STRUKTURY DANYCH I ZŁOŻONOŚĆ OBLICZENIOWA

Zadanie projektowe nr 2: Badanie efektywności algorytmów grafowych w zależności od rozmiaru instancji oraz sposobu reprezentacji grafu w pamięci komputera.

17 MAJA 2017

ŁUKASZ BROLL

225972

SPIS TREŚCI

WSTĘP TEORETYCZNY

ZŁOŻONOŚĆ OBLICZENIOWA

Czasową złożoność obliczeniową określamy jako ilość czasu niezbędnego do rozwiązania problemu w zależności od liczby danych wejściowych. Jest to jeden z najważniejszych parametrów charakteryzujących algorytm. Decyduje on o efektywności całego programu. Podstawowymi zasobami systemowymi uwzględnianymi w analizie algorytmów są czas działania oraz obszar zajmowanej pamięci. Na złożoność czasową składają się dwie wartości: **pesymistyczna**, czyli taka, która charakteryzuje najgorszy przypadek działania oraz **oczekiwana**. Złożoność czasową oznaczamy jako O(n).

REPREZENTACJA GRAFU W PAMIĘCI KOMPUTERA

LISTA SĄSIEDZTWA

Do reprezentacji grafu wykorzystano tablicę *n* elementową, gdzie *n* oznacza liczbę wierzchołków. Każdy element tej tablicy jest listą. Lista reprezentuje wierzchołek startowy. Na liście są przechowywane numery wierzchołków końcowych, czyli sąsiadów wierzchołka startowego, z którymi jest on połączony krawędzią oraz wagi krawędzi. Złożoność obliczeniowa struktury wynosi: O(n) , gdzie n – liczba wierzchołków grafu.

MACIERZ SĄSIEDZTWA

Graf reprezentowany jest za pomocą macierzy kwadratowej A o stopniu *n*, gdzie *n* oznacza liczbę wierzchołków w grafie. Odwzorowuje ona połączenia wierzchołków między krawędziami. Wiersze macierzy sąsiedztwa odwzorowują zawsze wierzchołki startowe krawędzi, a kolumny odwzorowują wierzchołki końcowe krawędzi. Komórka A[*i*,*j*], która znajduje się w *i*-tym wierszu i *j*-tej kolumnie odwzorowuje krawędź łączącą wierzchołek startowy v_i z wierzchołkiem końcowym v_j. Jeśli A[*i*,*j*] ma wartość różną od 0, to dana krawędź istnieje, a jej wartość oznacza wagę krawędzi. Jeśli A[*i*,*j*] ma wartość 0, to wierzchołek v_i nie łączy się krawędzią z wierzchołkiem v_j. Złożoność obliczeniowa struktury wynosi: $O(n^2)$, gdzie n – liczba wierzchołków grafu.

ZAIMPLEMENTOWANE ALGORYTMY

PROBLEM NAJKRÓTSZEJ ŚCIEZKI

ALGORYTM DIJKSTRY

Algorytm znajduje w grafie najkrótsze ścieżki pomiędzy wybranym wierzchołkiem a wszystkimi pozostałymi, wyliczając również koszt przejścia każdej z tych ścieżek, czyli sumę wag krawędzi na ścieżce. Algorytm ma złożoność czasową $O(n^2)$ przy wykorzystaniu wyszukiwania liniowego podczas szukania wierzchołków o najmniejszym koszcie dojścia.

ALGORYTM BELLMANA-FORDA

Idea algorytmu opiera się na metodzie relaksacji. W odróżnieniu od algorytmu Dijkstry, poprawność algorytmu Bellmana-Forda nie opiera się na założeniu, że wagi w grafie są nieujemne (nie może jednak występować cykl o łącznej ujemnej wadze osiągalny ze źródła). Za tę ogólność płaci się jednak wyższą złożonością czasową. Działa on w czasie O(n*m).

MINIMALNE DRZEWO ROZPINAJĄCE

ALGORYTM KRUSKALA

Algorytm polega na łączeniu wielu poddrzew w jedno za pomocą krawędzi o najmniejszej wadze. W rezultacie powstałe drzewo będzie minimalne. Na początek należy posortować wszystkie krawędzie w porządku niemalejącym. Po tej czynności można przystąpić do tworzenia drzewa. Proces ten nazywa się rozrastaniem lasu drzew. Wybieramy krawędzie o najmniejszej wadze i jeśli wybrana krawędź należy do dwóch różnych drzew należy je scalić (dodać do lasu). Krawędzie wybieramy tak długo, aż wszystkie wierzchołki nie będą należały do jednego drzewa. Złożoność obliczeniowa algorytmu wynosi $O(m * \log_2 n)$.

ALGORYTM PRIMA

Na początku, algorytm dodaje do zbioru A reprezentującego drzewo krawędź o najmniejszej wadze, łączącą wierzchołek początkowy v z dowolnym wierzchołkiem. W każdym kolejnym kroku procedura dodaje do A najlżejszą krawędź wśród krawędzi łączących wierzchołki już odwiedzone z nieodwiedzonymi. W przypadku, gdy struktura A jest kolejką priorytetową opartą na kopcu binarnym, czasowa złożoność obliczeniowa operacji wynosi $O(m * \log_2 n)$.

GENEROWANIE GRAFU

Metoda generująca graf przyjmuje jako parametry ilość wierzchołków i gęstość grafu. Celem zachowania spójności grafu, jego budowa rozpoczyna się od utworzenia drzewa rozpinającego i znacznie większych wagach niż kolejne, dodatkowe krawędzie. Na samym początku z wzorów matematycznych obliczana jest maksymalna ilość krawędzi jaka ma zostać dodana do grafu z uwzględnieniem jego gęstości, a licznik krawędzi jest zerowany. Następnie w pętli, razem z inkrementacją ilości krawędzi losowana jest waga krawędzi i uruchamiana funkcja dodająca krawędź do grafu.

Funkcja dodająca krawędź na początku sprawdza czy waga krawędzi nie przekracza dopuszczalnego zakresu (max 900), następnie sprawdzane są kolejne warunki jednocześnie na liście i macierzy tj, czy krawędź już istnieje w macierzy lub liście, czy wierzchołek ma już sąsiadów, czy to pierwsza krawędź wierzchołka itd. Na tym etapie utworozne zostało drzewa rozpinające grafu.

Kolejnym krokiem jest zapełnianie grafu krawędziami zgodnie z wyznaczoną gęstością. Losowane są kolejne wierzchołki, do których próbuje się dodać krawędź. Jeśli wylosowany wierzchołek spełnia warunki dodania krawędzi, licznik krawędzi wzrasta i graf jest generowany. Sposób ten zapewnia generowanie spójnej i zmiennej struktury.

PLAN EKSPERYMENTU

- Wagi krawędzi oraz liczba wierzchołków i krawędzi są liczbami całkowitymi. W przypadku losowego generowania grafu, liczba krawędzi jest obliczana na podstawie gęstości podanej przez użytkownika i zaokrąglona w górę do liczby całkowitej.

- Pomiaru czasów wykonania każdego z trzech algorytmów dokonano dla ilości wierzchołków n = {60, 80, 100, 120, 140}. Dla każdej ilości wierzchołków rozważano 4 różne gęstości – 25%, 50%, 75% oraz 99%. Przedstawiony w sprawozdaniu czas wykonania algorytmu dla danej gęstości i ilości wierzchołków jest czasem uśrednionym ze 100 dokonanych pomiarów, a każdy ze 100 pomiarów odbywał się na nowym, losowo wygenerowanym grafie.

- W przypadku problemu najkrótszej ścieżki, podczas mierzenia czasów, wierzchołek, z którego rozpocznie się algorytm był pierwszym elementem tabeli.

- W przypadku problemu najkrótszej ścieżki, krawędzie grafu wejściowego traktowane są jako skierowane, natomiast w przypadku problemu minimalnego drzewa rozpinającego, jako nieskierowane.

- Pomiarów czasu dokonano za pomocą funkcji:
  *BOOL QueryPerformanceCounter (__out LARGE_INTEGER *lpPerformanceCount);*
  Źródło licznika: http://cpp0x.pl/forum/temat/?id=21331

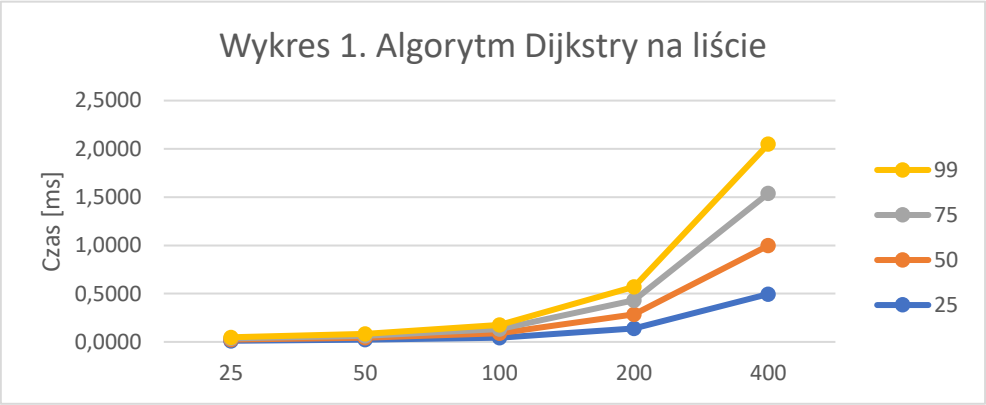
- Do pomiaru czasu w przypadku problemu MST były wliczane algorytmy kopiujące zawartość listy i macierzy do kopca i drzew.

WYNIKI EKSPERYMENTU

NAJKRÓTSZA ŚCIEŻKA W GRAFIE

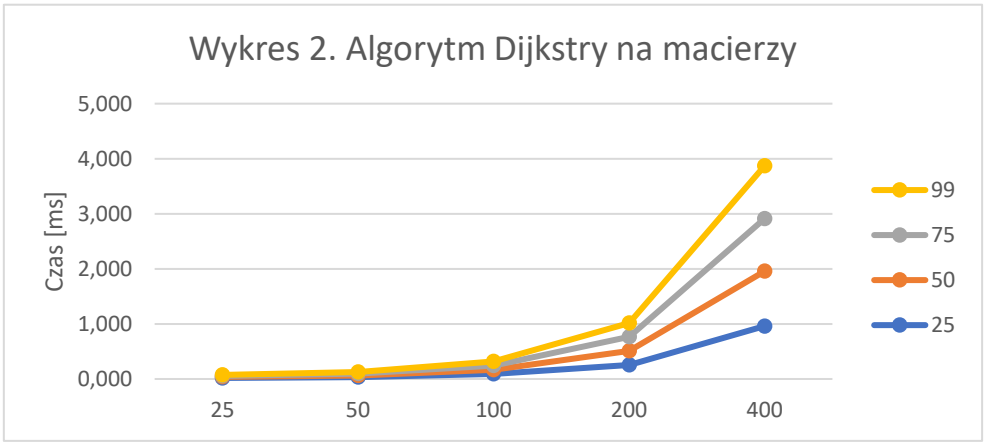
ALGORYTM DIJKSTRY

NA LIŚCIE



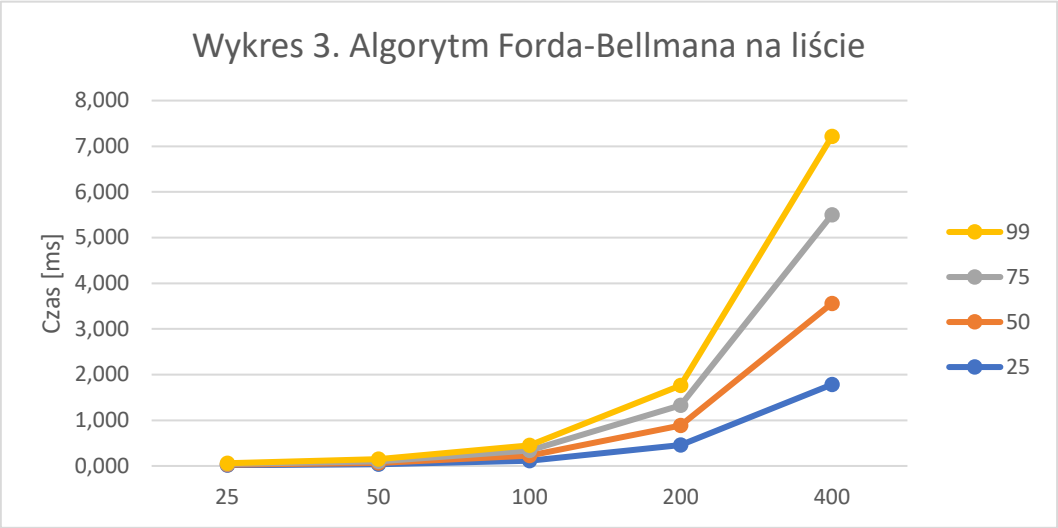
	Gęstość [%]			
Rozmiar	**25**	**50**	**75**	**99**
25	0,0118	0,0121	0,0123	0,0118
50	0,0243	0,0191	0,0190	0,0208
100	0,0446	0,0442	0,0439	0,0435
200	0,1390	0,1477	0,1436	0,1392
400	0,4953	0,5025	0,5396	0,5107

NA MACIERZY



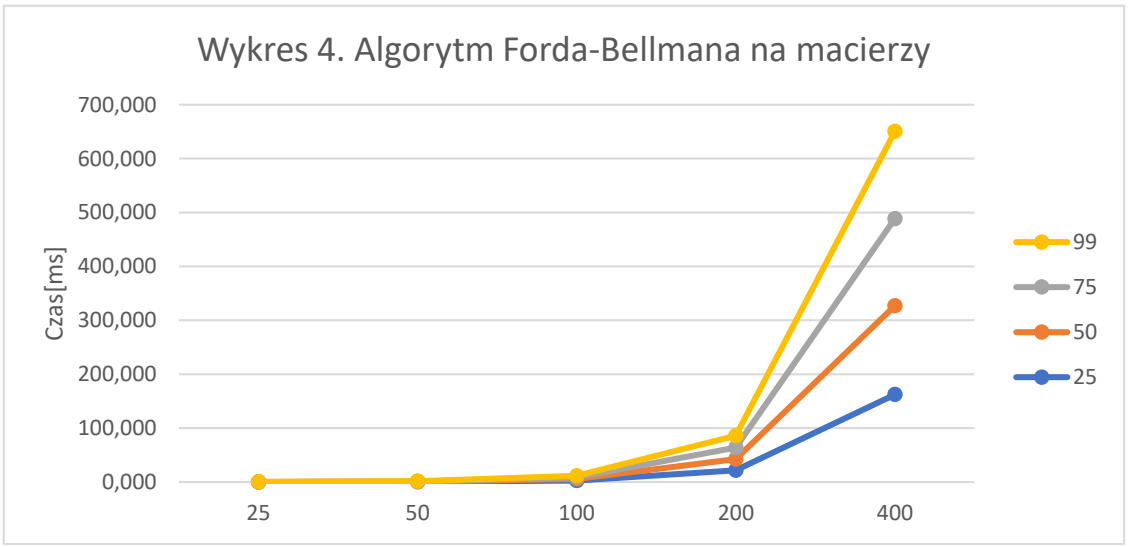
	Gęstość [%]			
Rozmiar	**25**	**50**	**75**	**99**
25	0,019	0,019	0,018	0,019
50	0,032	0,031	0,033	0,034
100	0,094	0,076	0,075	0,075
200	0,253	0,259	0,252	0,251
400	0,960	0,997	0,955	0,958

NA LIŚCIE



	Gęstość [%]			
Rozmiar	**25**	**50**	**75**	**99**
25	0,015	0,014	0,015	0,016
50	0,033	0,033	0,033	0,050
100	0,113	0,116	0,109	0,112
200	0,456	0,430	0,443	0,434
400	1,782	1,776	1,939	1,716

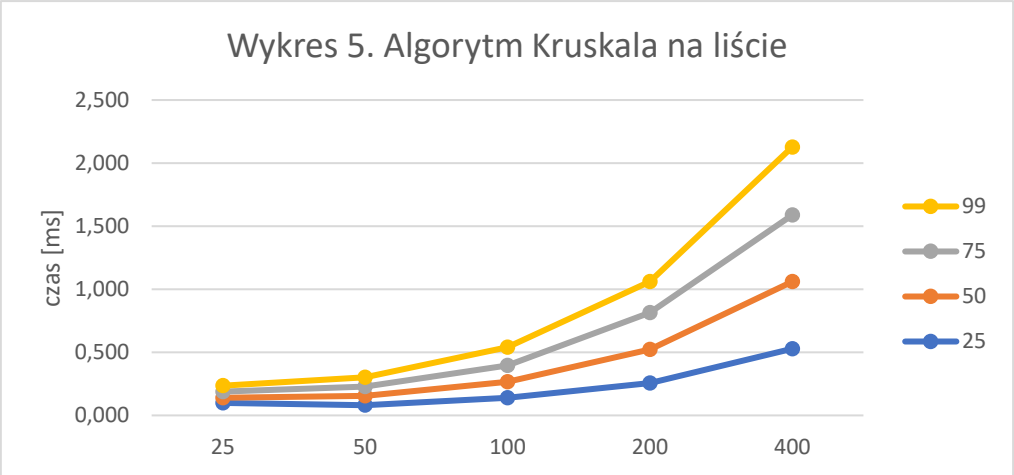
NA MACIERZY



	Gęstość [%]			
Rozmiar	**25**	**50**	**75**	**99**
25	0,066	0,065	0,074	0,066
50	0,398	0,413	0,461	0,390
100	2,813	3,149	3,032	2,909
200	21,730	21,082	21,272	21,853
400	162,230	164,799	161,671	161,704

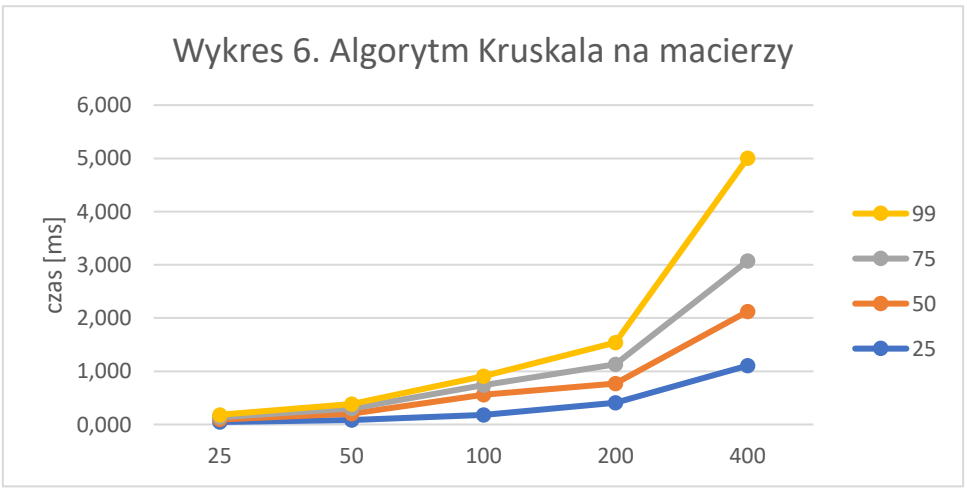
ALGORYTM KRUSKALA

NA LIŚCIE



	Gęstość [%]			
Rozmiar	**25**	**50**	**75**	**99**
25	0,100	0,040	0,049	0,047
50	0,081	0,075	0,073	0,073
100	0,138	0,128	0,129	0,144
200	0,256	0,267	0,293	0,245
400	0,528	0,532	0,528	0,540

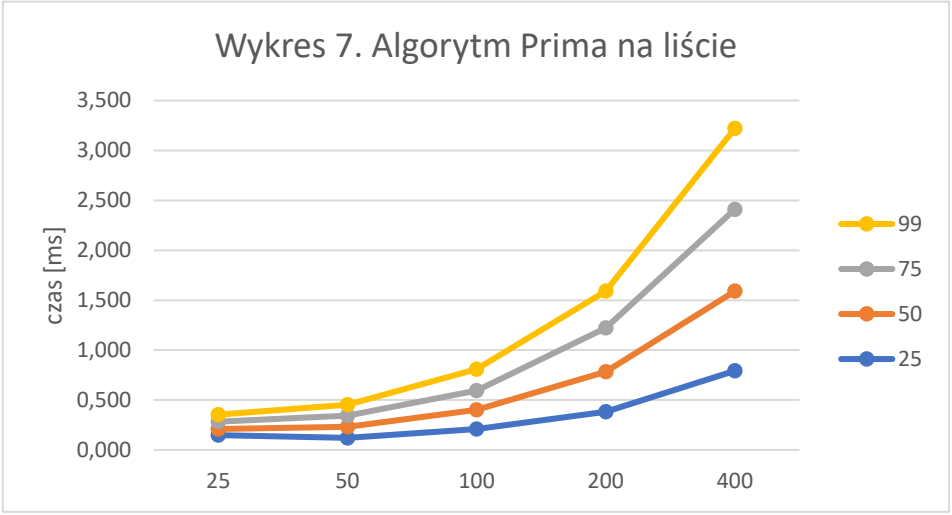
NA MACIERZY



	Gęstość [%]			
Rozmiar	**25**	**50**	**75**	**99**
25	0,047	0,046	0,041	0,048
50	0,085	0,120	0,097	0,083
100	0,180	0,378	0,183	0,165
200	0,407	0,364	0,358	0,410
400	1,104	1,017	0,949	1,932

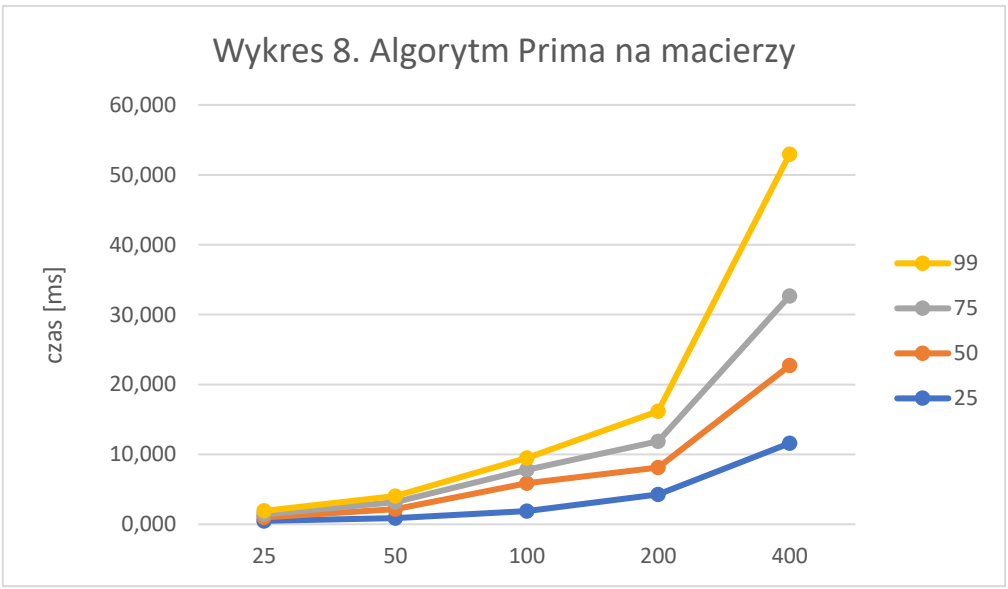
ALGORYTM PRIMA

NA LIŚCIE



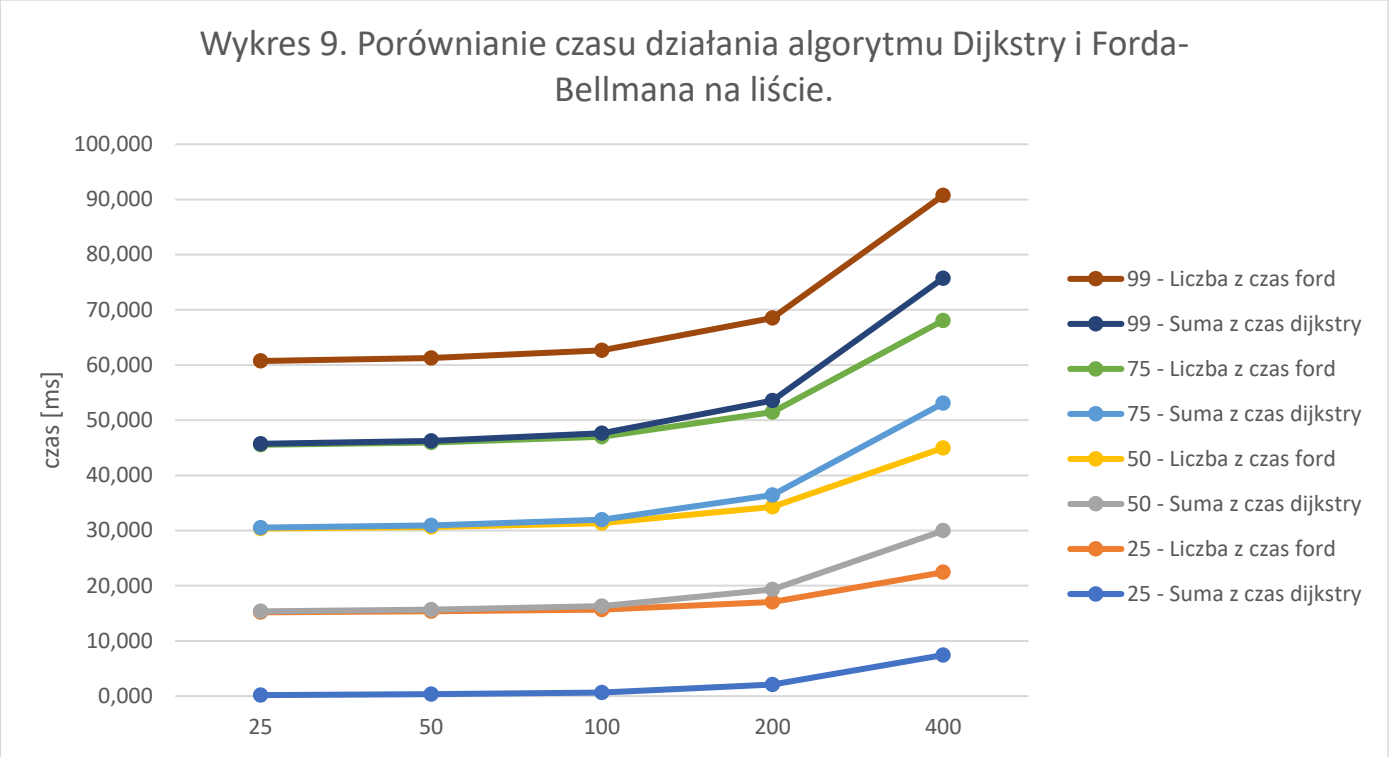
	Gęstość [%]			
Rozmiar	**25**	**50**	**75**	**99**
25	0,150	0,060	0,073	0,070
50	0,121	0,112	0,109	0,109
100	0,208	0,192	0,193	0,216
200	0,384	0,401	0,439	0,368
400	0,792	0,798	0,820	0,810

NA MACIERZY



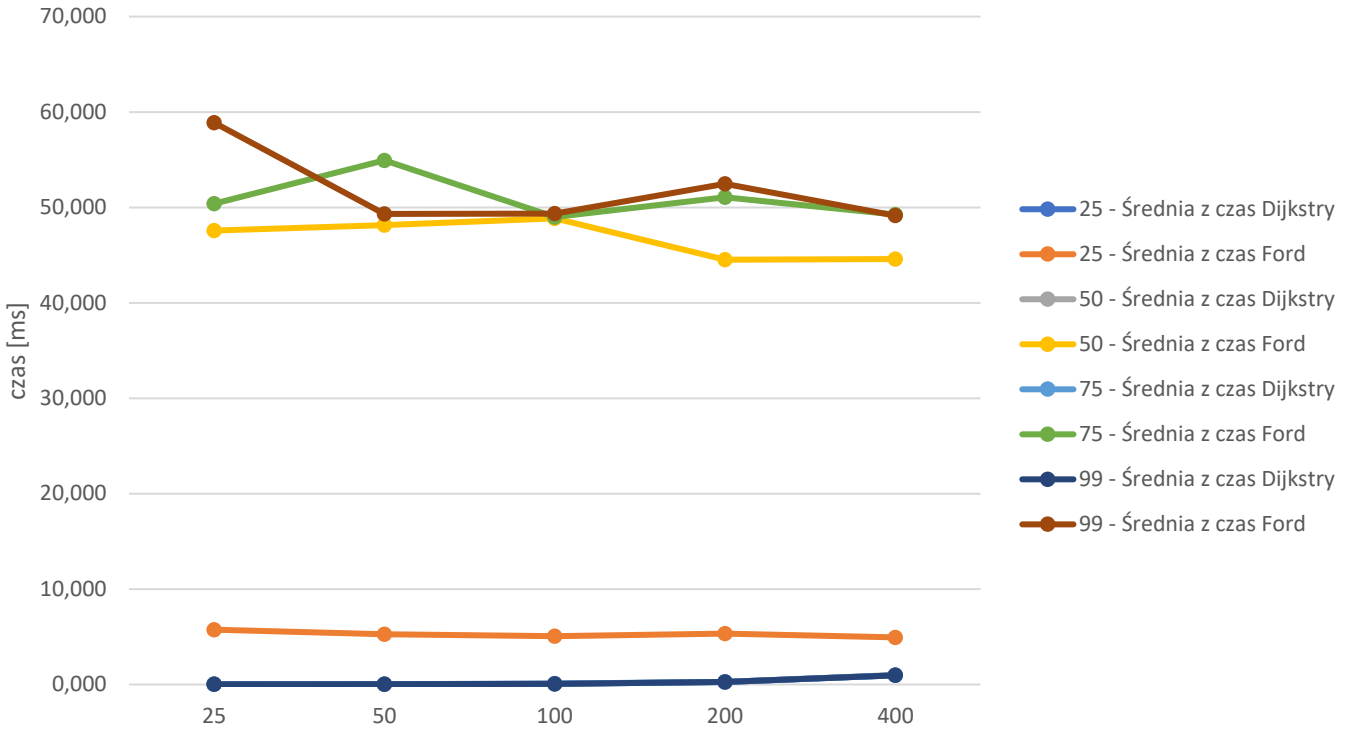
	Gęstość [%]			
Rozmiar	**25**	**50**	**75**	**99**
25	0,492	0,488	0,435	0,502
50	0,891	1,258	1,014	0,867
100	1,892	3,968	1,917	1,736
200	4,273	3,822	3,763	4,307
400	11,595	11,119	9,967	20,285

PODSUMOWANIE WYNIKÓW



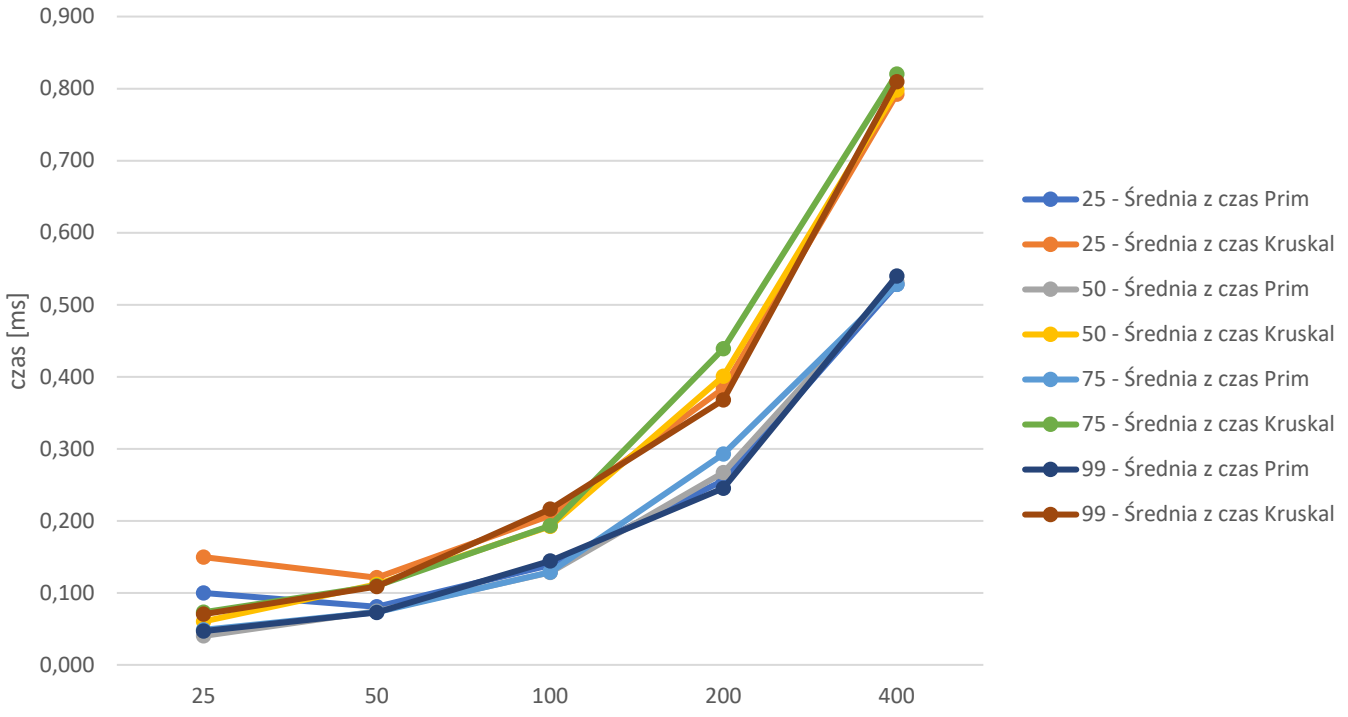
	Gęstość [%]							
	25		50		75		99	
Rozmiar	Średnia z czas dijkstry	Średnia z czas ford	Średnia z czas dijkstry	Średnia z czas ford	Średnia z czas dijkstry	Średnia z czas ford	Średnia z czas dijkstry	Średnia z czas ford
25	0,012	0,148	0,012	0,517	0,012	0,621	0,012	0,758
50	0,024	0,127	0,019	0,433	0,019	0,620	0,021	0,634
100	0,045	0,121	0,044	0,557	0,044	0,609	0,044	0,598
200	0,139	0,146	0,148	0,424	0,144	0,583	0,139	0,615
400	0,495	0,155	0,502	0,505	0,540	0,583	0,511	0,612

Wykres 10. Porównanie czasu działania algorytmu Dijkstry i Forda-Bellmana na macierzy.

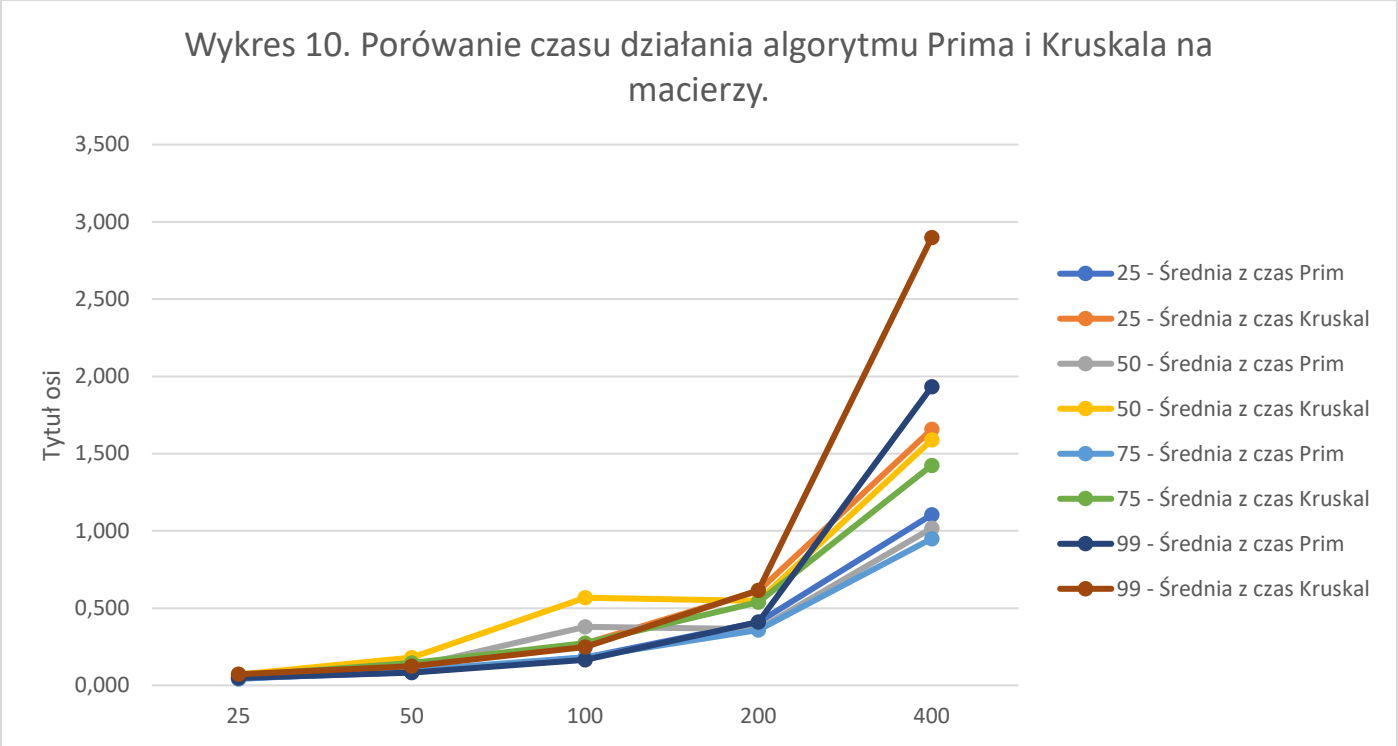


	Gęstość [%]							
	25		50		75		99	
Rozmiar	Średnia z czas Dijkstry	Średnia z czas Ford	Średnia z czas Dijkstry	Średnia z czas Ford	Średnia z czas Dijkstry	Średnia z czas Ford	Średnia z czas Dijkstry	Średnia z czas Ford
25	0,019	5,744	0,019	47,583	0,018	50,406	0,019	58,882
50	0,032	5,264	0,031	48,126	0,033	54,934	0,034	49,302
100	0,094	5,052	0,076	48,851	0,075	48,995	0,075	49,359
200	0,253	5,332	0,259	44,522	0,252	51,057	0,251	52,486
400	0,960	4,937	0,997	44,571	0,955	49,261	0,958	49,137

Wykres 10. Porównanie czasu działania algorytmu Prima i Kruskala na liście.



	Gęstość [%]							
	25		50		75		99	
Rozmiar	Średnia z czas Prim	Średnia z czas Kruskal	Średnia z czas Prim	Średnia z czas Kruskal	Średnia z czas Prim	Średnia z czas Kruskal	Średnia z czas Prim	Średnia z czas Kruskal
25	0,100	0,150	0,040	0,060	0,049	0,073	0,047	0,070
50	0,081	0,121	0,075	0,112	0,073	0,109	0,073	0,109
100	0,138	0,208	0,128	0,192	0,129	0,193	0,144	0,216
200	0,256	0,384	0,267	0,401	0,293	0,439	0,245	0,368
400	0,528	0,792	0,532	0,798	0,528	0,820	0,540	0,810



	Gęstość [%]							
	25		**50**		**75**		**99**	
Rozmiar	**Średnia z czas Prim**	**Średnia z czas Kruskal**	**Średnia z czas Prim**	**Średnia z czas Kruskal**	**Średnia z czas Prim**	**Średnia z czas Kruskal**	**Średnia z czas Prim**	**Średnia z czas Kruskal**
25	0,047	0,070	0,046	0,070	0,041	0,062	0,048	0,072
50	0,085	0,127	0,120	0,180	0,097	0,145	0,083	0,124
100	0,180	0,270	0,378	0,567	0,183	0,274	0,165	0,248
200	0,407	0,610	0,364	0,546	0,358	0,538	0,410	0,615
400	1,104	1,656	1,017	1,588	0,949	1,424	1,932	2,898

WNIOSKI

Rzędy złożoności obliczeniowych zaimplementowanych algorytmów generalnie zgadzają się z danymi w literaturze pod względem zależności czasu wykonywania operacji od liczby wierzchołków i krawędzi grafu.

Widać znaczącą różnicę między czasem wykonywania się algorytmów w grafie reprezentowanym za pomocą list sąsiedztwa i za pomocą macierzy sąsiedztwa. Różnica między czasami wzrasta wraz z ilością elementów w grafie. Lista sąsiedztwa jest nieznacznie szybsza od macierzy jedynie przy małej gęstości grafu.

Największa różnica występuje w przypadku algorytmu Dijkstry oraz Bellmana Forda. Mimo, że algorytm Dijkstry wymaga przejrzenia wszystkich elementów macierzy, które wykonuje się w dwóch pętlach, a dla listy jest to jedna pętla przeglądająca sąsiadów wierzchołków., to sprawdzanie ujemnych cykli w związku z działaniem na ujemnych wagach, drastycznie wydłuża czas pracy tego algorytmu.

Analizując wykresy dotyczące algorytmu Prima i Kruskala, można stwierdzić, że mają one również rozkład logarytmiczny względem ilości krawędzi. Mimo zbudowania kolejki priorytetowej na kopcu w obu przypadkach, algorytm Kruskala wykonuje się szybciej.

Rozbieżności pomiędzy danymi literaturowymi a przedstawionymi wynikami eksperymentu mogą wynikać z niedokładności pomiaru czasu w systemie Windows, różnej zajętości procesora podczas wykonywania algorytmów, niedoskonałej implementacji struktur (lista, tablica, kopiec, kolejka, stos) oraz, przede wszystkim, wpływających na efektywność błędów w implementacji algorytmów. Ponadto, rzeczywiste kształty wykresów mogą nie być widoczne przy tak małej ilości danych.

BIBLIOGRAFIA

1. Cormen Thomas H., Leiserson Charles E., Rivest Ronald L., *Wprowadzenie do Algorytmów*, wyd. IV, Warszawa: Wydawnictwa Naukowo-Techniczne
2. Algorytm Dijkstry – Wikipedia, wolna encyklopedia http://pl.wikipedia.org/wiki/Algorytm_Dijkstry
3. Algorytm Prima – Wikipedia, wolna encyklopedia http://pl.wikipedia.org/wiki/Algorytm_Prima
4. 4.Algorytm Bellmana-Forda - Wikipedia, wolna encyklopedia https://pl.wikipedia.org/wiki/Algorytm_Bellmana-Forda
5. 5.Algorytm Kruskala - Wikipedia, wolna encyklopedia https://pl.wikipedia.org/wiki/Algorytm_Kruskala
6. 6.Macierz sąsiedztwa – Wikipedia, wolna encyklopedia https://pl.wikipedia.org/wiki/Macierz_s%C4%85siedztwa