Algorytmy sortowania

Nr ćwiczenia: 3

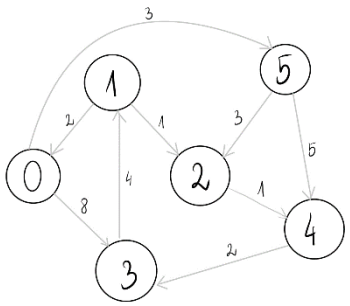
Temat ćwiczenia: Grafy

Wykonawca:	Mgr inż. Marcin Ochman
Imię i Nazwisko nr indeksu	Martyna Żukowska 252877
Termin zajęć: dzień tygodnia, godzina	Czwartek 15:15
Numer grupy ćwiczeniowej	E12-99j
Data oddania sprawozdania:	13.05.2021
Ocena końcowa	

1. Cel ćwiczeń

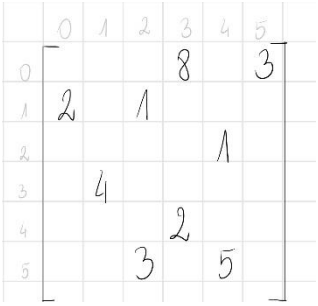
Należy zaimplementować 2 wymienione algorytmy szukania najszybszej ścieżki do wybranego wierzchołka grafu stworzonych za pomocą macierzy i listą. Następnie należy zbadać ich wydajność dla różnych rozmiarów grafów i różnej ich wielkości. Sprawozdanie ma zawierać wykresy czasu wykonaniu algorytmów w zależności od liczby danych do wygenerowania grafów.

2. Teoria



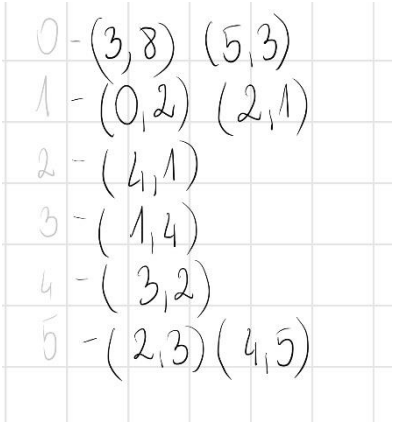
Rysunek 2 Schemat przykładowego Grafu

Graf możemy reprezentować za pomocą macierzy kwadratowej o stopniu ilości wierzchołków w grafie. Macierz ta to macierz sąsiedztwa, pokazuje ona połączenia wierzchołków krawędziami w następujący sposób: wiersze macierzy sąsiedztwa odwzorowują wierzchołki początkowe krawędzi, a kolumny wierzchołki jej końca. Odpowiednie komórki w macierzy mają wartość wagi połączenia wartości indeksu wiersza i kolumny komórki.



Rysunek 1 Macierz sąsiedztwa do grafu z Rysunku1

Innym sposobem reprezentowania grafu są listy sąsiedztwa reprezentowaną jako vektor o ilości elementów równym ilości wierzchołków grafu. Każdy element vektora jest listą (w naszym przypadku kolejnym vektorem). W niej przechowywane są informacje dotyczące połączeń. Wykorzystałam do tego jeszcze jeden vektor który przechowywał informacje o tym z którym z którym wierzchołkiem jest połączony wierzchołek badany oraz waga tego połączenia.



Rysunek 3 Lista sąsiedztwa dla grafu z Rysunku1

Algorytmy

Algorytm Dijkstry służy do znajdowania najkrótszej ścieżki do wybranego wierzchołka w grafie o nieujemnych wagach krawędzi. Mając dany graf z początkowym wierzchołkiem algorytm znajduje odległości od niego do wszystkich pozostałych wierzchołków przez najszybszą ścieżkę , przy okazji zliczając koszt tego przejścia. Algorytm Dijkstry jest algorytmem zachłannym, to znaczy że dokonuje najlepiej rokującego w danym momencie wyboru rozwiązania częściowego. Innymi słowy algorytm zachłanny nie dokonuje oceny czy w kolejnych krokach jest sens wykonywać dane działanie, dokonuje decyzji lokalnie optymalnej, dokonuje on wyboru wydającego się w danej chwili najlepszym, kontynuując działania wynikające z podjętej decyzji.

Złożoność obliczeniowa algorytmu Dijkstry zależy od liczby wierzchołków i krawędzi grafu. Dla:
- grafów gęstych (krawędzie = Θ(wierzchołki^2))
  Złożoność obliczeniowa = Ω(wierzchołki^2)

- grafów rzadkich (krawędzie = Θ(wierzchołki))
  Złożoność obliczeniowa = Ω(krawędzie ∗ log wierzchołki)

Algorytm Bellmana-Forda służy do wyznaczania najkrótszych ścieżek w grafie, nawet dla ujemnych wag połączenia(bez ujemnych cykli od wierzchołka startowego). Wyznacza najkrótsze ścieżki z wierzchołka startowego do pozostałych wierzchołków. W tym algorytmie korzystamy z metody relaksacji krawędzi.
Złożoność obliczeniowa algorytmu Bellmana-Forda zależy od liczby wierzchołków i krawędzi grafu jest równa Θ(wierzchołki*krawędzie)

Dla każdego przypadku zostały wykony pomiar. Zostały w nim generowane grafy, w których:

E - liczba krawędzi grafu (E ∈ {25%, 50%, 75%, 100%} ∗ Emax)
Emax = V ∗ (V − 1)
V - liczba wierzchołków grafu (V ∈ {10, 50, 100, 150, 250})

3. Eksperyment

3.1 Tabela pomiarów algorytmu Dijkstry

Lista	25% gęstości	50 % gęstości	75% gęstości	100% gęstości
10	0,000236239	0,000287745	0,000435358	0,000437752
50	0,003867190	0,004641030	0,007362590	0,007173320
100	0,012985400	0,013192100	0,012316300	0,016417600
150	0,019045700	0,025972000	0,028604300	0,039115400
250	0,027036900	0,054212800	0,098755100	0,151261000
macierz				
10	0,000244581	0,000298082	0,000279509	0,000294651
50	0,002484570	0,004828090	0,004629630	0,005076510
100	0,011723500	0,015267100	0,017683800	0,019008100
150	0,013476400	0,018573800	0,021783200	0,025512900
250	0,017052600	0,024840000	0,030945200	0,034370700

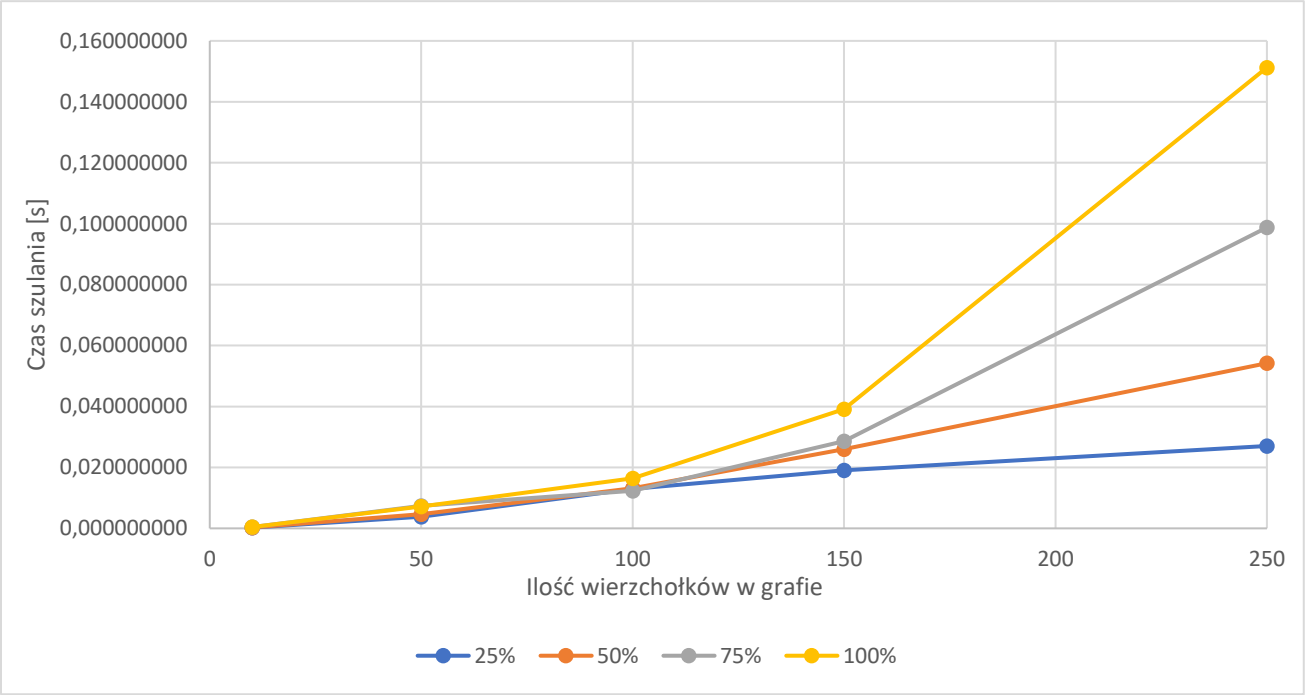
W tabeli zostały podane czasy (w sekundach) znalezienia najkrótszych ścieżek grafów o różnej gęstości i różnej ilości wierzchołków.

3.2 Tabela pomiarów algorytmu Bellmana-Forda

lista	0,25	0,5	0,75	1
10	0,000524286	0,000857812	0,001708890	0,001747610
50	0,032312100	0,047805600	0,068011300	0,103243000
100	0,171607000	0,409770000	0,768616000	1,225050000
150	0,630825000	1,745270000	3,317990000	5,435150000
250	3,972370000	11,487900000	24,583700000	37,798300000
macierz				
10	0,000436987	0,000929597	0,001146620	0,001223690
50	0,020728100	0,038555000	0,042442600	0,052309100
100	0,123293000	0,215436000	0,298167000	0,385951000
150	0,379835000	0,707765000	1,004050000	1,308350000
250	1,849500000	3,507110000	5,218020000	6,601940000

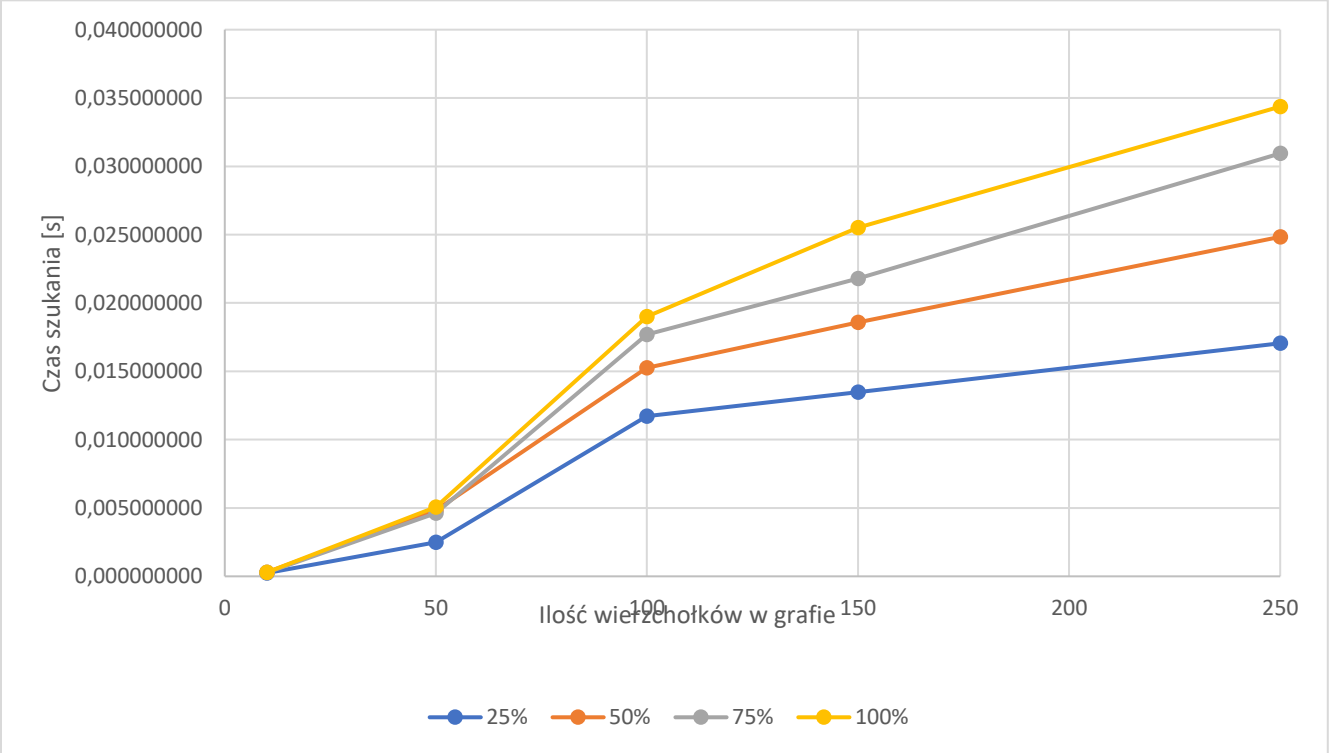
W tabeli zostały podane czasy (w sekundach) znalezienia najkrótszych ścieżek grafów o różnej gęstości i różnej ilości wierzchołków.

3.3 Wykres algorytmu Dijkstry dla grafu w postaci listy



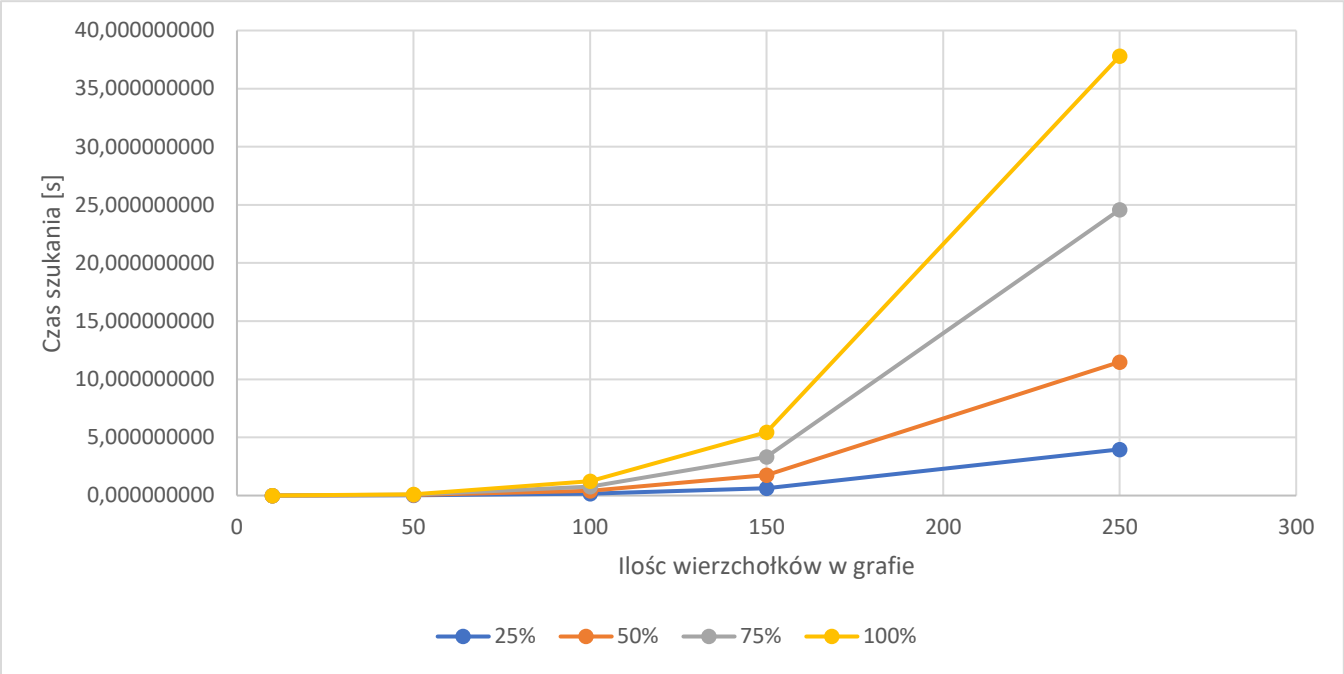
Istnieje wykładnicza zależność między kolejnymi czasami dla coraz większych grafów.

3.4 Wykres algorytmu Dijkstry dla grafu w postaci macierzy



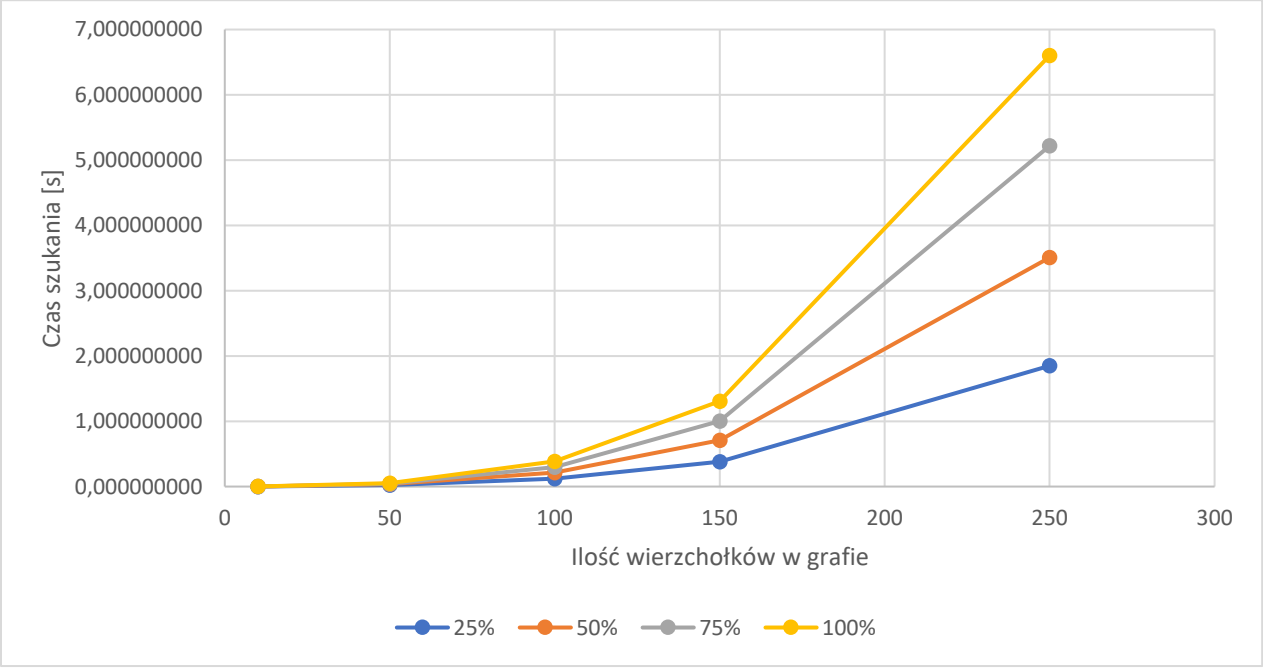
Isnieje wykładnicza zależność podobnie jak w przypadku list jednak te czasy są tak małe, a pomiary pojedyńcze, że we wartości są objęte błędem pomiaru.

3.3 Wykres algorytmu Bellmana-Forda dla grafu w postaci listy



Istnieje wykładnicza zależność między kolejnymi czasami dla coraz większych grafów.

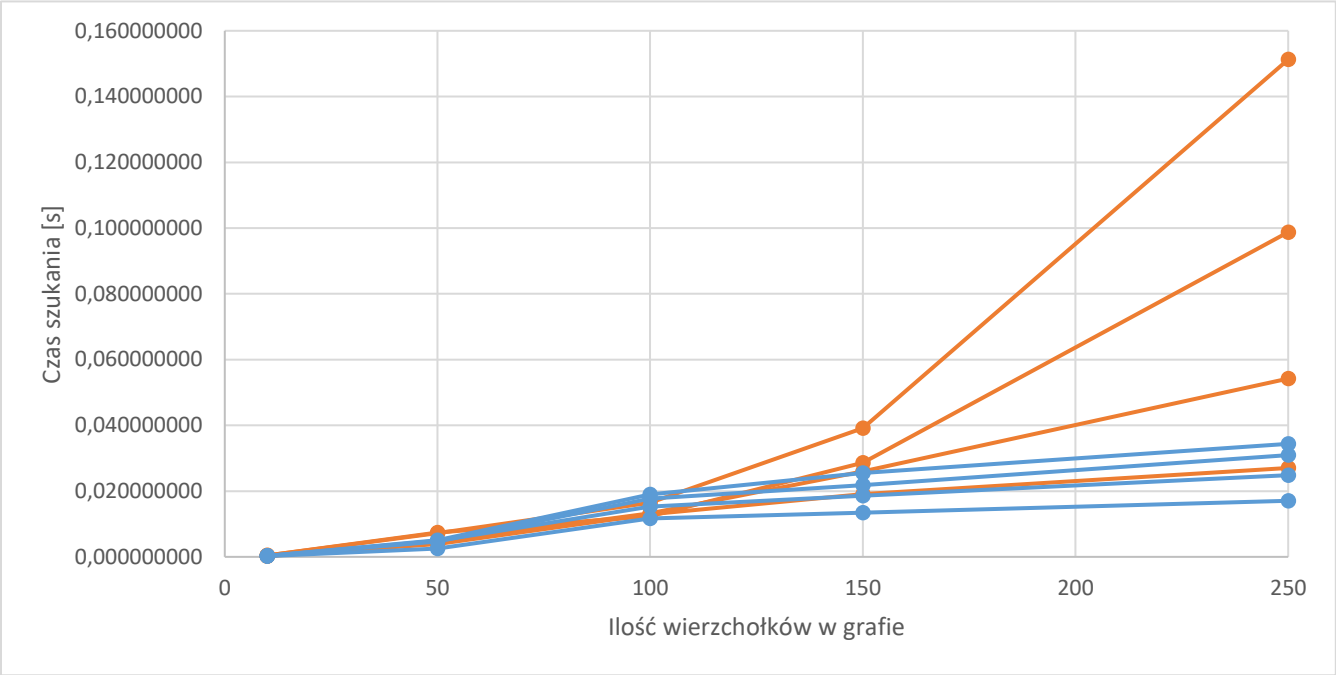
3.3 Wykres algorytmu Bellmana-Forda dla grafu w postaci macierzy



Istnieje wykładnicza zależność między kolejnymi czasami dla coraz większych grafów.

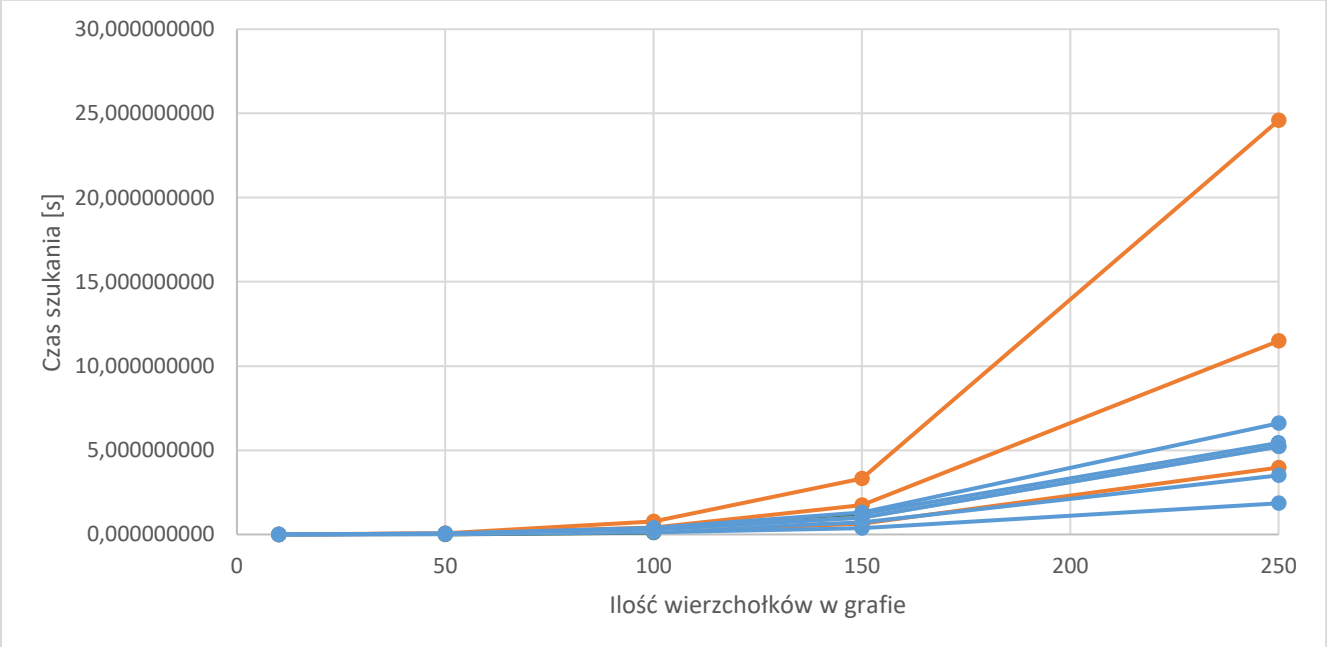
4. Wykres porównawczy

Dla algorytmu Dijkstry



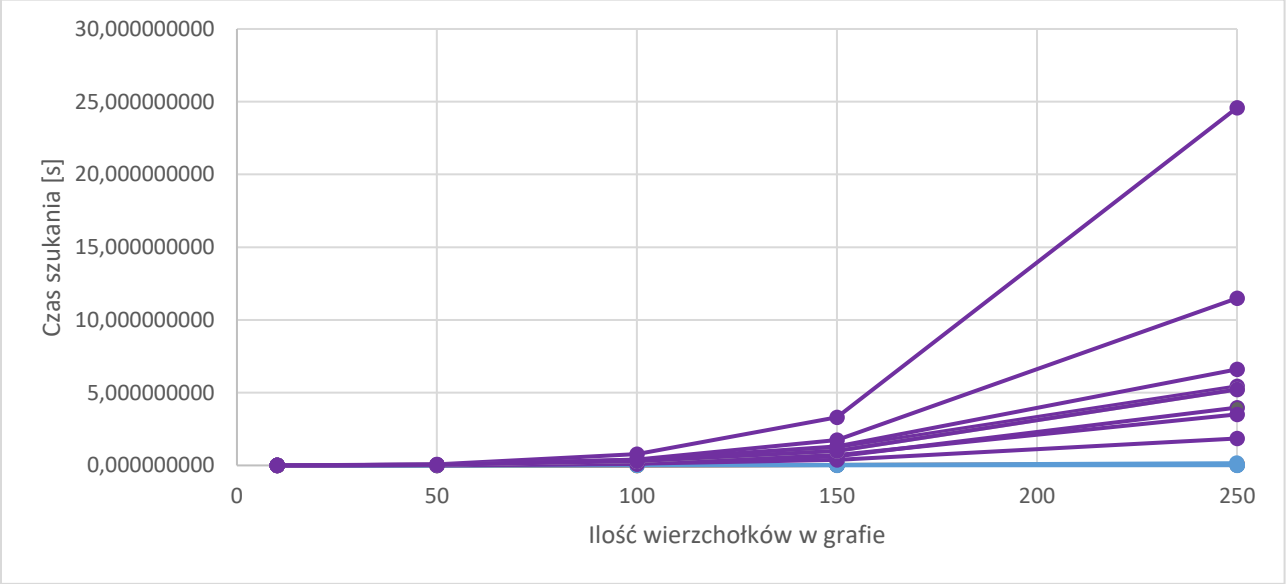
Na pomarańczowo zostały zaznaczone dane dotyczące grafów w postaci list, a na niebiesko w postaci macierzowej.

Dla algorytmu Bellmana-Forda



Na pomarańczowo zostały zaznaczone dane dotyczące grafów w postaci list, a na niebiesko w postaci macierzowej.

Dla algorytmów



Na filetowo zostały zaznaczone dane dotyczące algorytmu Bellmana-Forda, a na niebiesko algorytmu Dijkstry. Przez nałożenie na siebie niektórych danych nie wszystkie informacje widać

5. Wnioski

W obu algorytmach została zauważona ta sama zależność, to znaczy czas szukania najkrótszych ścieżek dla grafu na bazie macierzy jest znacznie szybszy niż dla grafu na bazie list. Czasy te są w obu przypadkach około 5 krotnie większe, co jest dość zauważalne zwłaszcza w większych grafach.

Tak samo łatwo spostrzec, że dla grafów o większej gęstości, a co za tym idzie większej ilości krawędzi czas znajdowania najszybszych ścieżek również jest dłuższy. Co jest bardzo logiczne. Taka zależność można zauważyć dla każdego algorytmu na bazie macierzy, jak i na bazie list.

Jeśli chodzi o porównanie dwóch algorytmów do siebie to algorytm Dijkstry jest znacznie szybszy niż algorytm Bellmana-Forda. W skrajnych przypadkach rozwiązuje problem nawet 5 razy szybciej.

Na tej podstawie można wywnioskować, że jeśli nie mamy do czynienia z ujemnymi wartościami najszybszym sposobem na znalezienie najszybszych ścieżek jest algorytm Dijkstry dla grafów na bazie macierzy (najlepiej o małej gęstości), a dla grafów z ujemnymi wartościami gdy jesteśmy zmuszeni skorzystać z Bellmana-Forda najbardziej efektywniej będzie również skorzystać z grafów na bazie macierzy.

Warto na koniec również wspomnieć, że błędy przy pomiarach mogą być spowodowane błędami pomiarów timera, jak i losowością wygenerowania grafów.

6. Bibliografia

https://pl.wikipedia.org/wiki/Algorytm Dijkstry
https://pl.wikipedia.org/wiki/Algorytm_Bellmana-Forda