Politechnika Wrocławska Wydział Informatyki i Telekomunikacji

Sprawozdanie

SDiZO – badanie efektywności algorytmów grafowych w zależności od rozmiaru instancji oraz sposobu reprezentacji grafu w pamięci komputera

Autor: Stanisław Strauchold 259142

Grupa: Piątek TP 11:15

Prowadzący: dr inż. Dariusz Banasiak

Termin oddania: 03.06.2022

1. Opis projektu

W tym projekcie zadaniem była implementacja algorytmów znajdywania najkrótszej ścieżki oraz minimalnego drzewa rozpinającego w grafie. Grafy miały być przechowywane w pamięci komputera przy użyciu następujących struktur danych: lista sąsiedztwa, macierz incydencji

Algorytmami zaimplementowanymi przeze mnie były:

- Algorytm Dijsktry
- Algorytm Prima
- Algorytm Bellmana-Forda

Wykorzystanym językiem programowania był C++. Implementacja algorytmów i struktur danych została wykonana bez wykorzystania gotowych bibliotek.

2. Złożoność obliczeniowa

W celu oszacowania złożoności obliczeniowej dla poszczególnych struktur posłużymy się następującymi parametrami:

- E – liczba krawędzi w grafie
- V – liczba wierzchołków w grafie

Zgodnie z literaturą, a dokładniej mówiąc z książką „Wprowadzenie do algorytmów" złożoności dla poszczególnych algorytmów prezentują się następująco

Algorytm Dijkstry

O złożoności decyduje sposób implementacji kolejki priorytetowej:

- przez zwykłą tablicę: O(E * V)

- poprzez kopiec: O(E * logV)

Algorytm Prima

Tak jak w poprzednim przypadku o złożoności decyduje implementacja kolejki priorytetowej:

- przez zwykłą tablicę: O(E * V)

-przez kopiec: O(E*logV)

Algorytm Bellmana-Forda

Dla każdego grafu złożoność obliczeniowa wynosi O (E*V)

3. Plan eksperymentu

Aby oszacować efektywność poszczególnych algorytmów w zależności od rozmiaru instancji oraz sposobu reprezentacji grafu w pamięci komputera konieczne było opracowanie modelu służącego do badań. Model zgodny z wytycznymi podanymi przez Prowadzącego zakładał wylosowanie grafu o określonej liczbie wierzchołków i procencie zagęszczenia krawędzi. Następnie dla każdego zestawu

danych dokonywany był 50-krotny pomiar czasu wykonywania danego algorytmu. Następnie z otrzymanych wyników liczona była średnia. Licznik czasu był inicjowany przed samym wywołaniem metody, a wartość końcowa była pobierana zaraz po zakończeniu działania algorytmu. W celu uzyskania jak największej dokładności, na czas pomiarów z algorytmów usunięto linijki odpowiedzialne za wypisanie wyniku operacji.

3.1. Metoda generowania grafu

Aby nasz eksperyment zakończył się sukcesem należało zapewnić, aby generowane przez nas losowo grafy były spójne oraz miały losową strukturę. Metoda polegała na obliczeniu potrzebnych do wylosowania krawędzi na podstawie obliczenia ilości krawędzi w grafie pełnym o podanej liczbie wierzchołków. Następnie otrzymana ilość była mnożona przez wartość w procentach zależną od oczekiwanego zagęszczenia grafów. Wynik był zaokrąglany w dół do części całkowitej.

Aby graf był spójny początkowo generowane były krawędzie łączące wierzchołek startowy z każdym innym wierzchołkiem. Następnie generowana była pozostała ilość dostępnych krawędzi. Na początku algorytmu losowany był wierzchołek startowy i końcowy oraz waga krawędzi. Potem algorytm sprawdzał czy w grafie nie istnieje już krawędź łącząca wylosowane wierzchołki. Jeśli nie, krawędź dodawana była do grafu. W przeciwnym wypadku losowanie było do skutku powtarzane.

4. Wyniki pomiarów

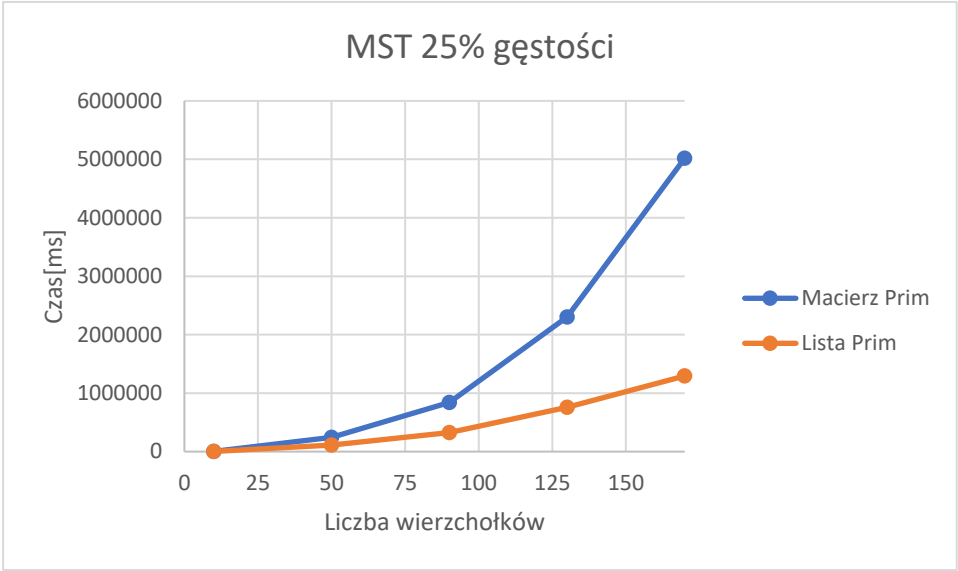
4.1. MST

% / V	25 [ms]	50 [ms]	75 [ms]	00 [ms]
10	3434	7780	10874	14730
50	240490	492944	857806	1323932
90	837730	2343556	4605116	7531120
130	2304920	6781838	13539452	22409428
170	5018752	15142908	38529520	89139850

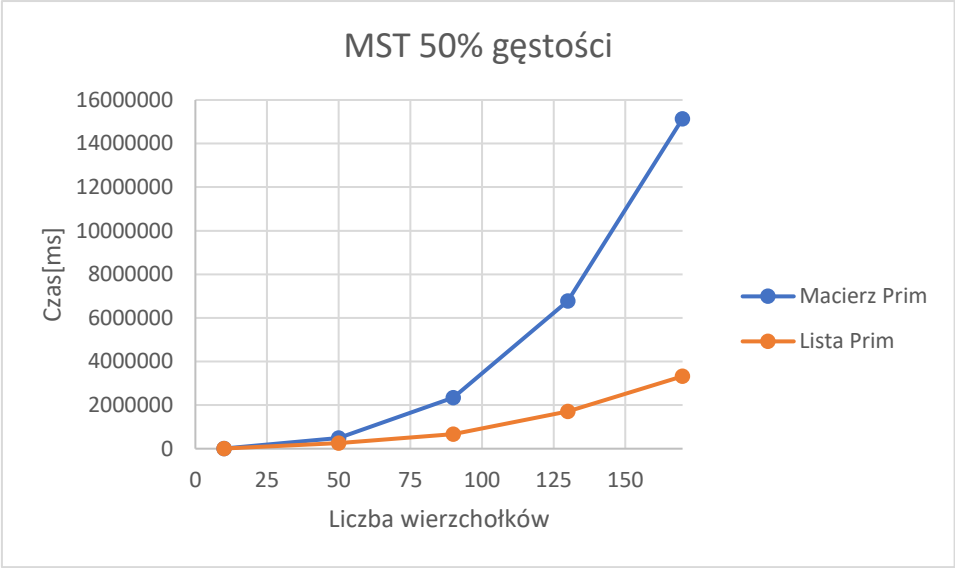
Tabela 1: Wyniki pomiarów dla algorytmu Prima w macierzy incydencji

% / V	25 [ms]	50 [ms]	75 [ms]	00 [ms]
10	3144	6666	8634	10604
50	108970	248308	348186	465186
90	324808	661190	1134646	1740854
130	760654	1711014	3081046	4860862
170	1293846	3322982	6317056	10178296

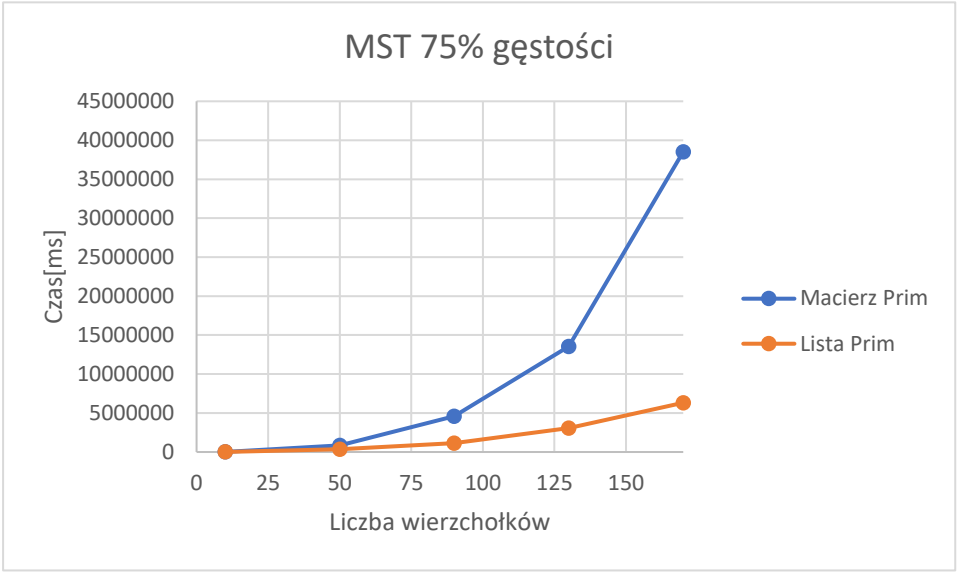
Tabela 2: Wyniki pomiarów dla algorytmu Prima w liście sąsiedztwa



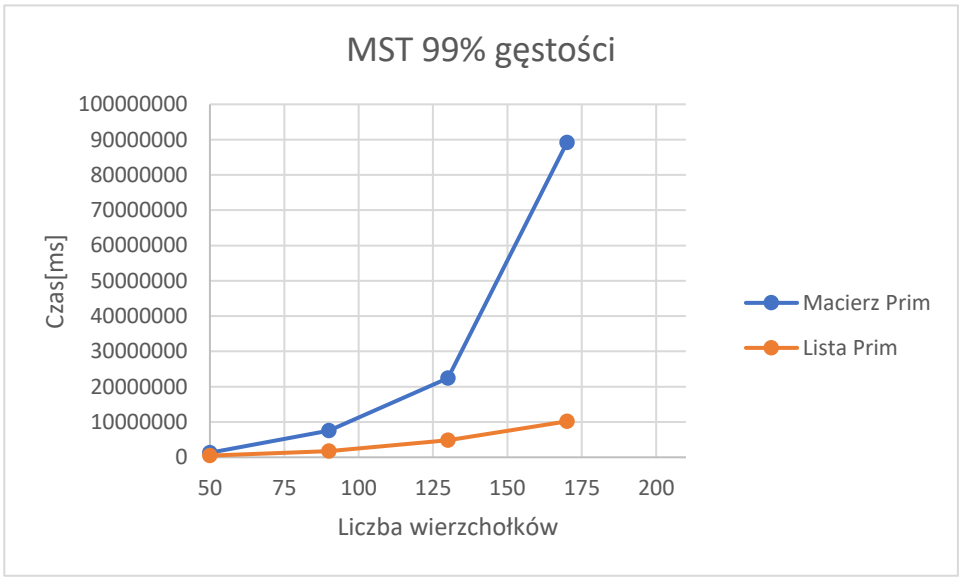
Wykres 1: Czas wykonywania algorytmu Prima w macierzy i liście w zależności od ilości wierzchołków dla gęstości 25%



Wykres 2: Czas wykonywania algorytmu Prima w macierzy i liście w zależności od ilości wierzchołków dla gęstości 50%



Wykres 3: Czas wykonywania algorytmu Prima w macierzy i liście w zależności od ilości wierzchołków dla gęstości 75%



Wykres 4: Czas wykonywania algorytmu Prima w macierzy i liście w zależności od ilości wierzchołków dla gęstości 99%

4.2. Problem najkrótszej ścieżki w grafie

% / V	25 [ms]	50 [ms]	75 [ms]	00 [ms]
10	3036	4776	6518	7724
50	31694	48684	6650	86452
90	58292	109618	156552	206140
130	120540	199390	291621	401500
170	175209	324216	513766	732542

Tabela 3: Wyniki pomiarów dla algorytmu Dijkstry w liście sąsiedztwa

% / V	25 [ms]	50 [ms]	75 [ms]	00 [ms]
10	3378	7270	9642	12480
50	130168	266928	446238	691104
90	430284	1171258	2311360	3827242
130	1165086	3403516	6740218	11316498
170	2511970	7674788	25405632	51672564

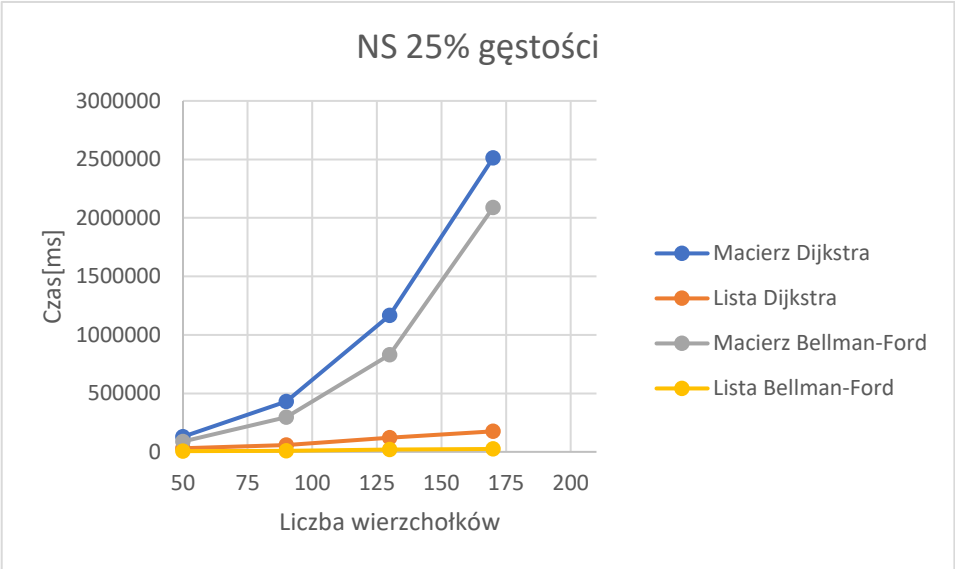
Tabela 4: Wyniki pomiarów dla algorytmu Dijkstry w macierzy incydencji

% / V	25 [ms]	50 [ms]	75 [ms]	00 [ms]
10	588	1362	2346	2756
50	5472	11010	18698	23770
90	9398	24722	41140	59772
130	19072	42228	75864	123538
170	24636	68770	123538	261826

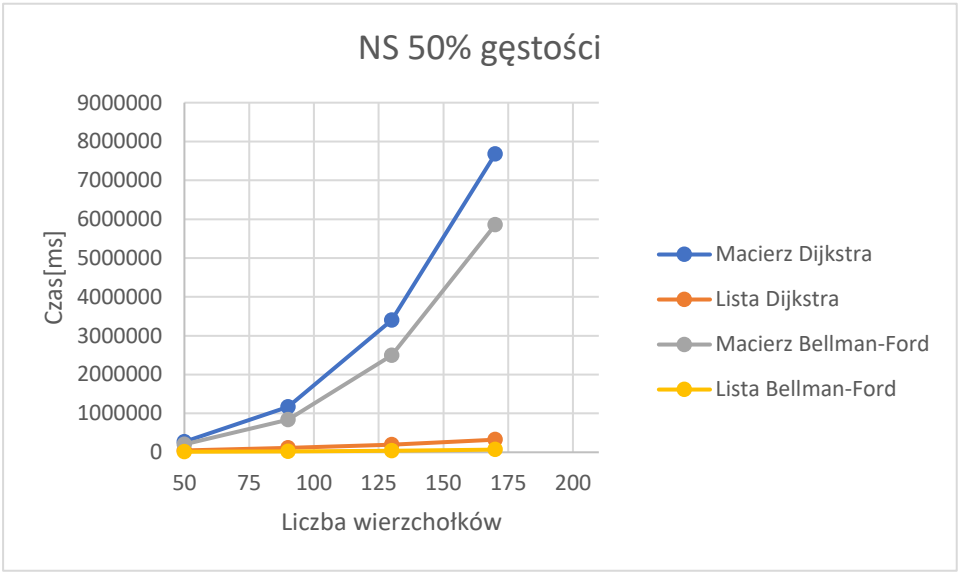
Tabela 5: Wyniki pomiarów dla algorytmu Bellmana-Forda w liście sąsiedztwa

% / V	25 [ms]	50 [ms]	75 [ms]	00 [ms]
10	1448	3676	5220	7154
50	88454	201316	330160	498044
90	295626	838602	1643584	2705736
130	828454	2491430	4940694	8258522
170	2087942	5854056	12526716	27313328

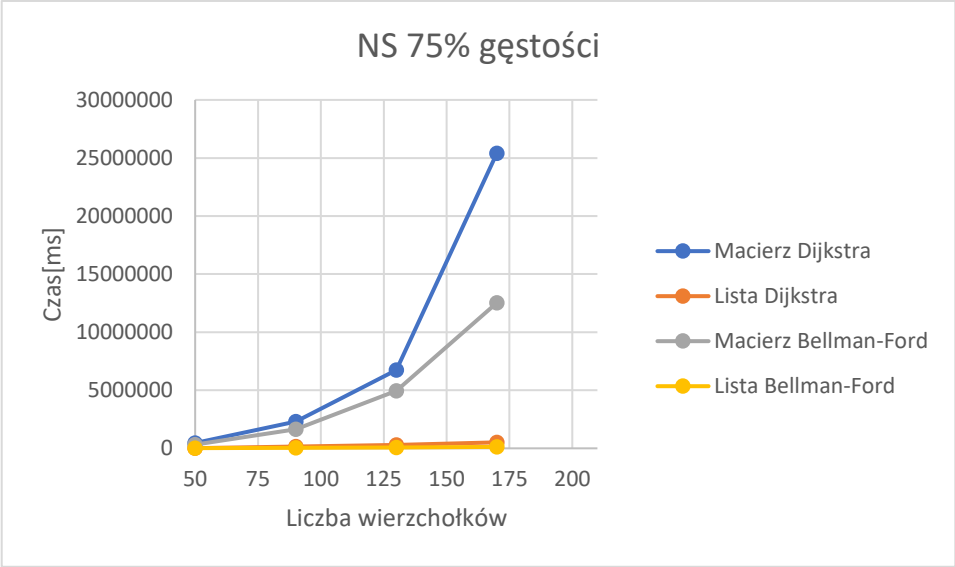
Tabela 5: Wyniki pomiarów dla algorytmu Bellmana-Forda w macierzy incydencji



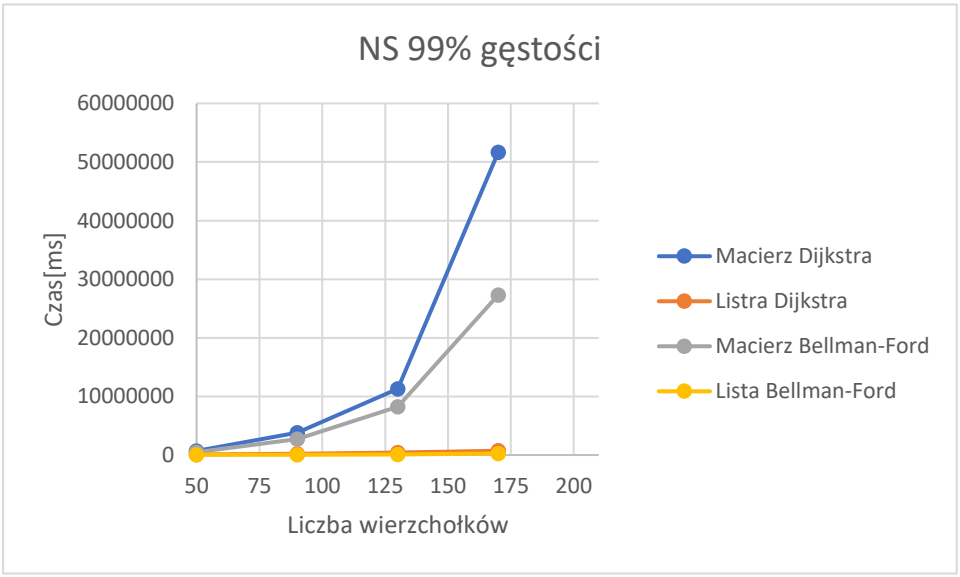
Wykres 4: Czas wykonywania algorytmów Dijkstry i Bellmana-Forda w macierzy i liście w zależności od ilości wierzchołków dla gęstości 25%



Wykres 5: Czas wykonywania algorytmów Dijkstry i Bellmana-Forda w macierzy i liście w zależności od ilości wierzchołków dla gęstości 50%

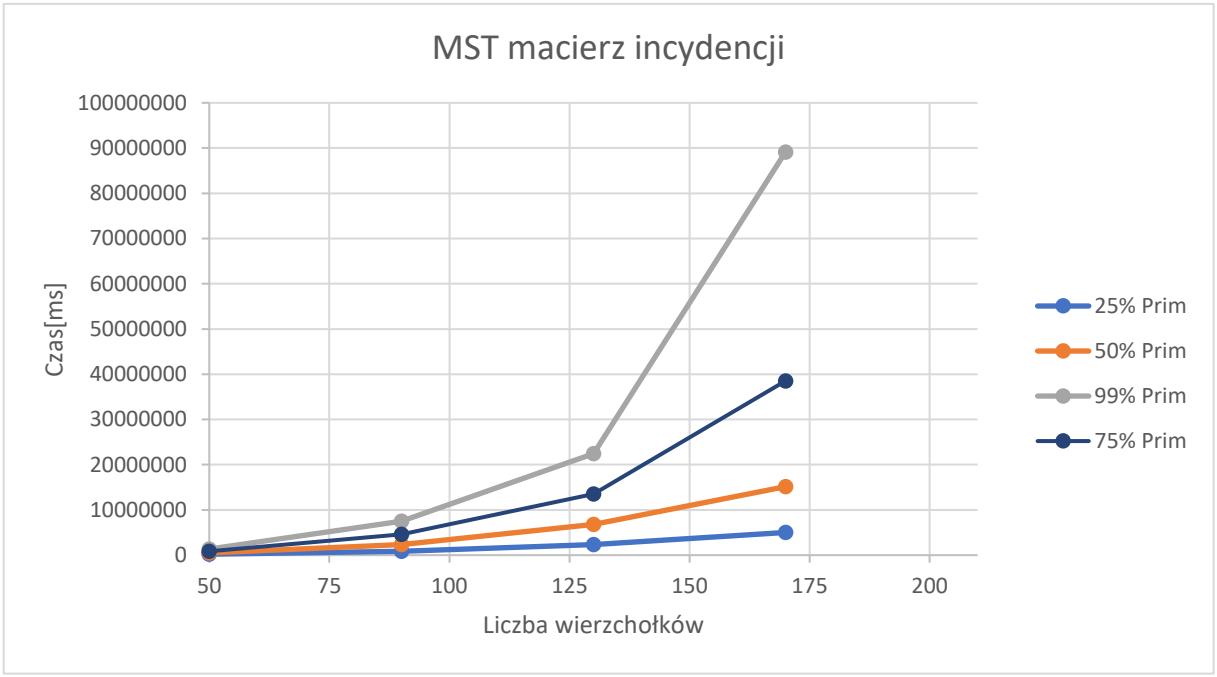


Wykres 6: Czas wykonywania algorytmów Dijkstry i Bellmana-Forda w macierzy i liście w zależności od ilości wierzchołków dla gęstości 75%

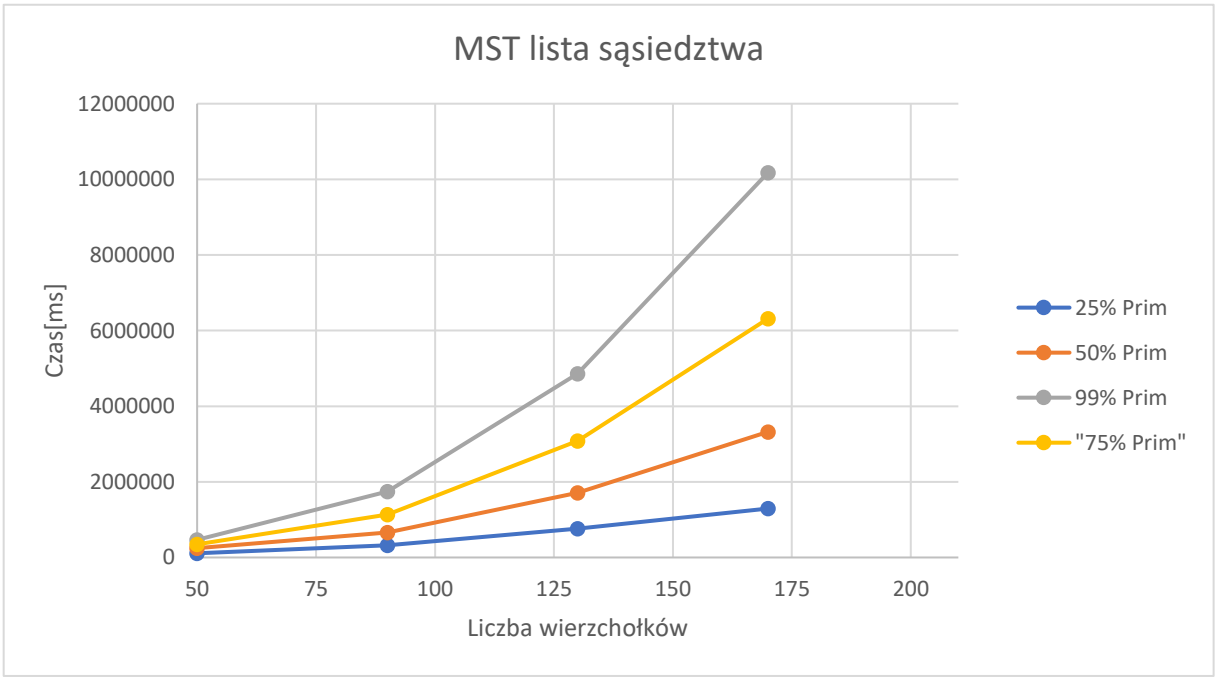


Wykres 7: Czas wykonywania algorytmów Dijkstry i Bellmana-Forda w macierzy i liście w zależności od ilości wierzchołków dla gęstości 99%

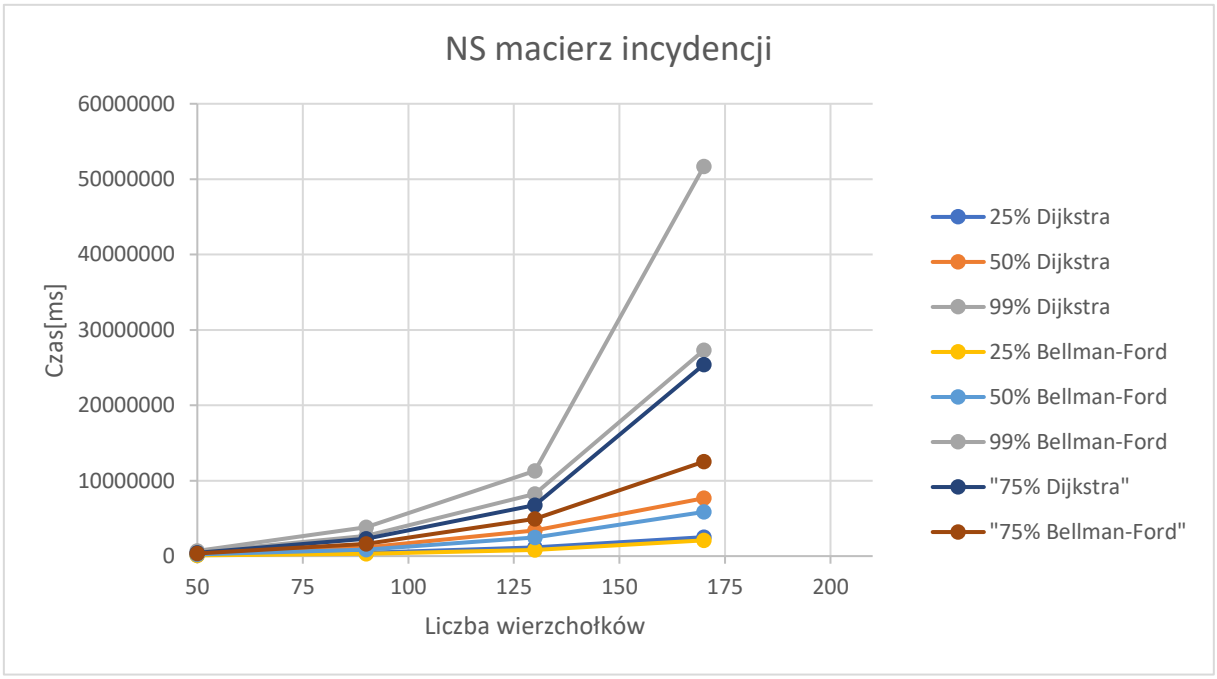
4.3. Osobne wykresy dla każdej gęstości grafu



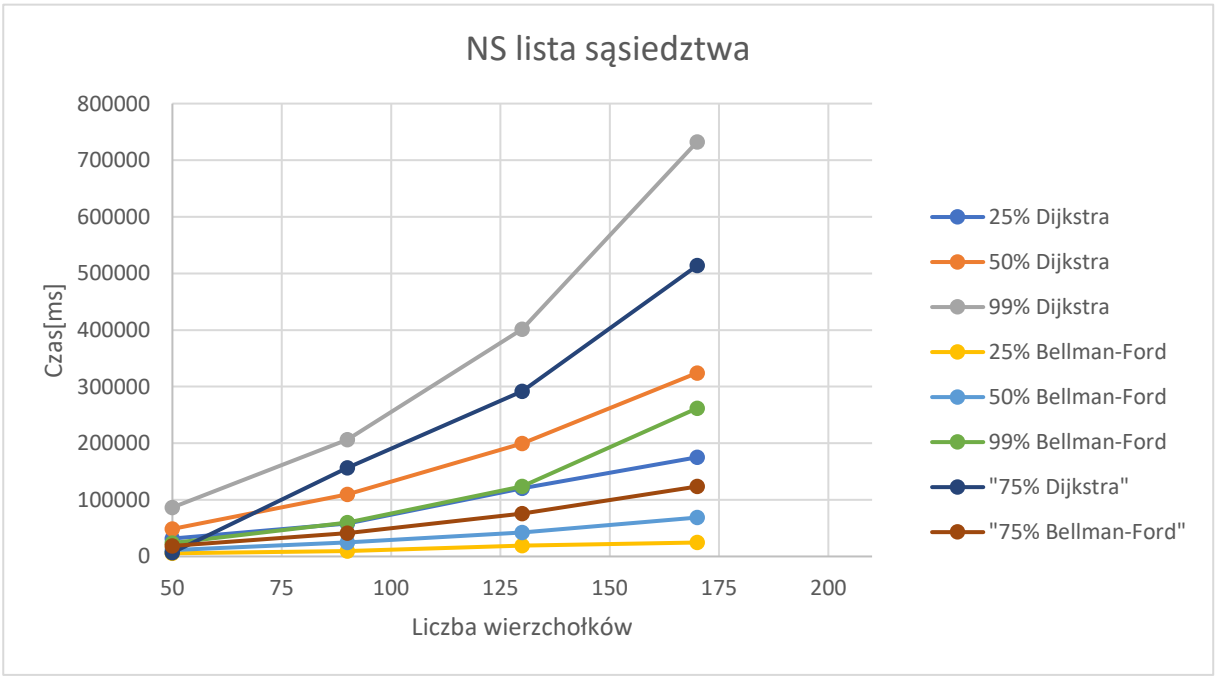
Wykres 8: Czas wykonywania algorytmu Prima w macierzy incydencji dla różnych gęstości w zależności od ilości wierzchołków w grafie



Wykres 9: Czas wykonywania algorytmu Prima w liście sąsiedztwa dla różnych gęstości w zależności od ilości wierzchołków w grafie



Wykres 10: Czas wykonywania algorytmów Dijkstry i Bellmana-Forda w macierzy incydencji dla różnych gęstości w zależności od ilości wierzchołków w grafie



Wykres 11: Czas wykonywania algorytmów Dijkstry i Bellmana-Forda w liście sąsiedztwa dla różnych gęstości w zależności od ilości wierzchołków w grafie

5. Wnioski

Dla wszystkich algorytmów lista sąsiedztwa dawała lepsze wyniki niż macierz incydencji. Powodem tego może być dłuższy proces szukania odpowiedniej krawędzi grafu, który w moim programie odbywa się poprzez przeszukiwanie macierzy. W liście sąsiedztwa aby pobierać krawędzie wystarczy je kolejno odczytywać ze struktury. Najszybszym z zaimplementowanych algorytmów jest algorytm Bellmana Forda.