Piotr Józefek 272311
grupa 18 parzyste czwartek, 11:15 - 13:00
Dariusz Banasiak

Wprowadzenie

Złożoność pamięciowa macierzy incydencji wynosi: O(V*E)
Złożoność pamięciowa listy następników wynosi: O(E)

Algorytm Prima

Lista następników: $O((V + E)logV)$

Inicjalizacja: O(V), gdzie V to liczba wierzchołków.
Dodanie wszystkich wierzchołków do kolejki priorytetowej: O(V log V), zakładając użycie kopca binarnego jako struktury kolejki priorytetowej.
Pętla główna (dla każdego wierzchołka): O(V)

- Usunięcie elementu z kolejki priorytetowej: O(log V)
- Przetworzenie sąsiadów: O(E log V), gdzie E to liczba krawędzi, ponieważ dla każdej krawędzi może zajść konieczność aktualizacji kolejki priorytetowej.

Macierz incydencji: $O(V^2\ logV)$

Inicjalizacja: O(V), gdzie V to liczba wierzchołków.

Dodanie wszystkich wierzchołków do kolejki priorytetowej: O(V log V), zakładając użycie kopca binarnego jako struktury kolejki priorytetowej.

Pętla główna dla każdego wierzchołka: O(V)

- Usunięcie elementu z kolejki priorytetowej: O(log V)
- Przetworzenie sąsiadów: O(V^2 log V), ponieważ przeszukiwanie macierzy incydencji dla każdego wierzchołka może zająć czas proporcjonalny do liczby wierzchołków, a dla każdego przetworzenia musimy zaktualizować kolejkę priorytetową.

Algorytm Kruskala

Lista następników: $O(E\ logE)$
Zebranie wszystkich krawędzi z listy następników:

- Przejście przez wszystkie wierzchołki i ich listy sąsiedztwa: O(E), gdzie E to liczba krawędzi.

Sortowanie krawędzi:

- Sortowanie E krawędzi: O(E log E).

Inicjalizacja zbiorów rozłącznych

- Inicjalizacja: O(V), gdzie V to liczba wierzchołków.
- Inicjalizacja zbiorów rozłącznych: O(E* V),

Macierz incydencji: $O(V^2 * logV)$
Zebranie wszystkich krawędzi z macierzy incydencji:

- Przejście przez macierz incydencji: O(V^2), gdzie V to liczba wierzchołków.

Sortowanie krawędzi:

- Sortowanie E krawędzi: O(E log E).

Inicjalizacja zbiorów rozłącznych

- Inicjalizacja: O(V).
- Operacje inicjalizacji zbiorów rozłącznych: O(E *V).

Algorytm Dijkstry

Lista następników: $O((V + E)logV) \approx O(V\ logV)$
Inicjalizacja

- Ustawienie odległości do wszystkich wierzchołków na nieskończoność i dodanie startowego wierzchołka do kolejki priorytetowej: O(V), gdzie V to liczba wierzchołków.

Korzystanie z kolejki priorytetowej z kopcem
- Dodanie i usunięcie wierzchołków z kolejki priorytetowej: O(V log V).
- Aktualizacja odległości do sąsiednich wierzchołków: O(E log V), gdzie E to liczba krawędzi.

Macierz incydencji: $O(V^2)$
Inicjalizacja

- Ustawienie odległości do wszystkich wierzchołków na nieskończoność i dodanie startowego wierzchołka do kolejki priorytetowej: O(V).

Korzystanie z kolejki priorytetowej z kopcem:

- Dodanie i usunięcie wierzchołków z kolejki priorytetowej: O(V log V).
- Aktualizacja odległości do sąsiednich wierzchołków:
    - Przeglądanie macierzy incydencji: O(V^2) dla każdego wierzchołka.
    - Dla każdego przetworzonego wierzchołka może zajść konieczność aktualizacji kolejki priorytetowej: O(V^2 log V).

Algorytm Forda-Bellmana

Lista następników: $O(V\ E)$
Inicjalizacja

- Ustawienie odległości do wszystkich wierzchołków na nieskończoność i ustawienie odległości startowego wierzchołka na 0: O(V), gdzie V to liczba wierzchołków.

Relaksacja krawędzi:

- Algorytm wykonuje V-1 iteracji, w każdej iteracji przegląda wszystkie krawędzie:
    - Przejście przez wszystkie wierzchołki: O(V).

    - Przejście przez listy następników dla każdego wierzchołka, aby przetworzyć wszystkie krawędzie: O(E), gdzie E to liczba krawędzi.

Macierz incydencji: $O(V^3)$
inicjalizacja

- Ustawienie odległości do wszystkich wierzchołków na nieskończoność i ustawienie odległości startowego wierzchołka na 0: O(V).

Relaksacja krawędzi:

- Algorytm wykonuje V-1 iteracji, w każdej iteracji przegląda wszystkie krawędzie:
    - Przejście przez wszystkie wierzchołki: O(V).
    - Przejście przez macierz incydencji, aby przetworzyć wszystkie krawędzie: O(V^2), gdzie V^2 to maksymalna liczba elementów w macierzy incydencji dla grafu pełnego.

Plan eksperymentu

Założeniem eksperymentu jest przeprowadzenie symulacji algorytmów algorytmów grafowych rozwiązujących następujące problemy:
- a. wyznaczanie minimalnego drzewa rozpinającego (MST) - algorytm Prima oraz algorytm Kruskala,
-b. wyznaczanie najkrótszej ścieżki w grafie – algorytm Dijkstry oraz algorytm Forda-Bellmana,dla wybranych rozmiarów tablic.
Przyjęte rozmiary grafów to: 10, 20, 50, 100, 200, 500, 1000 elementów.

Dla każdego rozmiaru tablic zostały zmierzone czasy dla trzech gęstości grafu 25%, 50% i 99%

Liczba krawędzi była liczona w następujący sposób:
$Liczba\ krawędzi\ = Gęstość\ *\ Liczba\ wierzchołków\ *\ (Liczba\ wierzchołków\ -\ 1)\ /\ 2$

Dla dokładności pomiarów każdy algorytm został wykonany po 50 razy dla każdej gęstości i wielkości i sposobu zapisu grafu w pamięci.

Dla każdego pomiaru został wygenerowany nowy graf. Generowanie tablic zostało zrealizowane za pomocą biblioteki <random> oraz funkcji uniform_real_distribution.

Do mierzenia czasu została wykorzystana biblioteka <chrono> oraz funkcja high_resolution_clock. Dla każdego algorytmu został zapisany czas rozpoczęcia oraz zakończenia działania algorytmu. Na ich podstawie został obliczony średni czas wykonania.

Generowanie polega na operacjach na pomocniczej strukturze a następnie na przeniesieniu wartości do właściwych struktur grafu:

- listy sąsiedztwa zawierającej liczbe wierzchołków, krawędzi, tablice przechowującą liczbę krawędzi dla każdego wierzchołka oraz tablicy z własną strukturą My_pair przechowującą wierzchołek końcowy oraz wagę krawędzi
- macierzy incydencji zawierającej liczbe wierzchołków, krawędzi, dwuwymiarową tablice reperentującą macierz incydencji oraz tablice przechowującą wagi krawędzi

Na początku generowany jest graf rozpinający a następnie z pozostałych wierzchołków losowany jest wierzchołek początkowy, końcowy i waga.

Wyznaczanie minimalnego drzewa rozpinającego

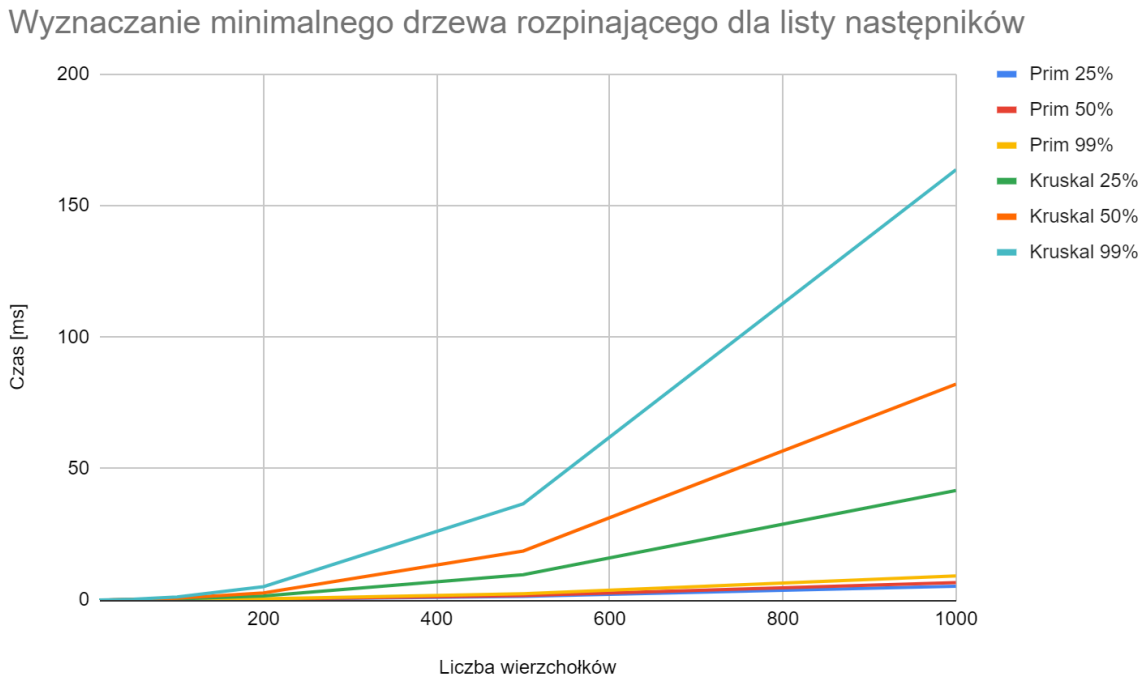
Typ 1

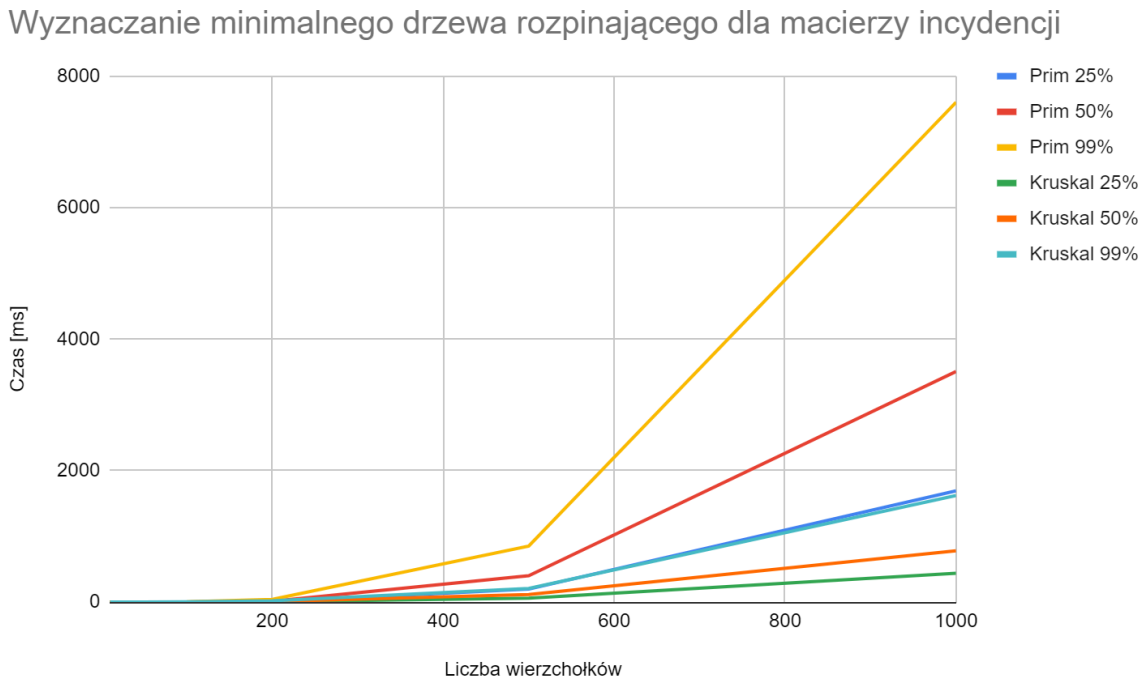
Lista [ms]	Algorytm Prima		
Liczba wierzchołków	Gęstość 25%	Gęstość 50%	Gęstość 99%
10	0.00752	0.00334	0.0061
20	0.0078	0.00832	0.01156
50	0.02612	0.27402	0.04026
100	0.0844	0.0926	0.14082
200	0.25712	0.32578	0.43706
500	1.40498	1.72466	2.35996
1000	5.25884	6.60694	9.19386

Macierz [ms]	Algorytm Prima		
Liczba wierzchołków	Gęstość 25%	Gęstość 50%	Gęstość 99%
10	0.00118	0.00274	0.00606
20	0.0077	0.01396	0.02876
50	0.0783	0.14582	0.30066
100	0.58138	1.19664	2.50652
200	4.75836	11.4799	40.0856
500	196.883	401.232	851.721
1000	1692.12	3506.73	7603.93

Lista [ms]	Algorytm Kruskala		
Liczba wierzchołków	Gęstość 25%	Gęstość 50%	Gęstość 99%
10	0.0033	0.00556	0.0133
20	0.01728	0.02866	0.05
50	0.1036	0.18014	0.30626
100	0.40314	0.68076	1.15604
200	1.54758	2.66792	5.08112
500	9.65448	18.638	36.6229
1000	41.692	82.1297	163.69

Macierz [ms]	Algorytm Kruskala		
Liczba wierzchołków	Gęstość 25%	Gęstość 50%	Gęstość 99%
10	0.00306	0.00938	0.01832
20	0.0232	0.04624	0.07896
50	0.17772	0.30904	0.53182
100	0.7916	1.4039	2.75096
200	3.88828	7.71436	17.9434
500	59.2511	113.425	205.574
1000	437.331	778.926	1619.2


Wyznaczanie minimalnego drzewa rozpinającego dla listy następników
200
150
100
50
0
Czas [ms]
200
400
600
800
1000
Liczba wierzchołków
Prim 25%
Prim 50%
Prim 99%
Kruskal 25%
Kruskal 50%
Kruskal 99%


Wyznaczanie minimalnego drzewa rozpinającego dla macierzy incydencji
8000
6000
4000
2000
0
Czas [ms]
200
400
600
800
1000
Liczba wierzchołków
Prim 25%
Prim 50%
Prim 99%
Kruskal 25%
Kruskal 50%
Kruskal 99%

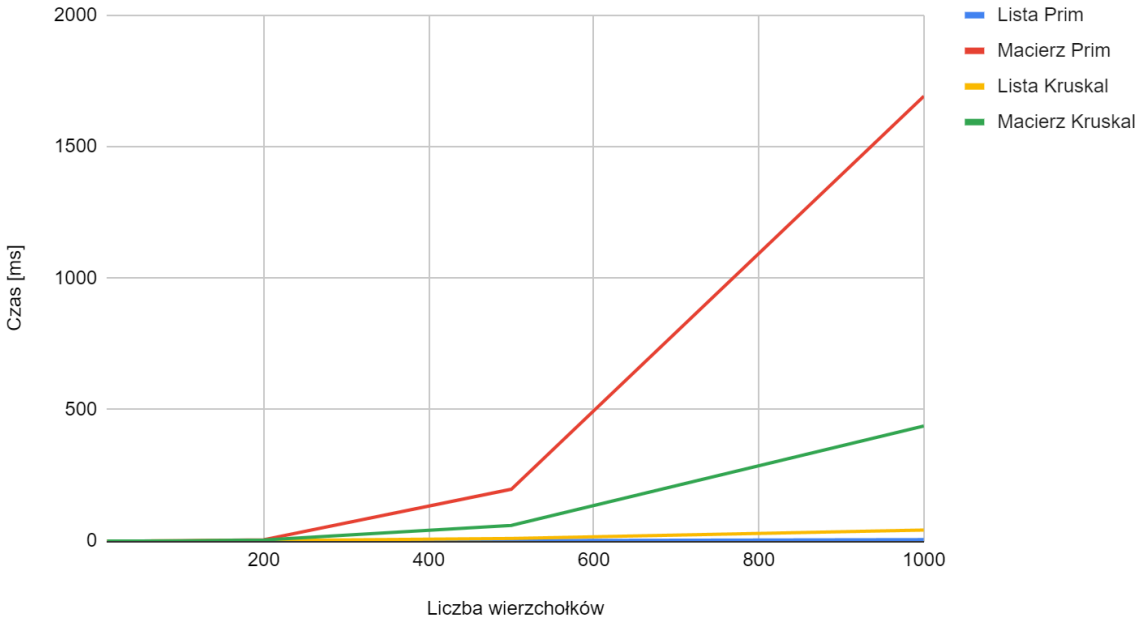
Typ 2

[ms]	Wyznaczanie minimalnego drzewa rozpinającego dla gęstości 25%			
Liczba wierzchołków	Lista Prim	Macierz Prim	Lista Kruskal	Macierz Kruskal
10	0.00752	0.00118	0.0033	0.00306
20	0.0078	0.0077	0.01728	0.0232
50	0.02612	0.0783	0.1036	0.17772
100	0.0844	0.58138	0.40314	0.7916
200	0.25712	4.75836	1.54758	3.88828
500	1.40498	196.883	9.65448	59.2511
1000	5.25884	1692.12	41.692	437.331

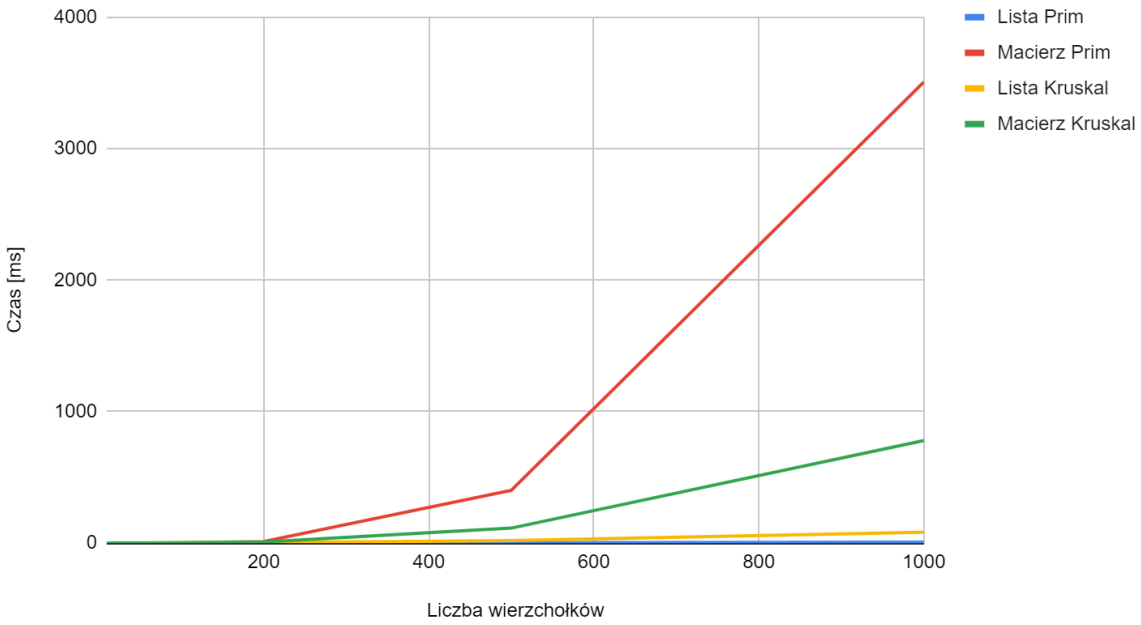
[ms]	Wyznaczanie minimalnego drzewa rozpinającego dla gęstości 50%			
Liczba wierzchołków	Lista Prim	Macierz Prim	Lista Kruskal	Macierz Kruskal
10	0.00334	0.00274	0.00556	0.00938
20	0.00832	0.01396	0.02866	0.04624
50	0.27402	0.14582	0.18014	0.30904
100	0.0926	1.19664	0.68076	1.4039
200	0.32578	11.4799	2.66792	7.71436
500	1.72466	401.232	18.638	113.425
1000	6.60694	3506.73	82.1297	778.926

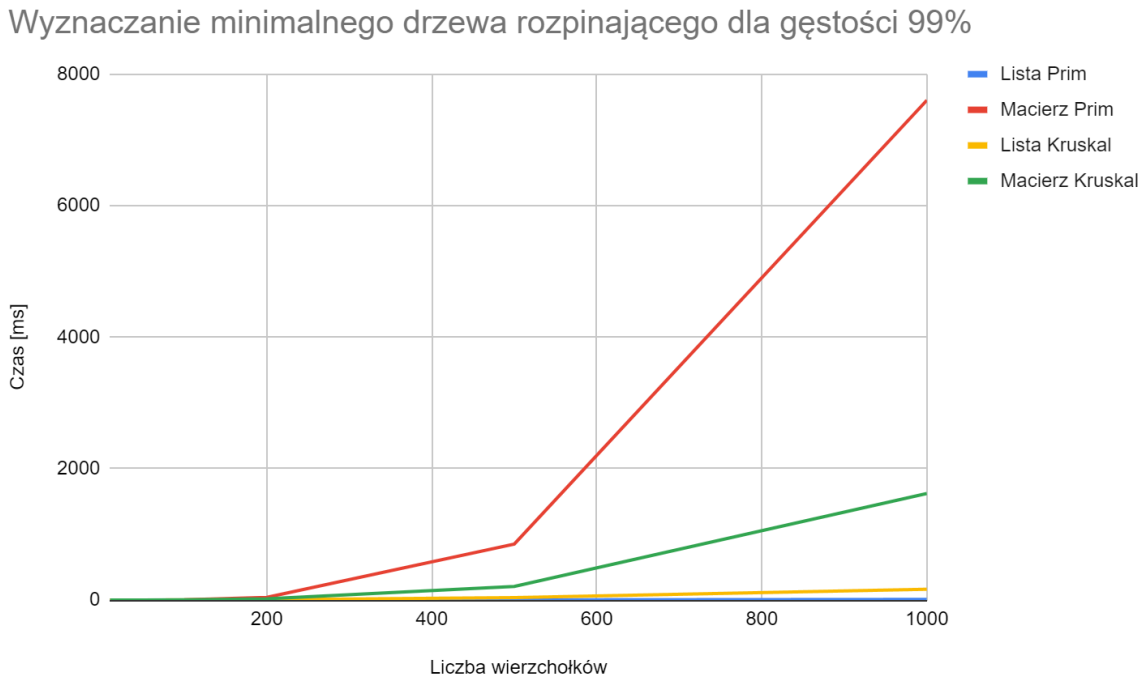
[ms]	Wyznaczanie minimalnego drzewa rozpinającego dla gęstości 99%			
Liczba wierzchołków	Lista Prim	Macierz Prim	Lista Kruskal	Macierz Kruskal
10	0.0061	0.00606	0.0133	0.01832
20	0.01156	0.02876	0.05	0.07896
50	0.04026	0.30066	0.30626	0.53182
100	0.14082	2.50652	1.15604	2.75096
200	0.43706	40.0856	5.08112	17.9434
500	2.35996	851.721	36.6229	205.574
1000	9.19386	7603.93	163.69	1619.2

Wyznaczanie minimalnego drzewa rozpinającego dla gęstości 25%



Wyznaczanie minimalnego drzewa rozpinającego dla gęstości 50%





Wyznaczanie najkrótszej ścieżki w grafie

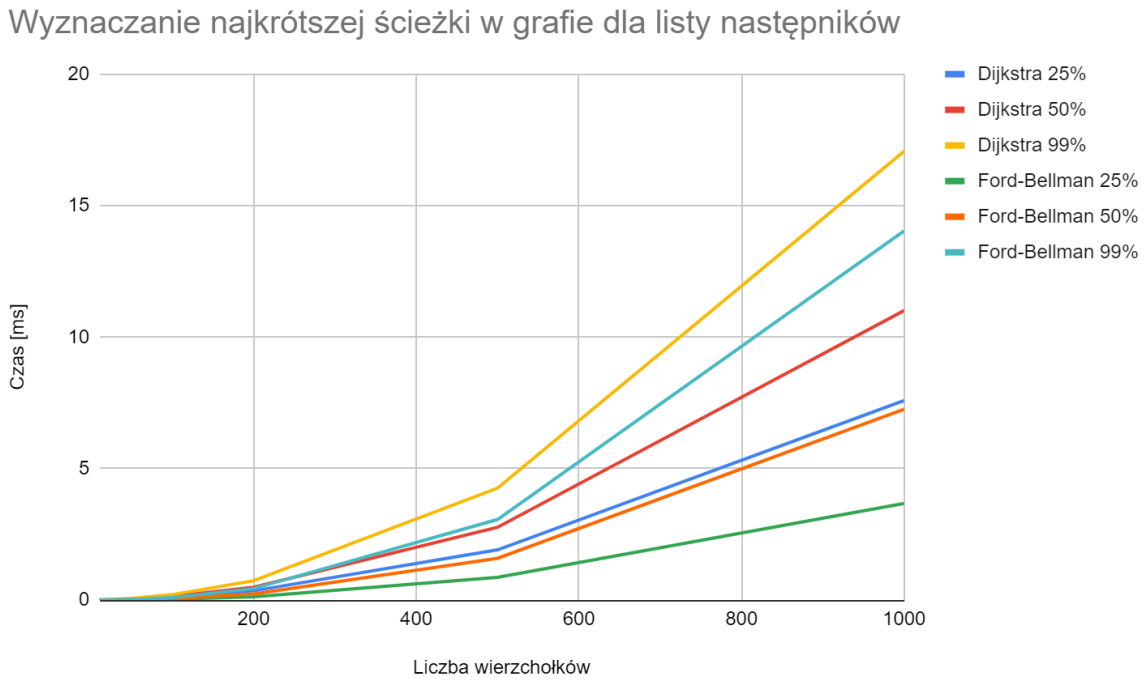
Typ 1

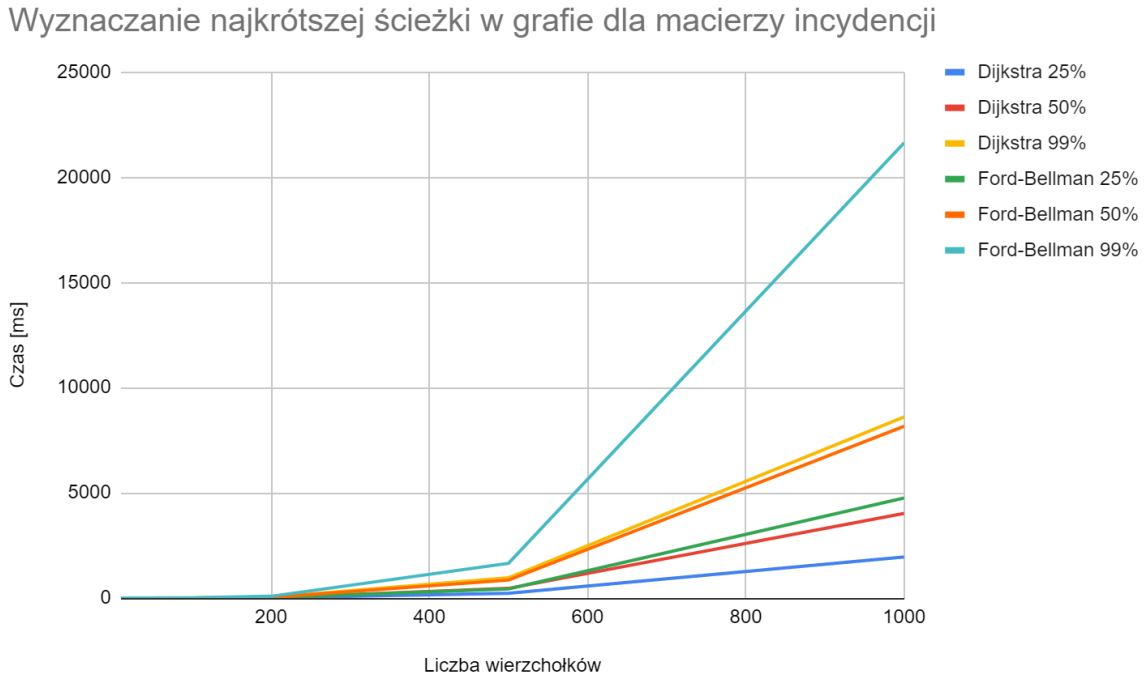
Lista [ms]	Algorytm Dijkstry		
Liczba wierzchołków	Gęstość 25%	Gęstość 50%	Gęstość 99%
10	0.00306	0.0031	0.00394
20	0.00818	0.00958	0.01288
50	0.03388	0.04294	0.05876
100	0.10544	0.14082	0.20444
200	0.35048	0.48804	0.73428
500	1.90958	2.76562	4.2575
1000	7.58578	11.0078	17.0711

Macierz [ms]	Algorytm Dijkstry		
Liczba wierzchołków	Gęstość 25%	Gęstość 50%	Gęstość 99%
10	0.00118	0.00212	0.00402
20	0.00708	0.01266	0.02292
50	0.07534	0.14568	0.29368
100	0.63802	1.30886	2.5449
200	5.22502	13.2799	42.2158
500	224.545	457.682	961.32
1000	1951.6	4026.97	8610.52

Lista [ms]	Algorytm Forda-Bellmana		
Liczba wierzchołków	Gęstość 25%	Gęstość 50%	Gęstość 99%
10	0.00002	0.00018	0.001
20	0.001	0.00164	0.0028
50	0.00662	0.01182	0.0212
100	0.027	0.05298	0.09256
200	0.12282	0.23126	0.43464
500	0.86458	1.58628	3.06326
1000	3.67232	7.25806	14.0389

Macierz [ms]	Algorytm Forda-Bellmana		
Liczba wierzchołków	Gęstość 25%	Gęstość 50%	Gęstość 99%
10	0.001	0.00196	0.00378
20	0.00672	0.01738	0.04104
50	0.1358	0.32514	0.66236
100	1.44074	3.32866	6.77374
200	15.2975	31.441	82.4594
500	431.965	851.627	1646.48
1000	4753.2	8168.76	21645.2


Wyznaczanie najkrótszej ścieżki w grafie dla listy następników
Czas [ms]
20
15
10
5
0
200
400
600
800
1000
Liczba wierzchołków
Dijkstra 25%
Dijkstra 50%
Dijkstra 99%
Ford-Bellman 25%
Ford-Bellman 50%
Ford-Bellman 99%


Wyznaczanie najkrótszej ścieżki w grafie dla macierzy incydencji
Czas [ms]
25000
20000
15000
10000
5000
0
200
400
600
800
1000
Liczba wierzchołków
Dijkstra 25%
Dijkstra 50%
Dijkstra 99%
Ford-Bellman 25%
Ford-Bellman 50%
Ford-Bellman 99%

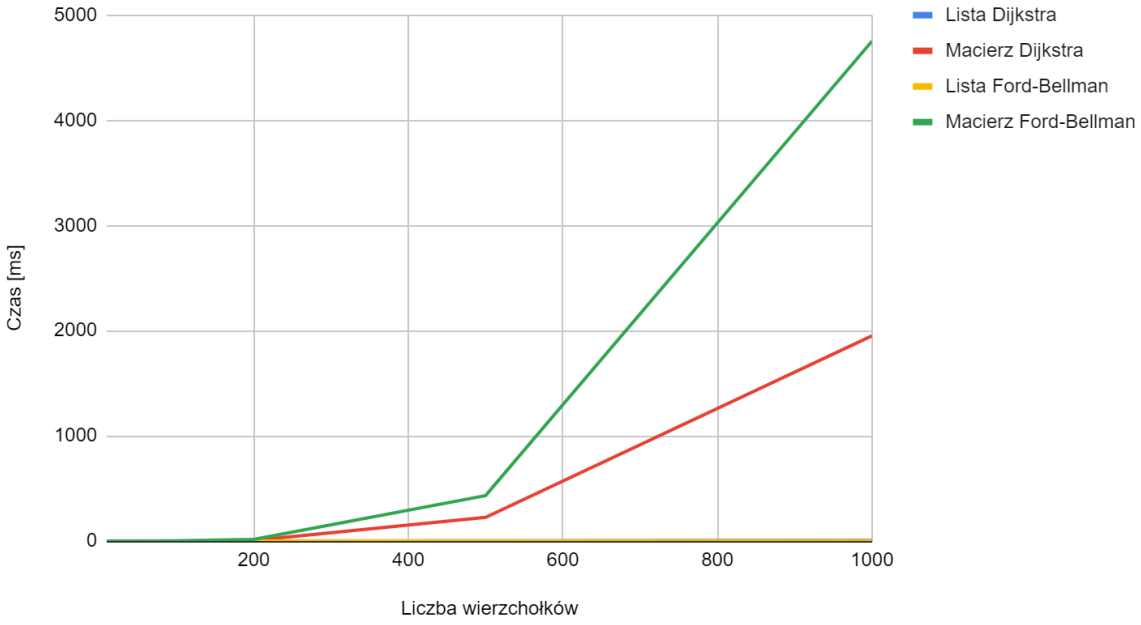
Typ 2

[ms]	Wyznaczanie najkrótszej ścieżki w grafie dla gęstości 25%			
Liczba wierzchołków	Lista Dijkstra	Macierz Dijkstra	Lista Ford-Bellman	Macierz Ford-Bellman
10	0.00306	0.00118	0.00002	0.001
20	0.00818	0.00708	0.001	0.00672
50	0.03388	0.07534	0.00662	0.1358
100	0.10544	0.63802	0.027	1.44074
200	0.35048	5.22502	0.12282	15.2975
500	1.90958	224.545	0.86458	431.965
1000	7.58578	1951.6	3.67232	4753.2

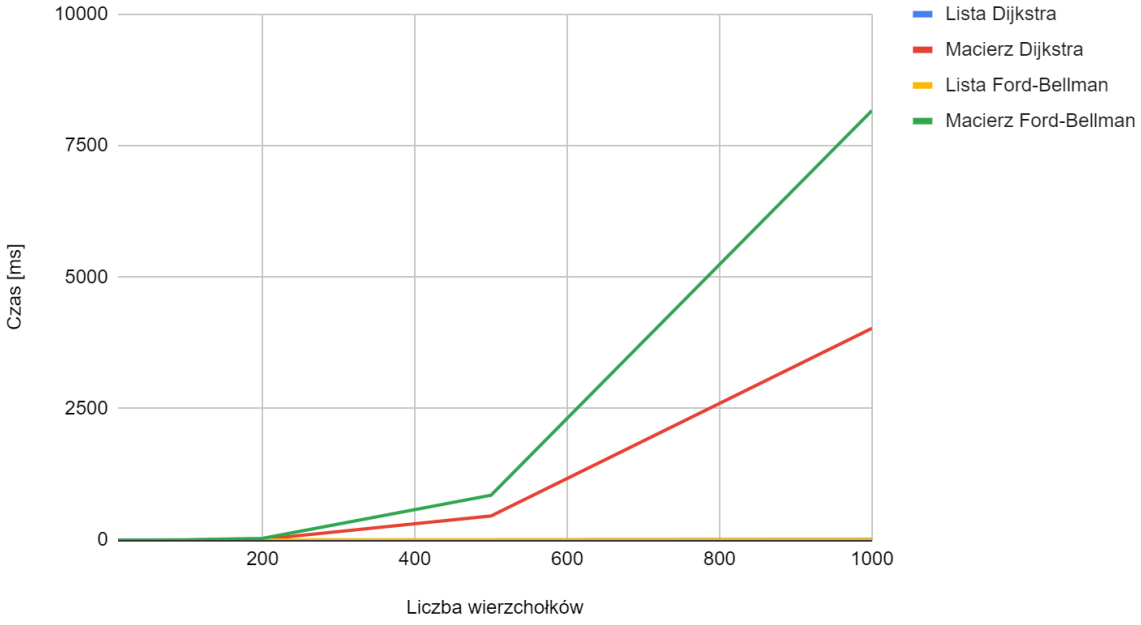
[ms]	Wyznaczanie najkrótszej ścieżki w grafie dla gęstości 50%			
Liczba wierzchołków	Lista Dijkstra	Macierz Dijkstra	Lista Ford-Bellman	Macierz Ford-Bellman
10	0.0031	0.00212	0.00018	0.00196
20	0.00958	0.01266	0.00164	0.01738
50	0.04294	0.14568	0.01182	0.32514
100	0.14082	1.30886	0.05298	3.32866
200	0.48804	13.2799	0.23126	31.441
500	2.76562	457.682	1.58628	851.627
1000	11.0078	4026.97	7.25806	8168.76

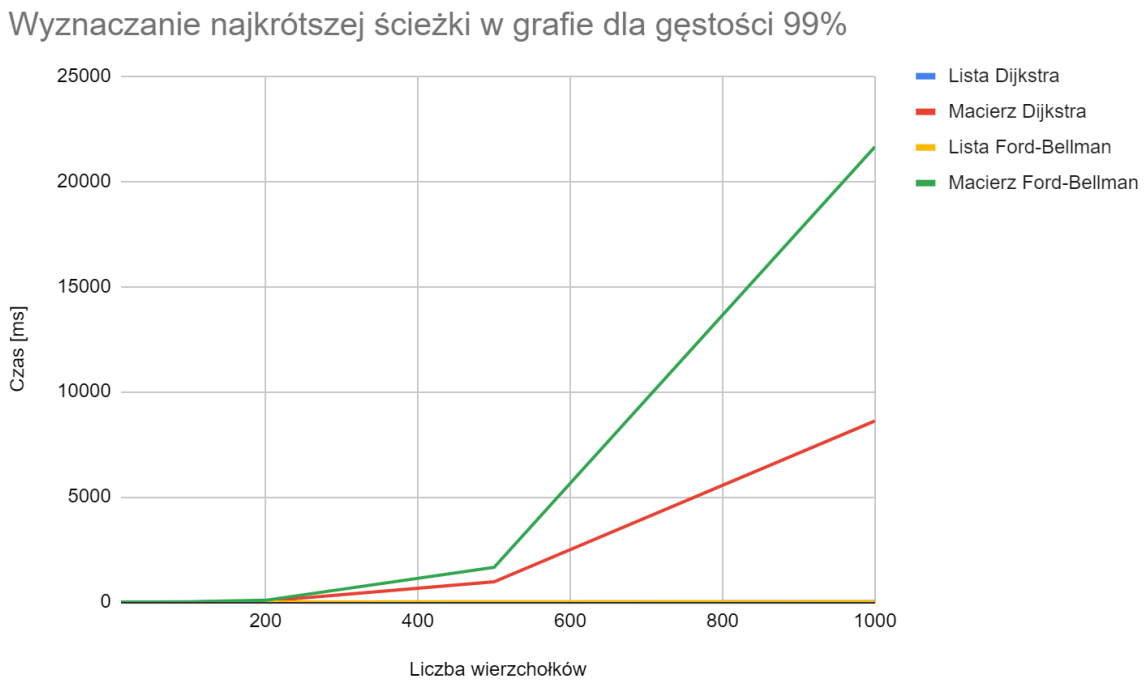
[ms]	Wyznaczanie najkrótszej ścieżki w grafie dla gęstości 99%			
Liczba wierzchołków	Lista Dijkstra	Macierz Dijkstra	Lista Ford-Bellman	Macierz Ford-Bellman
10	0.00394	0.00402	0.001	0.00378
20	0.01288	0.02292	0.0028	0.04104
50	0.05876	0.29368	0.0212	0.66236
100	0.20444	2.5449	0.09256	6.77374
200	0.73428	42.2158	0.43464	82.4594
500	4.2575	961.32	3.06326	1646.48
1000	17.0711	8610.52	14.0389	21645.2

Wyznaczanie najkrótszej ścieżki w grafie dla gęstości 25%



Wyznaczanie najkrótszej ścieżki w grafie dla gęstości 50%





Wnioski:

- Uzyskane wyniki pokrywają się w przybliżeniu z tymi wynikającymi z oczekiwanej teorii
- Dla problemu wyznaczenia najkrótszej ścieżki w grafie dla reprezentacji grafu w formie macierzy incydencji lepiej używać algorytmu algorytmu Kruskala a dla zapisu w formie listy następników algorytmu Prima
- Dla problemu minimalnego drzewa rozpinającego dla reperacji w formie macierzy incydencji lepiej używać algorytmu Dijkstry a dla zapisu w formie listy następników algorytm Forda-Bellmana
- Dla zapisu w formie macierzy incydencji wszystkie algorytmy były wolniejsze w porównaniu do listy następników
- Przy zwiększaniu gęstości grafu zwiększały się różnice w wydajności algorytmów

Literatura

- https://en.wikipedia.org/wiki/Prim%27s_algorithm#Time_complexity
- http://www.algorytm.org/algorytmy-grafowe/algorytm-prima.html
- https://en.wikipedia.org/wiki/Kruskal%27s_algorithm#Complexity
- https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#Running_time

- https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm#Algorithm
- https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/
-
-