Projektowanie i Analiza Algorytmów	
Kierunek *Informatyczne Systemy Automatyki*	**Termin** *Wtorek* $15^{15} - 16^{55}$
Imię, nazwisko, numer albumu *Mikołaj Nowak 280082*	**Data** *17.06.2025*
Link do projektu `https://github.com/devoid-of-thought/Szachy`	

Sprawozdanie Projekt nr 2

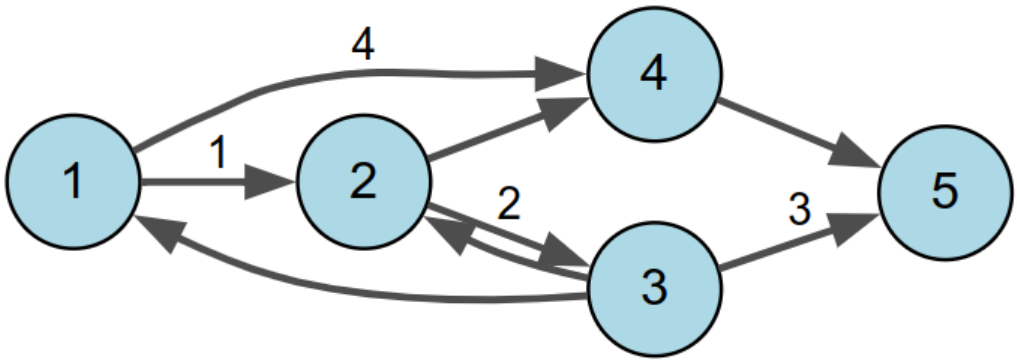
1 Wstęp

Celem niniejszego sprawozdania jest prze.

2 Opis algorytmów

2.1 Algorytm wyszukiwania wgłąb (DFS)

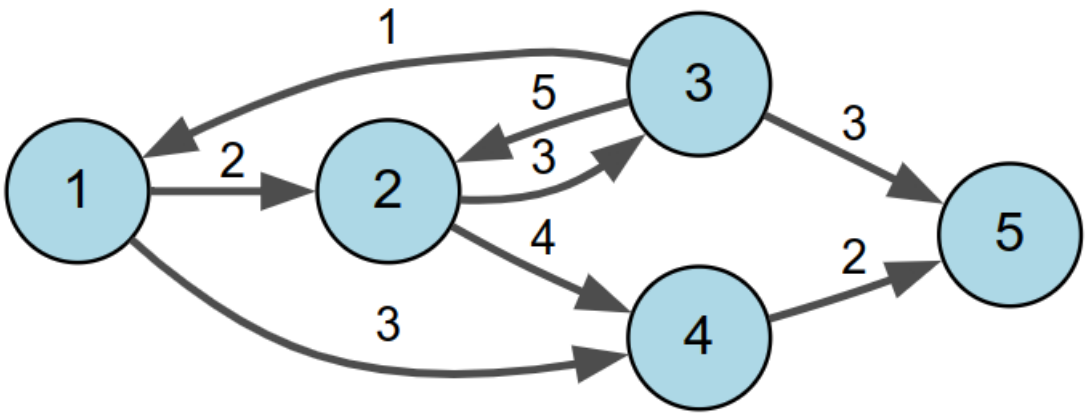
Zaimplementowany algorytm DFS zaczynając od pewnego punktu startowego będzie sprawdzał czy dany sąsiad znajduje się w tablicy połączeń i jeśli nie dodawał go do niej po czym rekurencyjnie odwoływał się do siebie. Teoretyczna złożoność obliczeniowa tego algorytmu to w najgorszym wypadku $O(V^2)$ ze względu na użycie macierzy sąsiedztwa, a dla listy sąsiedztwa $O(V + E)$.



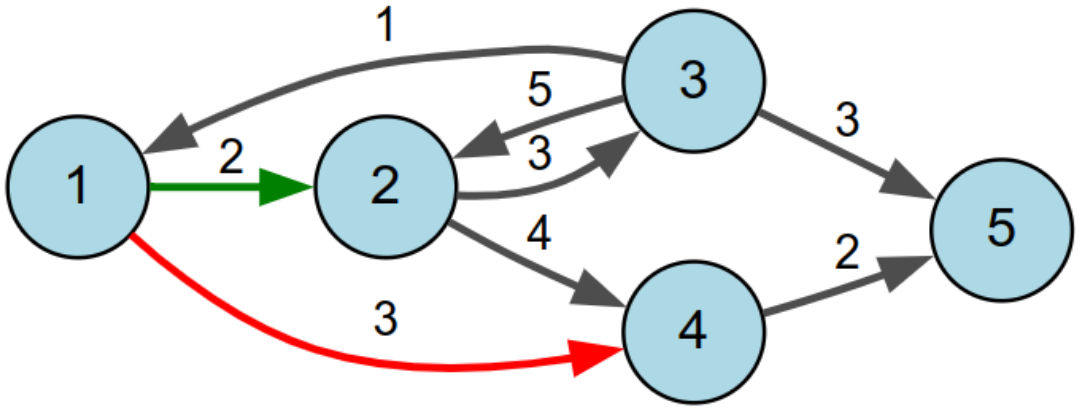
Rysunek 1: Wizualizacja algorytmu DFS na przykładowym grafie

2.2 Algorytm Dijkstry

Zaimplementowany algorytm Dijkstry polega na odwiedzeniu wierzchołków, do których odległość jest najmniejsza i nie zostały poprzednio odwiedzone poczynając od startu. Przy każdym odwiedzeniu wierzchołków mapujemy długość ścieżki do sąsiadów, a w przypadku gdy jest ona najmniejsza do startu zapisujemy w pamięci. Teoretyczna złożoność obliczeniowa dla tego algorytmu wynosily $O(V^2)$

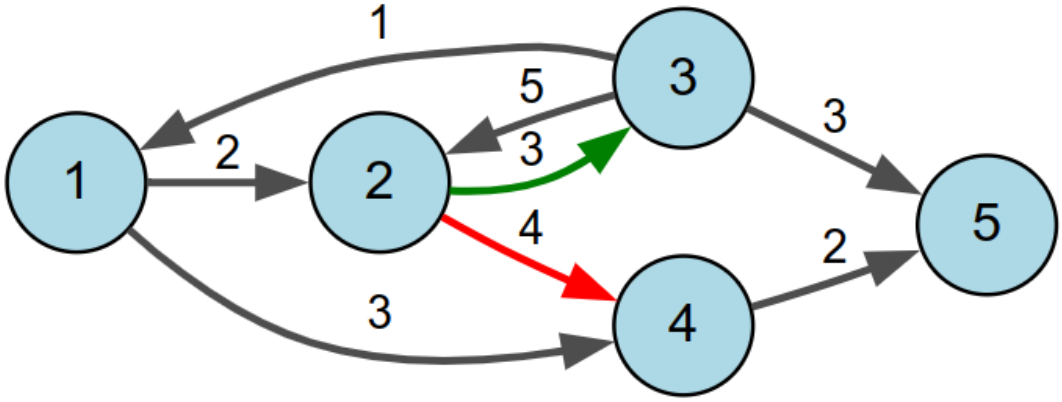


Rysunek 2: Przykładowy graf startowy. 1: 0, 2: INF, 3: INF, 4:INF, 5:INF



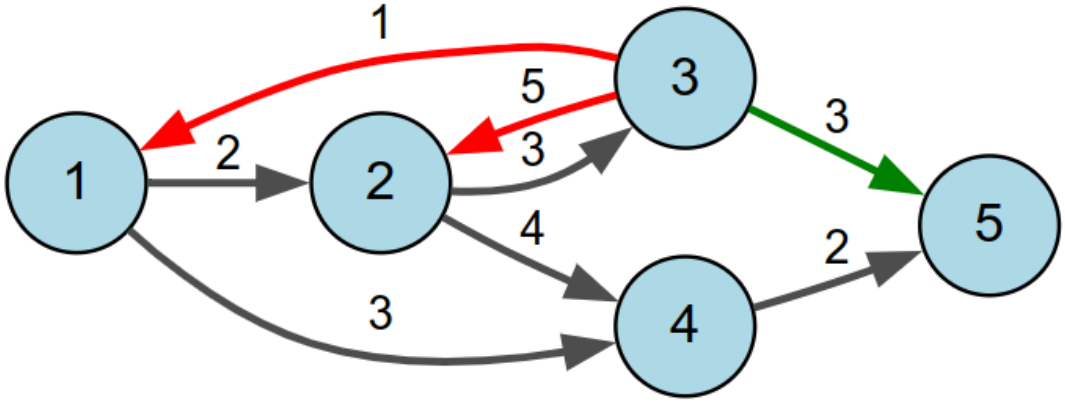
Rysunek 3: Pierwszy krok zapisanie w pamięci najkrótszą drogę do sąsiadów 1 i przejście do najbliższego wierchołka.

(a) 1: 0, 2: 2, 3: INF, 4:3, 5:INF



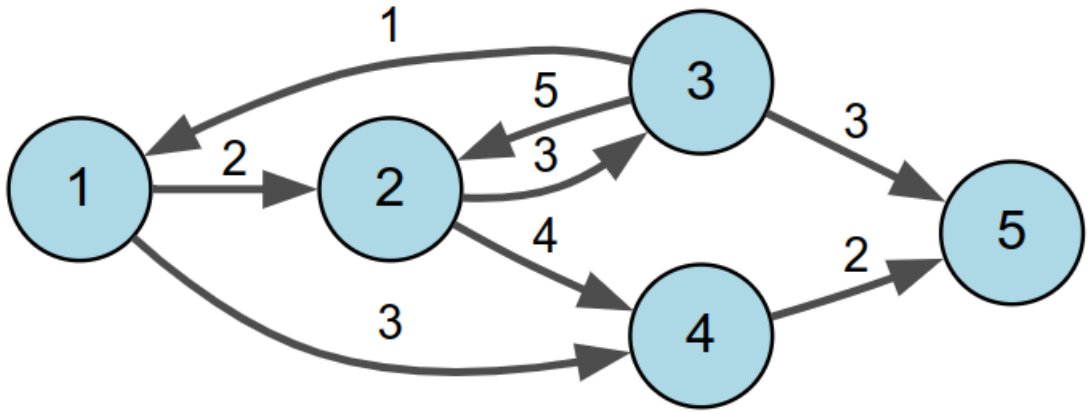
Rysunek 4: Drugi krok zapisanie w pamięci najkrótszą drogę od sąsiadów 2 od 1 i przejście z 2 do najbliższego wierchołka.

(a) 1: 0, 2: 2, 3: 5, 4:3, 5:INF



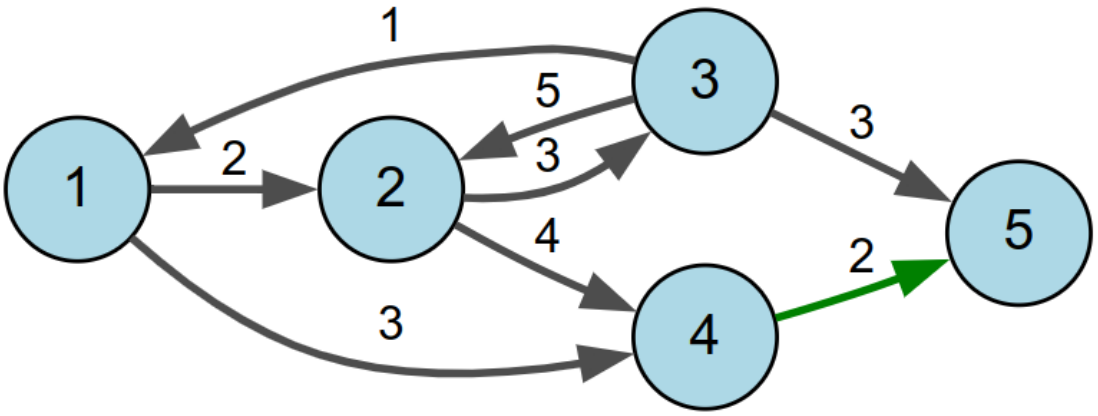
Rysunek 5: Trzeci krok zapisanie w pamięci najkrótszą drogę od sąsiadów 3 od 1 i przejście z 3 do najbliższego wierchołka.

(a) 1: 0, 2: 2, 3: 5, 4:3, 5:8



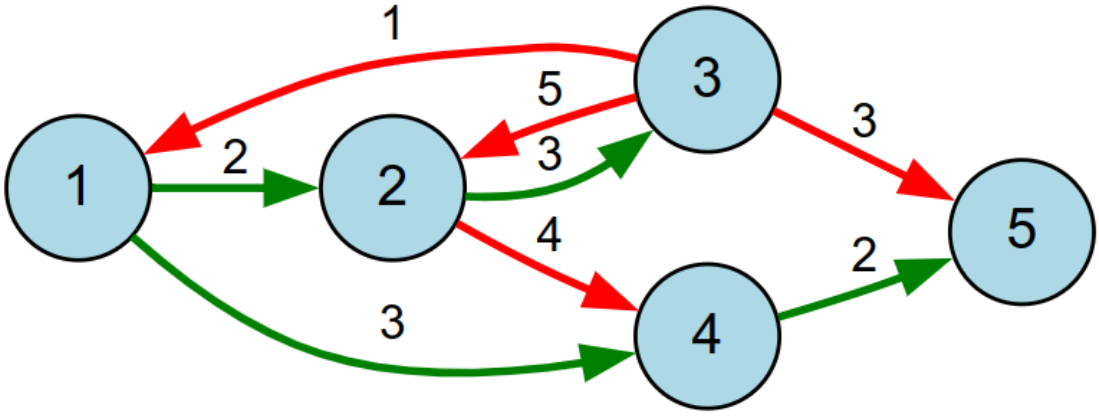
Rysunek 6: Czwarty krok zapisanie w pamięci najkrótszą drogę od sąsiadów 5 od 1 i przejście do kolejnego wierzchołka

(a) 1: 0, 2: 2, 3: 5, 4:3, 5:8



Rysunek 7: Piąty krok powrót do ostatniego nieodwiedzonego wierzchołka zapisanie w pamięci najkrótszej drogi od sąsiadów 4 do 1.

(a) 1: 0, 2: 2, 3: 5, 4:3, 5:5

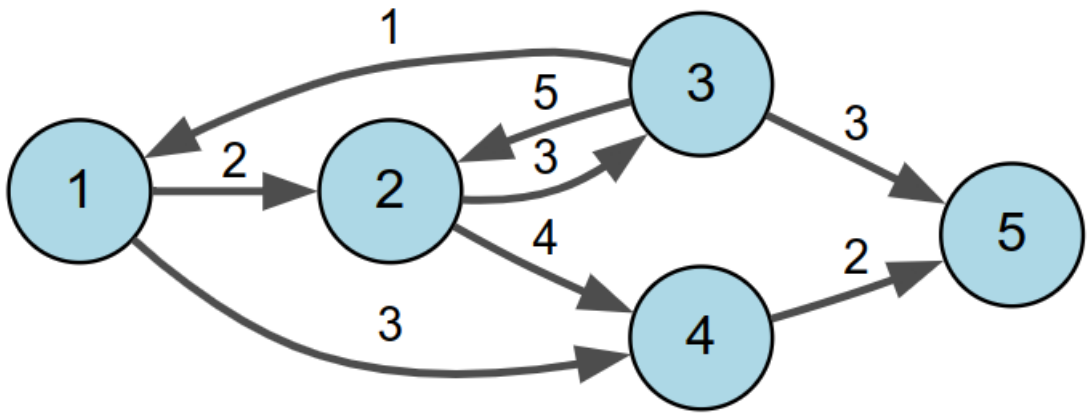


Rysunek 8: Koniec algorytmu na zielono najkrótsza trasa do każdego z wierzchołków.

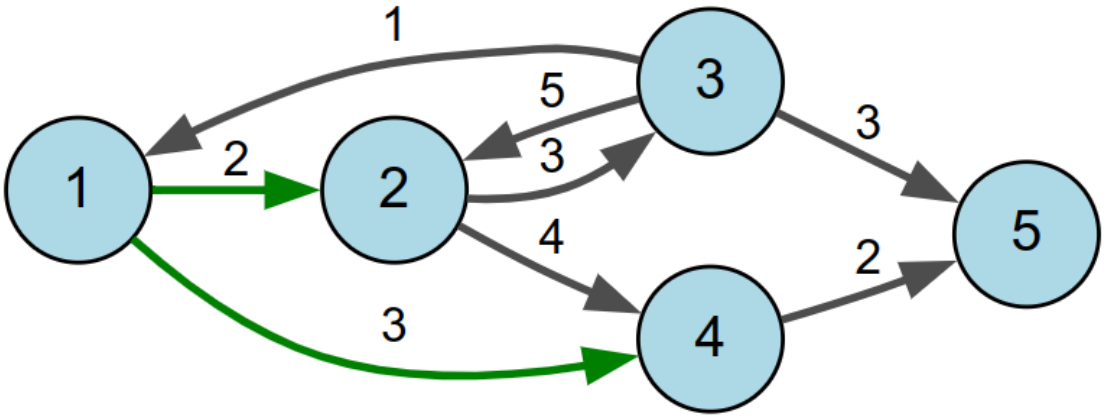
(a) 1: 0, 2: 2, 3: 5, 4:3, 5:5

2.3 Algorytm Bellmana-Forda

Zaimplementowany algorytm Bellmana-Forda polega na wielokrotnym przeglądaniu wszystkich możliwych par w grafie, za każdą iteracją zwracając uwagę na możliwe skrócenie ścieżki pomiędzy dwoma wierzchołkami. Złożonośc obliczeniowa tego algorytmu to $O(V^3)$ dla macierzy sąsiedztwa, a dla listy sąsiedztwa $O(V * E)$.

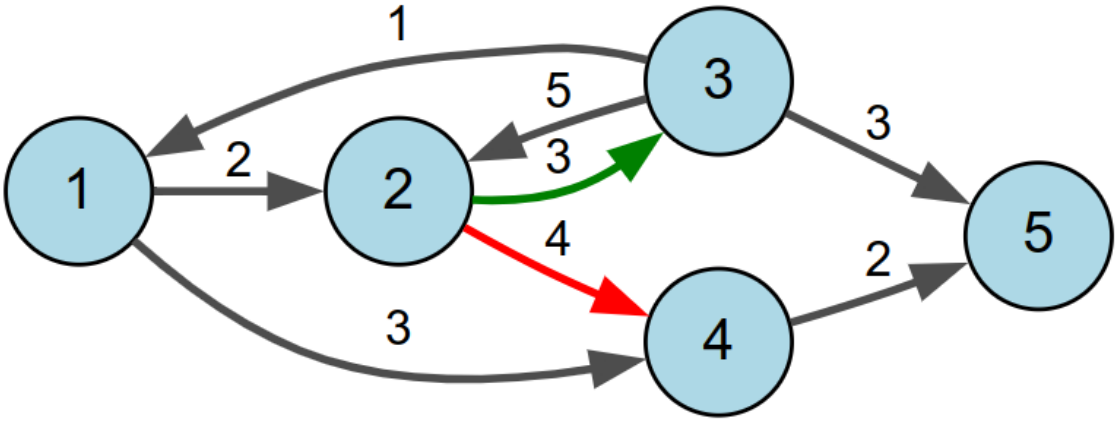


Rysunek 9: Przykładowy graf startowy. 1: 0, 2: INF, 3: INF, 4:INF, 5:INF



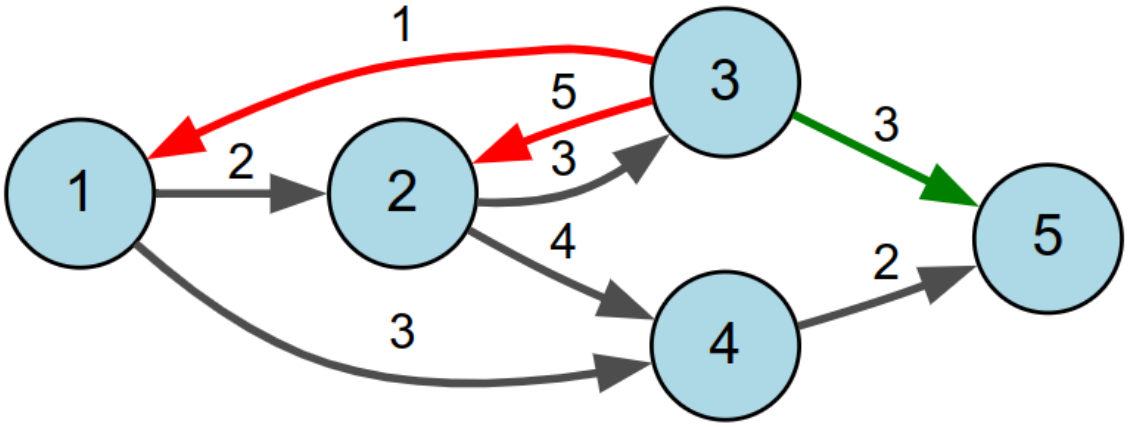
Rysunek 10: Pierwszy krok zapisanie w pamięci najkrótszą drogę do sąsiadów 1 i przejście do następnego w kolejności wierchołka.

(a) 1: 0, 2: 2, 3: INF, 4:3, 5:INF



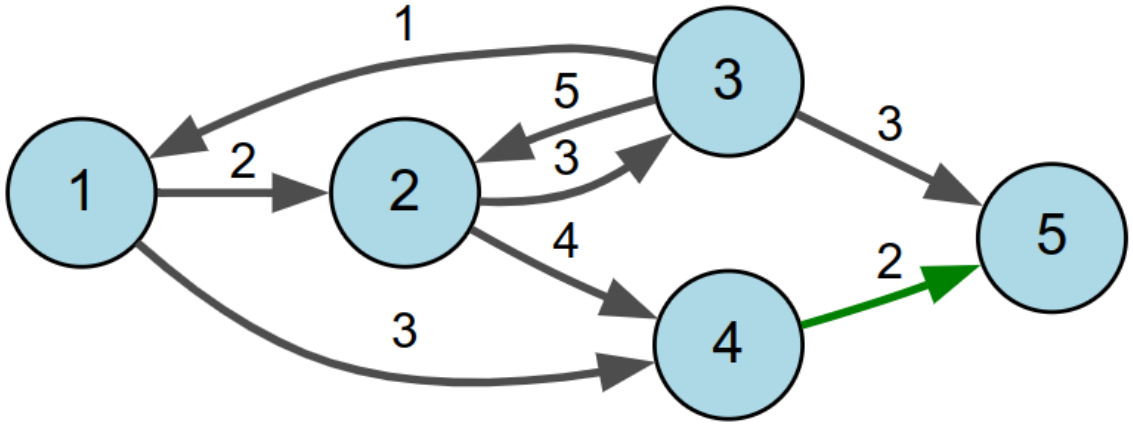
Rysunek 11: Drugi krok zapisanie w pamięci najkrótszą drogę do sąsiadów 2 od 1 i przejście do następnego w kolejności wierchołka.

(a) 1: 0, 2: 2, 3: 5, 4:3, 5:INF



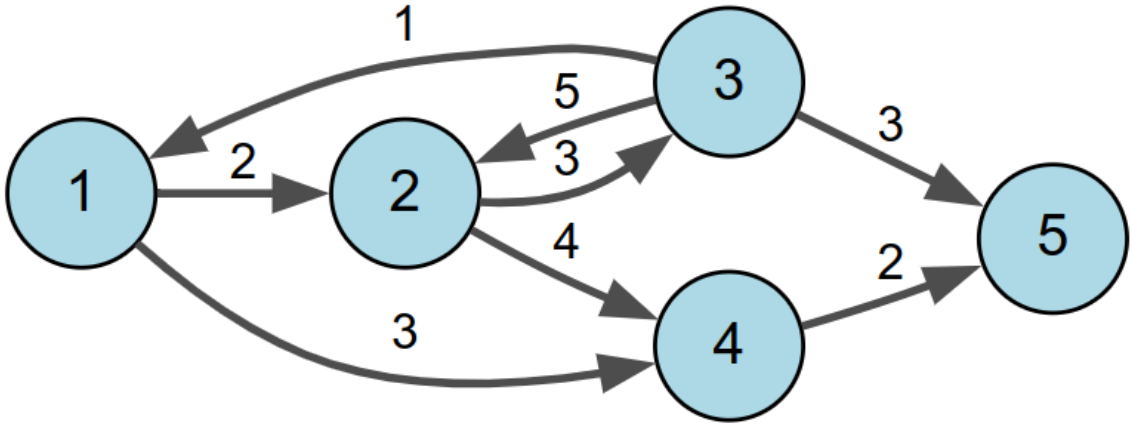
Rysunek 12: Trzeci krok zapisanie w pamięci najkrótszą drogę do sąsiadów 3 od 1 i przejście do następnego w kolejności wierchołka

(a) 1: 0, 2: 2, 3: 5, 4:3, 5:8



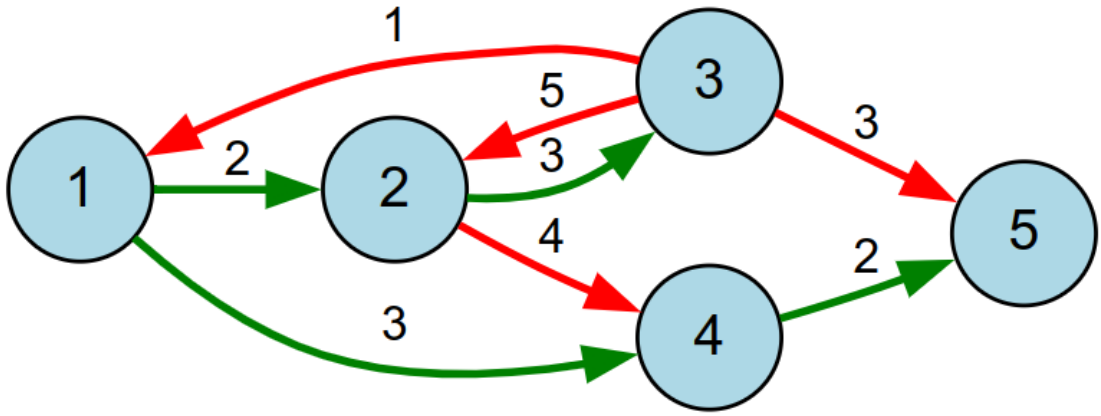
Rysunek 13: Czwarty krok zapisanie w pamięci najkrótszą drogę do sąsiadów 4 od 1 i przejście do następnego w kolejności wierchołka

(a) 1: 0, 2: 2, 3: 5, 4:3, 5:5



Rysunek 14: Piąty krok zapisanie w pamięci najkrótszą drogę do sąsiadów 5 od 1 i przejście do następnego w kolejności wierzchołka jakim jest 1. każdy kolejny krok to sprawdzenie kolejnych sąsiadów i ich ścieżek, przez ilość wierzchołków - 1 razy, ale nie zmieni on wyniku w tym przypadku.

(a) 1: 0, 2: 2, 3: 5, 4:3, 5:5

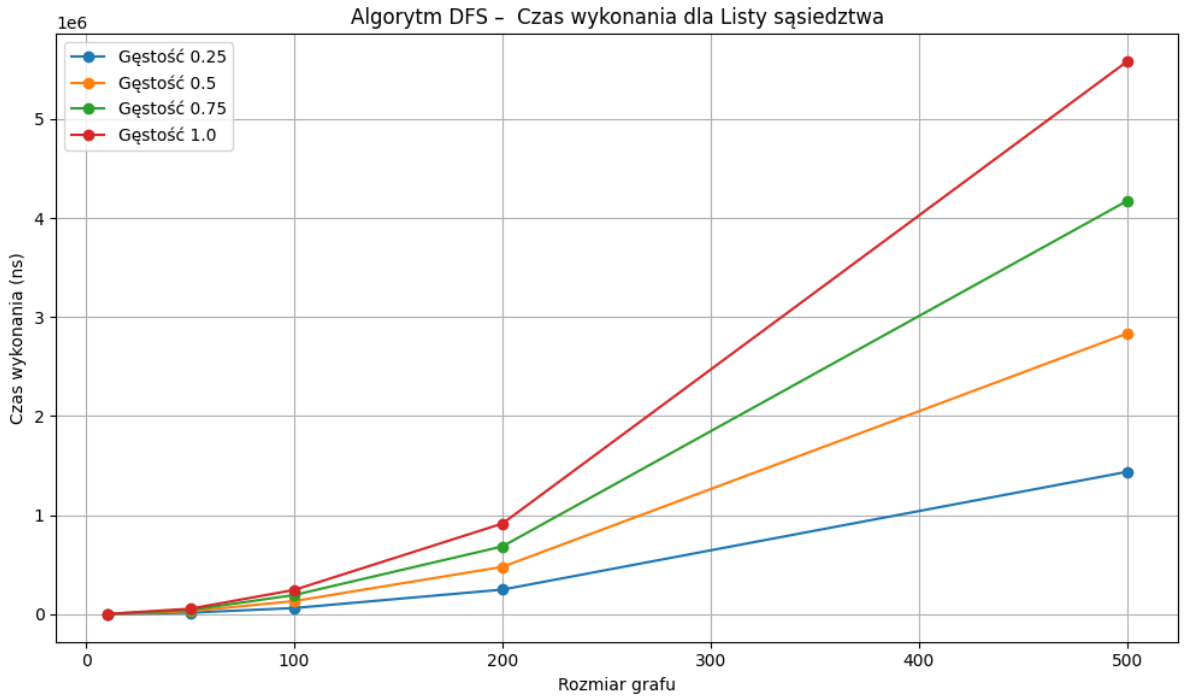


Rysunek 15: Koniec algorytmu na zielono najkrótsza trasa do każdego z wierzchołków.

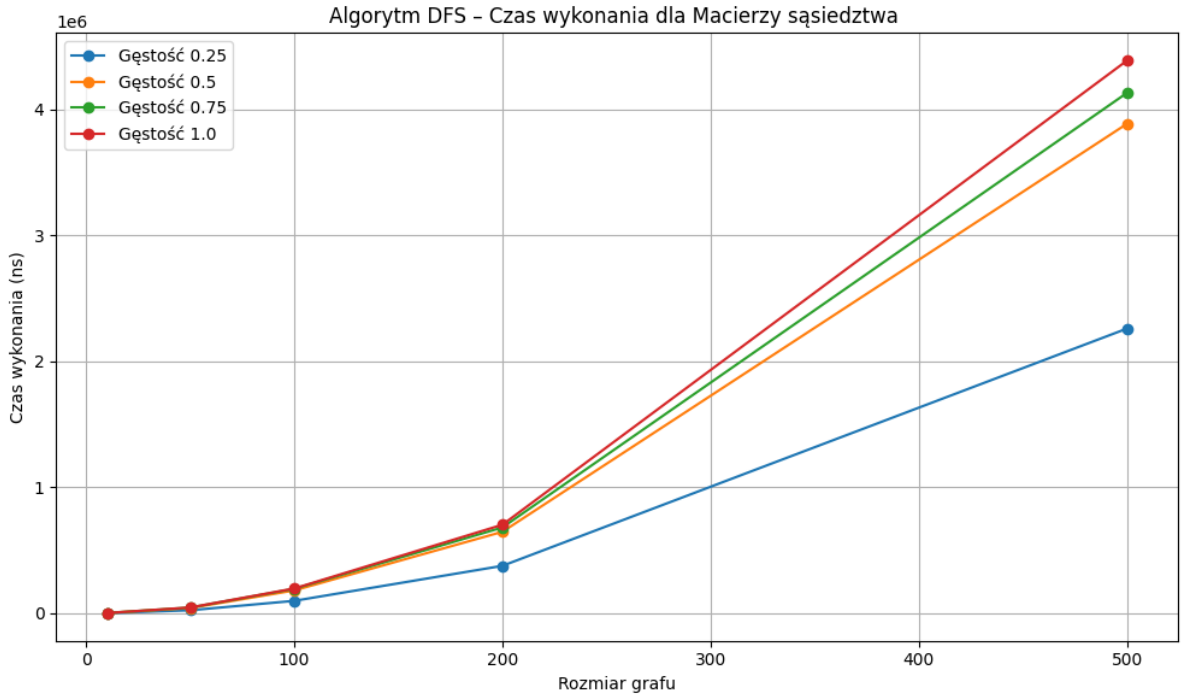
(a) 1: 0, 2: 2, 3: 5, 4:3, 5:5

3 Analiza i weryfikacja algorytmów

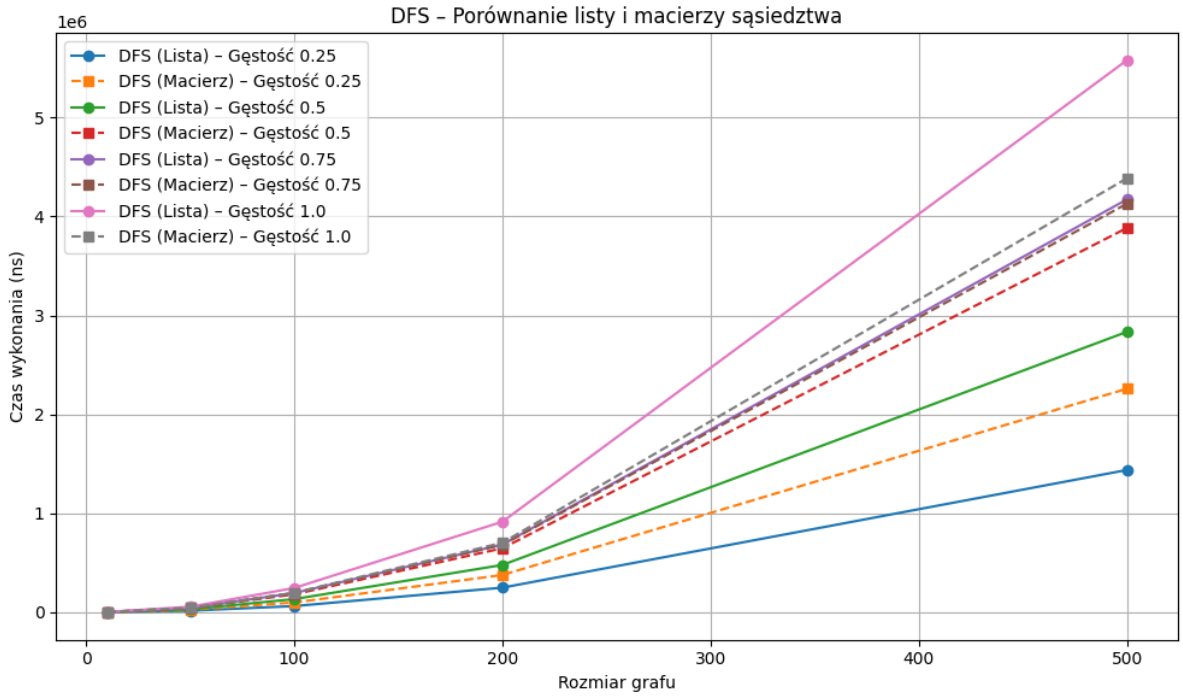
3.1 Algorytm DFS



Rysunek 16: Czas wykonania algorytmu DFS dla Listy sąsiedztwa

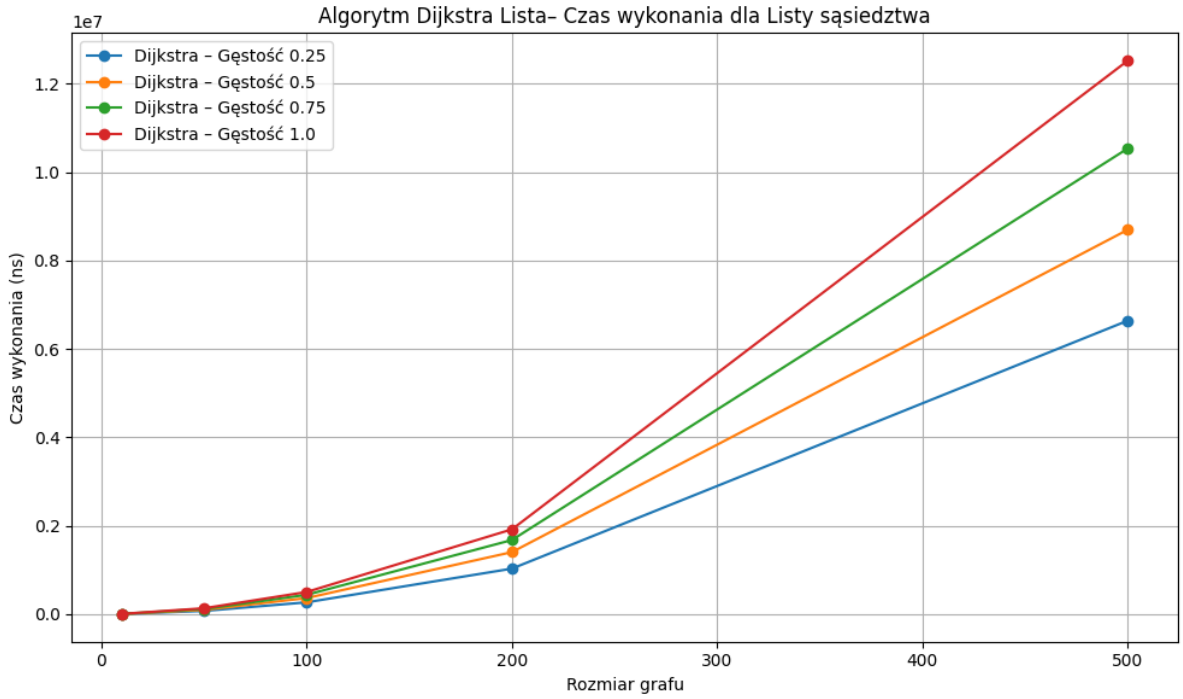


Rysunek 17: Czas wykonania algorytmu DFS dla Macierzy Sąsiedztwa

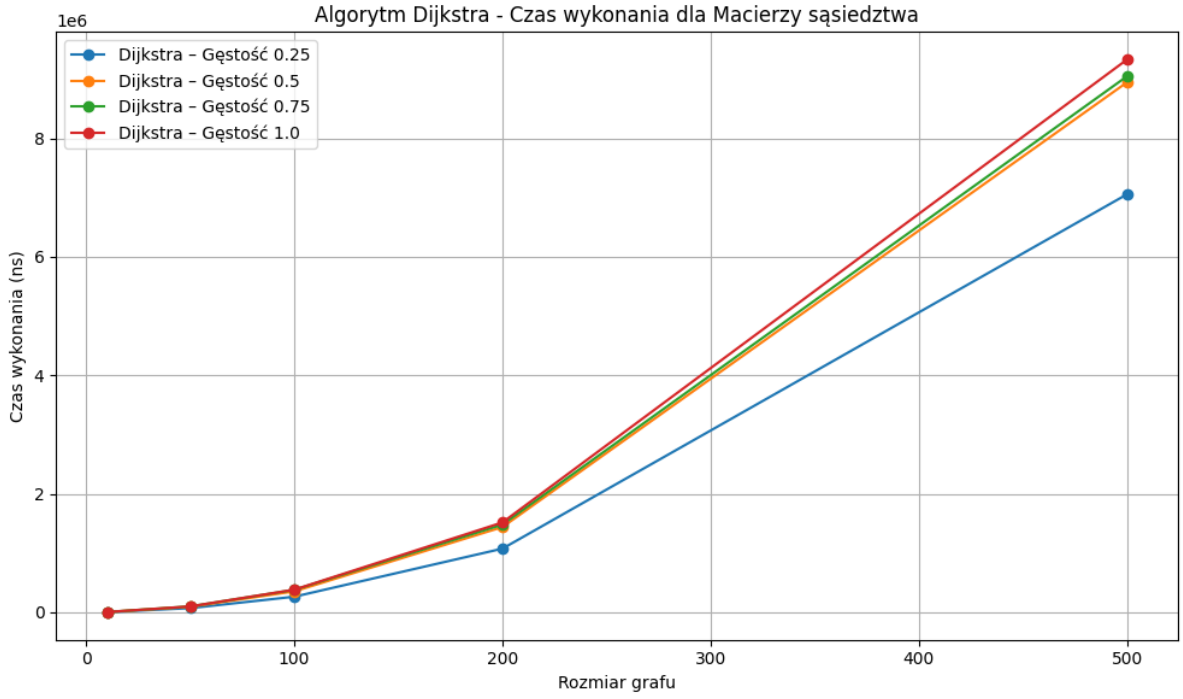


Rysunek 18: Porównanie implementacji DFS

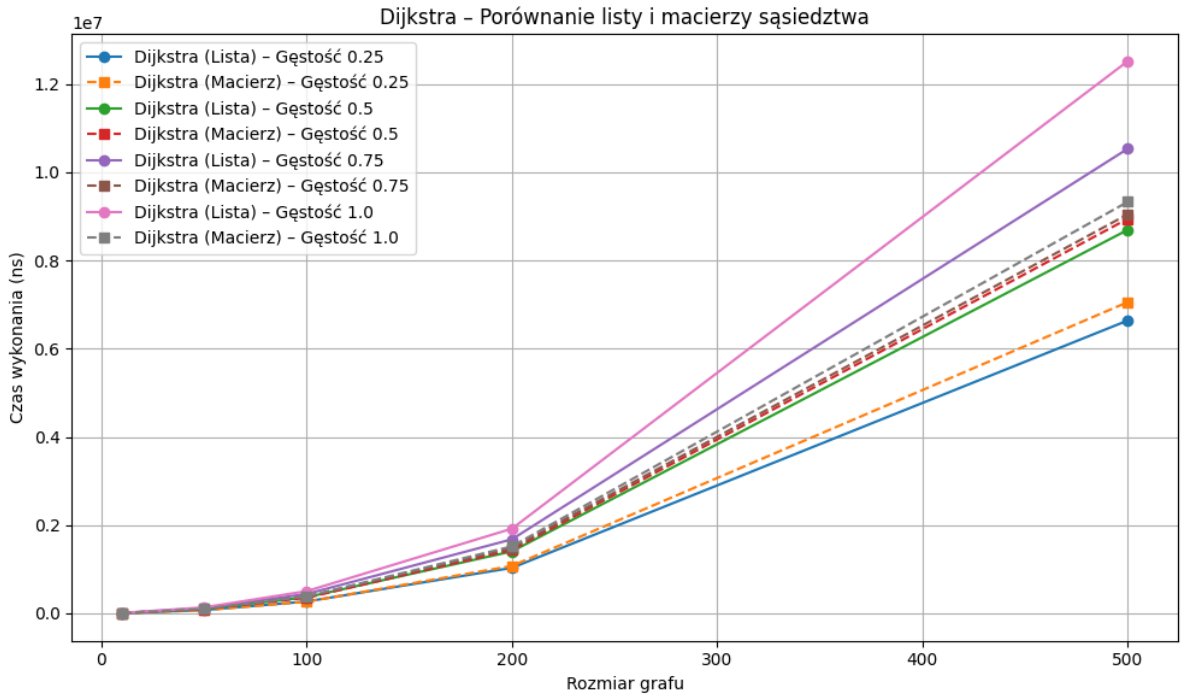
3.2 Algorytm Dijkstry



Rysunek 19: Czas wykonania algorytmu Dijkstry dla Listy Sąsiedztwa

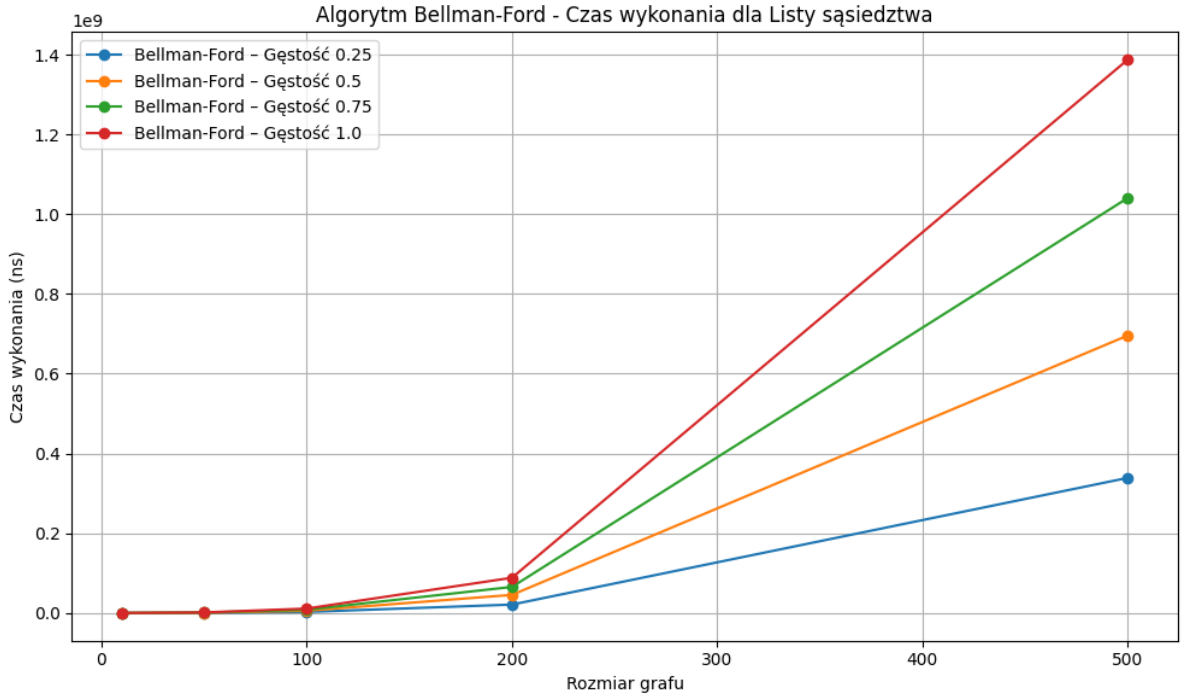


Rysunek 20: Czas wykonania algorytmu Dijkstry dla Macierzy sąsiedztwa

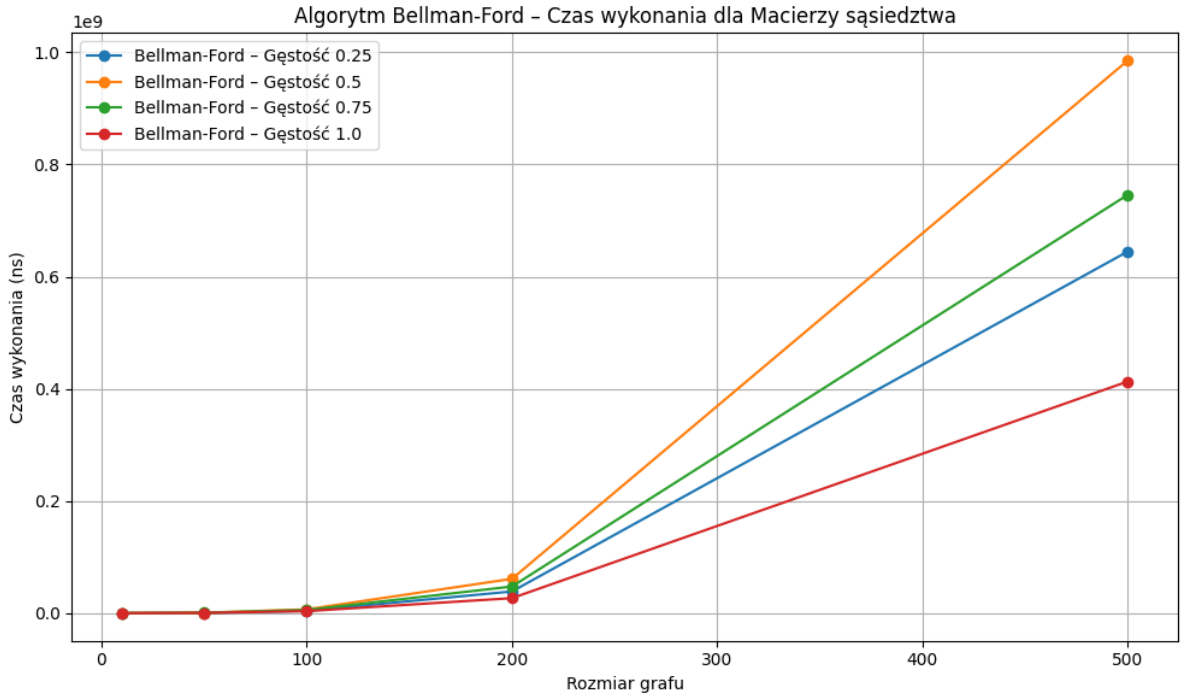


Rysunek 21: Porównianie implementacji dla Dijkstry

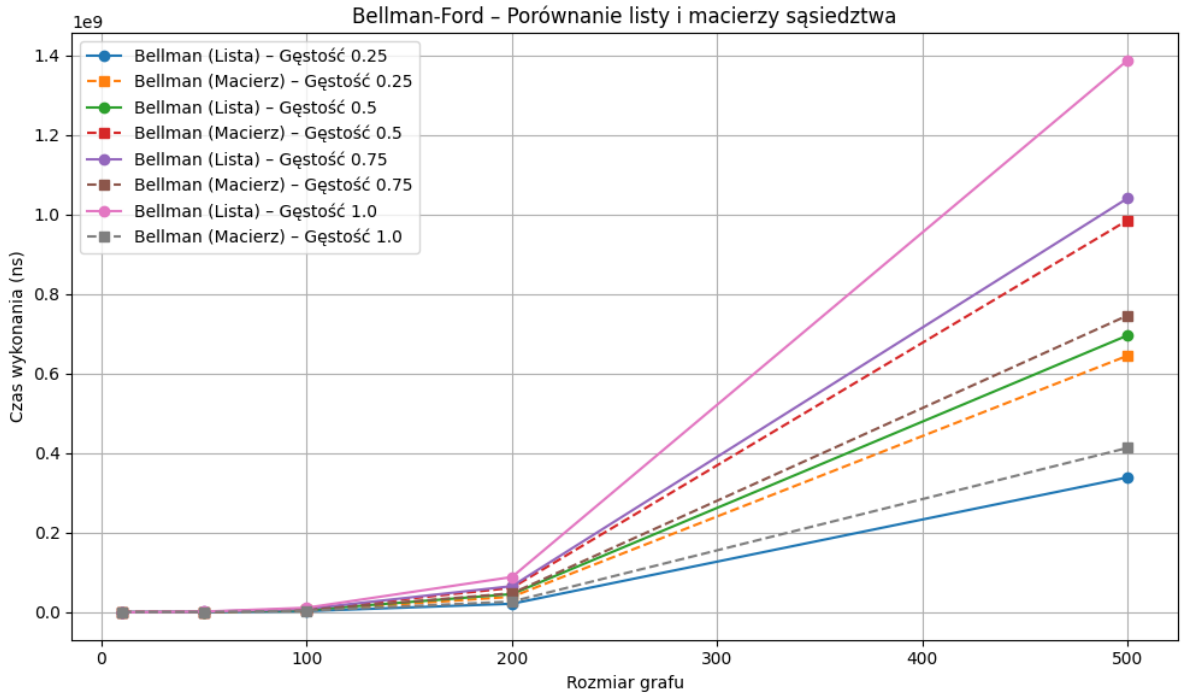
3.3 Algorytm Bellmana-Forda



Rysunek 22: Czas wykonania algorytmu Bellmana-Forda dla Listy Sąsiedztwa

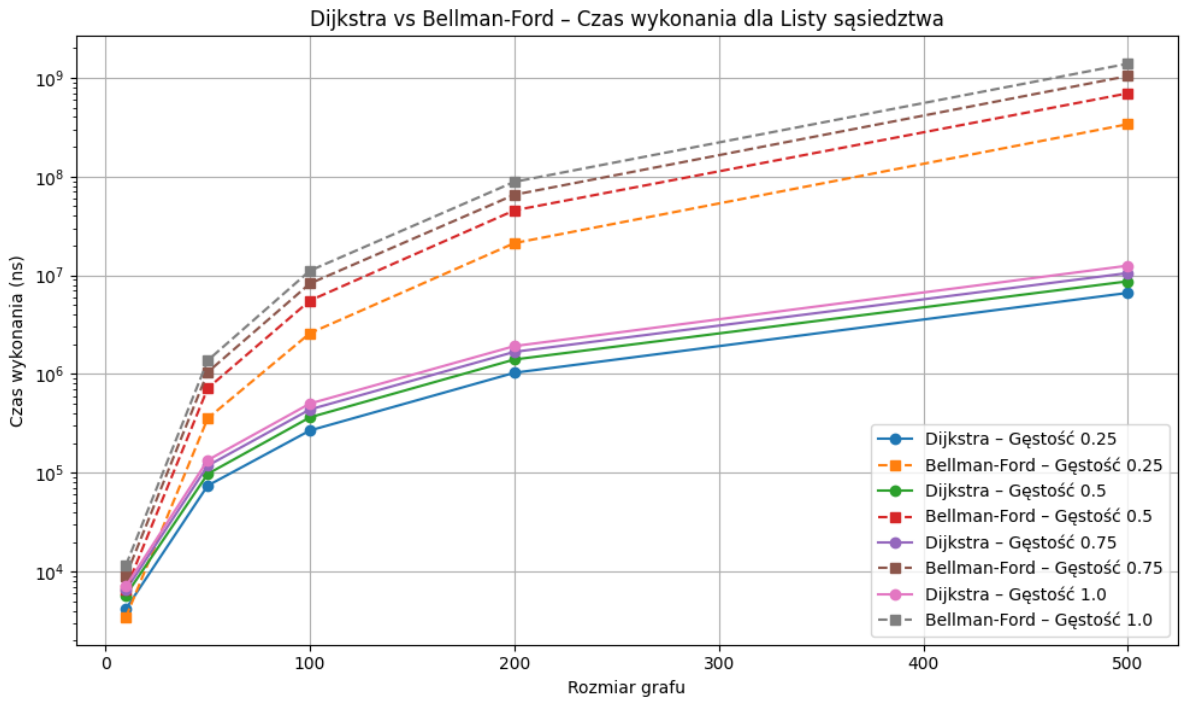


Rysunek 23: Czas wykonania algorytmu Bellmana-Forda dla Macierzy sąsiedztwa

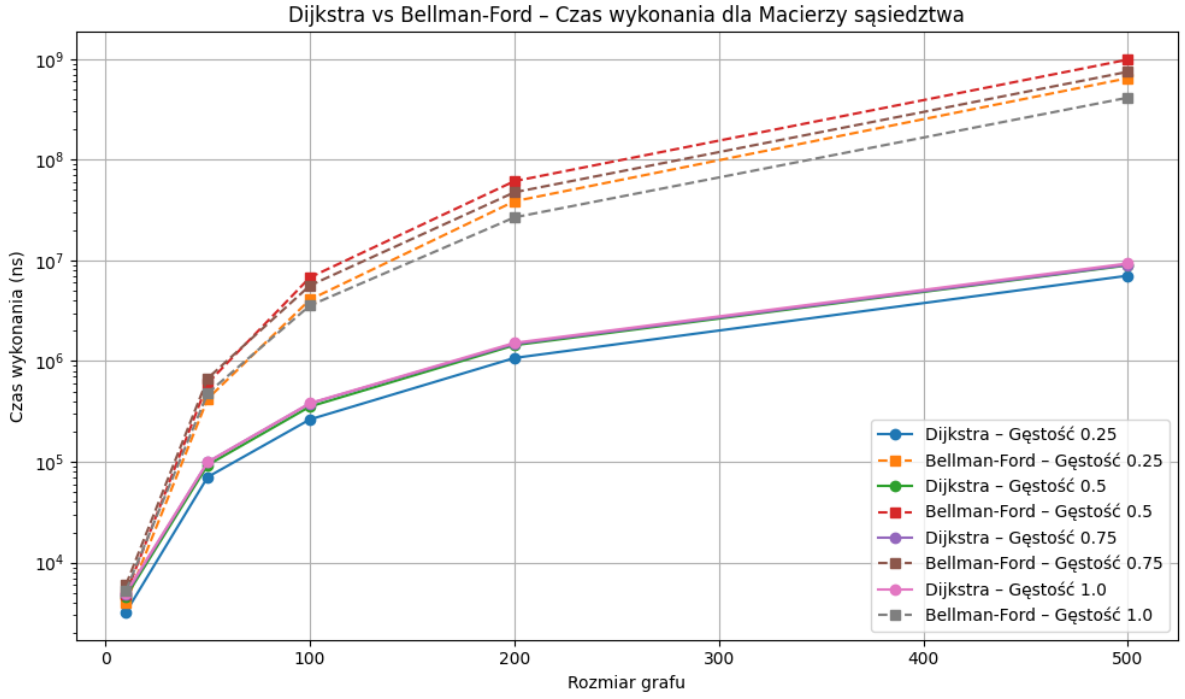


Rysunek 24: Porównianie implementacji dla Bellmana-Forda

3.4 Porównanie Dijkstra i Bellman-Ford

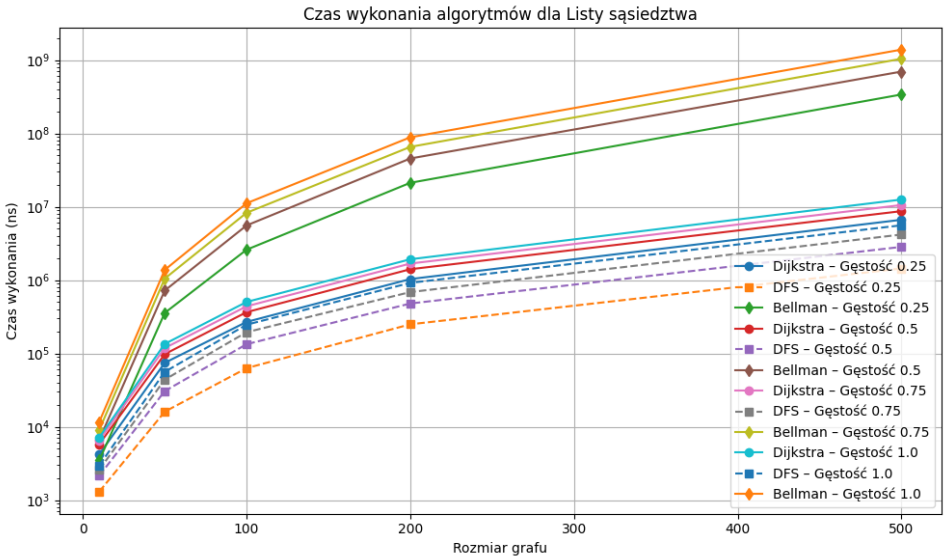


Rysunek 25: Porównanie algorytmów Djikstry i Bellmana-Forda dla Listy Sąsiedztwa



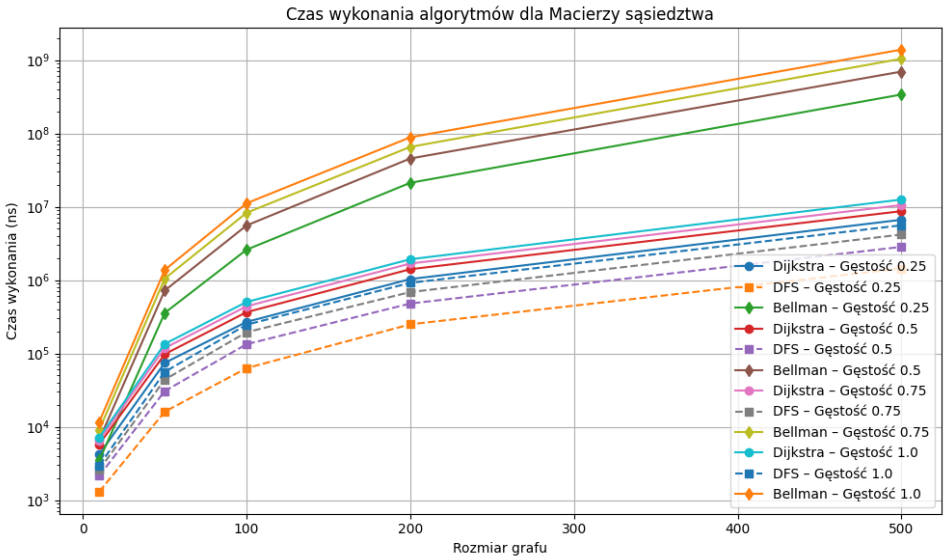
Rysunek 26: Porównanie algorytmów Djikstry i Bellmana-Forda dla Macierzy sąsiedztwa

3.5 Porównanie wszystkich algorytmów dla Listy sąsiedztwa



Rysunek 27: Porównanie wszystkich algorytmów dla Listy sąsiedztwa

3.6 Porównanie wszystkich algorytmów dla Macierz sąsiedztwa



Rysunek 28: Porównianie wszystkich algorytmów dla Macierz sąsiedztwa

4 Wnioski

- Dijkstra jest szybszym algorytmem do szukania ścieżki od Bellmana-Forda tak jak wskazywałaby na to teoretyczna złożoność obliczeniowa. $O(V^2)$ Dikstry kontra $O(V^3)$ Bellmana-Forda. Jego wada jest jednak fakt iż nie obsługuje ujemnych wag.
- Dikstra i Dfs mają zbliżone złożoności teoretyczne oba $O(V^2)$
- Gęstość grafu ma znaczący wpływ na czas wykonywania algorytmów