2020

Zadanie projektowe 2

BADANIE EFEKTYWNOŚCI ALGORYTMÓW GRAFOWYCH W ZALEŻNOŚCI OD ROZMIARU INSTANCJI ORAZ SPOSOBU REPREZENTACJI GRAFU W PAMIĘCI KOMPUTERA

MACIEJ BRONIKOWSKI 248838

1 WSTĘP

W programie zostało zrealizowanych 5 algorytmów:

1. Wyznaczanie minimalnego drzewa rozpinającego
    - Algorytm Prima
    - Algorytm Kruskala
2. Wyznaczanie najkrótszej ścieżki w grafie
    - Algorytm Dijkstry
    - Algorytm Forda-Bellmana
3. Wyznaczanie maksymalnego przepływu
    - Algorytm Forda Fulkersona

Zostały one zaimplementowane na dwóch strukturach:

- Reprezentacji macierzowej grafów (macierz sąsiedztwa)
- Reprezentacji listowej grafów (lista sąsiedztwa)

Do obsługi list, oraz kolejki została zaimplementowana klasa `List` z typem generycznym (ponieważ klasa ta wykorzystywana jest do przechowywania różnych obiektów w zależności od potrzeby), a do stworzenia kolejki priorytetowej implementacja kopca w klasie `Heap`.

Teoretyczne złożoności obliczeniowe struktur:

Algorytm	**Złożoność**
Prima	O(|E|log|V|)
Kruskala	O(|E|log|V|)
Dijkstry	O(|E| + |V|log|V|)
Forda-Bellmana	O(|V|*|E|)
Forda Fulkersona	O(|E|*|f|)

Gdzie E oznacza liczbę krawędzi, V liczbę wierzchołków, a f maksymalny przepływ w grafie.

2 PLAN EKSPERYMENTU

Za pomocą metody `Menu::random(size_t nodesLength, int fillPercent)`, dla której `nodesLength` oznacza ilość wierzchołków do wygenerowania, a `fillPercent` gęstość grafu (ilość krawędzi w stosunku do grafu pełnego podana w procentach) zostały wylosowane dane, dla których później zostaną obliczone czasy wykonywania się kolejnych algorytmów. W menu po wybraniu opcji „Testy” jest możliwość podania ilości wierzchołków w grafie, którego gęstość wyniesie kolejno 20%, 50% i 99%. Czas trwania algorytmu obliczany jest za pomocą biblioteki chrono `chrono` przy użyciu metody `std::chrono::high_resolution_clock::now()`. Założenia co do ilości wierzchołków grafów, ilości iteracji wykonania algorytmów, oraz gęstości:

Lp.	Ilość wierzchołków	Ilość iteracji
1	10	100
2	20	100
3	40	100
4	60	100
5	80	100
6	100	100
7	120	100

Wyniki eksperymentu, po wprowadzeniu danych zostają wyświetlone w oknie konsoli.

3 OPIS METODY GENEROWANIA GRAFU

Metoda `Menu::random(size_t nodesLength, int fillPercent)` w pierwszym etapie oblicza ilość krawędzi do wygenerowania na podstawie `nodesLength` oraz `fillPercent`. Kolejnym etapem jest wygenerowanie i wstawienie do listy wszystkich możliwych krawędzi. (Jeżeli graf jest skierowany to ilość tych krawędzi wynosi $e = \frac{v*(v-1)*p}{100}$, gdzie v wynosi ilość krawędzi, a p ilość wierzchołków. Dla grafów skierowanych wartość e jest dzielona dodatkowo przez 2, ponieważ krawędzie będą dodawane w dwóch kierunkach. Następnie metoda generuje losowe drzewo rozpinające zaczynając od wierzchołka 0, i losując krawędzie kolejnych wierzchołków V z puli V-1. (każdy kolejny wierzchołek ma możliwość połączenia z wierzchołkiem już połączonym. Dla grafu skierowanego krawędź dodawana jest w kierunku od wierzchołka już istniejącego w drzewie, do wierzchołka dodawanego. Jeżeli graf jest nieskierowany, to dodawana jest krawędź również w kierunku przeciwnym. Każda kolejno dodana krawędź jest usuwana z listy możliwych krawędzi, oraz odjęta zostaje ich ilość z ilości krawędzi do wygenerowania. Ostatnim już etapem generowania grafów jest wylosowanie spośród listy możliwych krawędzi tylu z nich, aby gęstość grafu była odpowiednia. (tu również, jeżeli graf jest nieskierowany dodawane są dodatkowo krawędzie w przeciwnym kierunku). Dla każdej krawędzi zostaje wylosowana również jej waga (dla grafów nieskierowanych waga krawędzi w przeciwnym kierunku zostaje taka sama).

Dodatkowo zostają również wylosowane dane dotyczące wierzchołka początkowego dla algorytmów najkrótszej ścieżki i maksymalnego przepływu, oraz wierzchołek końcowy dla algorytmu maksymalnego przepływu, który jest różny od tego startowego.

4 WYNIKI

4.1 TABELE WYNIKÓW

4.1.1 Wyznaczanie minimalnego drzewa rozpinającego

Macierz sąsiedztwa						
Algorytm	Prima			Kruskala		
Gęstość	20%	60%	99%	20%	60%	99%
ilość wierzchołków	[us]	[us]	[us]	[us]	[us]	[us]
10	1.24	1.06	1.62	1.08	2.11	3.43
20	3.13	3.77	4.07	3.65	9.95	18.1
40	7.77	13.65	8.12	17.2	46.79	78.92
60	17.33	25.51	14.76	37.57	118.29	197.29
80	29.47	46.01	25.78	72.51	239.21	381.15
100	46.43	64.61	29.43	114.59	404.27	630.11
120	64.7	93.1	42.69	172.27	571.97	993.18

Lista sąsiedztwa						
Algorytm	Prima			Kruskala		
Gęstość	20%	60%	99%	20%	60%	99%
ilość wierzchołków	[us]	[us]	[us]	[us]	[us]	[us]
10	1.02	1.12	1.48	1.18	2.68	6.13
20	2.24	3.32	4.74	5.24	26.37	60.2
40	6.18	12.01	18.12	42.89	265.31	693.46
60	13.85	33.57	53.77	156.77	1226.5	3204.63
80	28.59	69.65	118.54	448.39	3717.3	9865.97
100	49.09	133.4	210.35	1030.78	8919.9	23375.7
120	69.85	261.2	388.69	2071.35	17925.7	48252.9

4.1.2 Wyznaczanie najkrótszej ścieżki

Macierz sąsiedztwa						
Algorytm	Dijkstry			Forda-Bellmana		
Gęstość	20%	60%	99%	20%	60%	99%
ilość wierzchołków	[us]	[us]	[us]	[us]	[us]	[us]
10	1.05	1.03	1.94	2.3	7.3	11.69
20	2.54	2.57	5.01	10.62	36.49	63.55
40	8.42	13.18	8.97	67.89	243.93	434.73
60	19.7	29.72	17.49	210.91	1111.28	1302.05
80	36.09	51.66	29.84	508.69	2858.22	2887.44
100	53.56	77.92	37.56	1183.13	3881.3	5229.1
120	74.88	108.4	51.85	1524.54	6710.09	8660.59

Lista sąsiedztwa						
Algorytm	Dijkstry			Forda-Bellmana		
Gęstość	20%	60%	99%	20%	60%	99%
ilość wierzchołków	[us]	[us]	[us]	[us]	[us]	[us]
10	1.07	1.17	1.55	0.3	1.33	2.45
20	2.12	2.67	4.2	4.84	15.61	28.31
40	6.9	12.11	18.35	48.92	196.09	387.48
60	15.9	33.08	53.99	268.58	1169.77	2103.49
80	32.8	78.36	132.03	757.56	3807.35	6465.76
100	56.09	148.24	208.16	2059.81	7356.09	12745.9
120	86.29	252.86	331.09	2590.37	13704.4	23401

4.1.3 Wyznaczanie maksymalnego przepływu

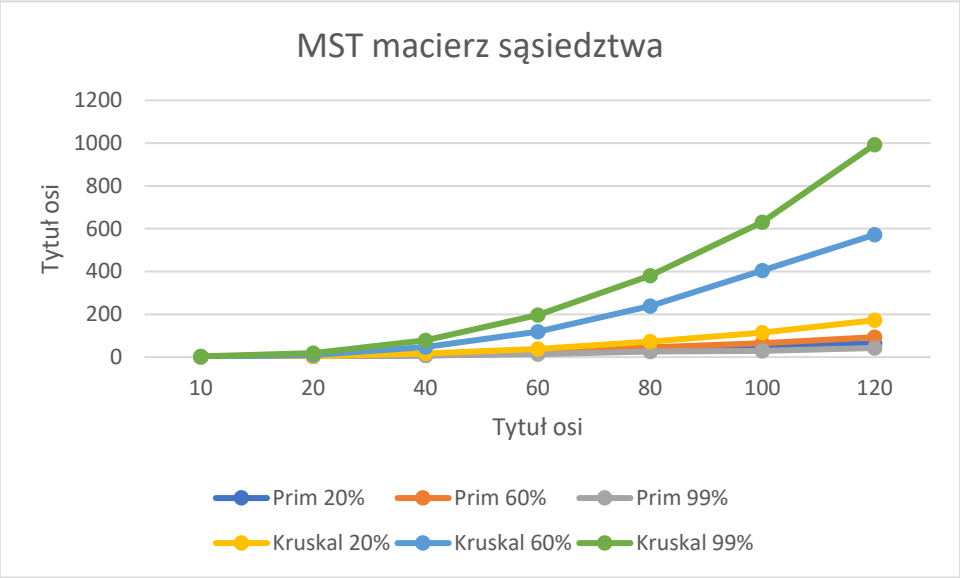
Macierz sąsiedztwa			
Algorytm	Forda Fulkersona		
Gęstość	20%	60%	99%
ilość wierzchołków	[us]	[us]	[us]
10	1.02	3.14	13.1
20	13.2	45.46	56.41
40	96.66	263.51	259.81
60	218.51	878.72	717.05
80	490.11	1772.99	1479.93
100	906.57	3456.62	2193.87
120	1416.67	5434.35	3463.76

Lista sąsiedztwa			
Algorytm	Forda Fulkersona		
Gęstość	20%	60%	99%
ilość wierzchołków	[us]	[us]	[us]
10	3.07	8.62	23.53
20	19.64	67.99	101.55
40	102.13	295.6	508.75
60	194.82	1043.36	1727.52
80	446.34	2176.98	3380.36
100	789.86	3471.04	5675.35
120	1073.07	5014.21	9407.04

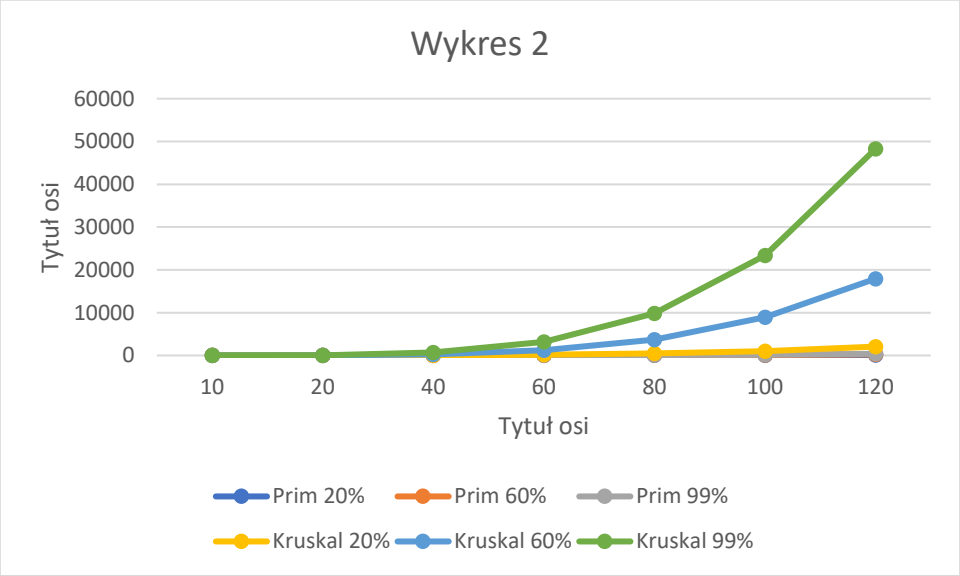
4.2 WYKRESY

4.2.1 Wykres zależności gęstości grafu i typu algorytmu

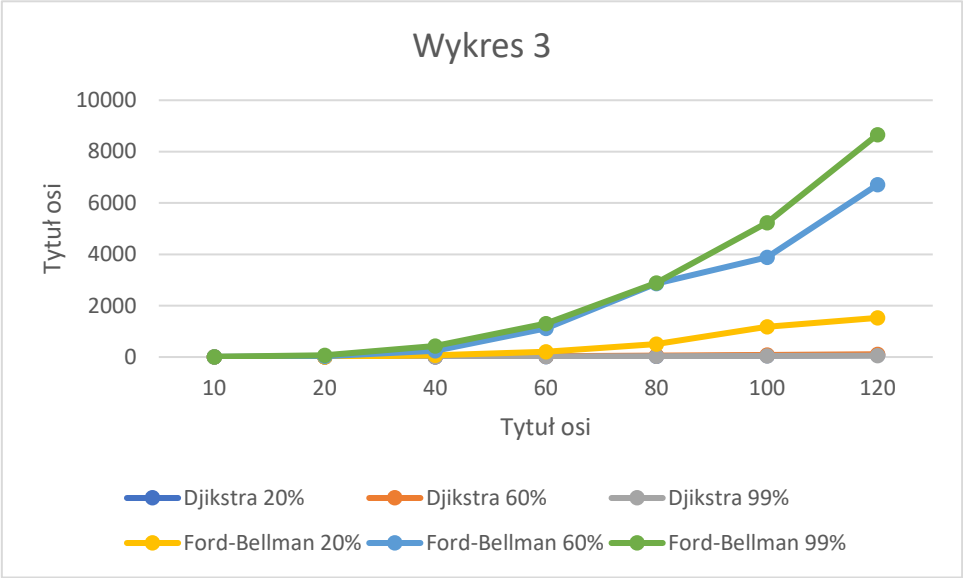
4.2.1.1 Algorytm minimalnego drzewa rozpinającego dla macierzy sąsiedztwa



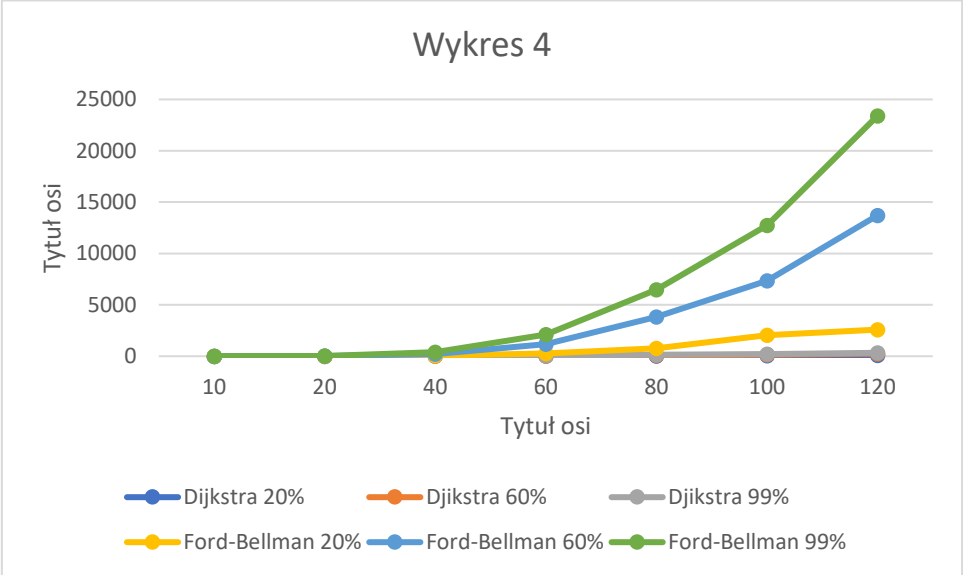
4.2.1.2 Algorytm minimalnego drzewa rozpinającego dla listy sąsiedztwa



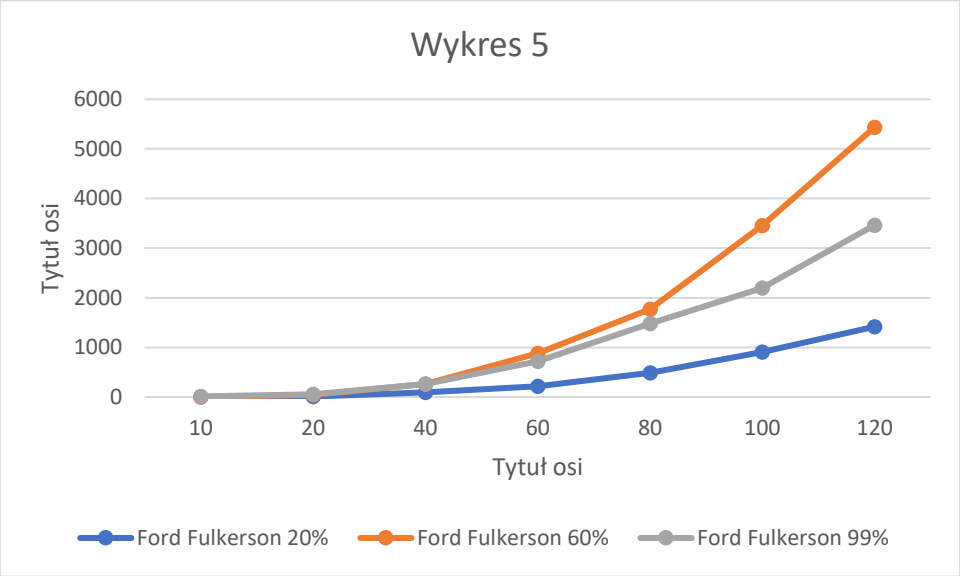
4.2.1.3 Algorytm najkrótszej ścieżki w grafie dla macierzy sąsiedztwa



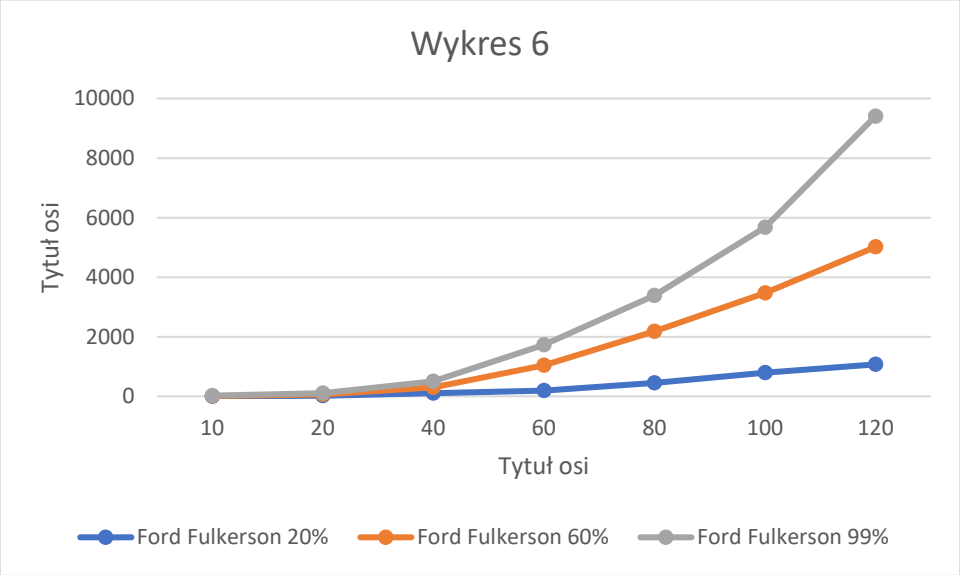
4.2.1.4 Algorytm najkrótszej ścieżki w grafie dla listy sąsiedztwa



4.2.1.5 *Wyznaczanie maksymalnego przepływu w grafie dla macierzy sąsiedztwa*

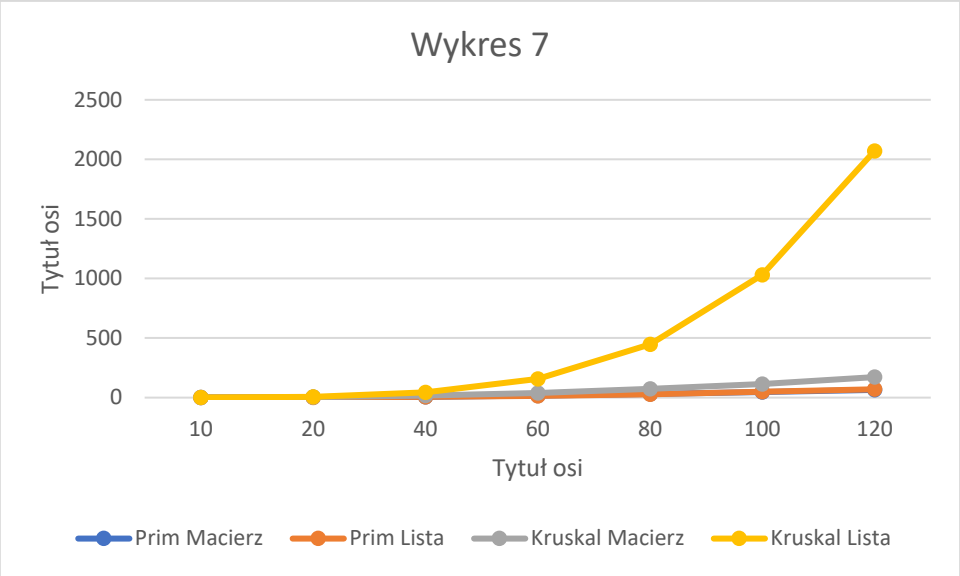


4.2.1.6 *Wyznaczanie maksymalnego przepływu w grafie dla listy sąsiedztwa*

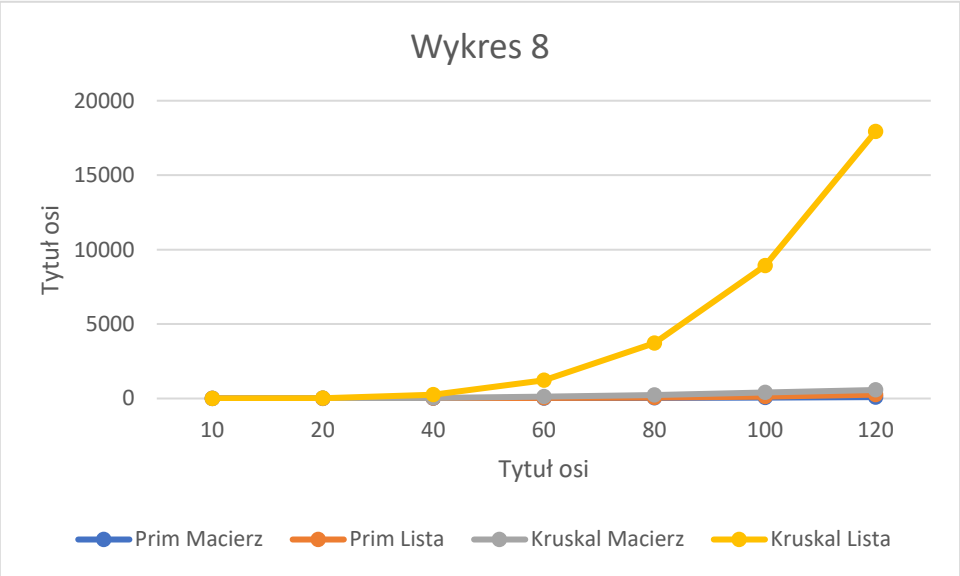


4.2.2 Wykresy zależności typu algorytmu i typu reprezentacji grafu

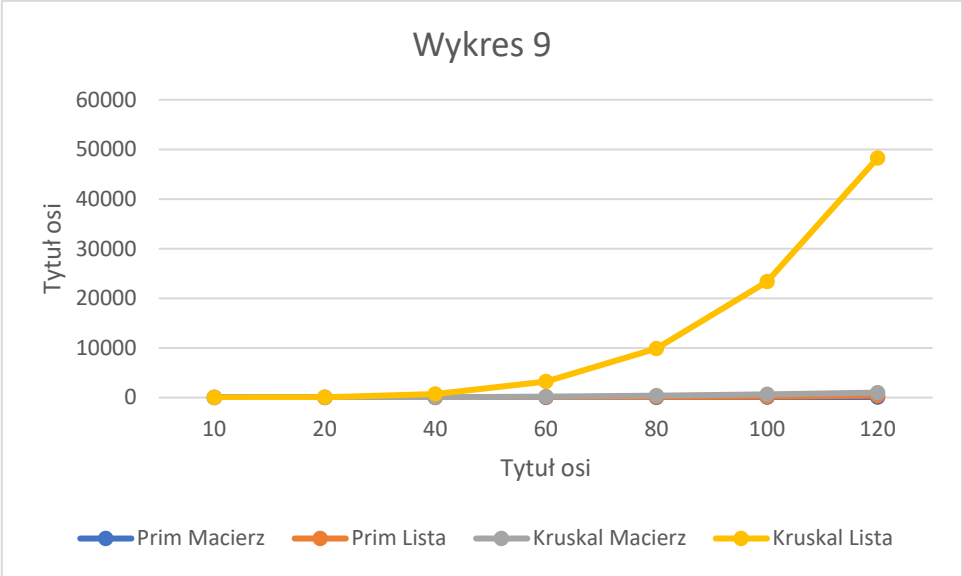
4.2.2.1 Algorytm minimalnego drzewa rozpinającego dla 20 % gęstości



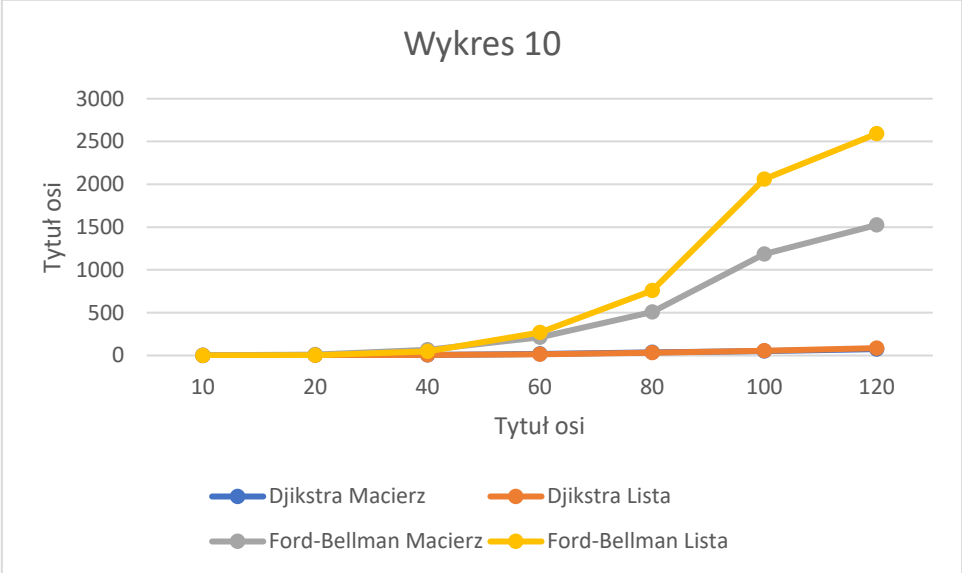
4.2.2.2 Algorytm minimalnego drzewa rozpinającego dla 60 % gęstości



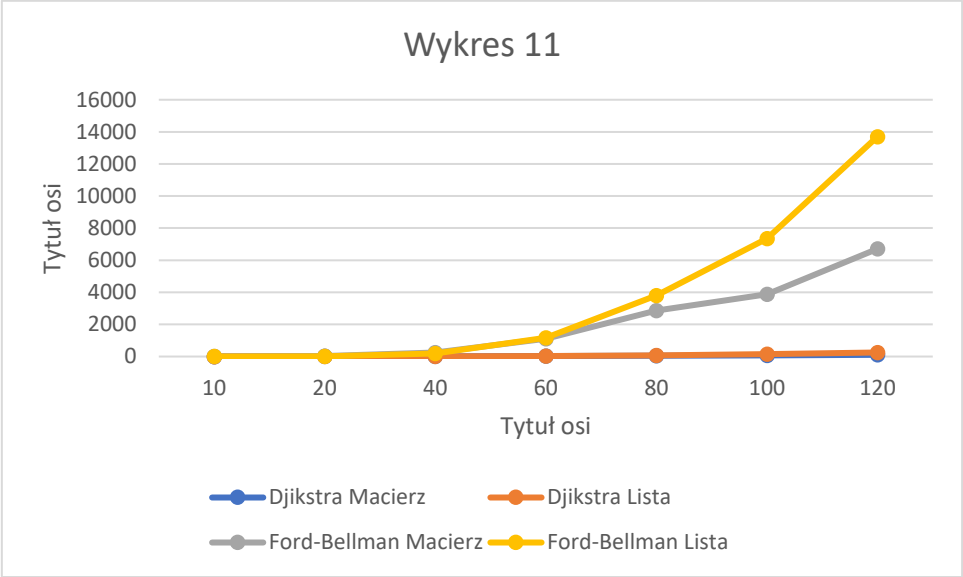
4.2.2.3 *Algorytm minimalnego drzewa rozpinającego dla 99 % gęstości*



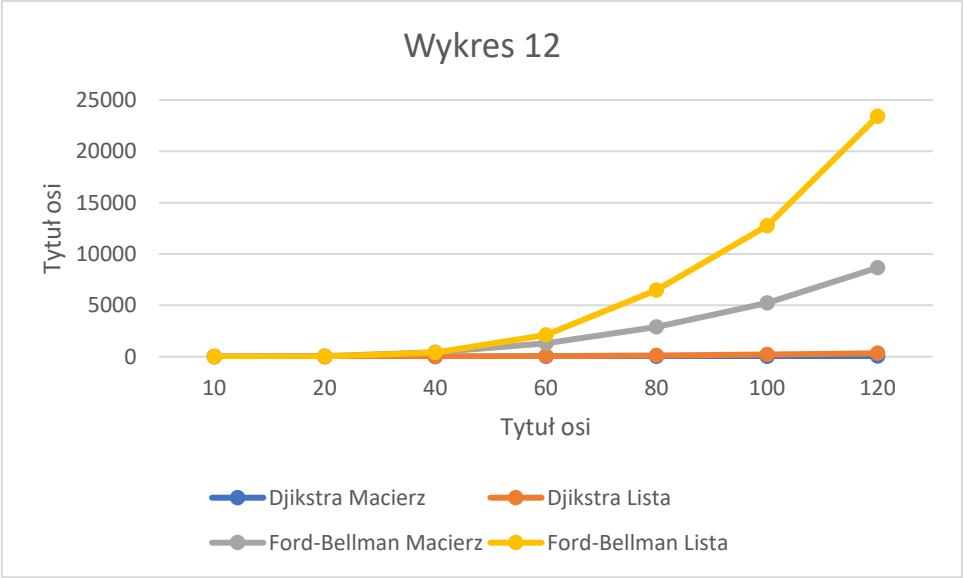
4.2.2.4 *Algorytm najkrótszej ścieżki w grafie dla 20 % gęstości*



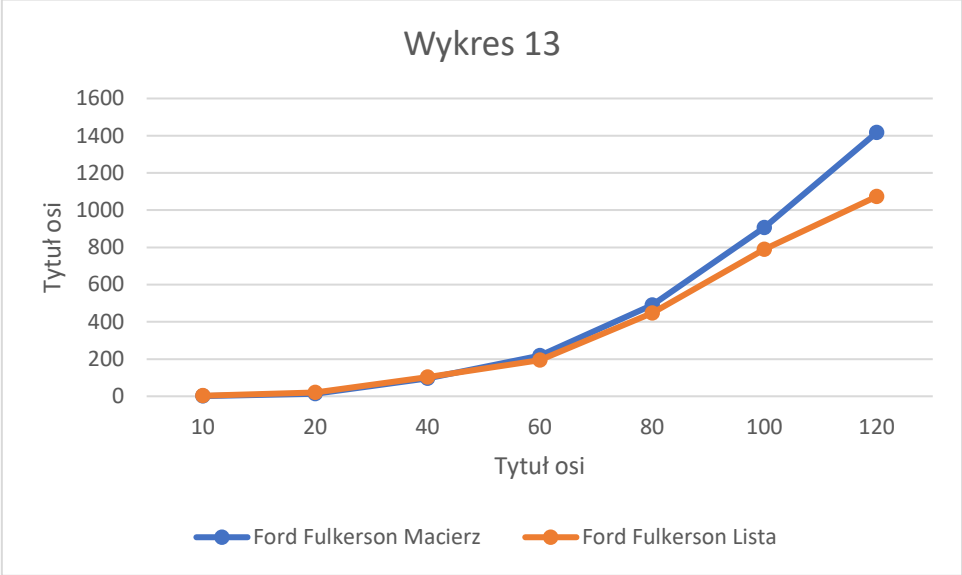
4.2.2.5 *Algorytm najkrótszej ścieżki w grafie dla 60 % gęstości*



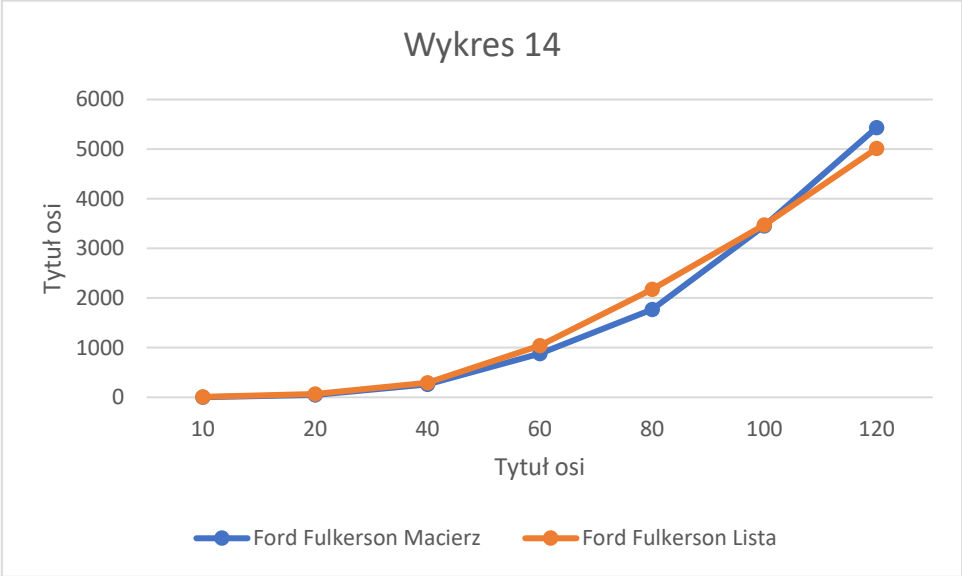
4.2.2.6 *Algorytm najkrótszej ścieżki w grafie dla 99% gęstości*



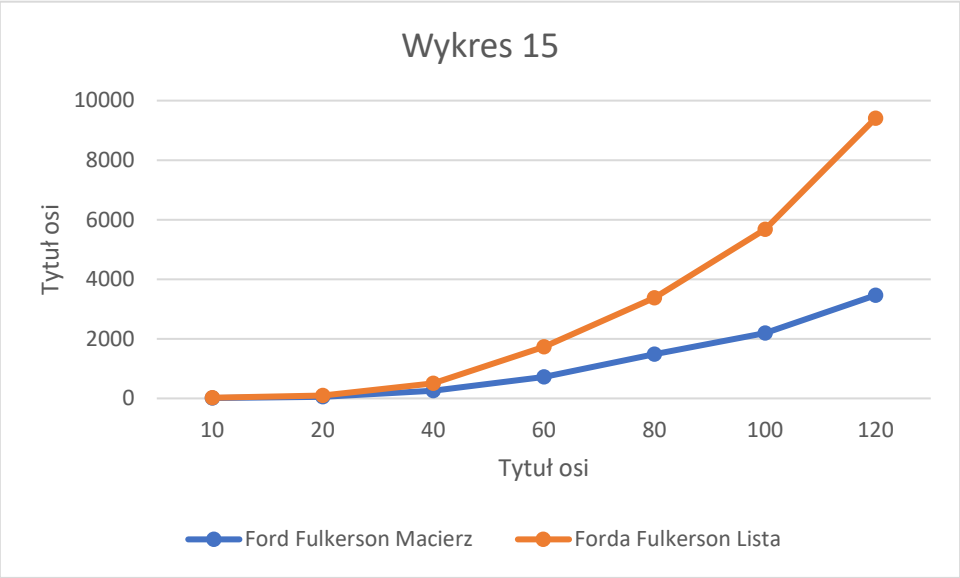
4.2.2.7 *Wyznaczanie maksymalnego przepływu w grafie dla 20% gęstości*



4.2.2.8 *Wyznaczanie maksymalnego przepływu w grafie dla 60% gęstości*



4.2.2.9 Wyznaczanie maksymalnego przepływu w grafie dla 99% gęstości



5 WNIOSKI

5.1 ALGORYTM MINIMALNEGO DRZEWA ROZPINAJĄCEGO

Algorytm Kruskala może nadawać się bardziej do bardzo prostych struktur. W tabelach 4.1.1 dla 10 wierzchołków i 20% gęstości algorytm ten ma lepsze wyniki w reprezentacji macierzowej, a w reprezentacji listowej jest delikatnie gorszy. Jednak w miarę wzrostu ilości wierzchołków oraz/lub krawędzi widać, że staje się o wiele mniej wydajny od algorytmu Prima, co widać na wykresie 4.2.1.1 i 4.2.1.2.

Reprezentacja macierzowa algorytmu Prima oraz Kruskala jest znacznie szybsza od reprezentacji listowej, co widać na wykresach 4.2.2.1, 4.2.2.2, 4.2.2.3. Znaczna różnica w algorytmie Kruskala wynika z trudności w dostępie do „krawędzi przeciwnej" (wynikającej z założenia co do listy sąsiedztwa, że krawędź nieskierowana jest reprezentowana przez dwie krawędzie skierowane, przeciwne), które trzeba pominąć podczas dodawania do kolejki priorytetowej. Różnica czasu dla algorytmu Prima może wynikać z tego, że dostęp do elementów list jest nieco trudniejszy, niż do elementów tablicy.

5.2 WYZNACZANIE NAJKRÓTSZEJ ŚCIEŻKI

W przypadku algorytmów wyznaczania najkrótszej ścieżki algorytm Djikstry, mimo zastosowania w algorytmie Forda-Bellmana dodatkowego uproszczenia w formie wyjścia z pętli głównej we wcześniejszym etapie, gdy nie ma poprawy, jest znacznie mniej wydajny, co można zobaczyć w tabelach 4.1.2 oraz na wykresach 4.2.1.3 i 4.2.1.4. Zaletą algorytmu Forda-Bellmana jest to, że może być użyty na grafach z wagami ujemnymi i nie dojdzie do przekłamania wag (algorytm Djikstry może pominąć pozornie dłuższą ścieżkę, na której końcu jest waga ujemna).

Dla algorytmu Djikstry nie ma znacznej różnicy między macierzą sąsiedztwa, a listą sąsiedztwa, co można zauważyć na wykresach 4.2.2.4, 4.2.2.5 i 4.2.2.6. Niewielka różnica może wynikać z nieco trudniejszego dostępu do elementów listy, niż elementów tablicy, podobnie jak przy algorytmie Prima. Jednak na tych samych wykresach. Algorytm Forda-Bellmana dla małych grafów pokazuję

przewagę reprezentacji listowej, jednak im większa ilość wierzchołków, tym większą przewagę zaczyna zyskiwać reprezentacja macierzowa.

5.3 ALGORYTM MAKSYMALNEGO PRZEPŁYWU

Reprezentacja macierzowa algorytmu Forda Fulkersona w postaci macierzy sąsiedztwa jest nieznacznie szybszą metodą w wyznaczaniu maksymalnego przepływu w odróżnieniu od reprezentacji listowej (drobne wyjątki widać dla małej gęstości przy większej ilości wierzchołków), co widać na tabelach 4.1.3, oraz na wykresach 4.2.2.8 i 4.2.2.9. Jednak odpowiednia implementacja w postaci Listy sąsiedztwa (która nie została użyta akurat w tym programie) w postaci przechowywania dodatkowych informacji na temat przepustowości i przepływu, co nie jest możliwe w reprezentacji macierzowej, gdzie trzeba utworzyć kilka struktur przechowujących dodatkowe dane.

5.4 ZATETY I WADY OBU REPREZENTACJI GRAFU

Reprezentacja macierzowa jest znacznie wygodniejsza w dostępie, jednak tracimy czas na sprawdzanie, czy dany element jest na pewno krawędzią. Dodatkową przewagą reprezentacji listowej jest łatwy dostęp do listy krawędzi, wszystkie krawędzie można znaleźć sprawdzając kolejne elementy tablicy i listy w danej komórce. Jednak wadą może być trudniejsze sprawdzenie, czy w reprezentacji listowej istnieje dana krawędź. Operacja taka zmusiłaby nas do sprawdzenia w najgorszym wypadku wszystkich elementów listy danej komórki tablicy list. Dla reprezentacji macierzowej wystarczy jedynie sprawdzić czy dana kolumna i wiersz tablicy są różne od 0.