Politechnika
Wrocławska

Algorytmy i złożoność obliczeniowa	
Temat: Badanie efektywności algorytmów grafowych w zależności od rozmiaru instancji oraz sposobu reprezentacji grafu w pamięci komputera	
Osoba wykonująca projekt: Mateusz Bukowski	
Termin zajęć: Czwartek (TP) 13:15-15:00	**Numer grupy:** 2

Spis treści

Wprowadzenie

Podczas przeprowadzania eksperymentu badane są **cztery algorytmy grafowe**:

- dwa algorytmy wyznaczania **najkrótszej ścieżki** w grafie:

  - Algorytm Dijkstry
  - Algorytm Forda-Bellmana

- dwa algorytmy wyznaczania **minimalnego drzewa rozpinającego** (MST):

  - Algorytm Prima
  - Algorytm Kruskala

Opis badanych algorytmów wraz z ich złożonościami obliczeniowymi:

1. Algorytm Dijkstry

Algorytm Dijkstry pozwala na znalezienie najkrótszej ścieżki między dwoma wybranymi wierzchołkami. Algorytm ten zakłada, że wszystkie krawędzie grafu są nieujemne. Algorytm Dijkstry wykorzystuje zachłanne podejście. W każdej iteracji wybierany jest wierzchołek z najkrótszą odległością dojścia, a następnie aktualizowany jest koszt dojścia do jego sąsiadów. Proces ten jest kontynuowany, aż wszystkie wierzchołki zostaną odwiedzone i koszt dojścia do każdego z nich zostanie ustalony jako najkrótszy możliwy. Złożoność obliczeniowa jest zależna od reprezentacji grafu w pamięci komputera. Jeśli graf jest zapisany w postaci macierzy incydencji to algorytm będzie miał złożoność obliczeniową $\boldsymbol{O}(\boldsymbol{V^3})$, natomiast jeśli graf jest zapisany w postaci listy sąsiedztwa, złożoność obliczeniowa algorytmu wyniesie $\boldsymbol{O}(\boldsymbol{V^2})$.

2. Algorytm Bellmana-Forda

Algorytm Bellmana-Forda pozwala na znalezienie najkrótszej ścieżki między dwoma wybranymi wierzchołkami. Algorytm ten umożliwia korzystanie z ujemnych wag krawędzi w grafie. Algorytm Bellmana-Forda opiera się na metodzie relaksacji krawędzi. Występuje V-1 iteracji, w której każda z krawędzi ulega relaksacji i koszty dojścia do danych wierzchołków są ustawiane na najkrótsze możliwe. Jeśli jednak podczas pojedynczej iteracji nie nastąpiła żadna korekta kosztów dojścia do wierzchołków, algorytm można szybciej zakończyć. Złożoność obliczeniowa algorytmu Bellmana-Forda wynosi $\boldsymbol{O}(\boldsymbol{V} * \boldsymbol{E})$.

3. Algorytm Prima

Algorytm Prima pozwala na znalezienie minimalnego drzewa rozpinającego (MST) badanego grafu. Algorytm Prima wykorzystuje zachłanne podejście. W każdej iteracji wybierany jest wierzchołek z najmniejszą wartością parametru key (waga najkrótszej krawędzi, która dochodzi do danego wierzchołka), a następnie aktualizowany jest parametr key każdego z jego sąsiadów. Proces ten jest kontynuowany, aż wszystkie wierzchołki zostaną odwiedzone i parametr key każdego z nich zostanie ustawiony jako najkrótszy możliwy. Złożoność obliczeniowa jest zależna od reprezentacji grafu w pamięci komputera. Jeśli graf jest zapisany w postaci macierzy incydencji to algorytm będzie miał złożoność obliczeniową $\boldsymbol{O}(\boldsymbol{V^3})$, natomiast jeśli graf jest zapisany w postaci listy sąsiedztwa, złożoność obliczeniowa algorytmu wyniesie $\boldsymbol{O}(\boldsymbol{V^2})$.

4. Algorytm Kruskala

Algorytm Kruskala pozwala na znalezienie minimalnego drzewa rozpinającego (MST) badanego grafu. W algorytmie tym, na początku sortowane są wszystkie krawędzie według wag, a następnie idąc po każdej krawędzi sprawdza się czy należące do niej wierzchołki należą do różnych poddrzew. Jeśli tak to krawędź dodawana jest do zbioru krawędzi MST, a dwa poddrzewa do których należały sprawdzane wierzchołki łączą się w jedno poddrzewo. Dalej sprawdzana jest reszta pozostałych krawędzi aż do wyczerpania zbioru albo aż zbiór krawędzi MST będzie zawierał V-1 krawędzi. Złożoność obliczeniowa Algorytmu Kruskala wynosi $\boldsymbol{O}(\boldsymbol{E}\,\mathbf{log}\,\boldsymbol{E})$.

Gdzie:

V – liczba wierzchołków grafu

E – liczba krawędzi grafu

Plan eksperymentu

Algorytmy grafowe

Podczas przeprowadzania eksperymentu badane są **cztery algorytmy grafowe**:

- dwa algorytmy wyznaczania **najkrótszej ścieżki** w grafie:

  - Algorytm Dijkstry
  - Algorytm Forda-Bellmana

- dwa algorytmy wyznaczania **minimalnego drzewa rozpinającego** (MST):

  - Algorytm Prima
  - Algorytm Kruskala

Każdy algorytm badany jest dla **dwóch różnych implementacji reprezentacji grafu** w pamięci komputera:

- reprezentacji macierzowej (**macierz incydencji**)
- reprezentacji listowej (**lista następników/poprzedników/sąsiedztwa**)

Rodzaje badanych grafów

Eksperyment badający algorytmy grafowe jest przeprowadzany dla siedmiu różnych rozmiarów grafów:

- grafy składające się z **10** wierzchołków
- grafy składające się z **20** wierzchołków
- grafy składające się z **40** wierzchołków
- grafy składające się z **50** wierzchołków
- grafy składające się z **80** wierzchołków
- grafy składające się z **100** wierzchołków
- grafy składające się z **200** wierzchołków

Dodatkowo sprawdzany jest wpływ **gęstości grafu** (stosunek liczby krawędzi grafu do liczby krawędzi grafu pełnego) dla każdej wielkości grafu. Algorytmy są badane dla grafów, których gęstości wynoszą:

- **25%**
- **50%**
- **99%**

Grafy tworzone są w sposób losowy. Na początku dla ilości wierzchołków podanych przez użytkownika generowane jest losowe drzewo rozpinające, a następnie dla podanej przez użytkownika gęstości grafu generowane są kolejne losowe, niepowtarzające się krawędzie.

Każdy algorytm grafowy jest badany dla dwóch różnych implementacji reprezentacji grafu w pamięci komputera oraz dla każdej wielkości oraz gęstości grafu, poprzez **50-krotne** wywołanie sprawdzanego algorytmu dla każdego z wygenerowanych grafów.

Badanie algorytmów polega na mierzeniu czasu wykonywania się algorytmu dla konkretnego grafu i konkretnej jego reprezentacji w pamięci komputera, powtórzenia tego procesu 50-krotnie, a następnie wyliczeniu średniej zmierzonych czasów.
Czasy mierzone są za pomocą funkcji **QueryPerformanceCounter** (od rozpoczęcia algorytmu do zakończenia algorytmu, nie licząc czasu generowania grafów).

Przebieg eksperymentu

Na początku odpowiednio generowany jest graf w zależności od badanej wielkości oraz gęstości grafu. Następnie dla stworzonego grafu w zależności od badanego problemu (MST lub najkrótsza ścieżka) wywoływany jest każdy algorytm po kolei dla każdej z reprezentacji grafu w pamięci komputera i mierzony jest czas działania algorytmu. Pomiary powtarzane są 50-krotnie, czyli badane jest 50 różnych grafów o zadanych parametrach. Na koniec dla każdego algorytmu wyliczany jest jego średni czas wykonywania się. Proces ten powtarzany jest dla każdej badanej wielkości i gęstości grafu.

Dodatkowe informacje:

Kod źródłowy algorytmów grafowych wraz z menu został napisany obiektowo w języku C++.

Wyniki

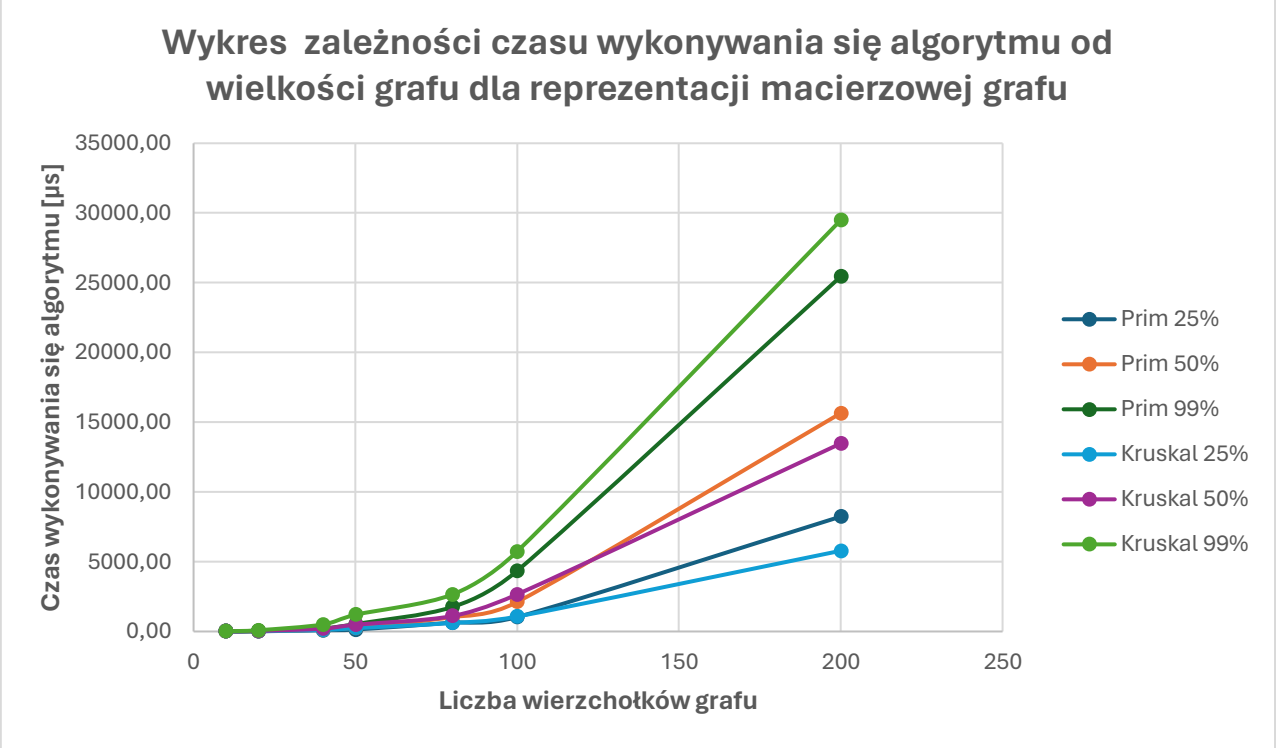
Wyznaczanie minimalnego drzewa rozpinającego (MST)

Porównanie algorytmu Prima i Kruskala dla dwóch różnych reprezentacji grafu w pamięci komputera

Reprezentacja macierzowa

Tabela 1: Tabela średnich czasów wykonywania się algorytmów wyznaczających MST dla badanych wielkości i gęstości grafów – reprezentacja macierzowa grafu

Średnie czasy [µs] wykonywania się algorytmów w zależności od gęstości i wielkości grafu							
Reprezentacja macierzowa grafu							
Algorytm-Gęstość\Liczba wierzchołków	**10**	**20**	**40**	**50**	**80**	**100**	**200**
Prim 25%	1,61	11,58	86,01	136,64	635,44	1031,95	8237,20
Prim 50%	1,86	15,58	131,49	274,91	1036,89	2114,62	15650,80
Prim 99%	3,13	29,13	223,80	532,88	1755,48	4345,61	25451,81
Kruskal 25%	3,14	18,77	109,04	235,31	610,24	1070,18	5783,98
Kruskal 50%	6,84	38,54	203,69	481,60	1104,12	2660,66	13479,67
Kruskal 99%	12,65	89,61	497,82	1194,98	2649,98	5737,95	29494,00

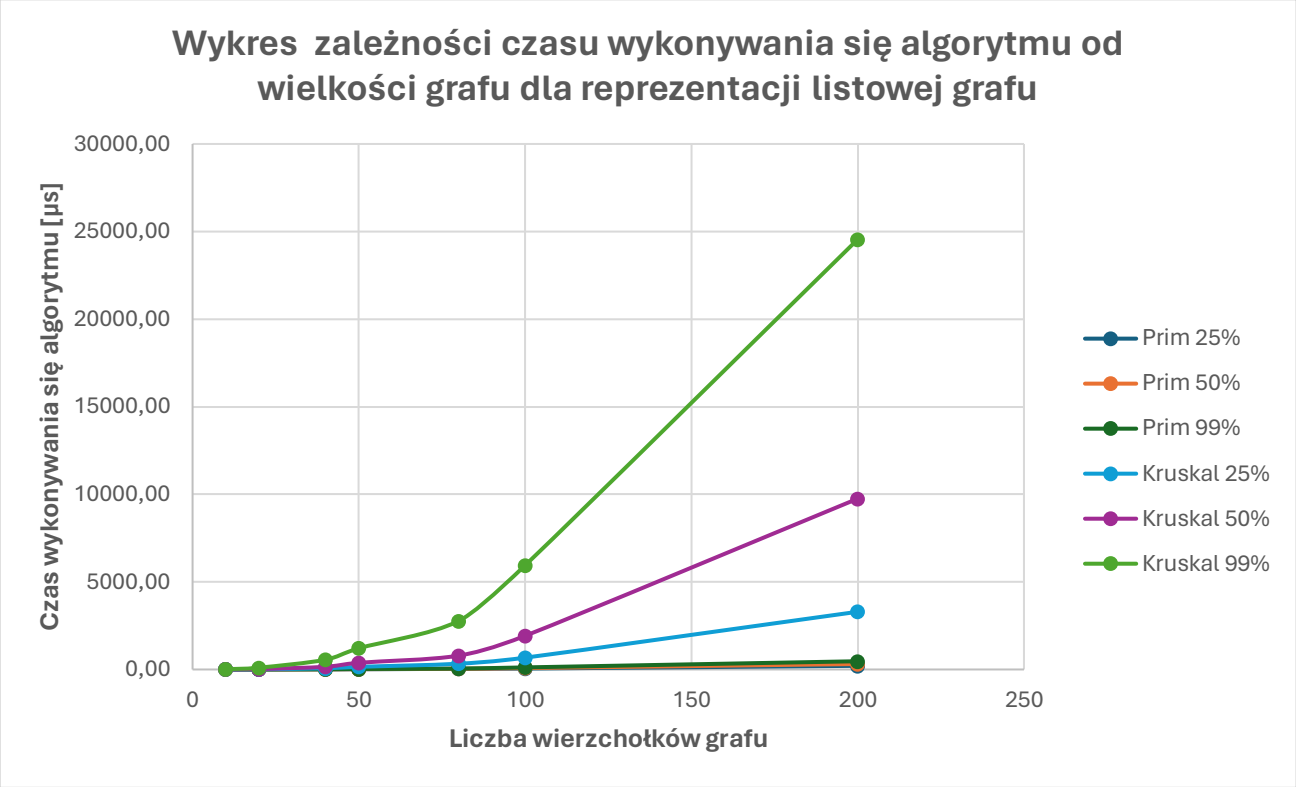


Rysunek 1: Wykres zależności czasu wykonywania się algorytmu wyznaczającego MST od wielkości grafu dla reprezentacji macierzowej grafu

Reprezentacja listowa

Tabela 2: Tabela średnich czasów wykonywania się algorytmów wyznaczających MST dla badanych wielkości i gęstości grafów – reprezentacja listowa grafu

Średnie czasy [µs] wykonywania się algorytmów w zależności od gęstości i wielkości grafu							
Reprezentacja listowa grafu							
Algorytm-Gęstość\Liczba wierzchołków	10	20	40	50	80	100	200
Prim 25%	0,63	2,13	8,96	12,65	38,38	53,76	209,05
Prim 50%	0,59	2,73	10,33	20,58	41,38	79,99	312,00
Prim 99%	0,70	2,63	11,68	30,34	50,83	118,69	471,52
Kruskal 25%	2,36	11,87	67,99	151,28	329,74	674,84	3293,98
Kruskal 50%	5,16	27,97	154,04	382,25	788,72	1921,99	9751,52
Kruskal 99%	15,46	104,98	557,90	1211,13	2765,27	5945,07	24551,52



Rysunek 2: Wykres zależności czasu wykonywania się algorytmu wyznaczającego MST od wielkości grafu dla reprezentacji listowej grafu

Wnioski

Analizując powyższe tabele wyników oraz wykresy, można zauważyć ogromną przewagę operowania na reprezentacji listowej grafu w porównaniu z reprezentacją macierzową grafu. Różnicę szczególnie widać dla algorytmu Prima, w którym dla reprezentacji listowej czasy wykonywania się algorytmu są kilkunastokrotnie krótsze w porównaniu z reprezentacją macierzową. Dla algorytmu Kruskala różnice także są widoczne, jednak są zdecydowanie mniejsze, co pokazuje ogromną przewagę wykorzystywania algorytmu Prima, gdy operuje się na grafach zapisanych w reprezentacji listowej.

Dla reprezentacji macierzowej, porównując wyniki pomiarów dla gęstości grafu 25% oraz 50% można zauważać, że dla większych grafów (powyżej 100-150 wierzchołków) algorytm Kruskala jest wydajniejszy niż algorytm Prima, natomiast dla gęstości 99% algorytm Kruskala jest mniej wydajny w porównaniu z algorytmem Prima. Można więc wystawić wniosek, że algorytm Prima jest bardziej efektywny dla gęstych grafów, natomiast algorytm Kruskala sprawdza się lepiej w grafach rzadkich.

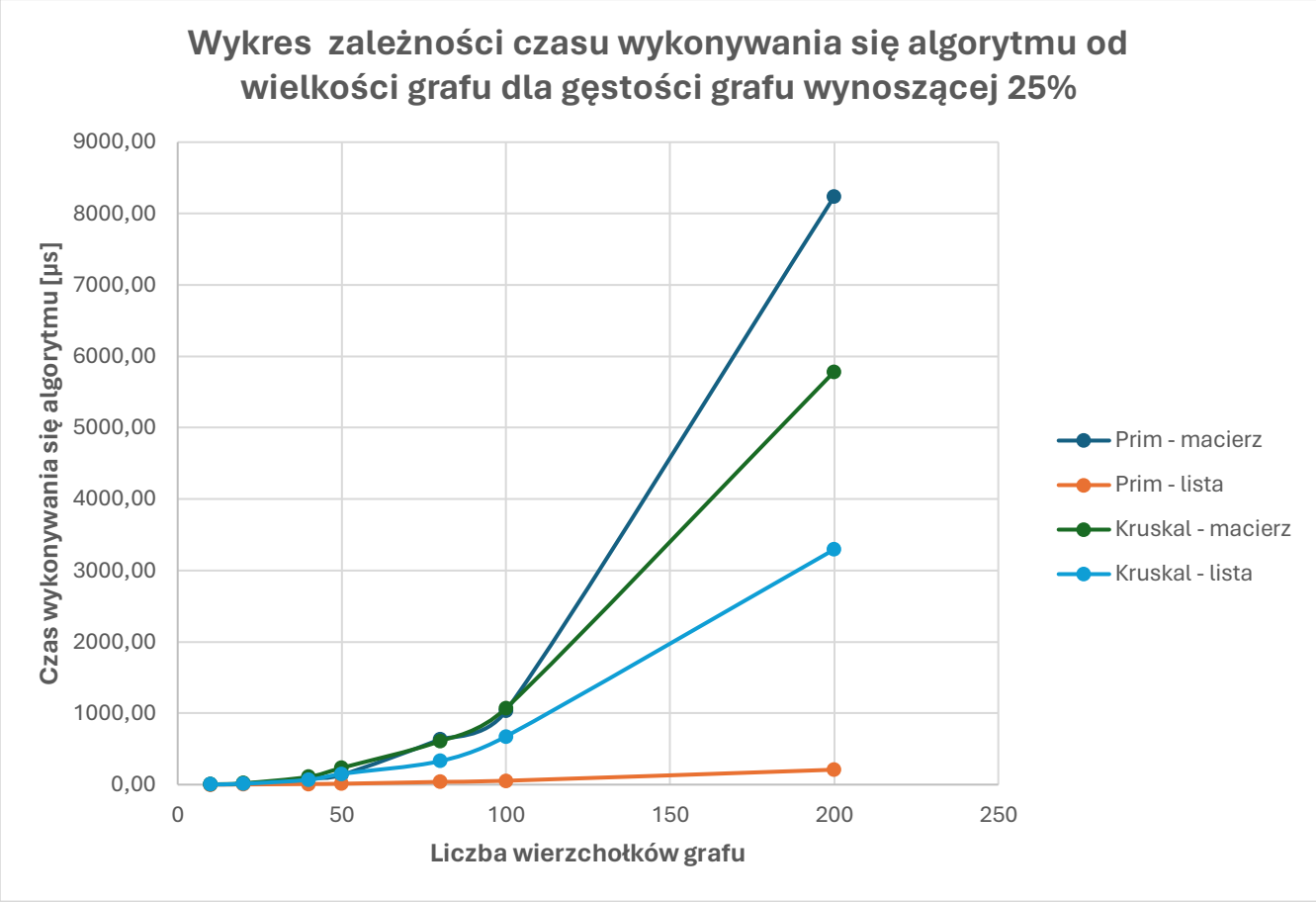
Zgodnie z oczekiwaniami, dla każdego algorytmu, czym większa gęstość badanego grafu, tym algorytm wykonuje się dłuższy czas.

Porównanie algorytmu Prima i Kruskala dla trzech różnych gęstości grafu

Gęstość grafu - 25%

Tabela 3: Tabela średnich czasów wykonywania się algorytmów wyznaczających MST dla badanych wielkości i reprezentacji grafów – gęstość grafu 25%

Średnie czasy [µs] wykonywania się algorytmów w zależności od reprezentacji i wielkości grafu							
Gęstość grafu: 25%							
Algorytm-Reprzentacja\Liczba wierzchołków	**10**	**20**	**40**	**50**	**80**	**100**	**200**
Prim - macierz	1,61	11,58	86,01	136,64	635,44	1031,95	8237,20
Prim - lista	0,63	2,13	8,96	12,65	38,38	53,76	209,05
Kruskal - macierz	3,14	18,77	109,04	235,31	610,24	1070,18	5783,98
Kruskal - lista	2,36	11,87	67,99	151,28	329,74	674,84	3293,98

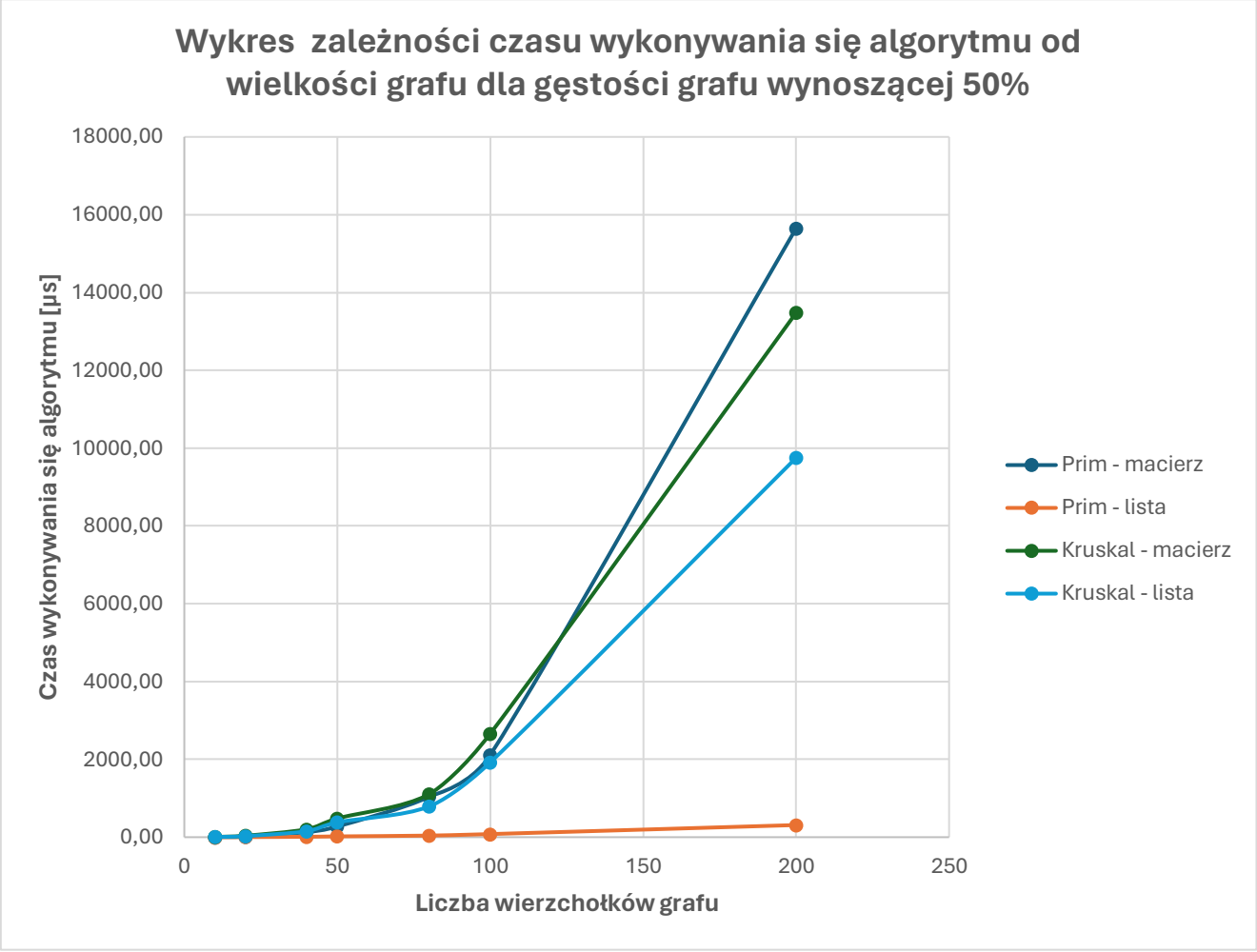


Rysunek 3: Wykres zależności czasu wykonywania się algorytmu wyznaczającego MST od wielkości grafu dla gęstości grafu 25%

Gęstość grafu - 50%

Tabela 4: Tabela średnich czasów wykonywania się algorytmów wyznaczających MST dla badanych wielkości i reprezentacji grafów – gęstość grafu 50%

Średnie czasy [µs] wykonywania się algorytmów w zależności od reprezentacji i wielkości grafu							
Gęstość grafu: 50%							
Algorytm-Reprzentacja\Liczba wierzchołków	10	20	40	50	80	100	200
Prim - macierz	1,86	15,58	131,49	274,91	1036,89	2114,62	15650,80
Prim - lista	0,59	2,73	10,33	20,58	41,38	79,99	312,00
Kruskal - macierz	6,84	38,54	203,69	481,60	1104,12	2660,66	13479,67
Kruskal - lista	5,16	27,97	154,04	382,25	788,72	1921,99	9751,52

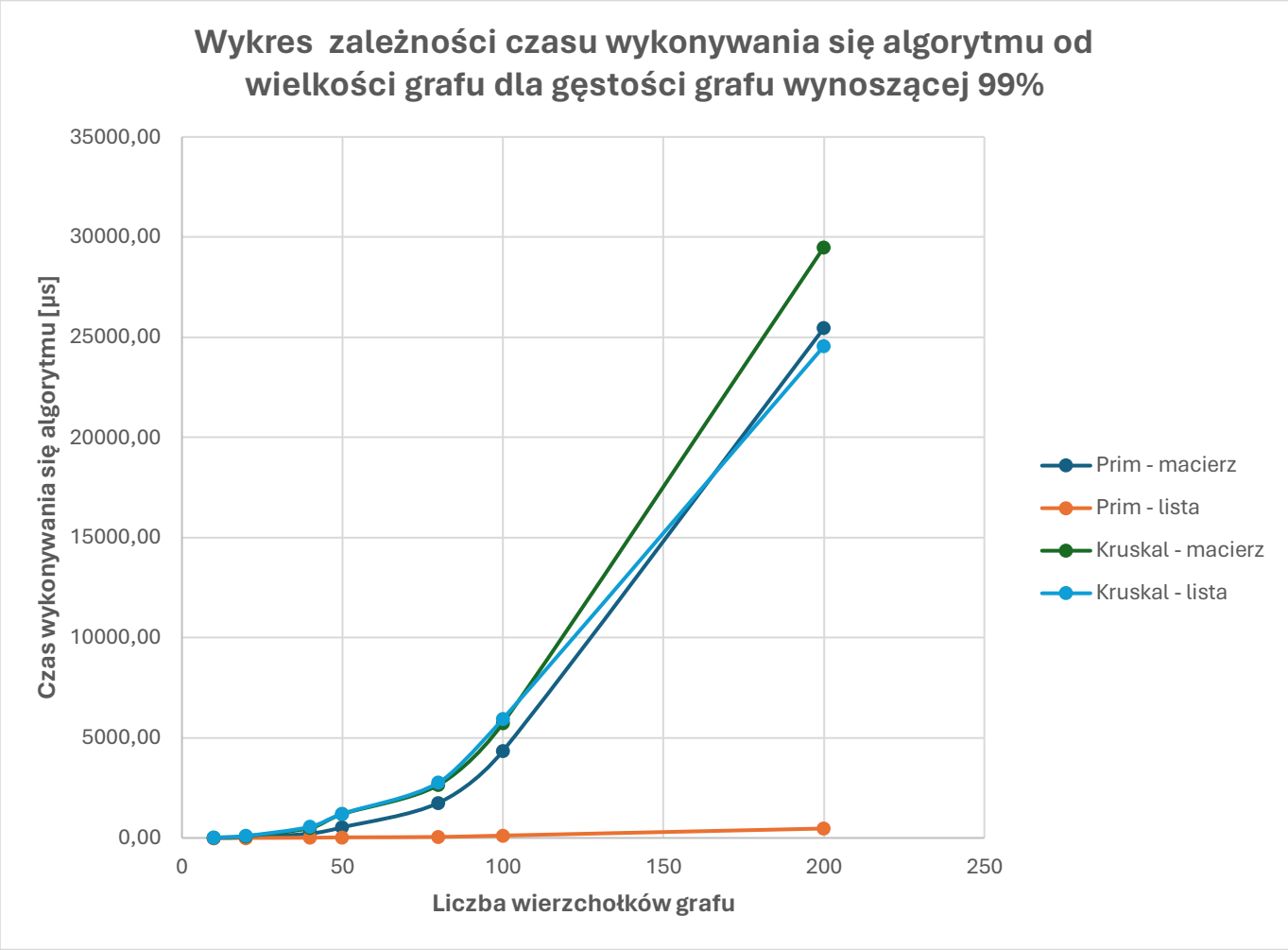


Rysunek 4: Wykres zależności czasu wykonywania się algorytmu wyznaczającego MST od wielkości grafu dla gęstości grafu 50%

Gęstość grafu - 99%

Tabela 5: Tabela średnich czasów wykonywania się algorytmów wyznaczających MST dla badanych wielkości i reprezentacji grafów – gęstość grafu 99%

Średnie czasy [µs] wykonywania się algorytmów w zależności od reprezentacji i wielkości grafu							
Gęstość grafu: 99%							
Algorytm-Reprzentacja\Liczba wierzchołków	10	20	40	50	80	100	200
Prim - macierz	3,13	29,13	223,80	532,88	1755,48	4345,61	25451,81
Prim - lista	0,70	2,63	11,68	30,34	50,83	118,69	471,52
Kruskal - macierz	12,65	89,61	497,82	1194,98	2649,98	5737,95	29494,00
Kruskal - lista	15,46	104,98	557,90	1211,13	2765,27	5945,07	24551,52



Rysunek 5: Wykres zależności czasu wykonywania się algorytmu wyznaczającego MST od wielkości grafu dla gęstości grafu 99%

Wnioski

Analizując powyższe tabele wyników oraz wykresy, można zauważyć, że dla każdej gęstości grafu dla listowej reprezentacji, oba algorytmy są znacznie wydajniejsze niż dla macierzowej reprezentacji.

Najszybszym algorytmem jest algorytm Prima dla reprezentacji listowej grafu. Dla każdej gęstości grafu, czas jego wykonywania się jest wielokrotnie krótszy od reszty algorytmów.

Dla grafów rzadkich (25% i 50%) zapisanych w pamięci komputera w reprezentacji macierzowej, algorytm Kruskala jest efektywniejszym algorytmem, natomiast dla grafów gęstych (99%) efektywniejszy jest algorytm Prima.

Zgodnie z oczekiwaniami, dla każdego algorytmu, czym większa gęstość badanego grafu, tym algorytm wykonuje się dłuższy czas.

Dla algorytmu Prima dla reprezentacji listowej, zależność między średnim czasem wykonywania się algorytmu a wielkością grafu jest kwadratowa. Jest to zgodne z teoretyczną złożonością obliczeniową – $\boldsymbol{O}(\boldsymbol{V^2})$.

Dla algorytmu Prima dla reprezentacji macierzowej, zależność między średnim czasem wykonywania się algorytmu a wielkością grafu jest sześcienna. Wyniki te zgodne są z teoretyczną złożonością obliczeniową – $\boldsymbol{O}(\boldsymbol{V^3})$.

Dla algorytmu Kruskala dla reprezentacji macierzowej oraz listowej, zależność między średnim czasem wykonywania się algorytmu a wielkością grafu jest w przybliżeniu liniowo logarytmiczna, co zgadza się teoretyczną złożonością obliczeniową – $\boldsymbol{O}(\boldsymbol{E}\log\boldsymbol{E})$.

Podsumowanie

Podczas wyznaczania minimalnego drzewa rozpinającego (MST), operowanie na grafach zapisanych w pamięci komputera w reprezentacji listowej jest znacznie korzystniejsze w porównaniu z operowaniem na grafach zapisanych w pamięci komputera w reprezentacji macierzowej.

Najwydajniejszym algorytmem szukającym minimalnego drzewa rozpinającego jest algorytm Prima dla reprezentacji listowej grafu. Jeśli jednak trzeba będzie operować na reprezentacji macierzowej grafu, dla grafów rzadkich warto skorzystać z algorytmu Kruskala, który jest efektywniejszy dla tych gęstości dla większych grafów. Natomiast dla grafów gęstych, warto skorzystać z algorytmu Prima, ponieważ charakteryzuje się on większą efektywnością dla tego rodzaju grafów.

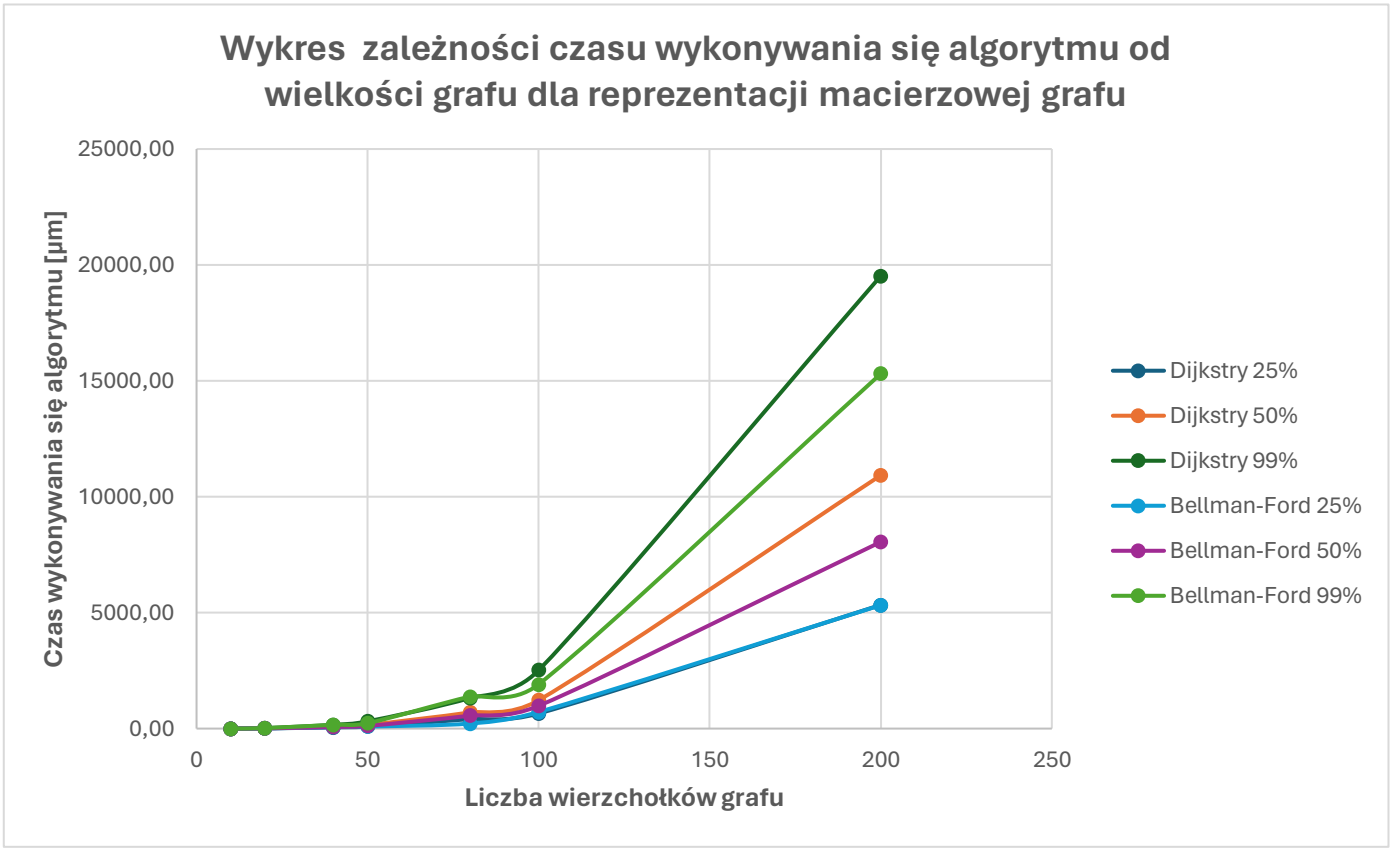
Wyznaczanie najkrótszej ścieżki w grafie

Porównanie algorytmu Dijkstry i Bellmana-Forda dla dwóch różnych reprezentacji grafu w pamięci komputera

Reprezentacja macierzowa

Tabela 6: Tabela średnich czasów wykonywania się algorytmów wyznaczających najkrótszą ścieżkę w grafie dla badanych wielkości i gęstości grafów – reprezentacja macierzowa grafu

Średnie czasy [µs] wykonywania się algorytmów w zależności od gęstości i wielkości grafu							
Reprezentacja macierzowa grafu							
Algorytm-Gęstość\Liczba wierzchołków	**10**	**20**	**40**	**50**	**80**	**100**	**200**
Dijkstry 25%	0,76	6,13	53,64	87,95	409,55	650,29	5333,55
Dijkstry 50%	1,51	12,01	91,45	160,27	698,22	1224,08	10926,15
Dijkstry 99%	2,49	21,44	163,29	312,82	1324,53	2519,04	19524,23
Bellman-Ford 25%	0,65	6,52	63,57	87,67	219,00	715,05	5323,04
Bellman-Ford 50%	1,31	9,06	68,04	117,17	557,71	980,66	8052,02
Bellman-Ford 99%	2,16	21,51	162,94	233,51	1381,59	1909,75	15305,10

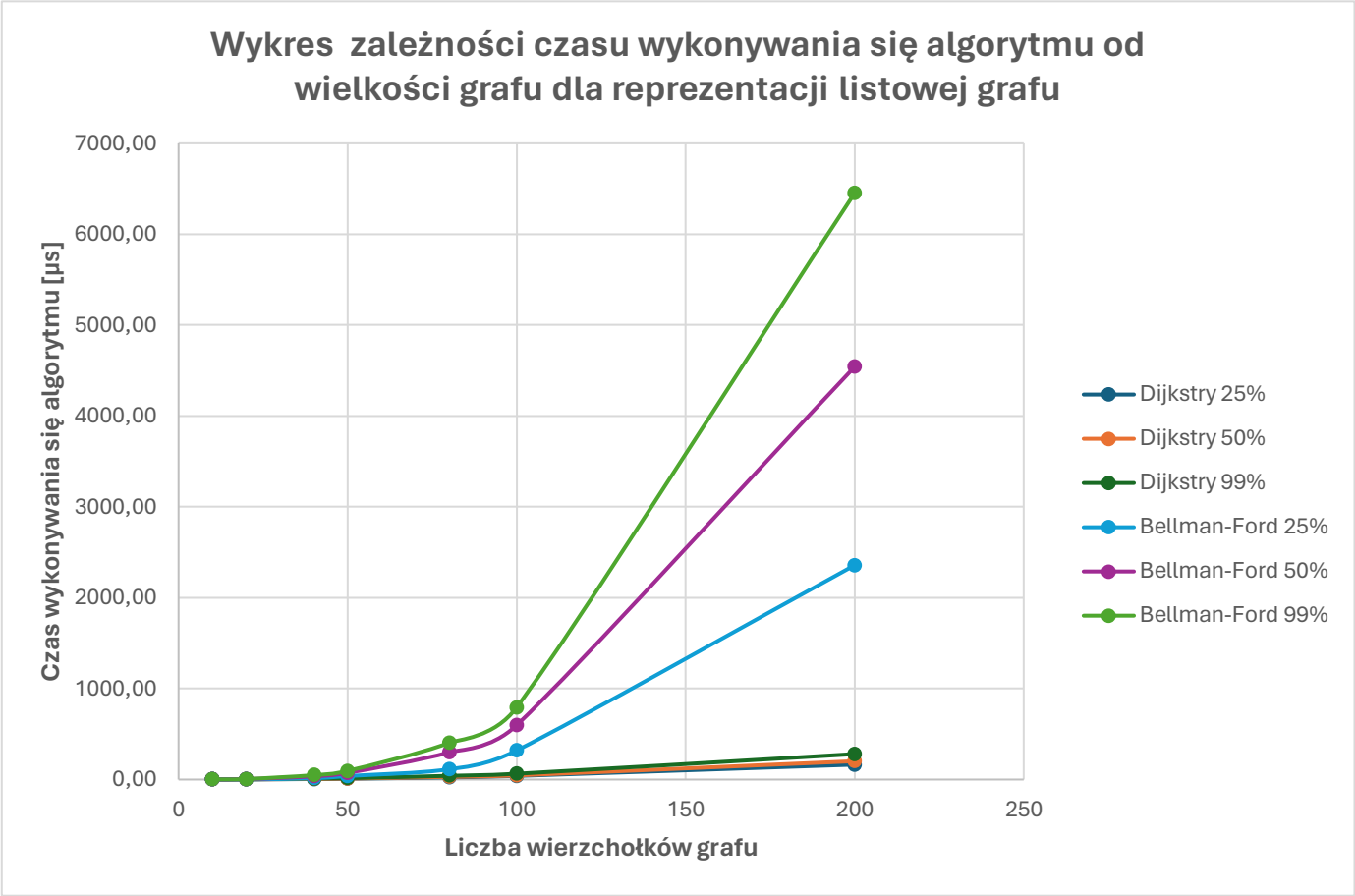


Rysunek 6: Wykres zależności czasu wykonywania się algorytmu wyznaczającego najkrótszą ścieżkę w grafie od wielkości grafu dla reprezentacji macierzowej grafu

Reprezentacja listowa

Tabela 7: Tabela średnich czasów wykonywania się algorytmów wyznaczających najkrótszą ścieżkę w grafie dla badanych wielkości i gęstości grafów – reprezentacja listowa grafu

Średnie czasy [µs] wykonywania się algorytmów w zależności od gęstości i wielkości grafu							
Reprezentacja listowa grafu							
Algorytm-Gęstość\Liczba wierzchołków	10	20	40	50	80	100	200
Dijkstry 25%	0,42	1,06	5,71	9,09	24,11	40,21	164,20
Dijkstry 50%	0,56	1,68	6,94	10,94	29,32	45,42	201,88
Dijkstry 99%	0,62	3,05	11,86	14,93	41,17	63,72	280,33
Bellman-Ford 25%	0,15	1,56	14,19	38,76	113,07	319,56	2358,38
Bellman-Ford 50%	0,53	4,22	35,31	73,50	299,01	597,25	4540,94
Bellman-Ford 99%	0,74	5,70	48,22	95,55	403,23	793,54	6452,22



Rysunek 7: Wykres zależności czasu wykonywania się algorytmu wyznaczającego najkrótszą ścieżkę w grafie od wielkości grafu dla reprezentacji listowej grafu

Wnioski

Analizując powyższe tabele wyników oraz wykresy, można zauważyć ogromną przewagę operowania na reprezentacji listowej grafu w porównaniu z reprezentacją macierzową grafu. Różnicę szczególnie widać dla algorytmu Dijkstry, w którym dla reprezentacji listowej czasy wykonywania się algorytmu są kilkunastokrotnie krótsze w porównaniu z reprezentacją macierzową. Dla algorytmu Bellmana-Forda różnice także są widoczne, jednak są zdecydowanie mniejsze, co pokazuje ogromną przewagę wykorzystywania algorytmu Dijkstry, gdy operuje się na grafach zapisanych w reprezentacji listowej.

Dla reprezentacji macierzowej, wydajniejszym algorytmem wyznaczającym najkrótszą ścieżkę w grafie jest algorytm Bellmana-Forda. Dla gęstości grafu 25% algorytm Dijkstry oraz Bellmana-Forda osiągają bardzo zbliżone do siebie wyniki, jednak dla większych gęstości, algorytm Bellmana-Forda staje się efektywniejszy.

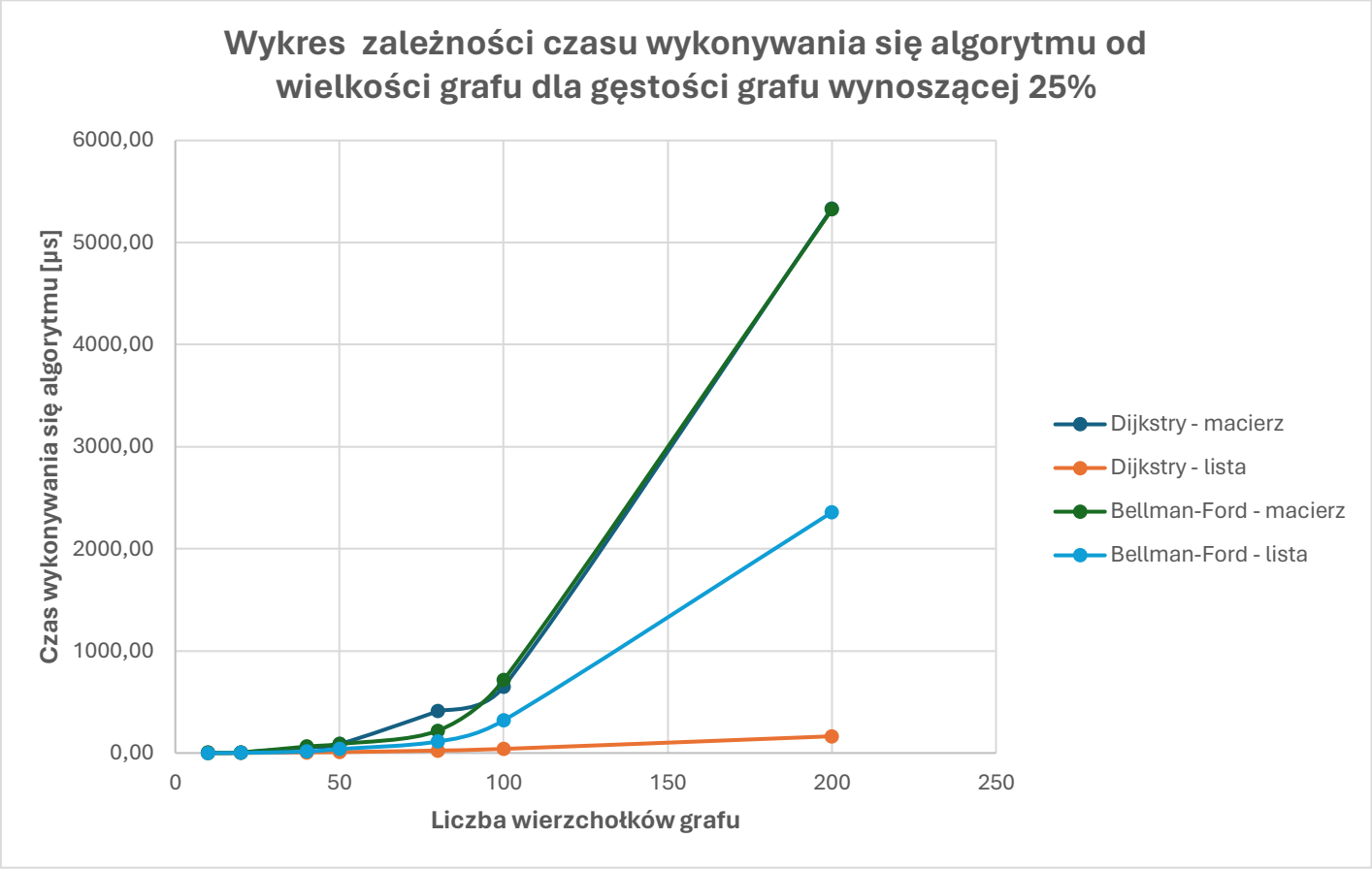
Zgodnie z oczekiwaniami, dla każdego algorytmu, czym większa gęstość badanego grafu, tym algorytm wykonuje się dłuższy czas.

Porównanie algorytmu Dijkstry i Bellmana-Forda dla trzech różnych gęstości grafu

Gęstość grafu - 25%

Tabela 8: Tabela średnich czasów wykonywania się algorytmów wyznaczających najkrótszą ścieżkę w grafie dla badanych wielkości i reprezentacji grafów – gęstość grafu 25%

Średnie czasy [µs] wykonywania się algorytmów w zależności od reprezentacji i wielkości grafu							
Gęstość grafu: 25%							
Algorytm-Reprzentacja\Liczba wierzchołków	**10**	**20**	**40**	**50**	**80**	**100**	**200**
Dijkstry - macierz	0,76	6,13	53,64	87,95	409,55	650,29	5333,55
Dijkstry - lista	0,42	1,06	5,708	9,088	24,114	40,212	164,202
Bellman-Ford - macierz	0,65	6,52	63,57	87,67	218,996	715,046	5323,04
Bellman-Ford - lista	0,15	1,562	14,186	38,764	113,074	319,558	2358,384

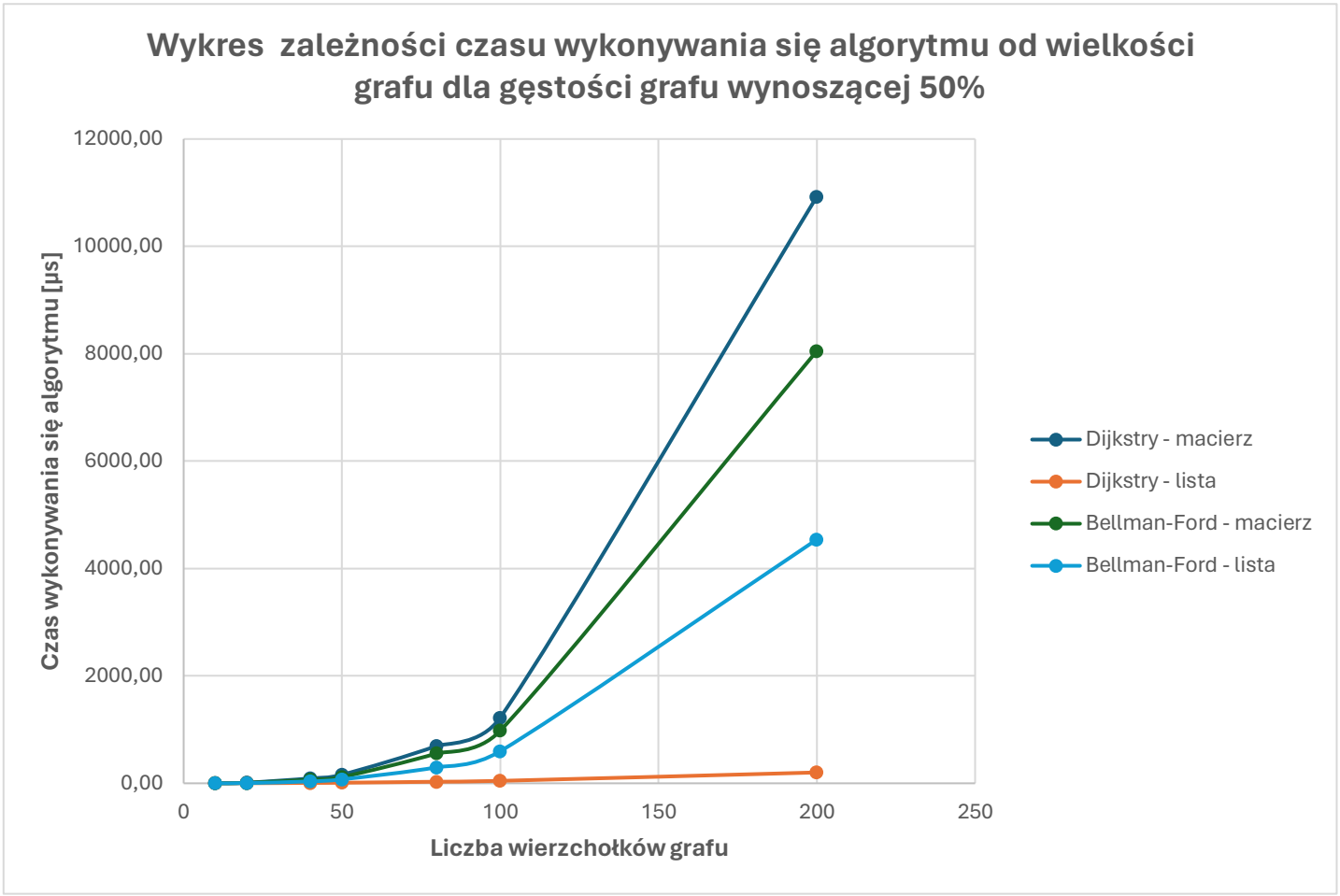


Rysunek 8: Wykres zależności czasu wykonywania się algorytmu wyznaczającego najkrótszą ścieżkę w grafie od wielkości grafu dla gęstości grafu 25%

Gęstość grafu - 50%

Tabela 9: Tabela średnich czasów wykonywania się algorytmów wyznaczających najkrótszą ścieżkę w grafie dla badanych wielkości i reprezentacji grafów – gęstość grafu 50%

Średnie czasy [µs] wykonywania się algorytmów w zależności od reprezentacji i wielkości grafu							
Gęstość grafu: 50%							
Algorytm-Reprzentacja\Liczba wierzchołków	10	20	40	50	80	100	200
Dijkstry - macierz	1,51	12,01	91,45	160,27	698,22	1224,08	10926,15
Dijkstry - lista	0,56	1,684	6,944	10,94	29,318	45,418	201,882
Bellman-Ford - macierz	1,31	9,06	68,04	117,168	557,712	980,66	8052,02
Bellman-Ford - lista	0,53	4,224	35,306	73,504	299,006	597,252	4540,936

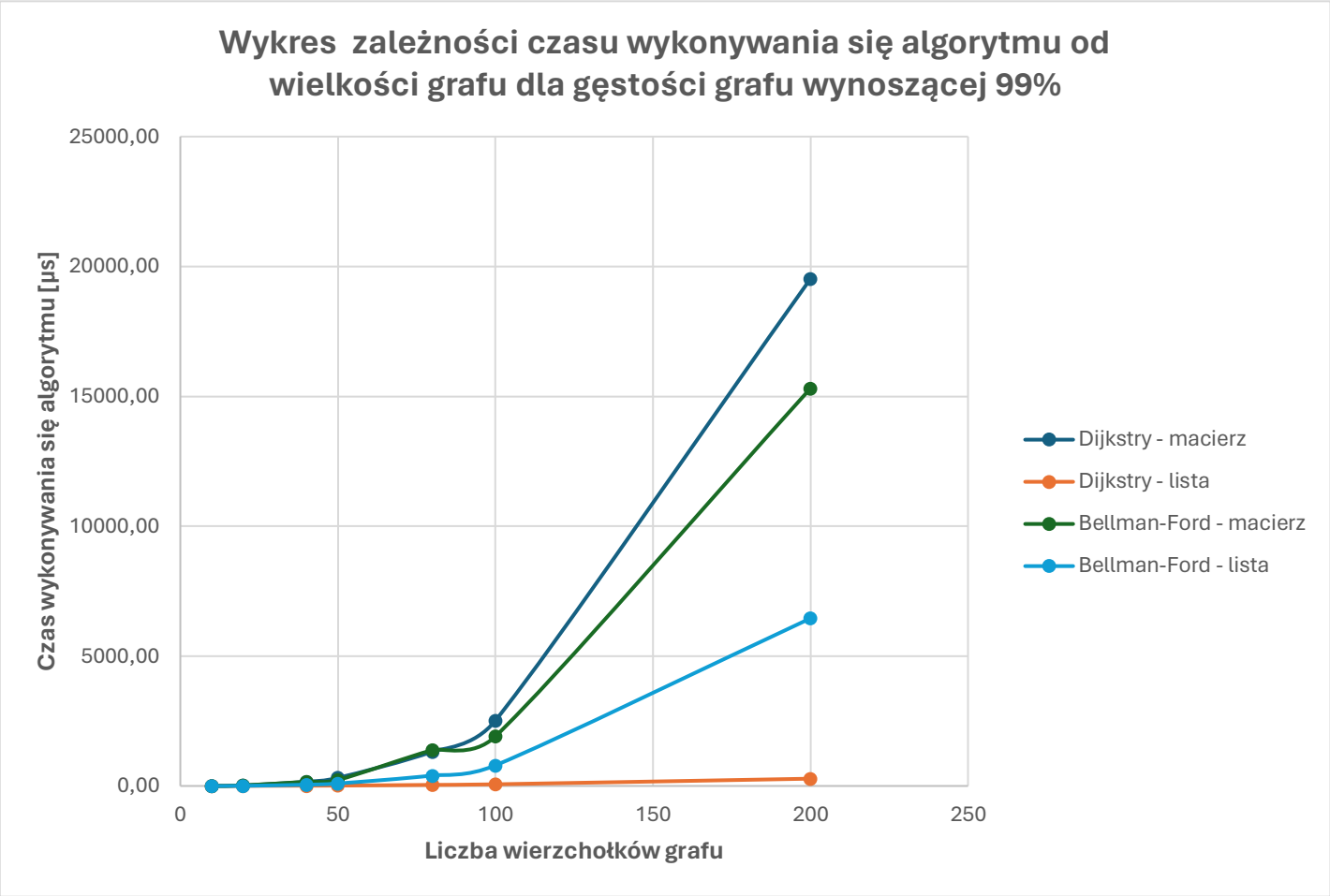


Rysunek 9: Wykres zależności czasu wykonywania się algorytmu wyznaczającego najkrótszą ścieżkę w grafie od wielkości grafu dla gęstości grafu 50%

Gęstość grafu - 99%

Tabela 10: Tabela średnich czasów wykonywania się algorytmów wyznaczających najkrótszą ścieżkę w grafie dla badanych wielkości i reprezentacji grafów – gęstość grafu 99%

Średnie czasy [µs] wykonywania się algorytmów w zależności od reprezentacji i wielkości grafu							
Gęstość grafu: 99%							
Algorytm-Reprzentacja\Liczba wierzchołków	10	20	40	50	80	100	200
Dijkstry - macierz	2,49	21,44	163,29	312,82	1324,53	2519,04	19524,23
Dijkstry - lista	0,62	3,054	11,864	14,93	41,168	63,72	280,328
Bellman-Ford - macierz	2,16	21,51	162,94	233,514	1381,586	1909,75	15305,1
Bellman-Ford - lista	0,74	5,696	48,22	95,554	403,23	793,54	6452,22



Rysunek 10: Wykres zależności czasu wykonywania się algorytmu wyznaczającego najkrótszą ścieżkę w grafie od wielkości grafu dla gęstości grafu 99%

Wnioski

Analizując powyższe tabele wyników oraz wykresy, można zauważyć, że dla każdej gęstości grafu dla listowej reprezentacji, oba algorytmy są znacznie wydajniejsze niż dla macierzowej reprezentacji.

Najszybszym algorytmem jest algorytm Dijkstry dla reprezentacji listowej grafu. Dla każdej gęstości grafu, czas jego wykonywania się jest wielokrotnie krótszy od reszty algorytmów.

Dla grafów o gęstości (50% i 99%) zapisanych w pamięci komputera w reprezentacji macierzowej, algorytm Bellmana-Forda jest efektywniejszym algorytmem. Dla grafów rzadkich (25%) oba algorytmy mają bardzo podobną efektywność.

Zgodnie z oczekiwaniami, dla każdego algorytmu, czym większa gęstość badanego grafu, tym algorytm wykonuje się dłuższy czas.

Dla algorytmu Dijkstry dla reprezentacji listowej, zależność między średnim czasem wykonywania się algorytmu a wielkością grafu jest kwadratowa. Jest to zgodne z teoretyczną złożonością obliczeniową – $\boldsymbol{O}(\boldsymbol{V^2})$.

Dla algorytmu Dijkstry dla reprezentacji macierzowej, zależność między średnim czasem wykonywania się algorytmu a wielkością grafu jest sześcienna. Wyniki te zgodne są z teoretyczną złożonością obliczeniową – $\boldsymbol{O}(\boldsymbol{V^3})$.

Dla algorytmu Bellmana-Forda dla reprezentacji macierzowej oraz listowej, czas wykonywania się algorytmu rośnie proporcjonalnie do wzrostu liczby wierzchołków oraz do wzrostu liczby krawędzi grafu, co zgadza się teoretyczną złożonością obliczeniową – $\boldsymbol{O}(\boldsymbol{V} * \boldsymbol{E})$.

Podsumowanie

Podczas wyznaczania najkrótszej ścieżki w grafie, operowanie na grafach zapisanych w pamięci komputera w reprezentacji listowej jest znacznie korzystniejsze w porównaniu z operowaniem na grafach zapisanych w pamięci komputera w reprezentacji macierzowej. Najwydajniejszym algorytmem wyznaczającym najkrótszą ścieżkę w grafie jest algorytm Dijkstry dla reprezentacji listowej grafu. Jeśli jednak trzeba będzie operować na reprezentacji macierzowej grafu, warto skorzystać z algorytmu Bellmana-Forda, który jest minimalnie efektywniejszy.

Bibliografia

[1] Prezentacje Doktora Jarosława Mierzwy

[2] https://www.youtube.com/@MichaelSambol

[3] Algorytmy i Struktury Danych - Grafy (eduinf.waw.pl)

[4] https://pl.wikipedia.org/wiki

[5] https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/