf na wierzchołki rozpatrzone i nierozpatrzone. Zaczyna od ustalonego wierzchołka. Dopóki istnieją nierozpatrzone wierzchołki: Ze wszystkich sąsiadów wierzchołków rozpatrzonych znajduje najbliższy (najmniejsza waga krawędzi) nierozpatrzony wierzchołek, dodaje go do wierzchołków rozpatrzonych oraz dodaje połączenie z tymże wierzchołkiem do zbioru krawędzi MST.

MST – Algorytm Kruskala – na początku każdy wierzchołek jest osobnym podgrafem. Sortuje wszystkie krawędzie grafu według wagi. Dopóki zbiór zawiera krawędzie, pobiera z niego krawędź łączącą wierzchołki v1 i v2. Jeżeli v1 i v2 należą do różnych podgrafów to dodaje tę krawędź do zbioru krawędzi MST, oraz łączy podgrafy v1 i v2 w jeden podgraf.

SPF – Algorytm Dijkstry – dzieli graf na wierzchołki rozpatrzone i nierozpatrzone. Zaczyna od ustalonego wierzchołka. Dopóki istnieją nierozpatrzone wierzchołki: Ze wszystkich sąsiadów rozpatrzonych wierzchołków wybiera nierozpatrzony wierzchołek o najmniejszej odległości od startowego wierzchołka, dodaje go do wierzchołków rozpatrzonych, oraz próbuje relaksować krawędzie jego sąsiadów.

SPF – Algorytm Bellmana-Forda – relaksuje każdą krawędź grafu *V-1* razy (*V* – liczba wierzchołków). Ponieważ najdłuższa możliwa ścieżka bez cyklu może mieć *V-1* krawędzi, krawędzie muszą być zbadane *V-1* razy aby zapewnić że znaleziono ścieżkę dla każdego wierzchołka.

Krótki opis badanych reprezentacji grafów:

Macierz incydencji – macierz o wymiarze *E* x *V* (*E* – liczba krawędzi, *V* - wierzchołków), w której każda kolumna reprezentuje jedną krawędź, a każdy wiersz jeden wierzchołek grafu. Zawartość komórki macierzy może wynosić:

- 0 – dany wierzchołek nie należy do danej krawędzi
- 1 – dany wierzchołek jest początkiem danej krawędzi
- -1 – dany wierzchołek jest końcem danej krawędzi

Jeśli graf jest nieskierowany to można uprościć wartości komórek:

- 0 – dany wierzchołek nie należy do danej krawędzi
- 1 – dany wierzchołek należy do danej krawędzi

Lista sąsiedztwa (następników) – graf zapisujemy jako *V* list, gdzie w każdej liście są przechowywane numery wierzchołków do których można się dostać z danego wierzchołka (np. w liście 3 są przechowywane wierzchołki do których można się dostać z wierzchołka nr 3).

Powyższe reprezentacje trzeba było lekko zmodyfikować aby mogły przechowywać również informacje o wagach krawędzi. Dla macierzy incydencji należało stworzyć osobną tablicę o rozmiarze *E* gdzie 'i'-ty element posiada informację o wadze 'i'-tej krawędzi. Dla list sąsiedztwa natomiast wystarczyło zmodyfikować pojedynczy element listy, oprócz numeru wierzchołka musi także przechowywać informację o wadze krawędzi do tego wierzchołka.

Złożoności algorytmów:

Algorytm	**Reprezentacja**	**Złożoność**
Prim	Lista sąsiedztwa	$O(E \log V)$
Prim	Macierz incydencji	$O(V^2)$
Kruskal	Lista sąsiedztwa	$O(E \log E)$
Kruskal	Macierz incydencji	$O(E \log E)$
Dijkstra	Lista sąsiedztwa	$O(E \log V)$
Dijkstra	Macierz incydencji	$O(E \log V)$
Bellman-Ford	Lista sąsiedztwa	$O(E \cdot V)$
Bellman-Ford	Macierz incydencji	$O(E \cdot V)$

Plan eksperymentu

Program został napisany w języku C++ w środowisku Visual Studio 2019, oraz w tymże środowisku skompilowany.

Badane rozmiary grafów to 20, 40, 60, 80, 100 wierzchołków, badane gęstości to 25%, 50%, 75%, 99% możliwych krawędzi.

Test każdego algorytmu, dla każdej reprezentacji, dla każdej gęstości, dla każdej liczby wierzchołków był wykonany 100 razy.
Czas mierzono za pomocą QueryPerformanceCounter, który mierzy czas z dokładnością do części dziesiętnej mikrosekundy. Wykorzystano fragment kodu z https://stackoverflow.com/questions/1739259/how-to-use-queryperformancecounter.

Do każdego testu był tworzony nowy graf z losowo wygenerowanymi krawędziami.

Generowanie grafu

Generowany graf był tworzony w poniższy sposób:

- wierzchołek start=0
- dopóki są wierzchołki do których nie da się dostać:
- wierzchołek end=wylosuj numer wierzchołka z wierzchołków które jeszcze nie były wylosowane
- dodaj do grafu krawędź start-end
- start=end

Gdy graf jest już spójny, tworzymy zbiór wszystkich możliwych krawędzi, a następnie dopóki nie mamy wystarczającej liczby krawędzi, losujemy jedną z możliwych krawędzi, jeśli nie ma takiej krawędzi w grafie, to ją dodajemy, w przeciwnym wypadku odrzucamy.

Powyższy sposób działa dla grafów nieskierowanych, aby działał również dla skierowanych, musimy po stworzeniu de facto drzewa rozpinającego graf, dodać jeszcze jedną krawędź, z ostatniego wierzchołka do pierwszego, aby uzyskać cykl, żeby była możliwość dostania się z każdego wierzchołka do każdego wierzchołka.

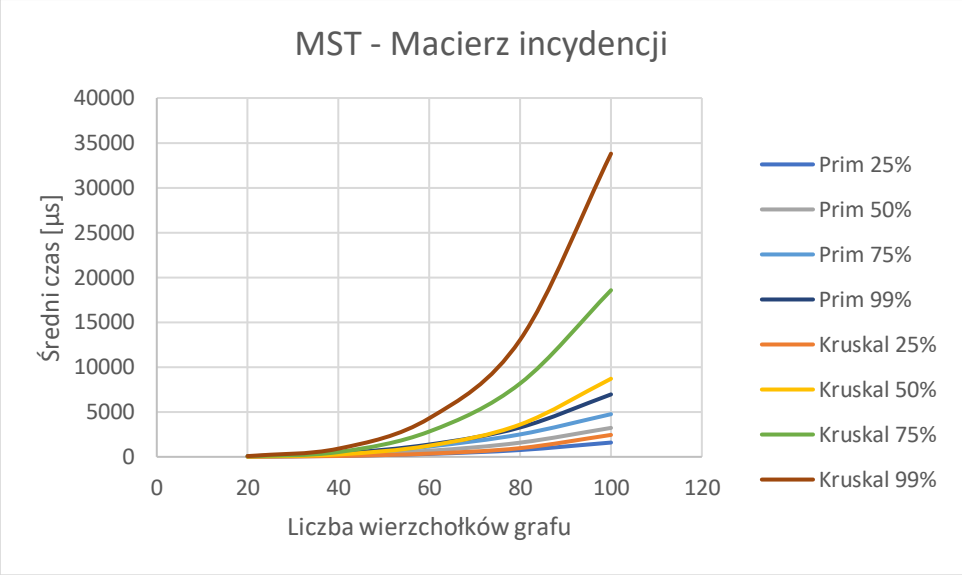
Każdej krawędzi nadano losową wagę od 1 do 999.

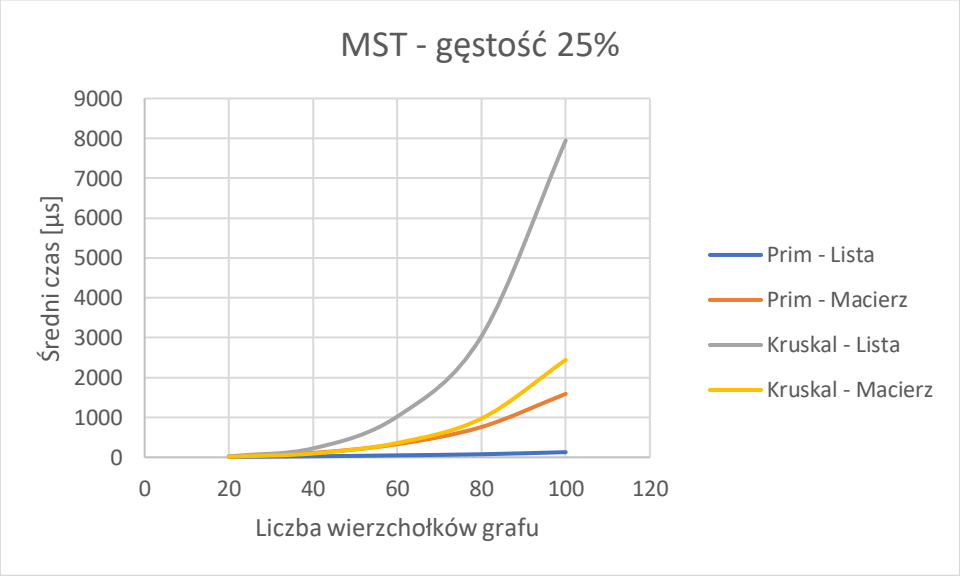
Wyniki

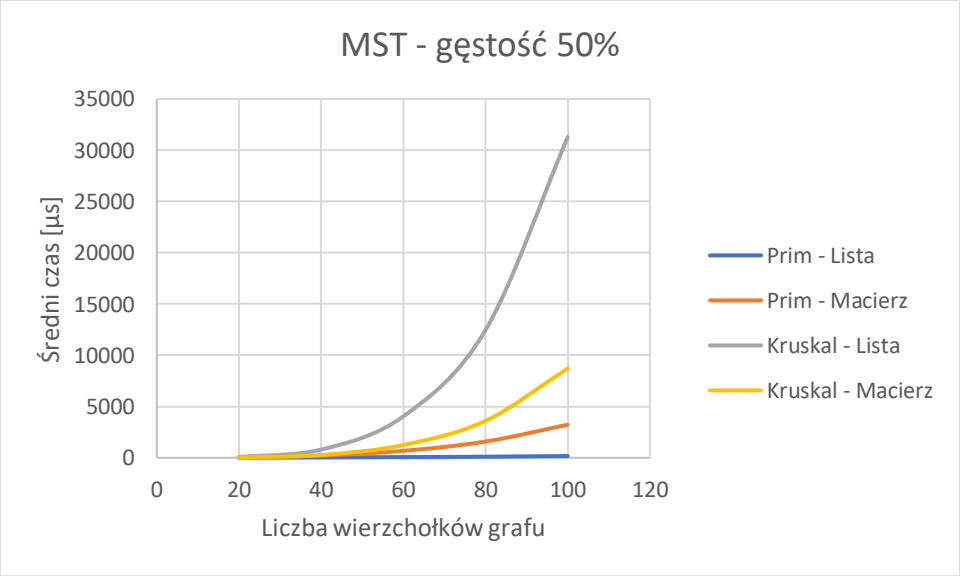
Wyniki są podane w mikrosekundach [µs]

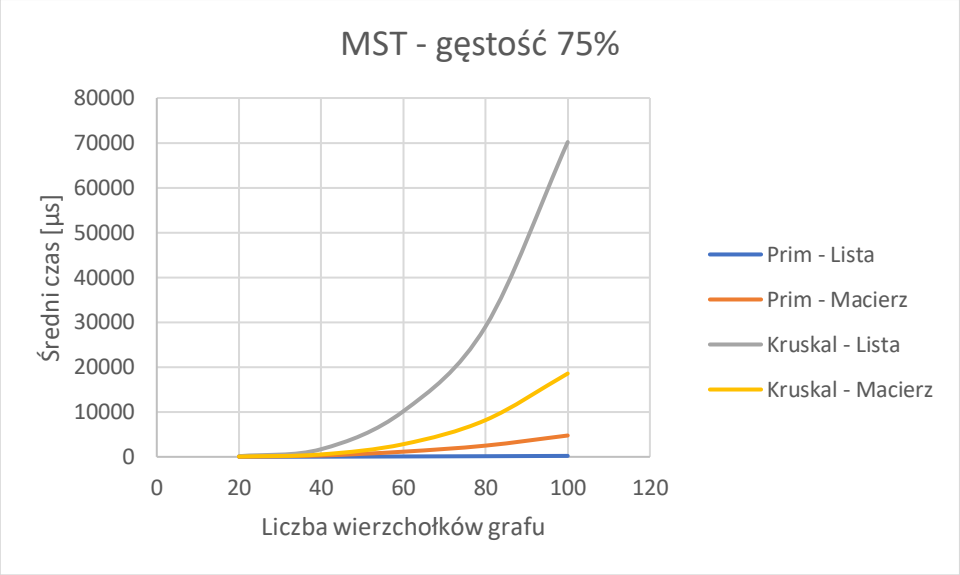
Problem MST:

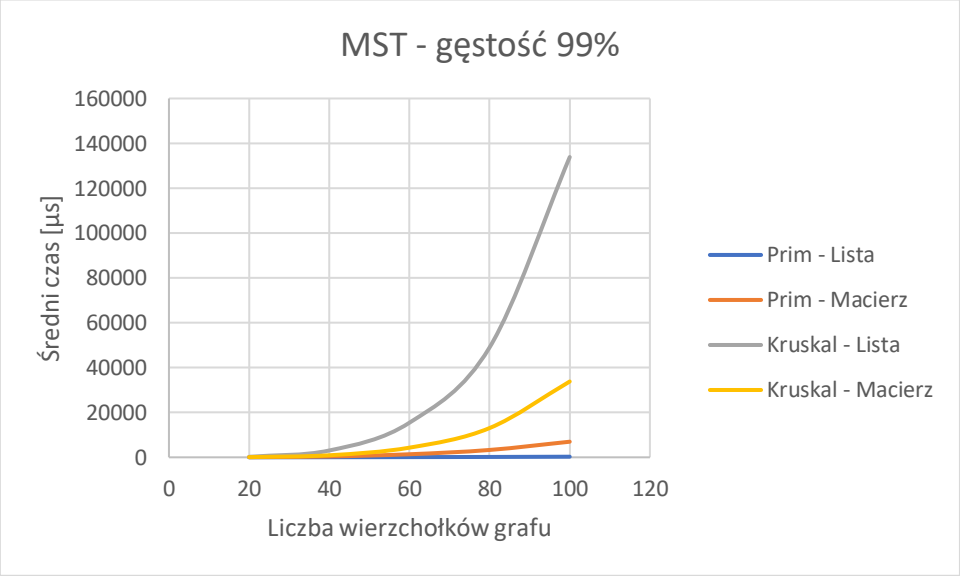
Liczba wierzchołków	Gęstość	Prim - lista	Prim - macierz	Kruskal - lista	Kruskal - macierz
20	25%	8,733	21,871	25,782	13,334
20	50%	12,283	38,338	72,711	31,752
20	75%	12,593	53,098	143,738	55,137
20	99%	15,119	67,445	223,106	81,749
40	25%	25,461	111,447	224,201	95,243
40	50%	31,919	220,433	805,063	268,708
40	75%	35,207	295,689	1706,53	536,482
40	99%	42,506	381,451	3047,85	913,804
60	25%	49,2	328,846	1022,19	360,386
60	50%	68,077	678,707	4042,08	1244,82
60	75%	89,966	1153,59	10170,8	2825,03
60	99%	95,781	1372,98	15429,5	4307,34
80	25%	77,231	760,871	3036,77	972,567
80	50%	104,309	1584,17	12457	3604,47
80	75%	142,901	2502,76	29048,4	8185,26
80	99%	166,53	3277,95	49022,4	13050,9
100	25%	129,162	1592,07	7946,7	2443,52
100	50%	174,63	3225,62	31279,3	8708,18
100	75%	214,172	4756,39	70183,1	18573,5
100	99%	278,882	6961,8	133885	33800,4


MST - Macierz incydencji
Średni czas [µs]
40000
35000
30000
25000
20000
15000
10000
5000
0
0
20
40
60
80
100
120
Liczba wierzchołków grafu
Prim 25%
Prim 50%
Prim 75%
Prim 99%
Kruskal 25%
Kruskal 50%
Kruskal 75%
Kruskal 99%


MST - gęstość 25%
Średni czas [µs]
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
120
Liczba wierzchołków grafu
Prim - Lista
Prim - Macierz
Kruskal - Lista
Kruskal - Macierz

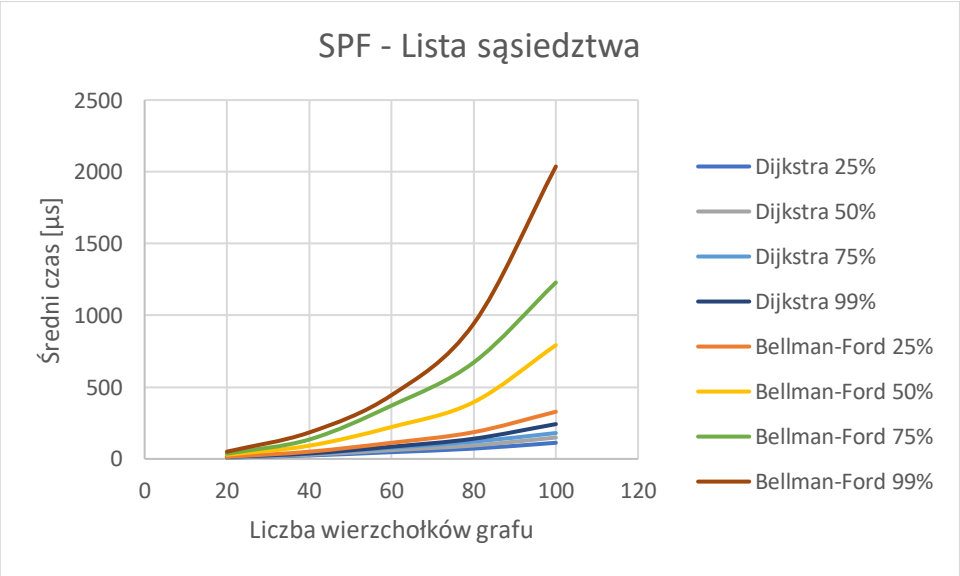

MST - gęstość 50%
Średni czas [µs]
35000
30000
25000
20000
15000
10000
5000
0
0
20
40
60
80
100
120
Liczba wierzchołków grafu
Prim - Lista
Prim - Macierz
Kruskal - Lista
Kruskal - Macierz

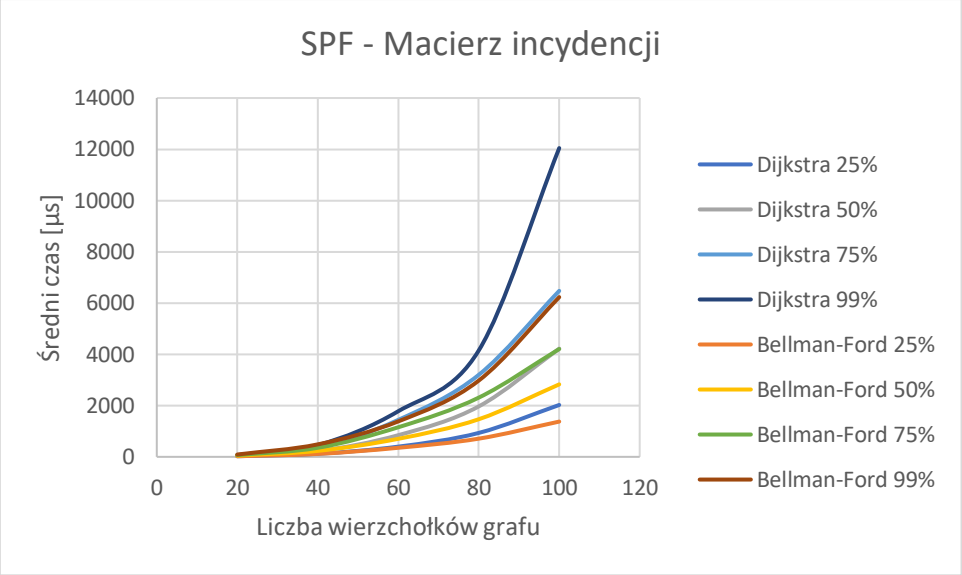

MST - gęstość 75%
Średni czas [µs]
80000
70000
60000
50000
40000
30000
20000
10000
0
0
20
40
60
80
100
120
Liczba wierzchołków grafu
Prim - Lista
Prim - Macierz
Kruskal - Lista
Kruskal - Macierz

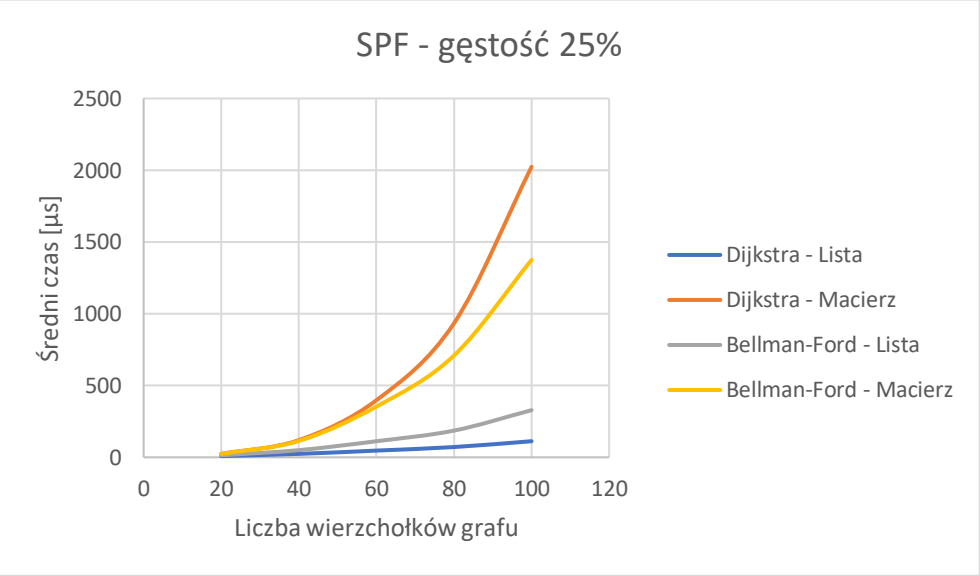

MST - gęstość 99%
Średni czas [µs]
160000
140000
120000
100000
80000
60000
40000
20000
0
0
20
40
60
80
100
120
Liczba wierzchołków grafu
Prim - Lista
Prim - Macierz
Kruskal - Lista
Kruskal - Macierz

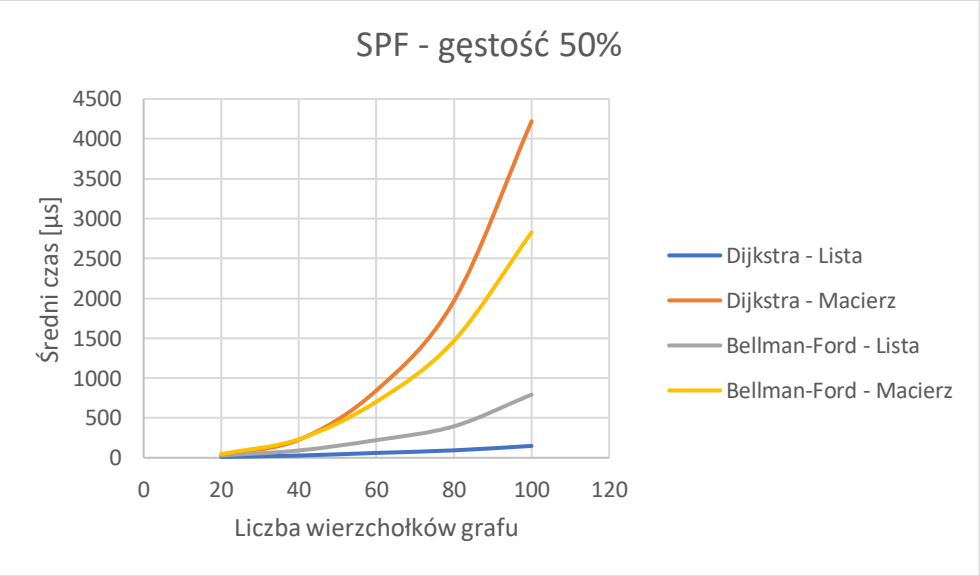
Problem SPF:

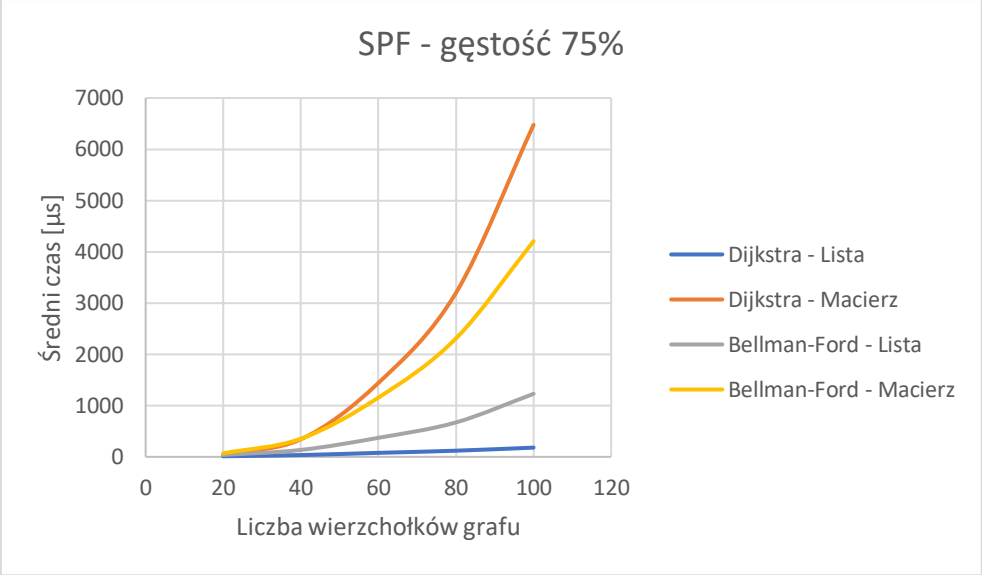
Liczba wierzchołków	Gęstość	Dijkstra - lista	Dijkstra - macierz	Bellman-Ford lista	Bellman-Ford macierz
20	25%	8,633	24,262	13,988	23,231
20	50%	10,635	38,554	28,696	46,485
20	75%	12,037	52,438	36,54	68,104
20	99%	14,212	67,889	50,443	90,38
40	25%	24,027	119,862	50,007	113,917
40	50%	28,125	225,976	91,729	232,715
40	75%	34,904	342,016	134,848	353,357
40	99%	38,136	458,149	183,845	486,998
60	25%	46,847	397,832	112,086	353,293
60	50%	61,048	844,954	221,661	703,663
60	75%	78,031	1441,64	370,883	1154,39
60	99%	84,913	1779,08	443,424	1382,64
80	25%	71,612	933,613	186,085	709,255
80	50%	93,34	1967,33	395,156	1467,13
80	75%	119,873	3199,3	671,52	2310,37
80	99%	140,502	4149,22	940,958	2975,11
100	25%	112,234	2024,05	328,427	1375,32
100	50%	148,854	4221	792,412	2827,24
100	75%	180,131	6475,7	1229,09	4207,04
100	99%	242,403	12046,8	2037,04	6231,69

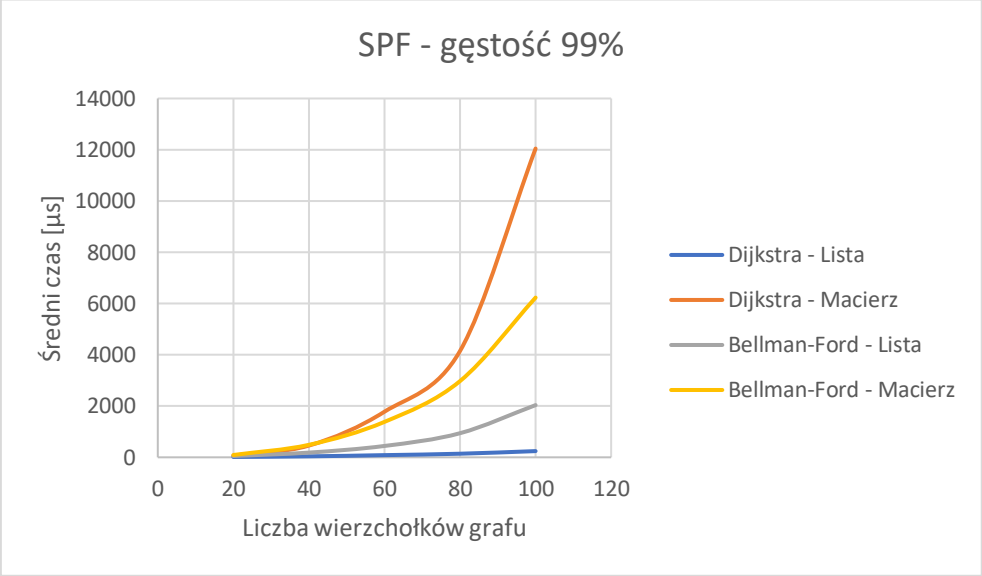



SPF - Macierz incydencji
Średni czas [µs]
14000
12000
10000
8000
6000
4000
2000
0
0
20
40
60
80
100
120
Liczba wierzchołków grafu
Dijkstra 25%
Dijkstra 50%
Dijkstra 75%
Dijkstra 99%
Bellman-Ford 25%
Bellman-Ford 50%
Bellman-Ford 75%
Bellman-Ford 99%


SPF - gęstość 25%
Średni czas [µs]
2500
2000
1500
1000
500
0
0
20
40
60
80
100
120
Liczba wierzchołków grafu
Dijkstra - Lista
Dijkstra - Macierz
Bellman-Ford - Lista
Bellman-Ford - Macierz


SPF - gęstość 50%
Średni czas [µs]
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
20
40
60
80
100
120
Liczba wierzchołków grafu
Dijkstra - Lista
Dijkstra - Macierz
Bellman-Ford - Lista
Bellman-Ford - Macierz





Podsumowanie

Algorytmy Prima, Dijkstry i Bellmana-Forda lepiej pracowały na listach następników, algorytm Kruskala natomiast lepiej pracował na macierzy incydencji. Algorytm Prima okazał się lepszy od Kruskala dla obu reprezentacji. Algorytm Dijkstry okazał się szybszy od Bellmana-Forda dla listy następników, jednakże w przypadku macierzy incydencji sytuacja się odwróciła.

Widać również że im większa gęstość grafu, tym bardziej opłaca się przedstawić graf jako macierz incydencji.

Źródła:

1. Materiały do ćwiczeń ze SDiZO dr inż. Jarosława Mierzwy
2. Wprowadzenie do algorytmów, Thomas H. Cormen
3. https://en.wikipedia.org/wiki/Prim%27s_algorithm
4. https://en.wikipedia.org/wiki/Kruskal%27s_algorithm
5. https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
6. https://en.wikipedia.org/wiki/Bellman–Ford_algorithm