STRUKTURY DANYCH I ZŁOŻONOŚĆ OBLICZENIOWA	Badanie efektywności algorytmów grafowych	
Szymon Leja Numer indeksu: 259047	Poniedziałek 13.15 – 15.00 TN	Dr inż. Zbigniew Buchalski

1. Cel ćwiczenia:

Celem projektu jest zaimplementowanie oraz dokonanie pomiaru czasu działania wybranych algorytmów grafowych rozwiązujących następujące problemy:

- Wyznaczanie minimalnego drzewa rozpinającego (MST) - algorytm Prima oraz algorytm Kruskala
- Wyznaczanie najkrótszej ścieżki w grafie - algorytm Dijkstry oraz algorytm Bellmana - Forda

Algorytmy te należało zaimplementować dla obu poniższych reprezentacji grafu w pamięci komputera:

- reprezentacja macierzowa (macierz incydencji)
- reprezentacja listowa (lista następników/poprzedników)

2. Wstęp teoretyczny

Reprezentacja grafu to sposób zapisu grafu umożliwiający jego obróbkę z użyciem programów komputerowych.

Lista sąsiedztwa

Reprezentacja grafów przez listy sąsiedztwa, jak sama nazwa wskazuje, polega na trzymaniu dla każdego wierzchołka listy jego wszystkich sąsiadów (następników albo poprzedników) oraz ich wag krawędzi.

- złożoność pamięciowa: $O(E)$
- przejrzenie wszystkich krawędzi: $O(E)$
- przejrzenie następników/poprzedników danego wierzchołka: maksymalnie $O(V)$ (tyle sąsiadów może mieć wierzchołek), ale średnio $O(E/V)$ (trzeba przejrzeć całą listę sąsiadów, których średnio wierzchołek ma właśnie E/V)
- sprawdzenie istnienia jednej krawędzi: tak jak przedstawiono wyżej w przypadku przeglądania następników/poprzedników (trzeba przejrzeć listę dla pewnego wierzchołka)

Jest to jedna z lepszych reprezentacji - minimalna możliwa złożoność pamięciowa, szybkie przeszukiwanie krawędzi wychodzących z danego wierzchołka, stosunkowo łatwa implementacja. Wadą jest jedynie długi

czas sprawdzenia istnienia pojedynczej krawędzi. Można go poprawić do $O(\log V)$ wprowadzając drzewa BST (albo nawet drzewa zrównoważone) zamiast list, ale to skomplikuje implementację.

Macierz incydencji

Macierz incydencji składa się z V wierszy (odpowiadającym wierzchołkom) i E kolumn (odpowiadającym krawędziom). Na „skrzyżowaniu" wierzchołka z krawędzią jest -1 gdy krawędź wychodzi z wierzchołka, +1 - krawędź wchodzi, 2 - pętla, 0 - brak incydencji. W przypadku, gdy uwzględniamy wagi krawędzi, na „skrzyżowaniu" wierzchołka z krawędzią jest uwzględniona waga tej krawędzi, która wchodzi lub wychodzi z wierzchołka.

- złożoność pamięciowa: $O(V * E)$
- przejrzenie wszystkich krawędzi: $O(E)$
- przejrzenie następników/poprzedników danego wierzchołka: $O(E)$
- sprawdzenie istnienia jednej krawędzi: $O(E)$

Reprezentacja nie wygląda na dobrą ze względu na dużą złożoność pamięciową.

Minimalne drzewo rozpinające

Drzewo rozpinające to takie, które zawiera wszystkie wierzchołki grafu oraz niektóre z jego krawędzi. Należy pamiętać, że drzewa nie zawierają cykli. Natomiast minimalne drzewo rozpinające jest drzewem rozpinającym, którego suma wag krawędzi jest najmniejsza ze wszystkich pozostałych drzew rozpinających danego grafu. W grafie może istnieć kilka drzew o tych własnościach.

Algorytm Prima

Dla każdego wierzchołka określamy, że jego koszt wynosi nieskończoność, a poprzednik jest nieokreślony. Wybranemu wierzchołkowi przypisujemy wartość 0, a następnie tworzymy kolejkę wierzchołków do rozpatrzenia. W kolejce dane są posortowane po koszcie elementu.
Następnie, dopóki kolejka nie jest pusta, wybieramy wierzchołek o najniższym koszcie i aktualizujemy jego sąsiadów, którzy występują w kolejce. Jeśli krawędź z wybranego wierzchołka z kolejki do sąsiada ma niższą wartość niż sąsiad ma koszt to aktualizujemy informacje. Zastosowanie algorytmu:

- Projektowanie sieci komputerowych
- Projektowanie sieci rurociągów wody pitnej lub gazu ziemnego.
- Umieszczenie wież mikrofalowych lub podobnych projektówZłożoność

obliczeniowa: $O(|V|^2)$, paięciowa: $O(|V|)$.

Założenia eksperymentu:

- Testujemy algorytm dla dwóch reprezentacji grafu. Graf zawsze się generuje w postacimacierzowej. W przypadku testowania listy sąsiedzictwa, graf przekształcamy w tą formę po zgenerowaniu.
- Nie liczymy czas generacji oraz przepisywanie grafu z jednej postaci do drugiej. Genracja grafu prepływa s następujący sposób: generujemy drzewo rozpinające, dalej wzależności od gęstości dodajemy krawędzie.
- Testujemy 7 punktów obserwowanych: 50, 60, 70, 80, 90, 100, 110 wierzchołków dlatrzech gęstości: 20%, 60%, 99%.
- Mierzymy czas 100 prób, zatem uśredniamy wynik.

Algorytm Kruskala

Algorytm Kruskala do znalezienia minimalnego drzewa rozpinającego wykorzystuje podejściechciwości. Algorytm ten traktuje graf jako las, a każdy jego węzeł jako pojedyncze drzewo.
Drzewo łączy się z innym tylko i tylko wtedy, gdy, ma najmniejszy koszt spośród wszystkichdostępnych opcji i nie narusza właściwości MDR.

Złożoność obliczeniowa: $O(E \log V)$, pamęciowa: $O(|V|)$.

Założenia eksperymentu:

- Testujemy algorytm dla dwóch reprezentacji grafu. Graf zawsze się generuje w postacimacierzowej. W przypadku testowania listy sąsiedzictwa, graf przekształcamy w tą formę po zgenerowaniu.
- Nie liczymy czas generacji oraz przepisywanie grafu z jednej postaci do drugiej. Genracja grafu prepływa s następujący sposób: generujemy drzewo rozpinające, dalej wzależności od gęstości dodajemy krawędzie.
- Testujemy 7 punktów obserwowanych: 50, 60, 70, 80, 90, 100, 110 wierzchołków dlatrzech gęstości: 20%, 60%, 99%.
- Mierzymy czas 100 prób, zatem uśredniamy wynik.

Najkrótsza ścieżka w grafie

Problem najkrótszej ścieżki to zagadnienie polegające na znalezieniu w grafie ważonym najkrótszego połączenia pomiędzy danymi wierzchołkami. Szczególnymi przypadkami tego problemu są problem najkrótszej ścieżki od jednego wierzchołka do wszystkich innych oraz problem najkrótszej ścieżki pomiędzy wszystkimi parami wierzchołków. Okazuje się, że żeby znaleźć najkrótszą ścieżkę pomiędzy dwoma wierzchołkami grafu trzeba (w pesymistycznym przypadku) znaleźć najkrótsze ścieżki od wierzchołka wyjściowego do wszystkich innych wierzchołków. Problem najkrótszej ścieżki od jednego z wierzchołków do wszystkich innych można więc zobrazować jako problem znalezienia najkrótszej drogi pomiędzy dwoma miastami. W takim wypadku wierzchołkami grafu są skrzyżowania dróg, krawędziami – drogi, a wagi krawędzi odwzorowują długość danego odcinka drogowego. Drugi szczególny przypadek problemu najkrótszej ścieżki występuje, gdy chcemy znaleźć najkrótsze ścieżki pomiędzy każdą parą wierzchołków. Możliwe jest zrobienie tego dla każdego wierzchołka, używając algorytmu znajdującego najkrótszą ścieżkę od jednego wierzchołka do wszystkich innych, jednak metoda ta okazuje się w praktyce niezbyt efektywna.

Algorytm Dijkstry

Algorytm Dijkstry służy do wyznaczania najmniejszej odległości od ustalonego wierzchołka *s* do wszystkich pozostałych w grafie skierowanym. W algorytmie tym pamiętany jest zbiór Q wierzchołków, dla których nie obliczono jeszcze najkrótszych ścieżek, oraz wektor D[i] odległości od wierzchołka *s* do *i*. Początkowo zbiór Q zawiera wszystkie wierzchołki a wektor D jest pierwszym wierszem macierzy wag krawędzi A.

Zastosowanie algorytmu:

- Występuje na mapach geograficznych.
- Znalezienie lokalizacji mapy, która odnosi się do wierzchołków grafu.
- Odległość między lokalizacją odnosi się do krawędzi.
- Jest używany w trasowaniu IP, aby najpierw znaleźć otwartą najkrótszą ścieżkę.
- Stosowany jest w sieci telefonicznej.

Złożoność obliczeniowa: $O(|E|+|V|log\,|V|)$, pamęciowa: $O(|V|)$.

Założenia eksperymentu:

- Testujemy algorytm dla dwóch reprezentacji grafu. Graf zawsze się generuje w postacimacierzowej. W przypadku testowania listy sąsiedzictwa, graf przekształcamy w tą formę po zgenerowaniu.
- Nie liczymy czas generacji oraz przepisywanie grafu z jednej postaci do drugiej. Genracja grafu prepływa s następujący sposób: generujemy drzewo rozpinające, dalej wzależności od gęstości dodajemy krawędzie.
- Testujemy 7 punktów obserwowanych: 50, 60, 70, 80, 90, 100, 110 wierzchołków dlatrzech gęstości: 20%, 60%, 99%.
- Mierzymy czas 100 prób, zatem uśredniamy wynik.
- Graf jest skirowany. Znajdujemy scieżkę do wszystkich punktów od zerowego

Pierwszy wariant jest optymalny dla grafów gęstych, drugi jest szybszy dla grafów rzadkich, trzeci jest bardzo rzadko używany ze względu na duży stopień skomplikowania i niewielki w porównaniu z nim zysk czasowy.

Algorytm Bellmana - Forda

Algorytm ten służy do wyznaczania najmniejszej odległości od ustalonego wierzchołka *s* do wszystkich pozostałych w grafie skierowanym bez cykli o ujemnej długości.
Warunek nieujemności cyklu jest spowodowany faktem, że w grafie o ujemnych cyklach najmniejsza odległość między niektórymi wierzchołkami jest nieokreślona, ponieważ zależy od liczby przejść w cyklu.

Zastosowanie algorytmu:

- Występuje na mapach geograficznych.
- Znalezienie lokalizacji mapy, która odnosi się do wierzchołków grafu.
- Odległość między lokalizacją odnosi się do krawędzi.
- Jest używany w trasowaniu IP, aby najpierw znaleźć otwartą najkrótszą ścieżkę.
- Stosowany jest w sieci telefonicznej.

Złożoność obliczeniowa: $O(|E| * |V|)$, pamęciowa: $O(|V|)$.

Założenia eksperymentu:

- Testujemy algorytm dla dwóch reprezentacji grafu. Graf zawsze się generuje w postaci macierzowej. W przypadku testowania listy sąsiedzictwa, graf przekształcamy w tą formę po zgenerowaniu.
- Nie liczymy czas generacji oraz przepisywanie grafu z jednej postaci do drugiej. Genracja grafu prepływa s następujący sposób: generujemy drzewo rozpinające, dalej w zależności od gęstości dodajemy krawędzie.
- Testujemy 7 punktów obserwowanych: 50, 60, 70, 80, 90, 100, 110 wierzchołków dla trzech gęstości: 20%, 60%, 99%.
- Mierzymy czas 100 prób, zatem uśredniamy wynik.
- Graf jest skirowany. Znajdujemy scieżkę do wszystkich punktów od zerowego

3. Plan projektu

Zadanie polegało na pomiarze czasu działania poszczególnych algorytmów w zależności od rozmiaru grafu ijego gęstości. Badania zostały wykonane dla 5 różnych liczb wierzchołków: 50, 100, 150, 200, 250 oraz
następujących gęstości grafu: 25%, 50%, 75%, 99%. Dla każdego zestawu została wygenerowana seria losowych instancji. W sprawozdaniu zostały umieszczone uśrednione

wyniki z danej serii. Wszystkie struktury są alokowane dynamiczne, a przepustowość krawędzi jest liczbą całkowitą. Program umożliwia również sprawdzenie poprawności zaimplementowanych operacji. Czas mierzony jest poprzez pobranie czasu systemowego przed i po wykonaniu algorytmu, a następnie obliczeniu różnicy.

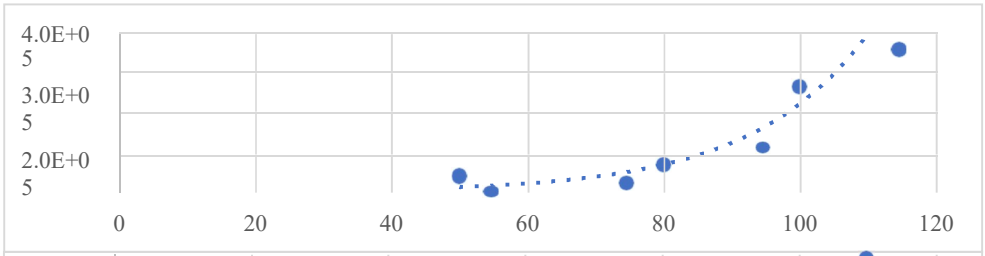
4. Wykresy

- **Algorytm Prima**

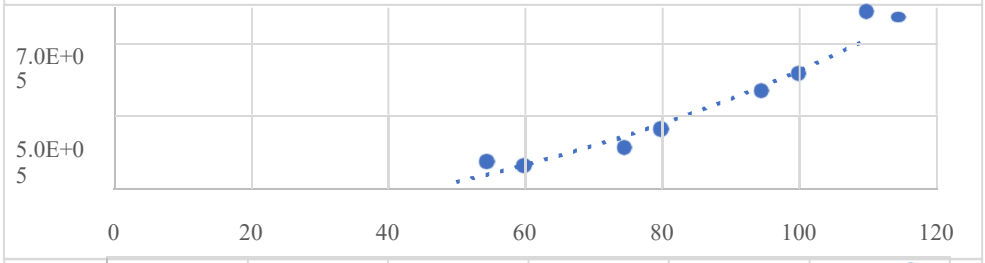
Gęstość

Macierz sąsiedzctwa

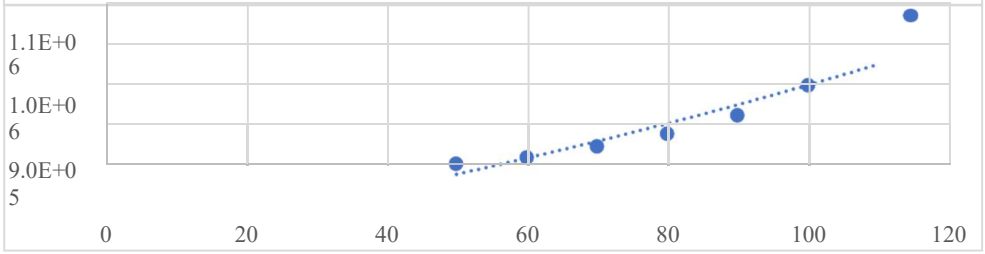
20%



60%



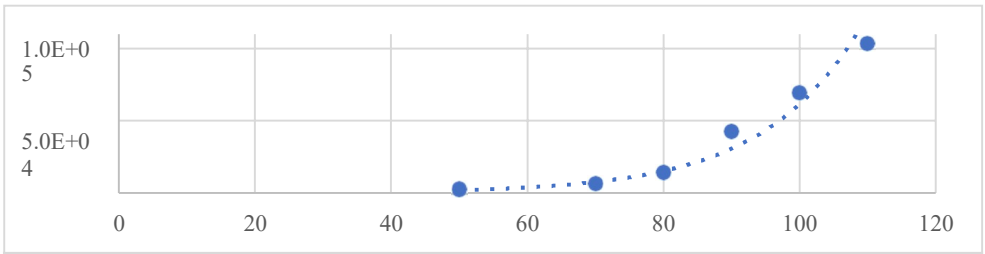
99%



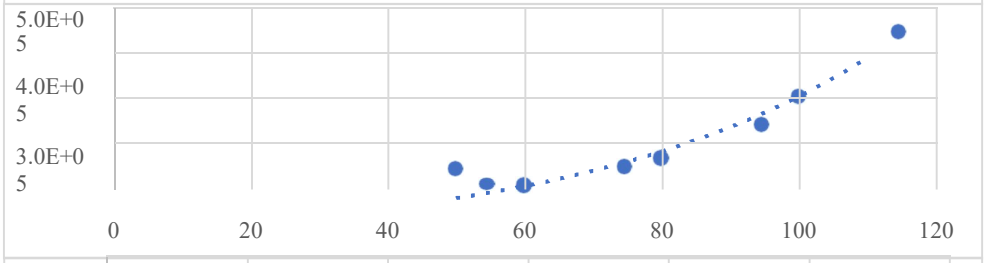
Gęstość

Lista sąsiadów

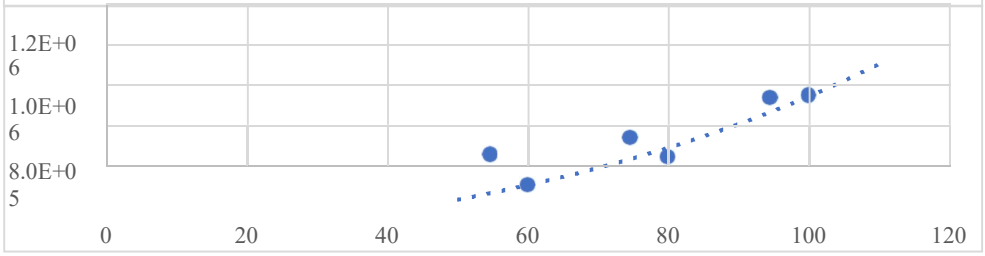
20%



60%



99%

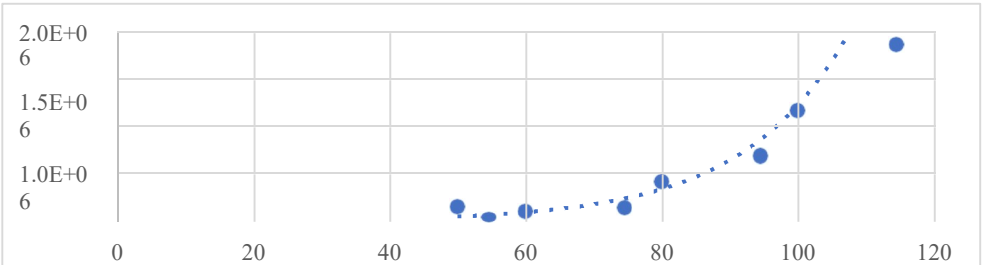


- **Algorytm Kruskala**

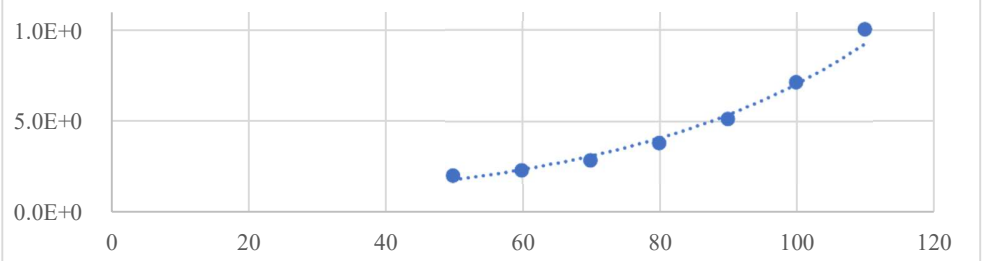
Gęstość

Macierz sąsiedzictwa

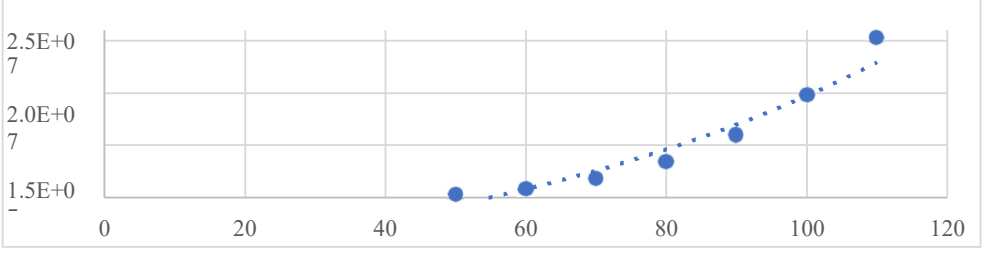
20%



60%



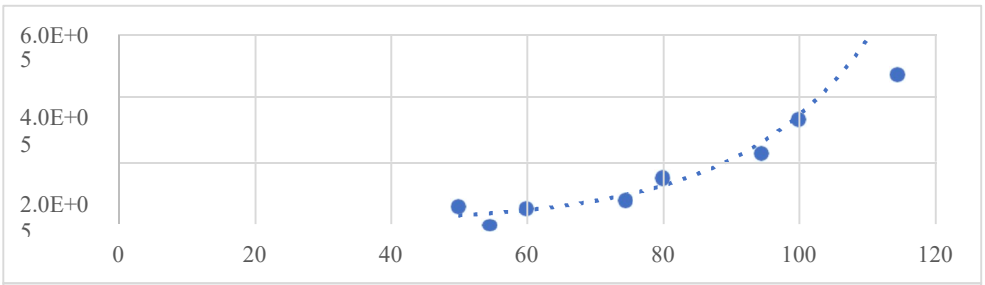
99%



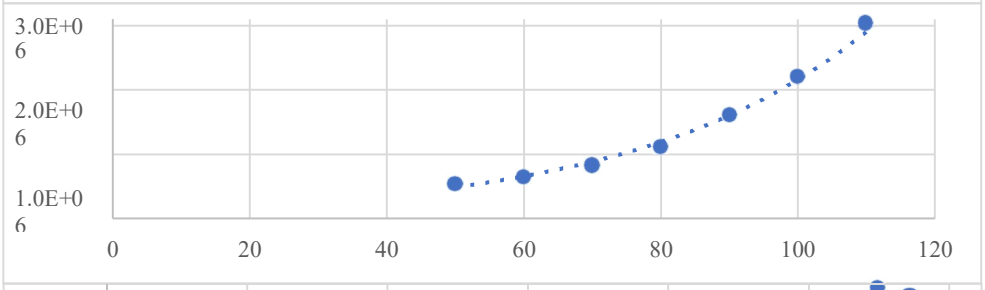
Gęstość

Lista sąsiadów

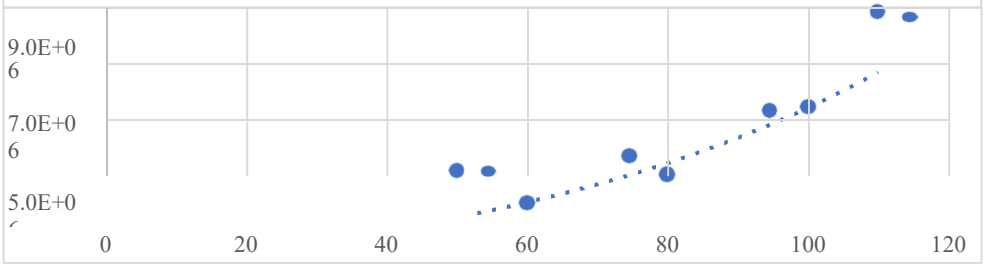
20%



60%



99%

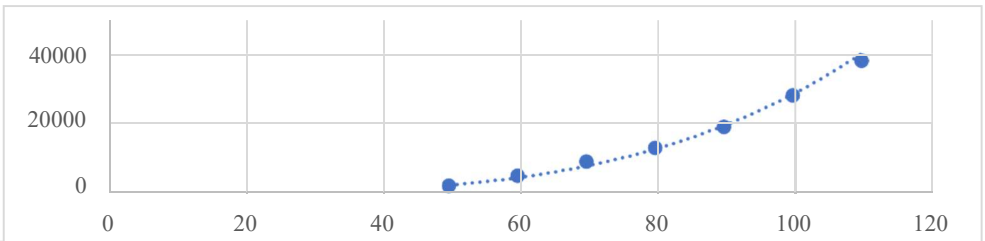


- **Algorytm Dijkstry**

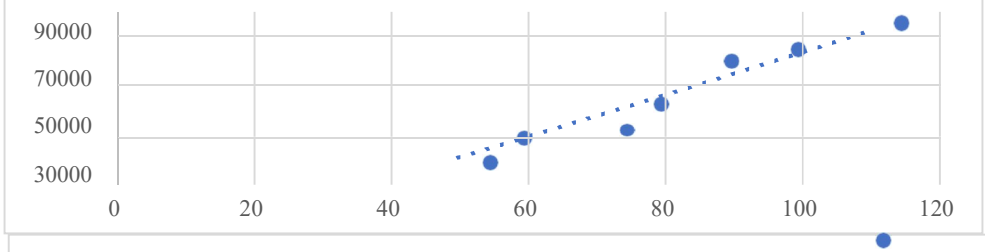
Gęstość

Lista sąsiadów

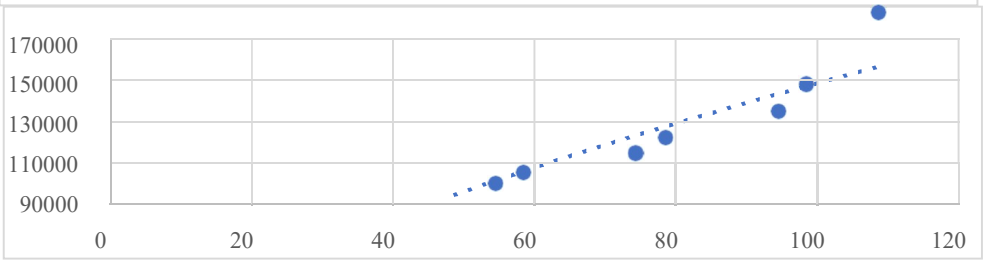
20%



60%



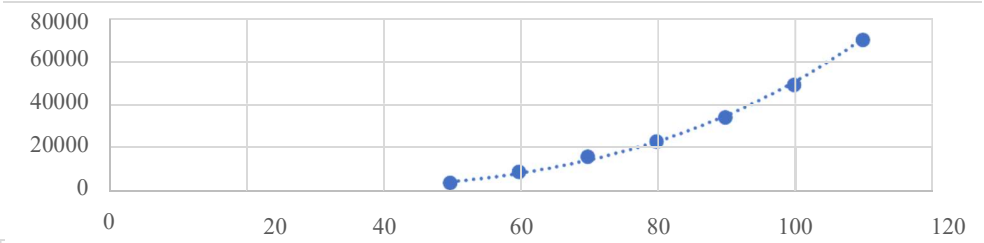
99%



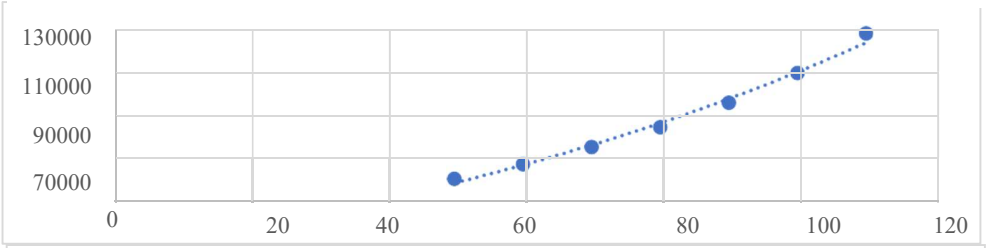
Gęstość

Macierz sąsiedztwa

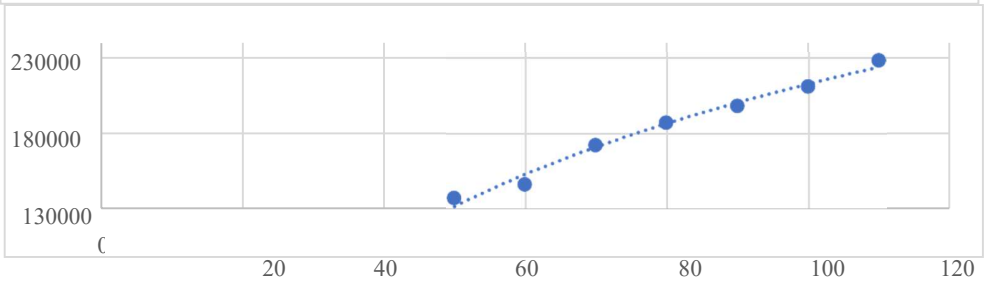
20%



60%



99%

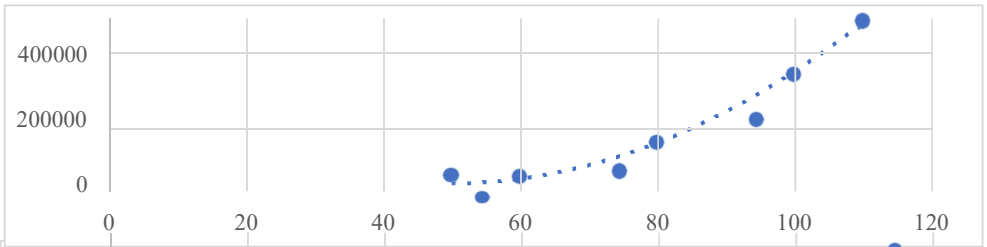


- Algorytm Bellmana-Forda

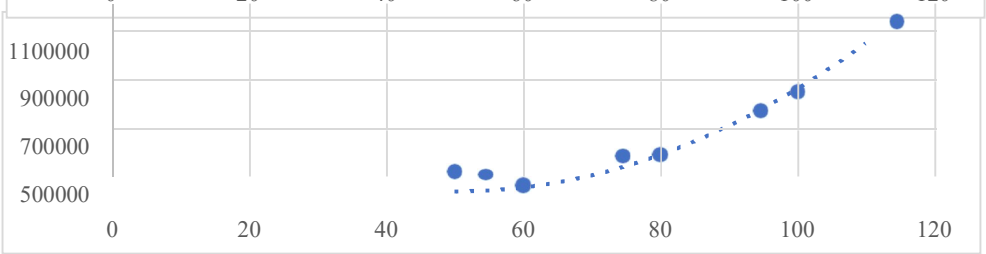
Gęstość

Macierz sąsiedztwa

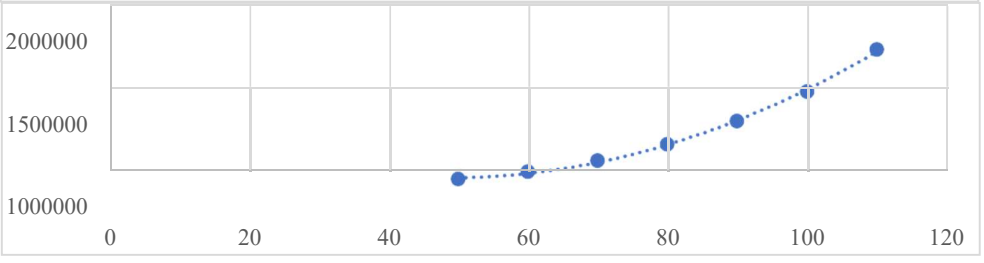
20%



60%



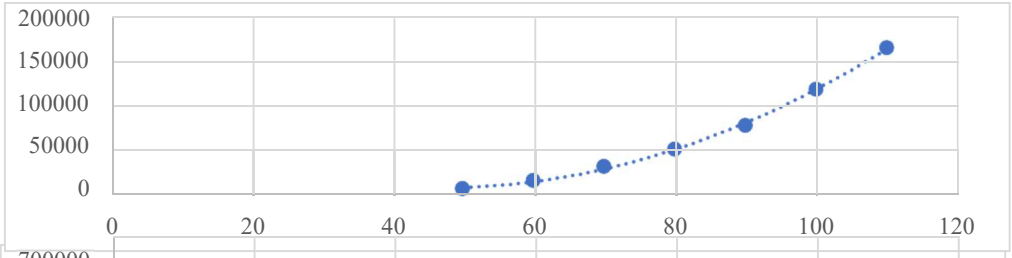
99%



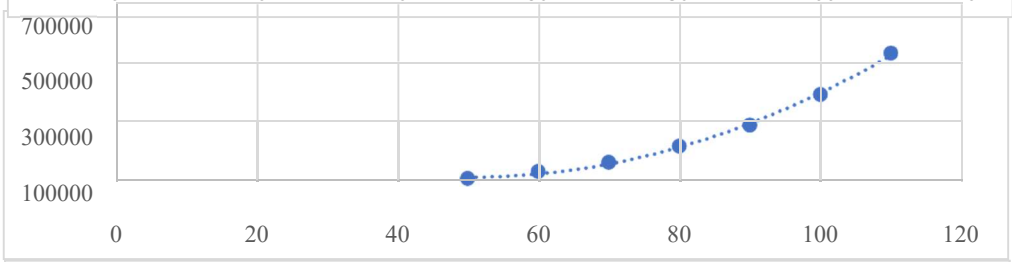
Gęstość

Lista sąsiadów

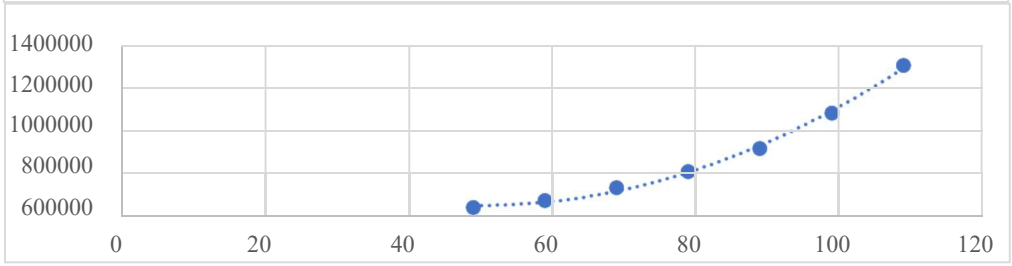
20%



60%



99%



5. Wnioski

Wyniki uzyskane podczas badania algorytmów pozwalają nam stwierdzić, że ich teoretyczna złożność pokrywa się z praktyką. Pewne wahania, wynikają z działania procesora w komputerze na którym były przeprowadzone testy.

W większości wypadków algorytmy działające na podstawie listy uzyskiwały lepsze wyniki od algorytmów działających na macierzach, ponieważ lista jest prostszą strukturą niż macierz incydencji, co pozwalało wykonywać mniej operacji.

Kolejnym wnioskiem jest to, że wszystkie algorytmy przy większej gęstości wydłużały swój czas działania, jednak np. W algorytmie Dijsktry widać, że mimo tego, że dla gęstości 99% wykres czasu działania algorytmu przypomina funkcje liniową, a dla mniejszych gęstości funkcje kwadratową – wynika to z jego złożoności obliczeniowej