Marcin Całkowski
nr 218191
Śr. 13.15

Struktury danych i złożoność obliczeniowa

Zadanie projektowe nr 2

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

Prowadzący: Zbigniew Buchalski

Politechnika Wrocławska

1. Wstęp

Tematem projektu jest przedstawienie działania algorytmów grafowych badających najkrótszą drogę w grafie (algorytm Dijkstry) oraz najmniejsze drzewo rozpinające (algorytm Prima). Wyniki eksperymentu zawierają wyniki różnych grafów, w zależności od jego wielkości, gęstości oraz typu przechowywania go w pamięci komputera (lista sąsiedztwa oraz macierz incydencji).

2. Podstawowe zagadnienia teoretyczne

Grafem nazywamy zbiór wierzchołków i krawędzi które łączą poszczególne wierzchołki. Mogą one reprezentować różne rzeczy takie jak połączenia poszczególnych modułów, dróg, sieci czy relacji między obiektami przedstawionymi jako poszczególne wierzchołki. W wielu przypadkach są one najlepszą metodą zapisania, a także pokazania złożoności obiektu, lub ich grup, dlatego bardzo przydatny staje się ich zapis w pamięci komputera.

Istnieje cała teoria matematyczna, zwana Teorią Grafów, która zajmuje się tymi strukturami, ich własnościami oraz zasadami wykonywania na nich różnych działań. Za ojca grafów uważany jest **Leonard Euler**, szwajcarski matematyk, który dokonał bardzo wielu odkryć matematycznych, np. w rachunku różniczkowym i całkowym.

Chodź w życiu codziennym jesteśmy przyzwyczajeni do zapisu grafów na kartce, taka reprezentacja jest bardzo trudna w komputerze. Dlatego też grafy w komputerze zapisujemy w innej postaci, żeby ułatwić ich wyświetlanie, wykonywanie na nich działań ale przede wszystkim zapis ich w pamięci komputera. Zadaniem tego projektu jest właśnie porównanie szybkości wykonywania algorytmów na grafach zapisanych w postaci macierzy incydencji oraz listy sąsiedztwa.

2.1 Oznaczenia i pojęcia

G = (V, E) – graf o zbiorze wierzchołków V = v0…vn-1 i krawędziach E = e0…em-1.

(u, v) ϵ E – krawędź zaczynająca się w wierzchołku u i kończąca wierzchołkiem v.

w(u, v) – waga krawędzi zaczynającej się w wierzchołku u i kończącej wierzchołkiem v.

Graf spójny – w przypadku grafu skierowanego to graf, w którym dla każdej pary wierzchołków istnieje łącząca je ścieżka. Jeśli chodzi o graf skierowany; warunkiem koniecznym spójności jest spójność grafu podstawowego, czyli bez kierunków na krawędziach. Jeśli tylko ten warunek jest spełniony, graf jest słabo spójny. Dodatkowo, graf skierowany jest silnie spójny, jeśli między każdymi dwoma wierzchołkami istnieje ścieżka (przy uwzględnieniu kierunków krawędzi).

Drzewo rozpinające – drzewo, które zawiera wszystkie wierzchołki grafu G oraz niektóre jego krawędzie. Jeśli graf ma n wierzchołków, drzewo rozpinające ma n wierzchołków i n-1 krawędzi.

Minimalne drzewo rozpinające – drzewo rozpinające o najmniejszej sumie wag krawędzi ze wszystkich drzew rozpinających grafu G.

Gęstość grafu – stosunek liczby krawędzi do maksymalnej liczby krawędzi.

2.2 Reprezentacja grafu w pamięci komputera

Macierz incydencji jest macierzą A o wymiarze n x m, gdzie n oznacza liczbę wierzchołków grafu, a m liczbę jego krawędzi. Każdy wiersz tej macierzy odwzorowuje jeden wierzchołek grafu. Każda kolumna odwzorowuje jedną krawędź. Złożoność pamięciowa to O(V * E) (oznaczenia: V – wierzchołki, E - krawędzie). Operacje przejrzenia wszystkich krawędzi, sąsiadów danego wierzchołka i sprawdzenie, czy dana krawędź istnieje wykonują się w czasie O(E).

Lista sąsiedztwa jest reprezentacją grafu, w której wykorzystujemy tablicę n elementową A, gdzie n oznacza liczbę wierzchołków. Każdy element tej tablicy jest listą. Lista reprezentuje wierzchołek startowy. Na liście są przechowywane numery wierzchołków końcowych, czyli sąsiadów wierzchołka startowego, z którymi jest on połączony krawędzią. Listy sąsiedztwa są efektywnym pamięciowo sposobem reprezentacji grafu w pamięci komputera, ponieważ zajmują pamięć rzędu O(m), gdzie m oznacza liczbę krawędzi grafu. Listy sąsiedztwa pozwalają w prosty sposób reprezentować pętle oraz krawędzie wielokrotne, co sprawia, że są bardzo chętnie stosowane w algorytmach grafowych.

2.3 Opis algorytmów użytych w projekcie

W projekcie rozwiązano problemy wyznaczania minimalnego drzewa rozpinającego (MST) za pomocą algorytmu Prima oraz wyznaczania najkrótszej ścieżki w grafie za pomocą algorytmu Dijkstry. Wszystkie algorytmy realizowane są korzystając z dwóch już podanych wcześniej metod zapisu grafu w pamięci.

- Poszukiwanie najkrótszych ścieżek w grafie : **Algorytm Dijkstry**

Algorytm Dijkstry znajduje w grafie najkrótsze ścieżki pomiędzy wybranym wierzchołkiem a wszystkimi pozostałymi, wyliczając również koszt przejścia każdej z tych ścieżek, czyli sumę wag krawędzi na ścieżce. Algorytm ma złożoność czasową $O(V^2)$ przy wykorzystaniu wyszukiwania liniowego podczas szukania wierzchołków o najmniejszym koszcie dojścia. Czas ten może być zmniejszony dzięki wykorzystaniu kolejki priorytetowej opartej na kopcu binarnym. Wtedy w korzeniu przechowywany jest wierzchołek o najmniejszej wartości kosztu dojścia. Złożoność czasowa upraszcza się do $O(V * \log_2 V)$ – tyle, ile wynosi czas przywracania własności kopca.

- Wyznaczanie minimalnego drzewa rozpinającego: **Algorytm Prima**

Algorytm zajmujący się wyznaczaniem minimalnego drzewa rozpinającego w grafie. Na początku dodaje do zbioru A reprezentującego drzewo krawędź o najmniejszej wadze, łączącą wierzchołek początkowy v z dowolnym wierzchołkiem. W każdym kolejnym kroku procedura dodaje do A najlżejszą krawędź wśród krawędzi łączących wierzchołki już odwiedzone z nieodwiedzonymi. Jeśli struktura A jest kolejką priorytetową opartą na kopcu binarnym, czasowa złożoność obliczeniowa operacji wynosi O(m * log2n).

3. Plan eksperymentu

Eksperyment został wykonany w moim własnym programie którego kod został oparty na algorytmach znalezionych w materiałach z zajęć lub z sieci. Podstawą jego wykonania jest zbadanie czasu przeprowadzenia odpowiedniej operacji dla algorytmów oraz porównanie ich wyników w działaniu dla macierzy incydencji oraz listy sąsiedztwa Dane do testów czasowych zostały każdorazowo wylosowane aby dać pewność średnich wyników. Dla każdej struktury test przeprowadzany jest **100 razy**, uzyskany wynik czasowy jest średnią z wszystkich uzyskanych wyników.

Czas został zmierzony przy pomocy biblioteki **chrono** (dostępnej od standardu c++11). Pozwala ona na uzyskanie dobrego pomiaru czasu bez komplikacji w kodzie, co znacząco ułatwia testowanie wszystkich struktur. Wszystkie wyniki czasowe przedstawione są w **milisekundach**. Maszyna testowa wyposażona jest w procesor **Intel i7** z taktowaniem **2.4 GHz** oraz **8GB** pamięci RAM, a program uruchamiany był w wersji 64 bitowej.

Testy wykonywane są dla grafów o kolejnych ilościach krawędzi: **30, 60, 90, 120, 150**. Dodatkowo każda z wielkości została przetestowana dla czterech różnych gęstości krawędzi w grafie: **25%, 50%, 75%** i **99%**.

4. Wyniki eksperymentu

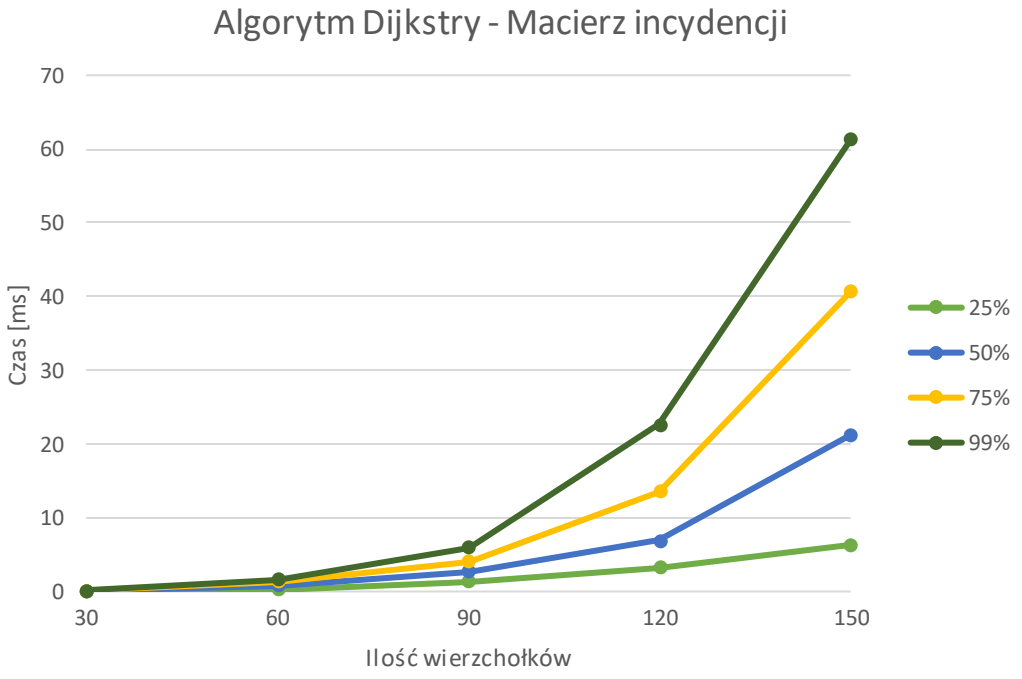
Na następnej stronie przedstawiono w tabelach oraz wykresach wyniki uzyskane podczas badań dla danego algorytmu w zależności od przechowywanej struktury, ilości wierzchołków oraz gęstości grafu.

4.1. **Algorytm Dijkstry**

a) Macierz incydencji

Macierz incydencji		
Liczba wierzchołków	Gęstość [%]	Czas [ms]
30	25%	0,04737
	50%	0,09360
	75%	0,13939
	99%	0,18373
60	25	0,32956
	50	0,80673
	75	1,34778
	99	1,59392
90	25	1,33572
	50	2,73184
	75	4,12710
	99	5,90622
120	25	3,26851
	50	6,92588
	75	13,56130
	99	22,61200
150	25	6,32510
	50	21,23150
	75	40,65250
	99	61,24950

Tabela 1: Wyniki czasowe algorytmu Dijkstry dla grafów zapisanych jako macierz incydencji

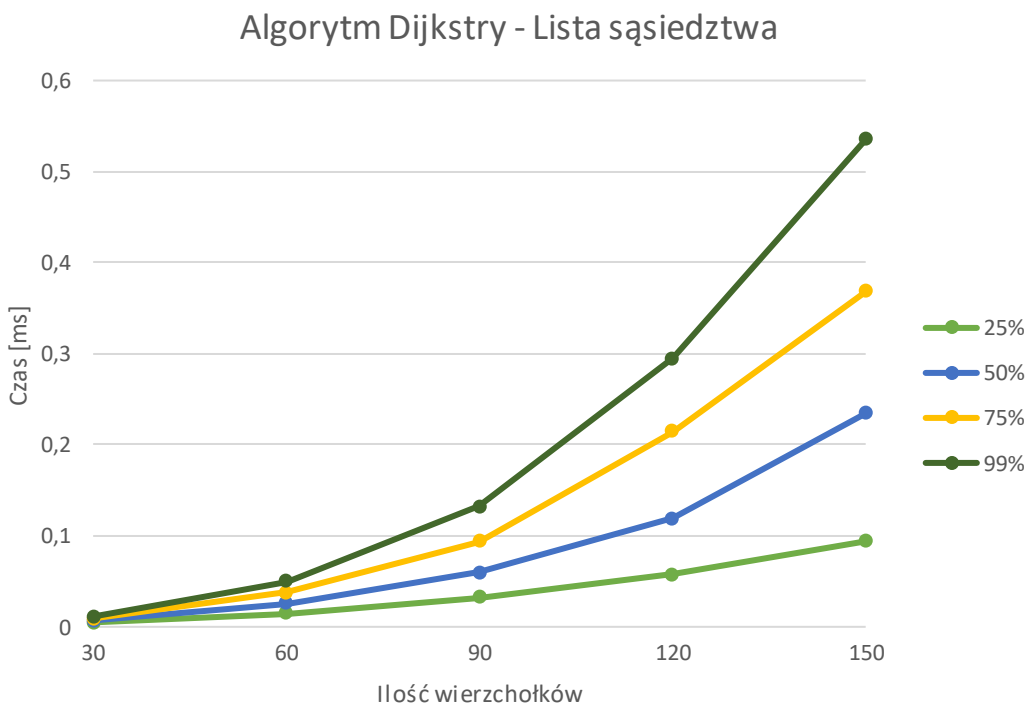


Wykres 1: Wyniki czasowe algorytmu Dijkstry na podstawie tabeli 1

b) Lista sąsiedztwa

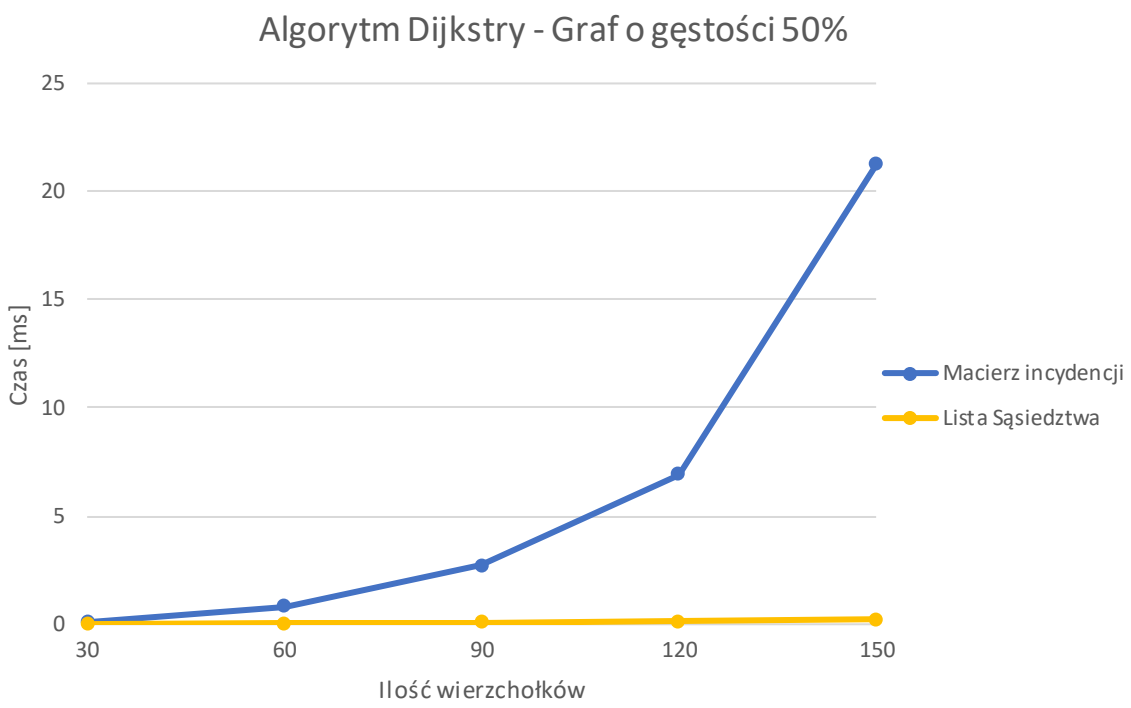
Lista Sąsiedztwa		
Liczba wierzchołków	Gęstość [%]	Czas [ms]
30	25	0,00488
	50	0,00704
	75	0,00922
	99	0,01129
60	25	0,01516
	50	0,02584
	75	0,03774
	99	0,04925
90	25	0,03261
	50	0,05954
	75	0,09319
	99	0,13199
120	25	0,05752
	50	0,11950
	75	0,21467
	99	0,29493
150	25	0,09388
	50	0,23434
	75	0,36800
	99	0,53513

Tabela 2: Wyniki czasowe algorytmu Dijkstry dla grafów zapisanych jako lista sąsiedztwa



Wykres 2: Wyniki czasowe algorytmu Dijkstry na podstawie tabeli 2

c) Porównanie wyników pomiędzy grafami zapisanymi jako Macierz oraz Lista dla grafu o gęstości krawędzi równej 50%



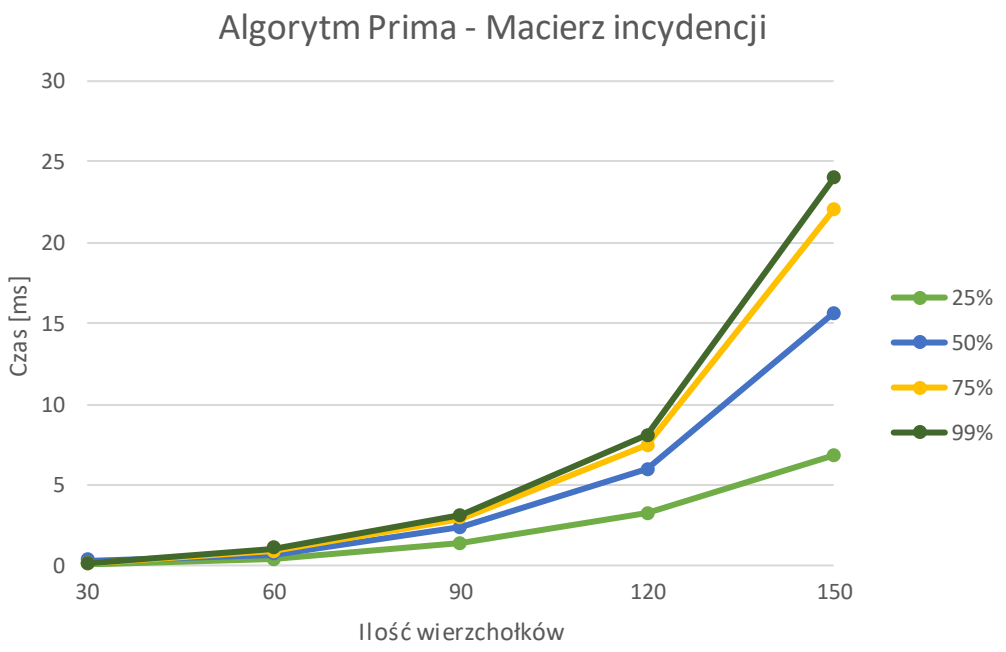
Wykres 3: Porównanie wyników algorytmu Dijkstry dla grafów o gęstości 50% przechowywanych jako Macierz incydencji oraz lista sąsiedztwa.

4.2. **Algorytm Prima**

a) Macierz incydencji

Macierz incydencji		
Liczba wierzchołków	Gęstość [%]	Czas [ms]
30	25	0,060422
	50	0,280138
	75	0,12247
	99	0,130356
60	25	0,37649
	50	0,700046
	75	0,837942
	99	1,04249
90	25	1,34567
	50	2,32139
	75	2,91049
	99	3,11217
120	25	3,20313
	50	5,94827
	75	7,40449
	99	8,02044
150	25	6,81643
	50	15,6339
	75	22,0373
	99	23,9782

Tabela 3: Wyniki czasowe algorytmu Prima dla grafów zapisanych jako macierz incydencji

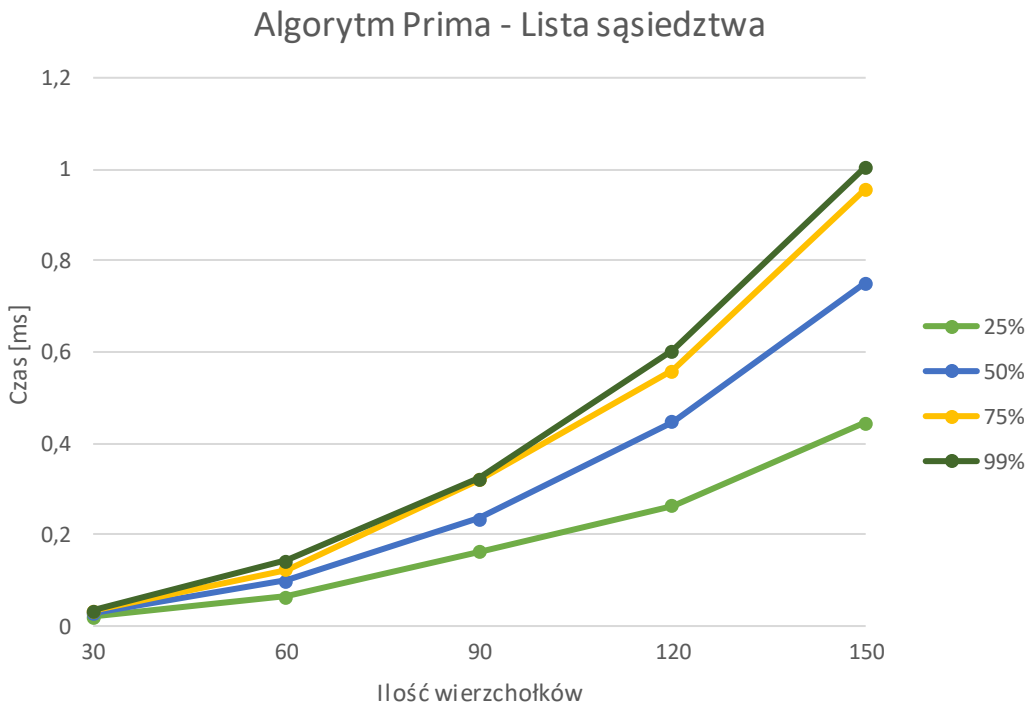


Wykres 4: Wyniki czasowe algorytmu Prima na podstawie tabeli 3

b) Lista sąsiedztwa

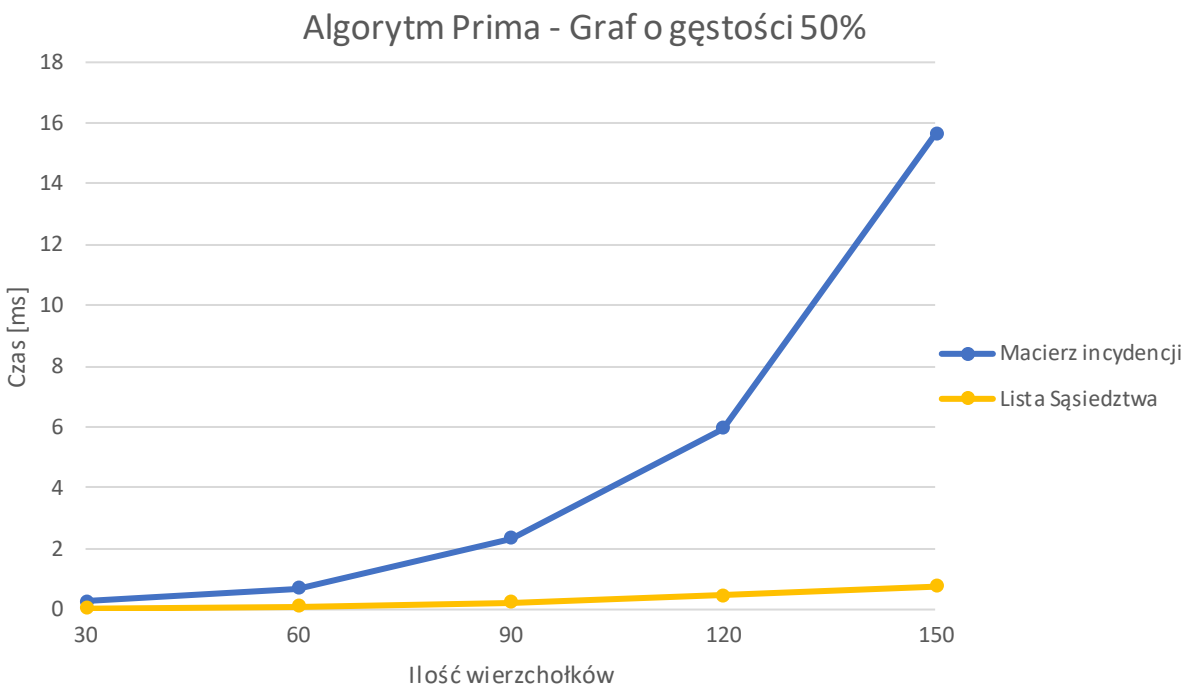
Lista Sąsiedztwa		
Liczba wierzchołków	Gęstość [%]	Czas [ms]
30	25	0,00488
	50	0,00704
	75	0,00922
	99	0,01129
60	25	0,01516
	50	0,02584
	75	0,03774
	99	0,04925
90	25	0,03261
	50	0,05954
	75	0,09319
	99	0,13199
120	25	0,05752
	50	0,11950
	75	0,21467
	99	0,29493
150	25	0,09388
	50	0,23434
	75	0,36800
	99	0,53513

Tabela 4: Wyniki czasowe algorytmu Prima dla grafów zapisanych jako lista sąsiedztwa



Wykres 5: Wyniki czasowe algorytmu Prima na podstawie tabeli 4

c) Porównanie wyników pomiędzy grafami zapisanymi jako Macierz oraz Lista dla grafu o gęstości krawędzi równej 50%



Wykres 6: Porównanie wyników algorytmu Prima dla grafów o gęstości 50% przechowywanych jako Macierz incydencji oraz lista sąsiedztwa.

5. Wnioski

Przeprowadzony eksperyment zgadza się mniej więcej z rzędami obliczeń dla badanych algorytmów. Zarówno w przypadku algorytmu Dijkstry do problemu najkrótszej ścieżki, jak i Prima dla MST otrzymane wyniki mniej więcej zgadzają się ze złożonościami z literatury. Warto jednak wziąć pod uwagę fakt, że podczas testowania mogły występować błędy pomiarowe, spowodowane niedokładnością pomiaru czasu, działaniu procesów systemowych w tle czy brak całkowitego zoptymalizowania napisanych algorytmów. Nie mniej wyniki testów są zadowalające, co wskazuje na przyzwoitą implementację algorytmów w programie.

Z obserwacji otrzymanych wyników można zauważyć dużą rozbieżność pomiędzy strukturą Macierzy incydencji a Listą sąsiedztwa, która wzrasta wraz ze zwiększającą się ilością danych (wykresy 3 i 6). Różnica ta wynika z faktu, że podczas przeglądania grafu w przypadku macierzy musimy przejrzeć kolejno wszystkie indeksy wierzchołków dla danej krawędzi (wykorzystując pętlę podwójną co znacząco wydłuża jego działanie), natomiast w przypadku listy mamy je podane od razu jako zmienne dla niej (wystarczy przejrzeć sąsiadów dla danego wierzchołka). Jest to również zgodne z literaturą, ponieważ Lista sąsiedztwa o wiele lepiej sprawdza się w przypadku odczytu danych ze struktury.