Małgorzata Pietras
Automatyka i Robotyka
235794

dr inż. Łukasz Jeleń
środa 7:30

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt 2
data oddania: 08.05.2019

1 Wstęp

Projekt drugi polegał na zaimplementowaniu grafu przechowującego elementy danego typu oraz funkcji wykonujących na nim potrzebne operacje. Graf zaimplementowany został za pomocą dwóch metod: listy sąsiedztwa i macierzy sąsiedztwa. Następnie przeprowadzona była analiza efektywności dwóch algorytmów: Kruskala i Prima-Jarnika. Liczby wierzchołków w grafach to: 10, 50, 100, 500 i 1000, a gęstości: 25%, 50%, 75% i 100%. Podane wyniki zostały uśrednione po 100 iteracjach.

2 Opis badanych algorytmów

Algorytmy Prima i Kruskala służą do obliczania minimalnego drzewa rozpinającego dany graf. Jest to drzewo, które zawiera wszystkie wierzchołki grafu i minimalizuje sumę wag wszystkich jego krawędzi. Problem ten oznacza się skrótem MST.

W algorytmie Kruskala na samym początku inicjowane jest puste minimalne drzewo rozpinające, a wszystkie węzły grafu stanowią osobne klastry. Następnie w kolejce umieszczane są krawędzie i sortowane za pomocą wag. W pętli krawędzie są odrzucane lub dodawane do drzewa, a na końcu klastry łączone są w jeden. W algorytmie Kruskala do zbioru dodawane są te krawędzie, które mają najmniejsze wagi i łączą dwie różna składowe. Jest to algorytm zachłanny, ponieważ na każdym kroku jest dodawana krawędź na najmniejszej możliwej wadze. Czas działanie algorytmu zależy od wyboru struktury danych. Złożoność obliczeniowa inicjacji O(m log n). Dalsze działania mają złożoność obliczeniową O(log n), więc przewidywalna efektywność to O(m log n).

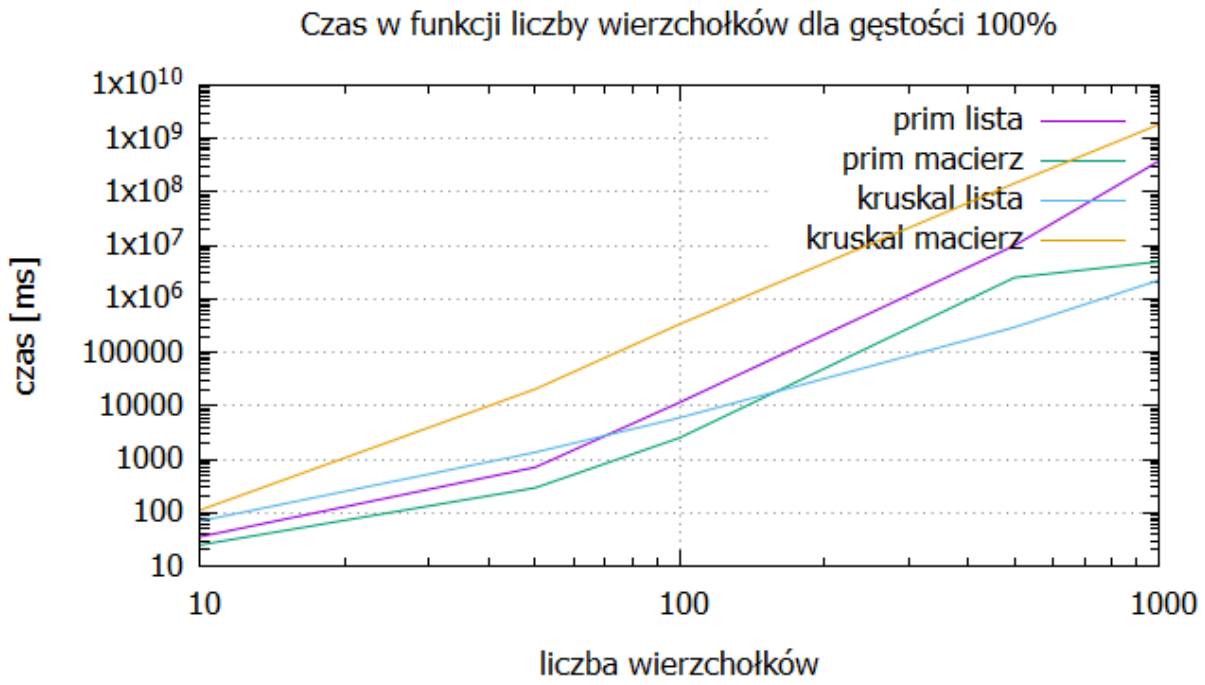
W algorytmie Prima-Jarnika na początku drzewo składa się z dowolnie wybranego wierzchołka-korzenia, które rozrasta się aż do połączenia wszystkich wierzchołków grafu. W czasie wykonywania algorytmu wszystkie wierzchołki grafu znajdują się w kolejce, w której kluczami są wagi. Kluczem na korzeniu jest 0, a jeżeli jakaś krawędź nie istnieje, wówczas za jej wagę przyjmuje się bardzo duża wartość (nieskończoność). Dodawane do drzewa krawędzie usuwane są z kolejki. Algorytm pracuje do momentu, w którym drzewo zawiera wszystkie wierzchołki grafu. W algorytmie Prima dodawane są krawędzie, która mają najmniejszą wagę i łączą wyznaczone już drzewo z wierzchołkiem spoza drzewa. Algorytm jest zachłanny, ponieważ w każdym kroku drzewo rozszerzane jest o krawędź, której waga jest najmniejsza. Efektywność tego algorytmu również zależy od wybranej metody implementacji. Całkowity czas działania jest asymptotycznie taki sam - O(m log n), jak czas wykonania dla algorytmu Kruskala,

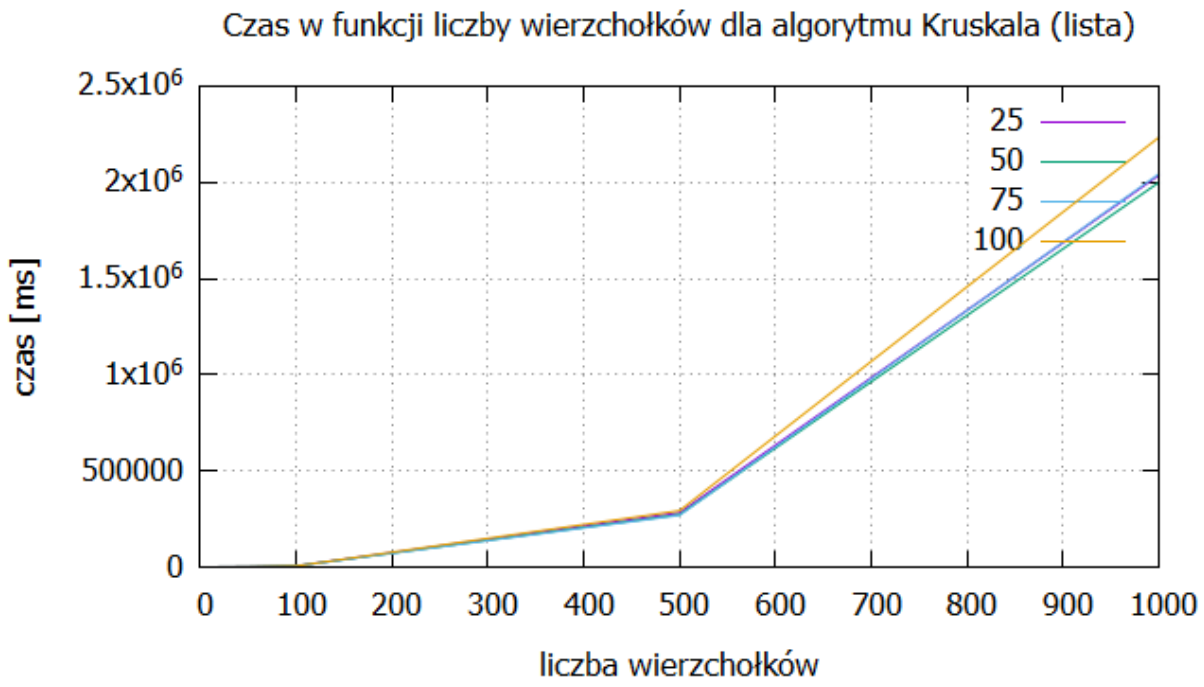
3 Przebieg eksperymentu i uzyskane wyniki

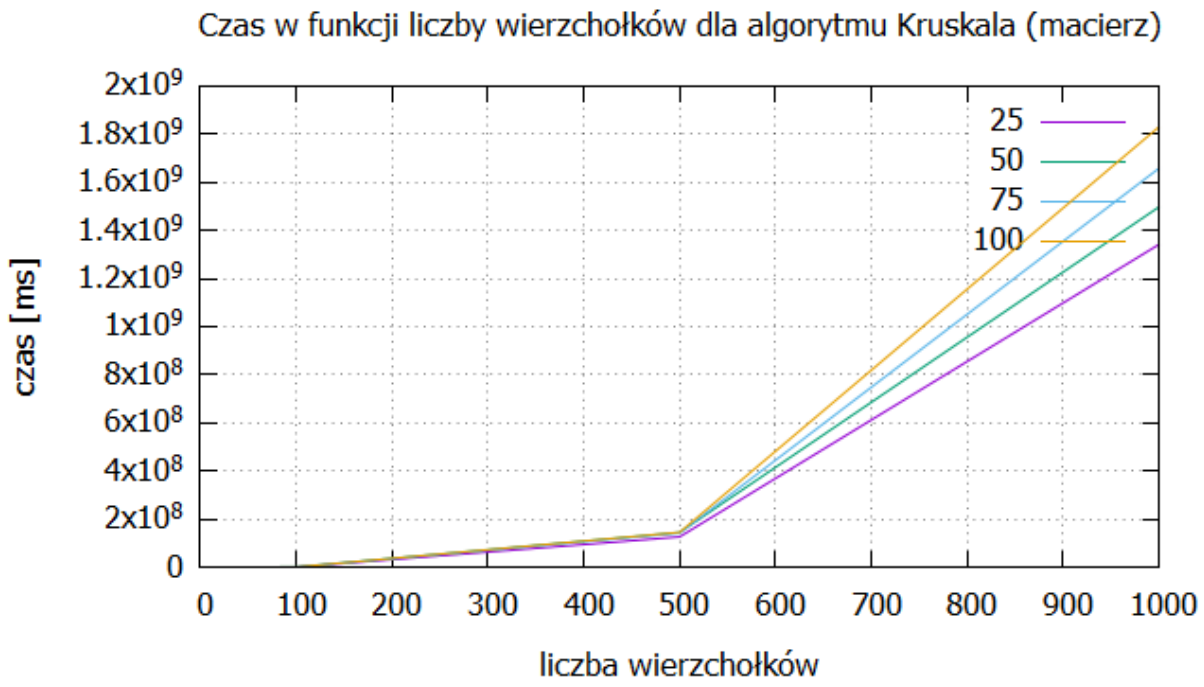
Przeprowadzone obliczenia polegały na badaniu czasu działania algorytmów na dwóch implementacjach: listy i macierzy sąsiedztwa w zależności od liczby wierzchołków oraz gęstości. Uśrednione po 100 losowych wywołaniach pomiary czasu wyrażone w milisekundach zostały zapisane do plików w celu dalszej analizy. Wyniki w postaci wykresów oraz tablic zostały pokazane poniżej.

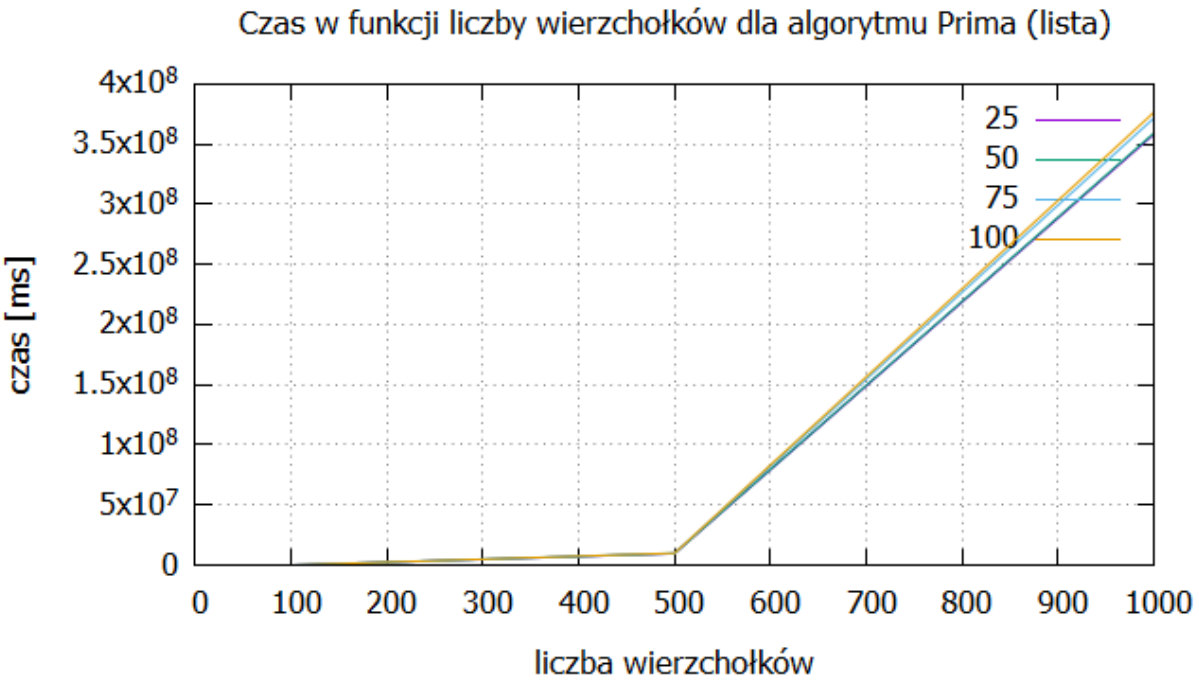
[ms]	ALGORYTM KRUSKALA LISTA				AGORYTM KRUSKALA MACIERZ			
	25%	50%	75%	100%	25%	50%	75%	100%
10	67	71	72	70	117	96	99	112
50	1263	1293	1289	1346	13640	14620	17616	20492
100	6369	6178	6053	5964	195777	235451	288471	336115
5000	282150	273157	269652	293356	125551408	143845394	141019348	144389108
10000	2035990	199875072	2042605	2233095	1340953438	1496843197	1657231765	1829088145

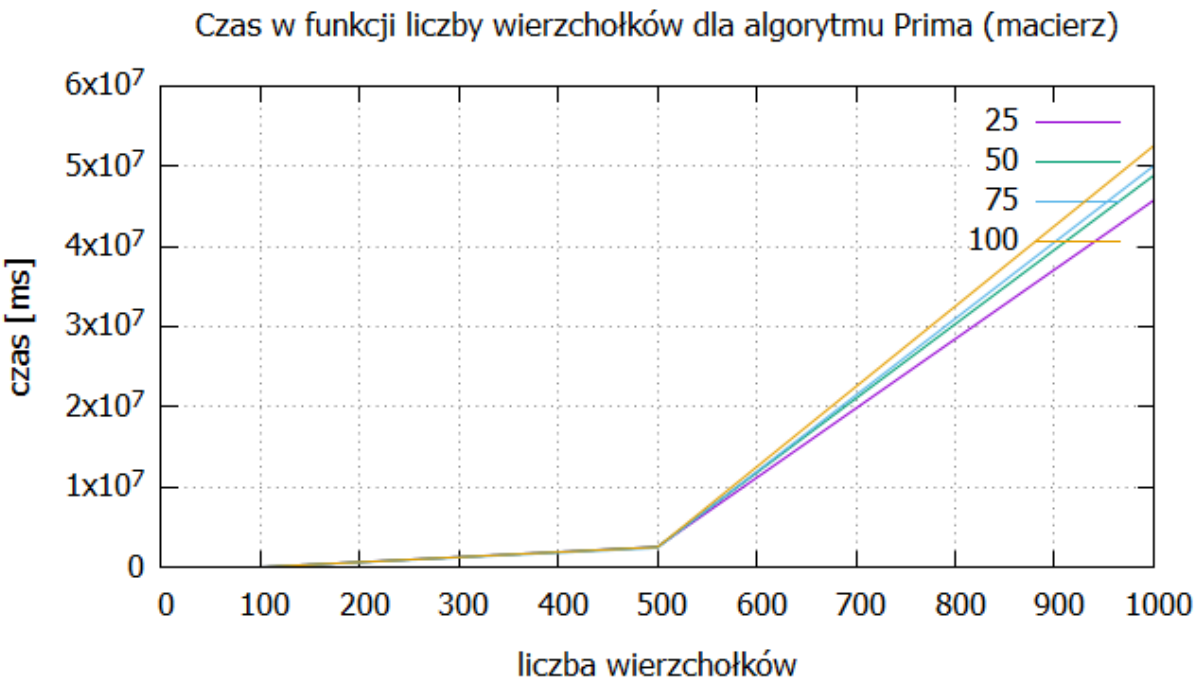
[ms]	ALGORYTM PRIMA LISTA				AGORYTM PRIMA MACIERZ			
	25%	50%	75%	100%	25%	50%	75%	100%
10	36	39	37	36	21	27	26	25
50	716	705	712	707	286	313	296	291
100	12583	11568	11570	11531	2488	2572	2491	2514
5000	9566356	9682876	9874117	9946113	2509035	2518648	2385421	2496351
10000	358014250	359082921	371468422	37635754	45746856	48847251	50047405	52552454




Czas w funkcji liczby wierzchołków dla algorytmu Kruskala (lista)
czas [ms]
liczba wierzchołków
25
50
75
100
0
500000
1x10^6
1.5x10^6
2x10^6
2.5x10^6
0
100
200
300
400
500
600
700
800
900
1000


Czas w funkcji liczby wierzchołków dla algorytmu Kruskala (macierz)
czas [ms]
liczba wierzchołków
25
50
75
100
0
2x10^8
4x10^8
6x10^8
8x10^8
1x10^9
1.2x10^9
1.4x10^9
1.6x10^9
1.8x10^9
2x10^9
0
100
200
300
400
500
600
700
800
900
1000


Czas w funkcji liczby wierzchołków dla algorytmu Prima (lista)
czas [ms]
liczba wierzchołków
25
50
75
100
0
5x10^7
1x10^8
1.5x10^8
2x10^8
2.5x10^8
3x10^8
3.5x10^8
4x10^8
0
100
200
300
400
500
600
700
800
900
1000


Czas w funkcji liczby wierzchołków dla algorytmu Prima (macierz)
czas [ms]
liczba wierzchołków
25
50
75
100
0
1x10^7
2x10^7
3x10^7
4x10^7
5x10^7
6x10^7
0
100
200
300
400
500
600
700
800
900
1000

4 Podsumowanie i wnioski

- Najwolniejszy jest algorytm Kruskala działający na macierzy sąsiedztwa.
- Dla małej liczby wierzchołków najszybszy jest algorytm Prima na macierzy.
- Dla dużej liczby wierzchołków najszybszy jest algorytm Kruskala na liście.
- Każdy z algorytmów wykazał zbliżoną do oczekiwanej złożoność obliczeniową $O(m\ logn)$.
- Wpływ gęstości grafu na czas trwania algorytmu był mniejszy niż wpływ liczby wierzchołków.
- W każdym przypadku gęstość grafu stawała się istotna jedynie dla dużej liczby wierzchołków.
- Wraz ze zwiększaniem się gęstości grafu wzrastał czas obliczeń.

5 Literatura

1. Gręboszz J., Symfonia C++ standard
2. Goodrich M., Data Structures and Algorithms in C++
3. Cormen T., Wprowadzenie do algorytmów
4. http://www.algorytm.org/algorytmy-grafowe/algorytm-prima.html 3.05.2019
5. http://www.algorytm.org/algorytmy-grafowe/algorytm-kruskala.html 3.05.2019