Struktury Danych

Sprawozdanie

Projekt – Algorytm Prima - minimalne drzewo rozpinające

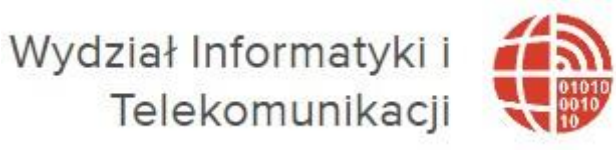
Jakub Piekarek

Indeks 264202

Prowadzący dr inż. Jarosław Rudy

Kod grupy K00-35b

Środa TP $15^{15} – 16^{55}$



1. Cel badań

   Celem naszych badań jest przeprowadzenie analizy złożoności czasowej działania algorytmu Prima dla różnych implementacji kolejki priorytetowej: opartej na liście oraz opartej na kopcu. W ramach badań wykorzystane zostały trzy popularne reprezentacje grafów: macierz sąsiedztwa, lista sąsiedztwa i listy krawędzi. Naszym głównym celem jest ustalenie, która z tych struktur danych pozwala na najszybsze działanie algorytmu Prima. Chcemy zbadać, jak różne implementacje kolejki priorytetowej wpływają na czas działania algorytmu dla różnych rozmiarów grafów oraz dla różnych typów reprezentacji grafu.

2. Metodologia

   2.1. Macierz sąsiedztwa

   Macierz sąsiedztwa jest dwuwymiarową tablicą o rozmiarze V x V, gdzie V to liczba wierzchołków grafu. Element macierzy na pozycji (i, j) reprezentuje wagę lub informację o krawędzi między wierzchołkiem i a j. Jeśli graf jest nieskierowany, macierz sąsiedztwa jest symetryczna wokół głównej przekątnej. Jeśli istnieje krawędź między wierzchołkiem i a j, odpowiedni element macierzy jest różny od zera, w przeciwnym razie jest równy zero. Macierz sąsiedztwa jest szczególnie przydatna w przypadku gęstych grafów, gdzie wiele krawędzi jest obecnych.

   2.2. Lista sąsiedztwa

   Lista sąsiedztwa polega na utworzeniu listy dla każdego wierzchołka w grafie. Dla każdego wierzchołka, lista zawiera informacje o jego sąsiadach, czyli innych wierzchołkach, do których prowadzą krawędzie. Może być reprezentowana jako tablica list, gdzie indeks tablicy odpowiada numerowi wierzchołka, a wartość tablicy to lista sąsiadów tego wierzchołka. Alternatywnie, można wykorzystać struktury danych takie jak listy lub zestawy dla każdego wierzchołka, aby przechowywać informacje o jego sąsiadach. Lista sąsiedztwa jest często stosowana w przypadku rzadkich grafów, gdzie jest niewiele krawędzi.

   2.3. Lista krawędzi

   Lista krawędzi reprezentuje graf poprzez wymienienie wszystkich krawędzi w grafie, wraz z ich wagami (jeśli graf jest ważony). Każda krawędź jest reprezentowana jako para (u, v), gdzie u i v to wierzchołki, które są połączone tą krawędzią. Dodatkowo, w przypadku grafów ważonych, można również przechowywać wagę krawędzi jako trzeci element pary. Lista krawędzi jest przydatna, gdy potrzebujemy łatwego dostępu do wszystkich krawędzi w grafie, bez konieczności przeszukiwania całej struktury grafu.

   2.4. Algorytm Prima

   Algorytm Prima jest algorytmem używanym do znajdowania minimalnego drzewa rozpinającego w ważonym grafie nieskierowanym. Minimalne drzewo rozpinające to podzbiór krawędzi grafu, który łączy wszystkie wierzchołki, tworząc drzewo bez cykli o minimalnej sumie wag krawędzi.

   Algorytm Prima działa na zasadzie stopniowego rozbudowywania minimalnego drzewa rozpinającego poprzez wybieranie krawędzi o najmniejszej wadze spośród dostępnych krawędzi. Początkowo drzewo składa się tylko z jednego wierzchołka, a następnie stopniowo dodawane są kolejne wierzchołki i krawędzie, aby połączyć wszystkie wierzchołki w minimalne drzewo rozpinające.

2.5. Generowanie grafów do przeprowadzenia badań

Funkcja generuje graf z losowymi krawędziami między wierzchołkami, aby stworzyć graf, w którym wszystkie wierzchołki są ze sobą połączone. Proces ten polega na wybieraniu losowych wierzchołków, tworzeniu krawędzi między nimi i oznaczaniu ich jako odwiedzone, aż do momentu, gdy cały graf będzie spójny. Na końcu zwracamy ten wygenerowany graf. Możliwe jest modyfikowanie liczby wierzchołków i krawędzi.

2.6. Dobranie danych do badań

Badania zostały przeprowadzone dla sześciu różnych wygenerowanych grafów o różnych gęstościach w celu oceny poprawności generatora. Poniżej przedstawiam liczbę wierzchołków oraz wartości gęstości dla każdego z grafów:

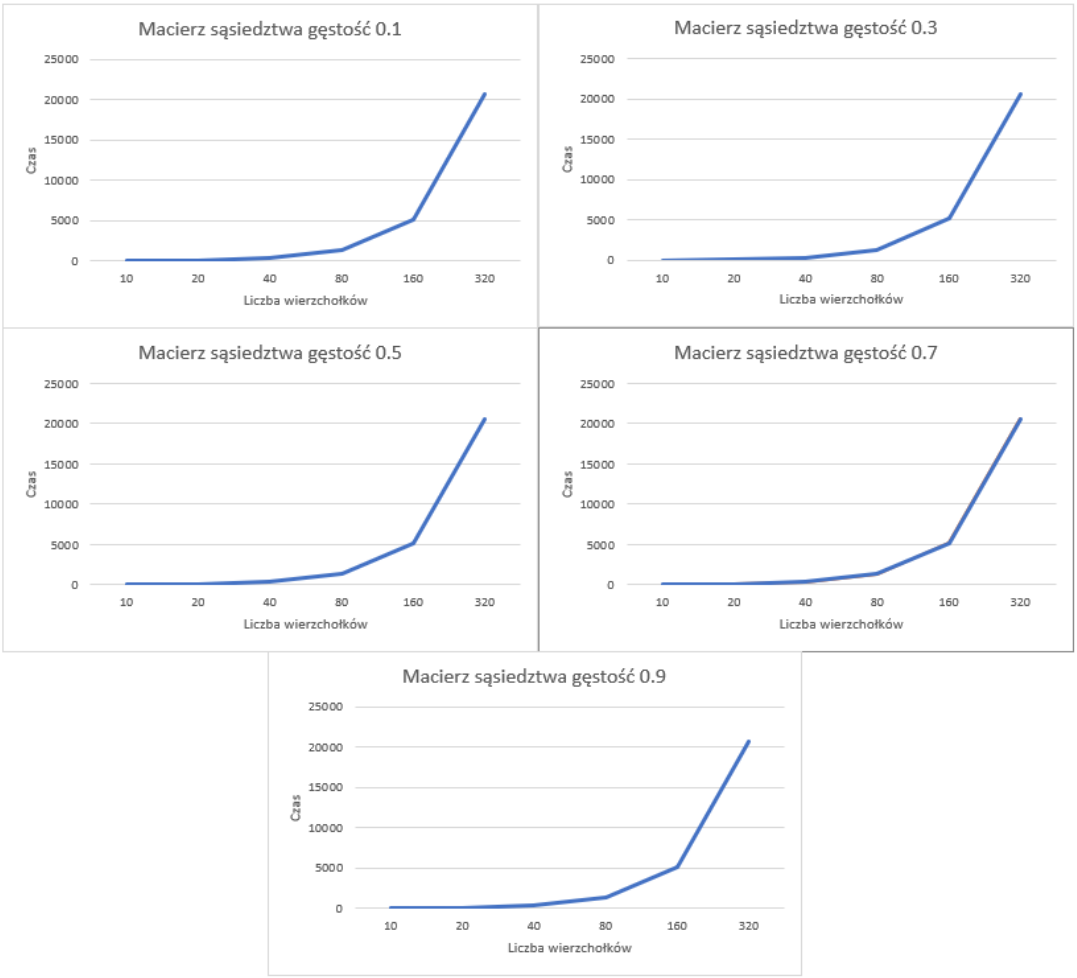
- Graf 1: 10 wierzchołków,
  Gęstości:
  - 0.1,
  - 0.3,
  - 0.5,
  - 0.7,
  - 0.9.
- Graf 2: 20 wierzchołków,
  Gęstości:
  - 0.1,
  - 0.3,
  - 0.5,
  - 0.7,
  - 0.9.
- Graf 3: 40 wierzchołków,
  Gęstości:
  - 0.1,
  - 0.3,
  - 0.5,
  - 0.7,
  - 0.9.
- Graf 4: 80 wierzchołków,
  Gęstości:
  - 0.1,
  - 0.3,
  - 0.5,
  - 0.7,
  - 0.9.
- Graf 5: 160 wierzchołków,
  Gęstości:
  - 0.1,
  - 0.3,
  - 0.5,
  - 0.7,
  - 0.9.

- Graf 6: 320 wierzchołków.
  Gęstości:
  - 0.1,
  - 0.3,
  - 0.5,
  - 0.7,
  - 0.9.

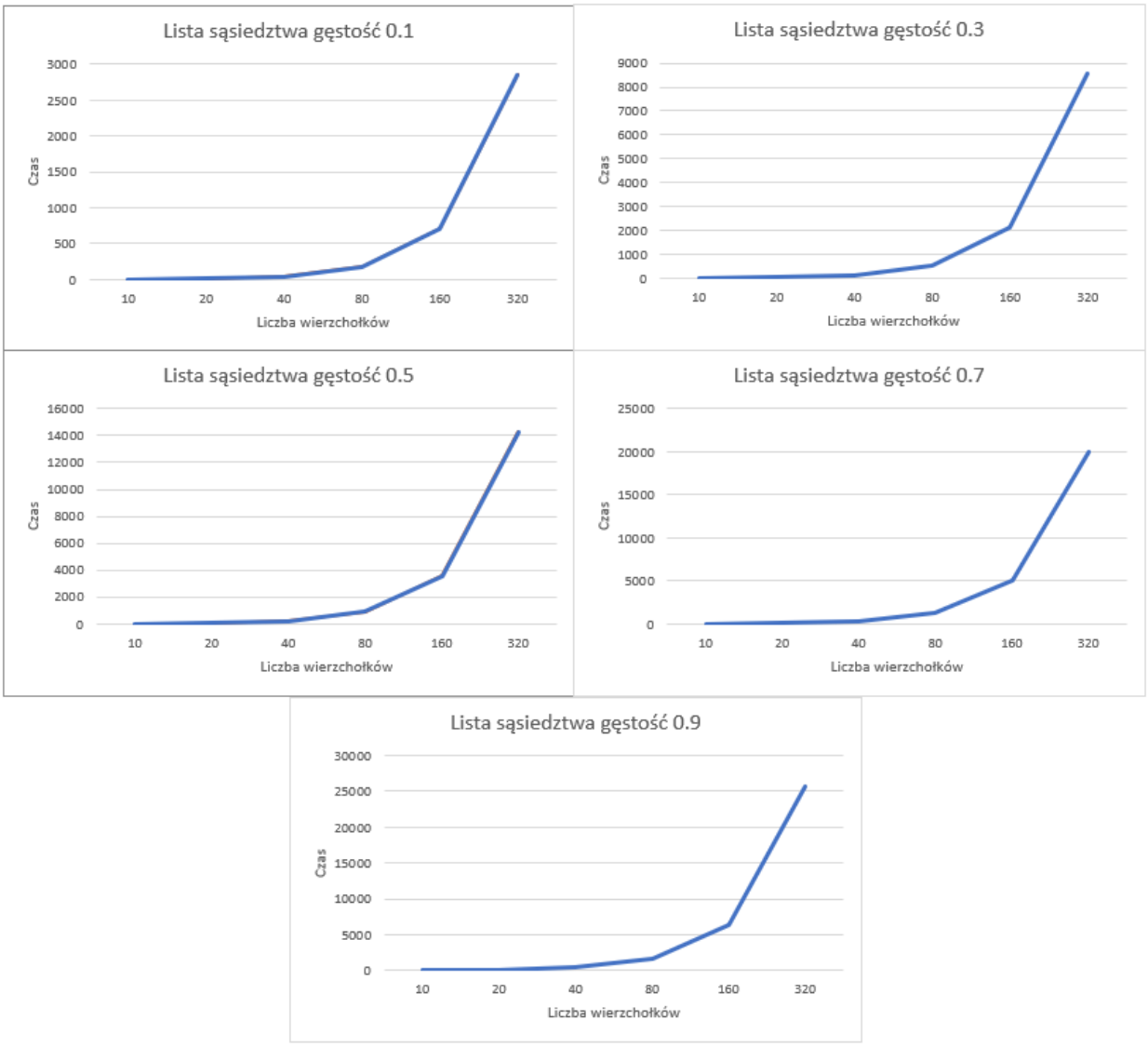
W trakcie badań, wartości liczby wierzchołków i gęstość były takie same dla obu przykładów implementacji kolejki priorytetowej.

3. Badania

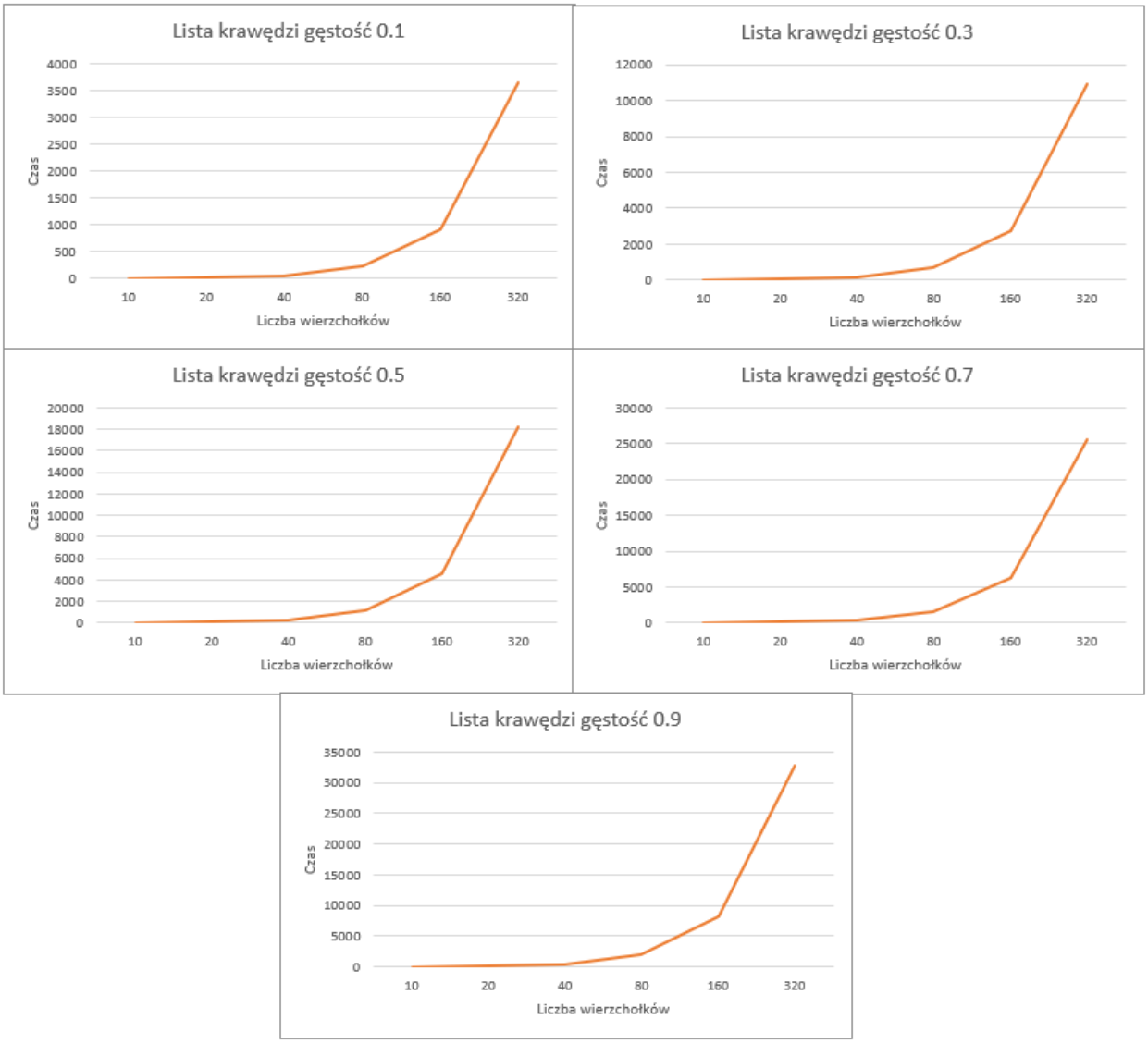
3.1. Badania przeprowadzone dla kolejki priorytetowej opartej na kopcu



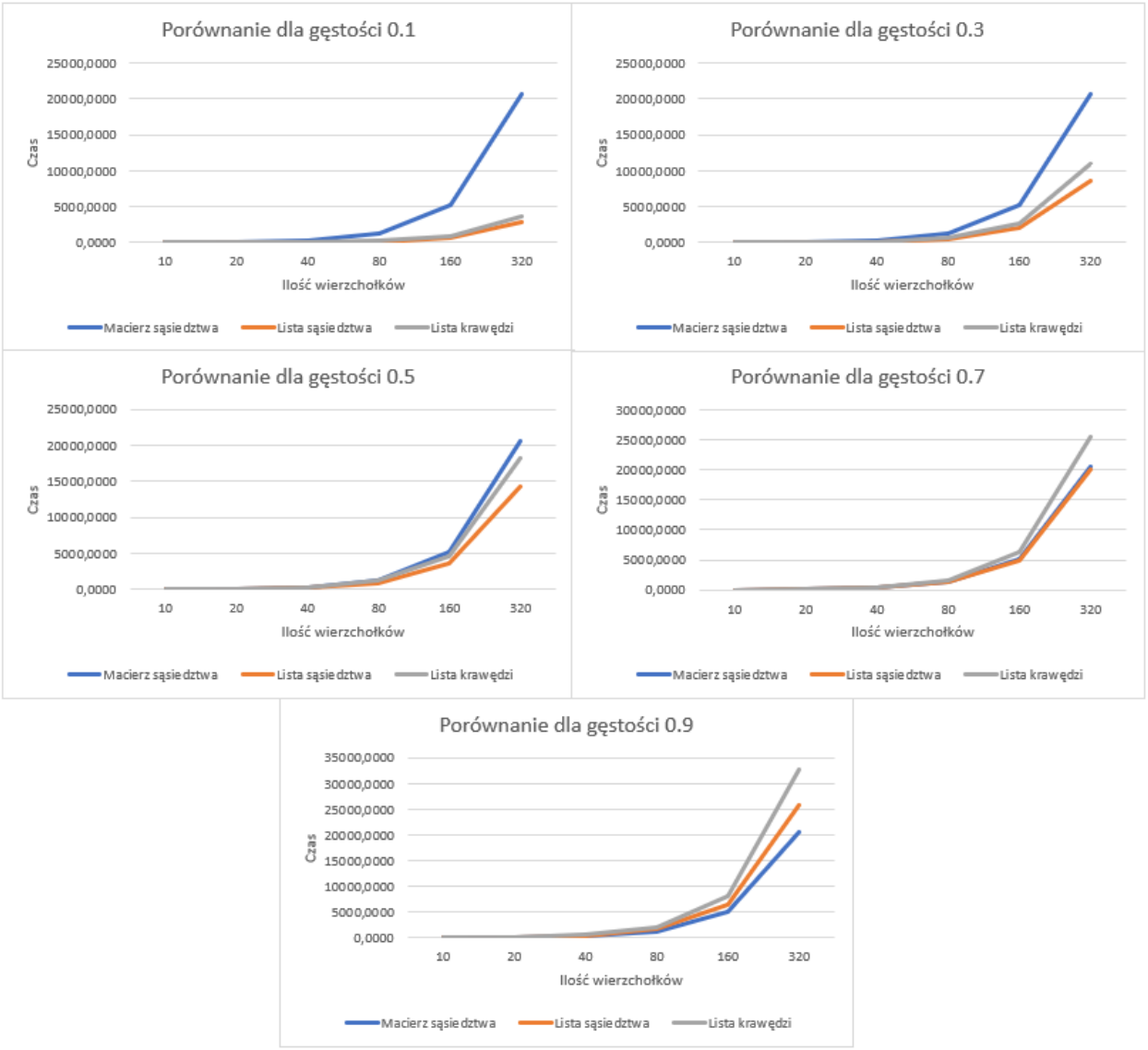
Rysunek 1 Wykresy dla macierzy sąsiedztwa o różnej gęstości



Rysunek 2 Wykresy dla listy sąsiedztwa o różnej gęstości

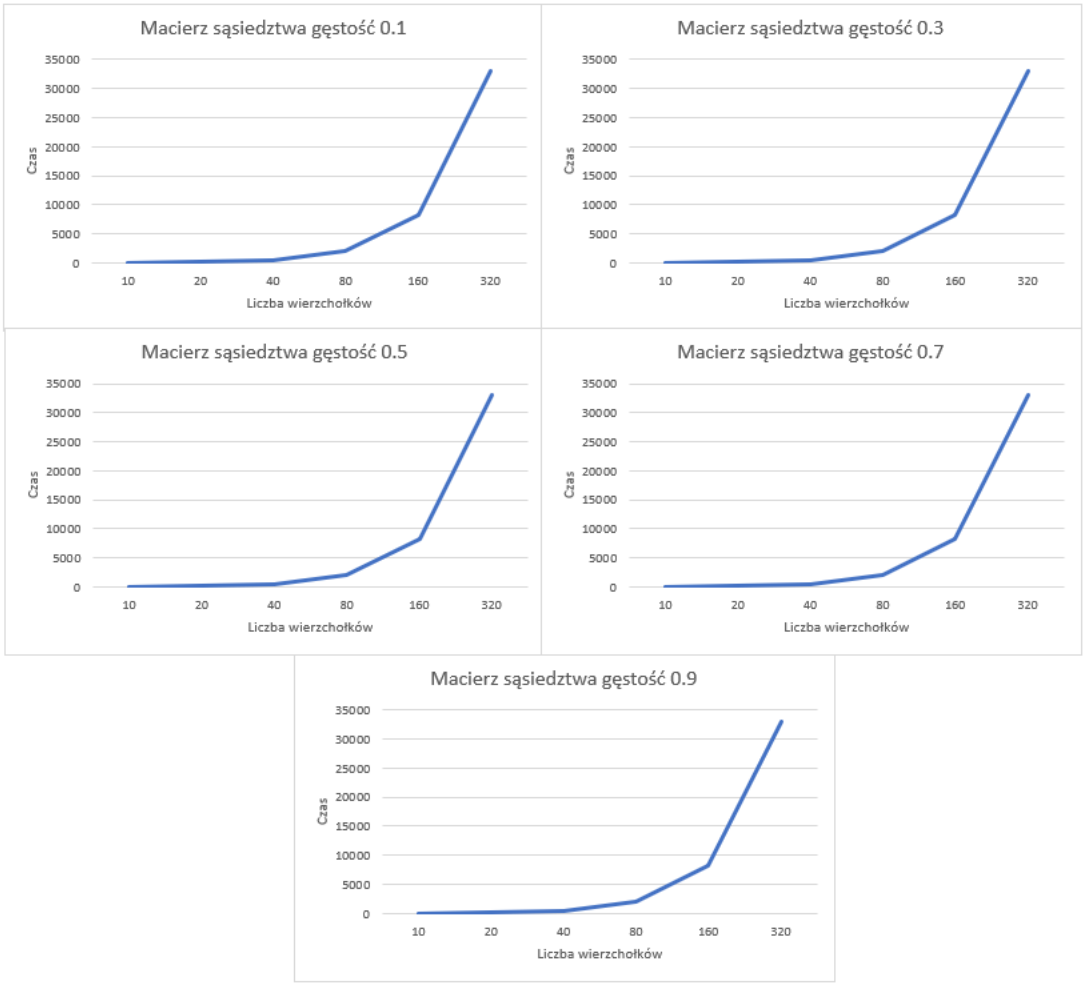


Rysunek 3 Wykresy dla listy krawędzi o różnej gęstości

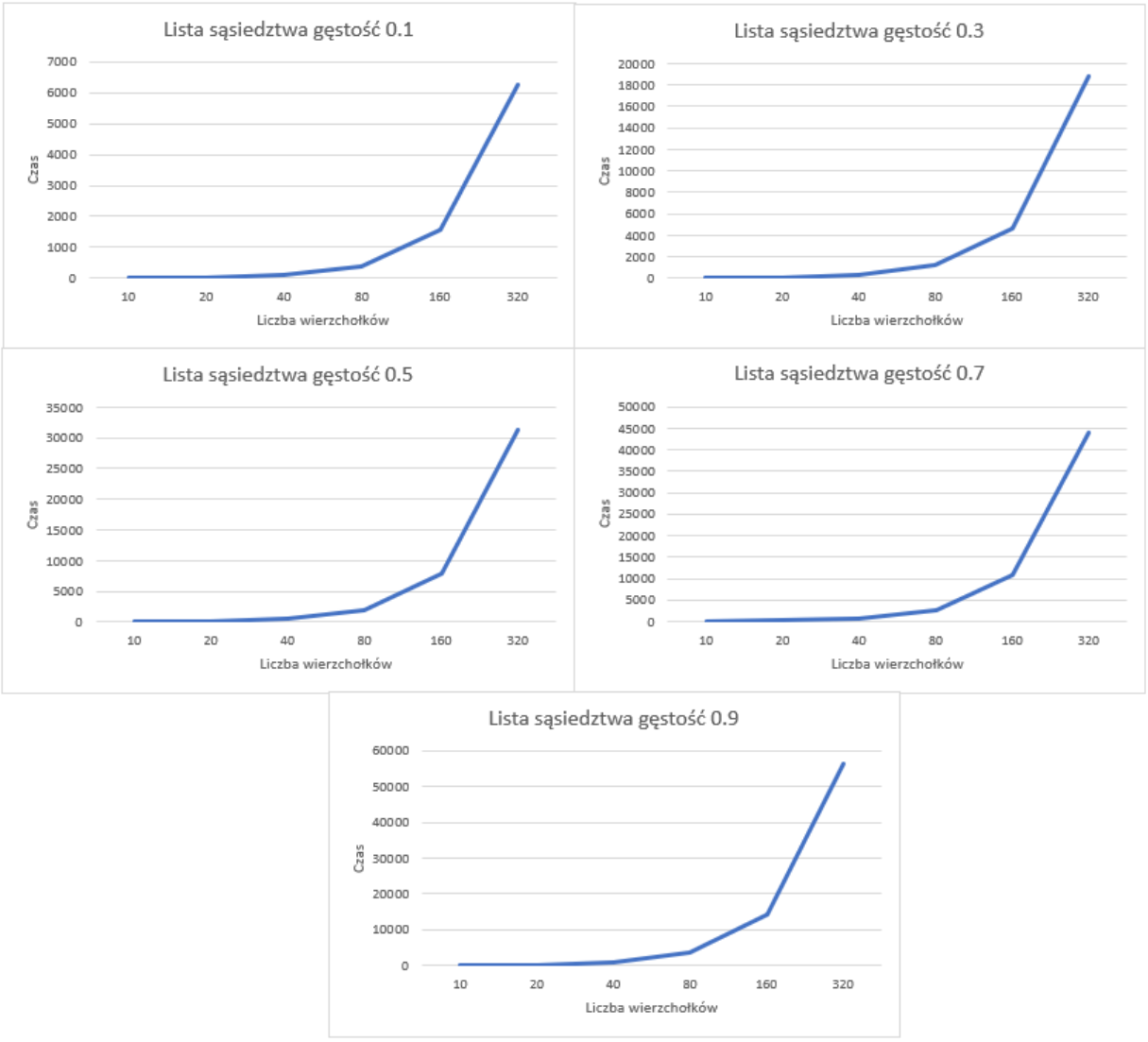


Rysunek 4 Porównanie reprezentacji grafów dla różnych gęstości

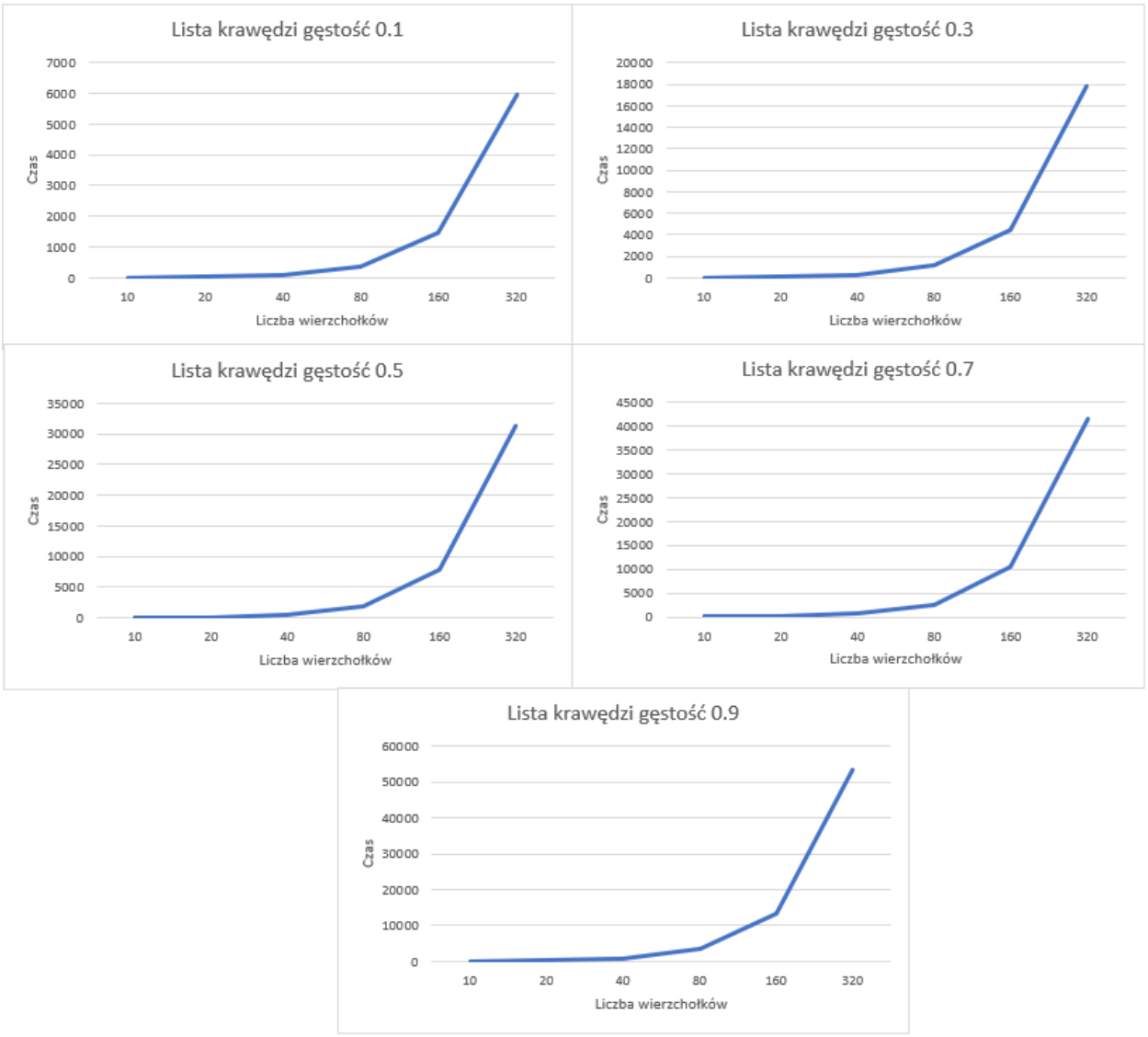
3.2. Badania przeprowadzone dla kolejki priorytetowej opartej na liście



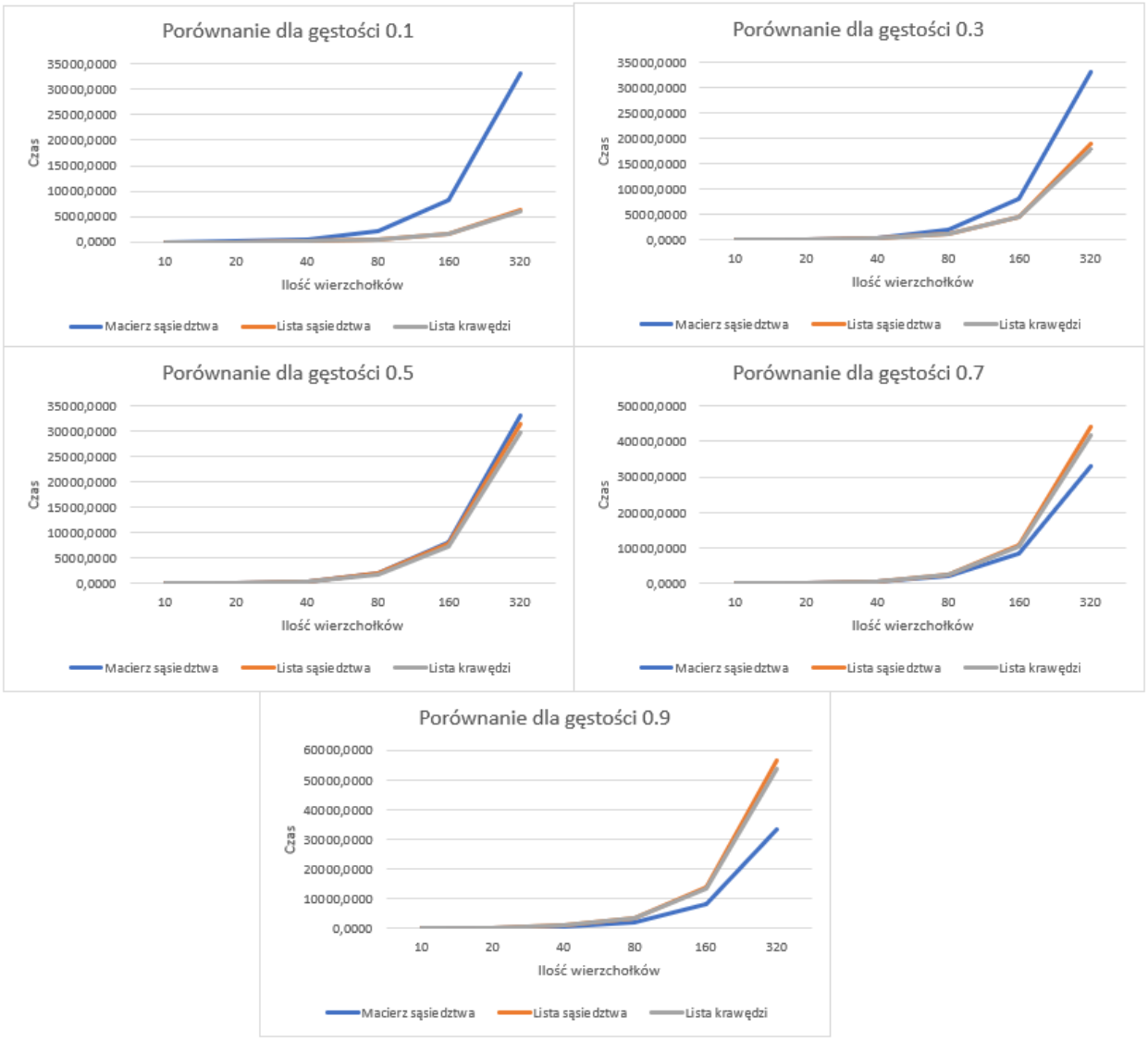
Rysunek 5 Wykresy dla macierzy sąsiedztwa dla różnych gęstości



Rysunek 6 Wykresy dla listy sąsiedztwa o różnych gęstościach



Rysunek 7 Wykresy dla listy krawędzi o różnych gęstościach



Rysunek 8 Porównanie dla reprezentacji grafów dla różnych gęstości

4. Złożoności obliczeniowe
   4.1. Algorytm Prima używający Kolejki Priorytetowej na kopcu:
      a. Macierz sąsiedztwa:
         - Tworzenie kopca: O(V) (gdzie V to liczba wierzchołków)
         - Dodawanie wierzchołka do drzewa rozpinającego: O(1)
         - Łącznie dla wszystkich wierzchołków: O(V)
         - Znajdowanie minimalnego elementu: O(log V)
         - Łącznie dla wszystkich krawędzi: O(E log V) (gdzie E to liczba krawędzi)
         - Łączna złożoność obliczeniowa dla macierzy sąsiedztwa: $O(V^2 + E \log V)$
      b. Lista sąsiedztwa:
         - Tworzenie kopca: O(V) (gdzie V to liczba wierzchołków)
         - Dodawanie wierzchołka do drzewa rozpinającego: O(1)
         - Łącznie dla wszystkich wierzchołków: O(V)
         - Znajdowanie minimalnego elementu: O(V log V)
         - Łącznie dla wszystkich krawędzi: O((V + E) log V) (gdzie E to liczba krawędzi)
         - Łączna złożoność obliczeniowa dla listy sąsiedztwa: O((V + E) log V)
      c. Lista krawędzi:
         - Tworzenie kopca: O(E) (gdzie E to liczba krawędzi)
         - Dodawanie wierzchołka do drzewa rozpinającego: O(1)
         - Łącznie dla wszystkich wierzchołków: O(V)
         - Znajdowanie minimalnego elementu: O(log E)
         - Łącznie dla wszystkich krawędzi: O(E log E)
         - Łączna złożoność obliczeniowa dla listy krawędzi: O(E log E + V)
   4.2. Algorytm Prima używający Kolejki Priorytetowej na liście:
      a. Macierz sąsiedztwa:
         - Tworzenie listy: O(V + E)
         - Dodawanie wierzchołka do drzewa rozpinającego: O(1)
         - Łącznie dla wszystkich wierzchołków: O(V)
         - Znajdowanie minimalnego elementu: O(V)
         - Łącznie dla wszystkich krawędzi: O((V + E) V)
         - Łączna złożoność obliczeniowa dla macierzy sąsiedztwa: $O(V^2 + EV)$
      b. Lista sąsiedztwa:
         - Tworzenie listy: O(V + E)
         - Dodawanie wierzchołka do drzewa rozpinającego: O(1)
         - Łącznie dla wszystkich wierzchołków: O(V)
         - Znajdowanie minimalnego elementu: O(V)
         - Łącznie dla wszystkich krawędzi: O((V + E) V)
         - Łączna złożoność obliczeniowa dla listy sąsiedztwa: $O(V^2 + EV)$
      c. Lista krawędzi:
         - Tworzenie listy: O(E)
         - Dodawanie wierzchołka do drzewa rozpinającego: O(1)
         - Łącznie dla wszystkich wierzchołków: O(V)
         - Znajdowanie minimalnego elementu: O(V)
         - Łącznie dla wszystkich krawędzi: O((V + E) V)
         - Łączna złożoność obliczeniowa dla listy krawędzi: $O(V^2 + EV)$

5. Wnioski

Algorytm Prima, niezależnie od wybranej implementacji, jest skutecznym narzędziem do znajdowania minimalnego drzewa rozpinającego w grafie ważonym. Użycie macierzy sąsiedztwa ułatwia implementację, ale może być nieefektywne dla gęstych grafów. Z kolei lista sąsiedztwa jest lepsza dla rzadkich grafów, jednak może być wolniejsza dla gęstych. Wykorzystanie listy krawędzi umożliwia łatwe uzyskanie minimalnego drzewa rozpinającego w postaci listy krawędzi.

Dla badanych danych jesteśmy w stanie ocenić, która reprezentacja grafu jest najoptymalniejsza. Jeśli chodzi o kolejki priorytetowe na kopcu, najlepszym wyborem będzie macierz sąsiedztwa lub lista sąsiedztwa. Natomiast dla kolejki priorytetowej na liście, nie ma znaczącej różnicy - łączna złożoność jest taka sama w każdym przypadku.

Podsumowując, złożoność obliczeniowa algorytmu Prima zależy od sposobu reprezentacji grafu oraz używanej implementacji kolejki priorytetowej. Wybór odpowiednich struktur danych może mieć duże znaczenie dla wydajności algorytmu. Zazwyczaj, w przypadku kopca, złożoność obliczeniowa jest bardziej optymalna niż w przypadku listy.