Dane studenta: Filip Kurasz 249458 07.05.2020

Nazwa kursu: Projektowanie algorytmów i metody sztucznej inteligencji

Dane prowadzącego: mgr inż. Marta Emirsajłow

Termin zajęć: piątek 13:15-15:00

SPRAWOZDANIE

PROJEKT NR 2

GRAFY

1. Wprowadzenie

   Należało zbadać efektywność jednego z algorytmów w zależności od sposobu reprezentacji grafu w postaci listy i macierzy. W badaniu użyto algorytmu Dijkstry, którego implementacja wymaga przyjęcia takiego grafu aby nie było w nim wag ujemnych pomiędzy wierzchołkami. Algorytm testowany był dla 5 różnych liczb wierzchołków oraz 4 różnych gęstości dla grafu zaimplementowanego w formie listy oraz macierzy. Łącznie wygenerowało to 20 różnych kombinacji, które następnie przedstawiono na wykresach w dalszej części tego sprawozdania. Liczba wierzchołków jaka była badana to: 10, 50, 100, 250, 500. Oraz gęstości grafu: 25%, 50%, 75%, 100%. Do każdej kombinacji generowano 100 uniwersalnych grafów które zapisywane były do folderu *Graph*, każdy jako plik tekstowy. Waga krawędzi w danym grafie uzależniona była od ilości wierzchołków (w zakresie 1-V np. dla 50 wierzchołków od 1 do 50). W programie nie wystąpiły elementy STL.

2. Opis algorytmu Dijkstry

   Algorytm Dijkstry służy do znajdowania najkrótszej ścieżki w grafie ważonym z wybranego wierzchołka startowego do wszystkich innych wierzchołków tego grafu lub od wybranego wierzchołka startowego do wybranego wierzchołka końcowego.

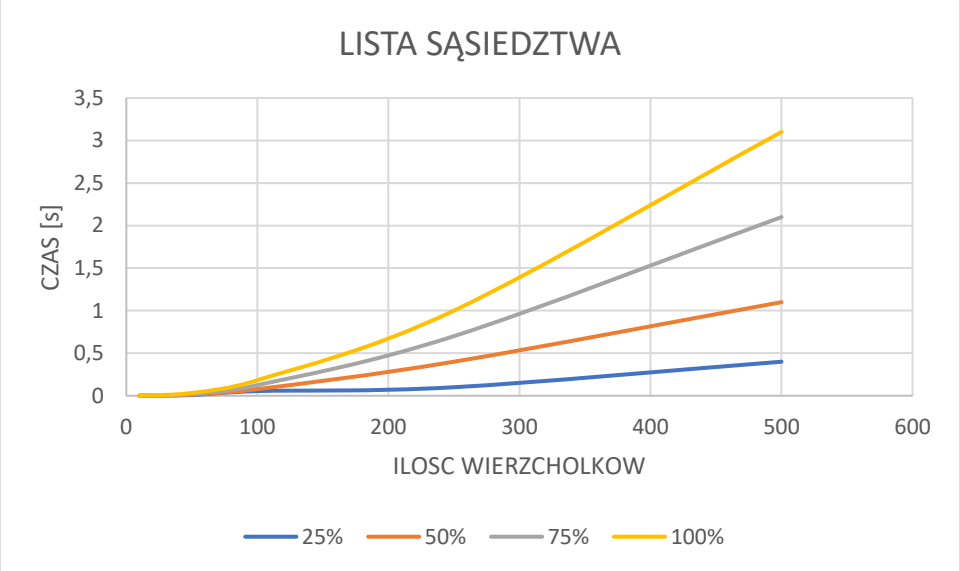
   Złożoność obliczeniowa algorytmu Dijkstry zależy od liczby wierzchołków i krawędzi grafu. O złożoności decyduje implementacja kolejki priorytetowej. Dla implementacji poprzez tablicę otrzymujemy złożoność $O(V^2)$. Poprzez kopiec złożoność $O(E \cdot logV)$. Poprzez kopiec Fibonacciego złożoność $O(E + V \cdot logV)$.

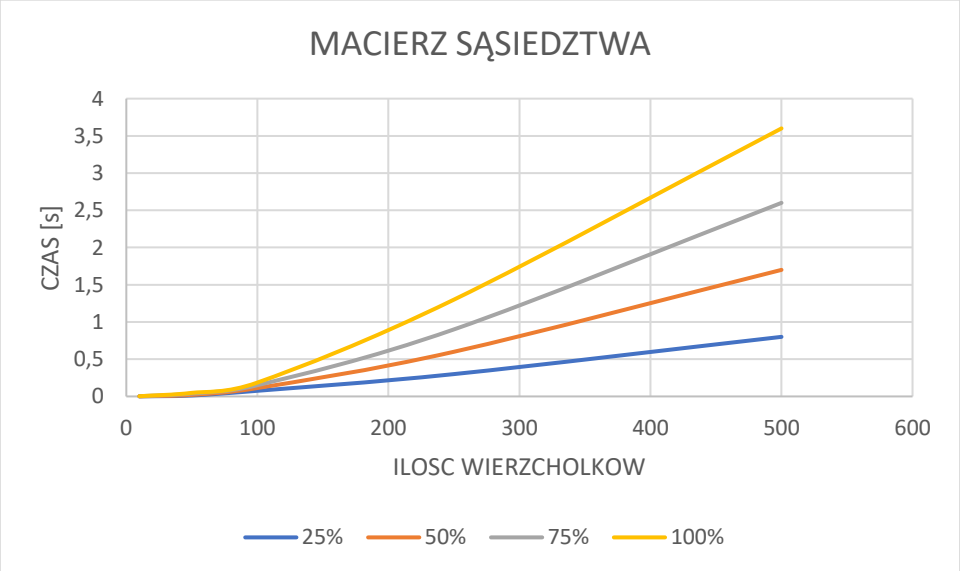
3. Przebieg badanego algorytmu

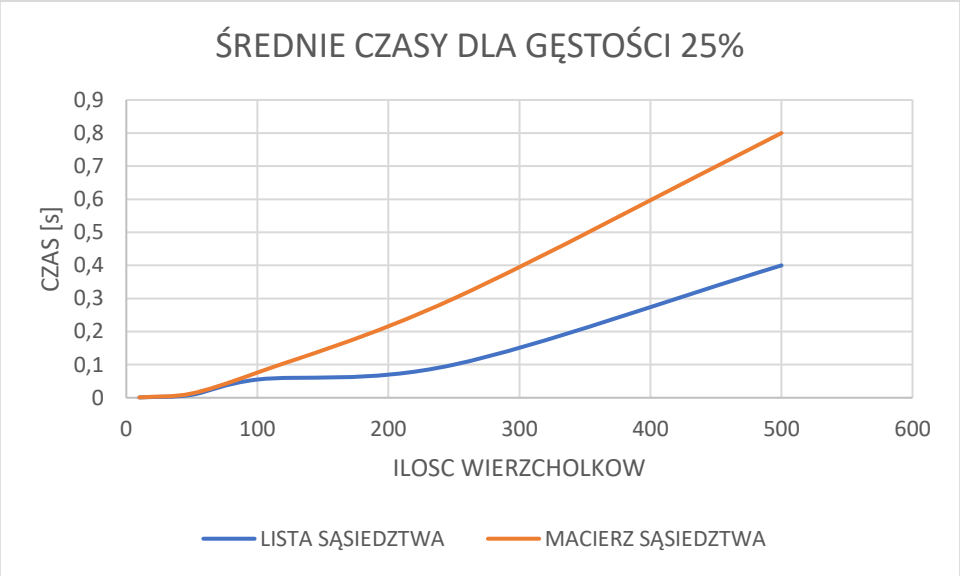
W obu tabelach podany jest średni czas w sekundach, a wykresy odnoszą się do wyników w tabelach.

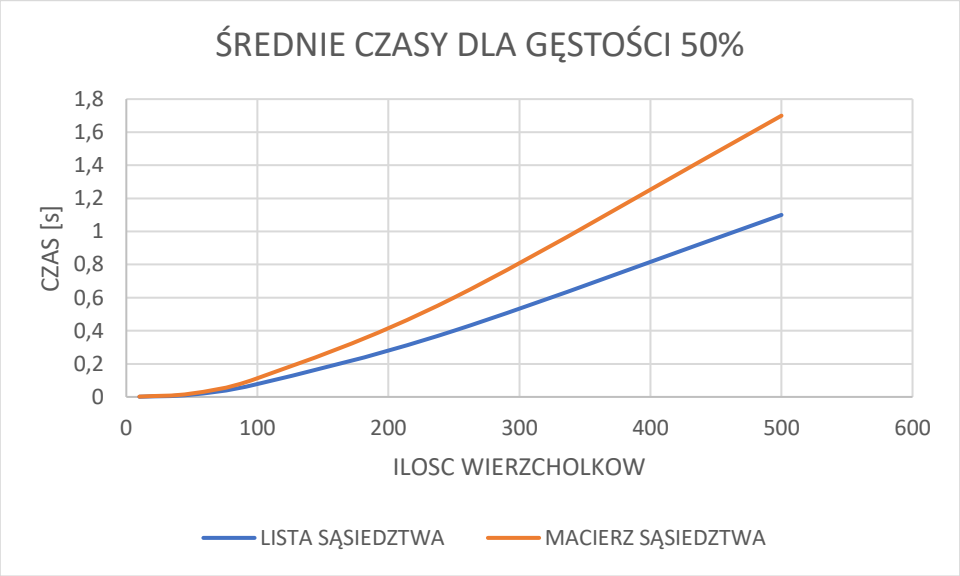
LISTA	10	50	100	250	500
25%	0,001	0,009	0,055	0,1	0,4
50%	0,001	0,014	0,078	0,4	1,1
75%	0,001	0,021	0,125	0,7	2,1
100%	0,001	0,033	0,183	1	3,1

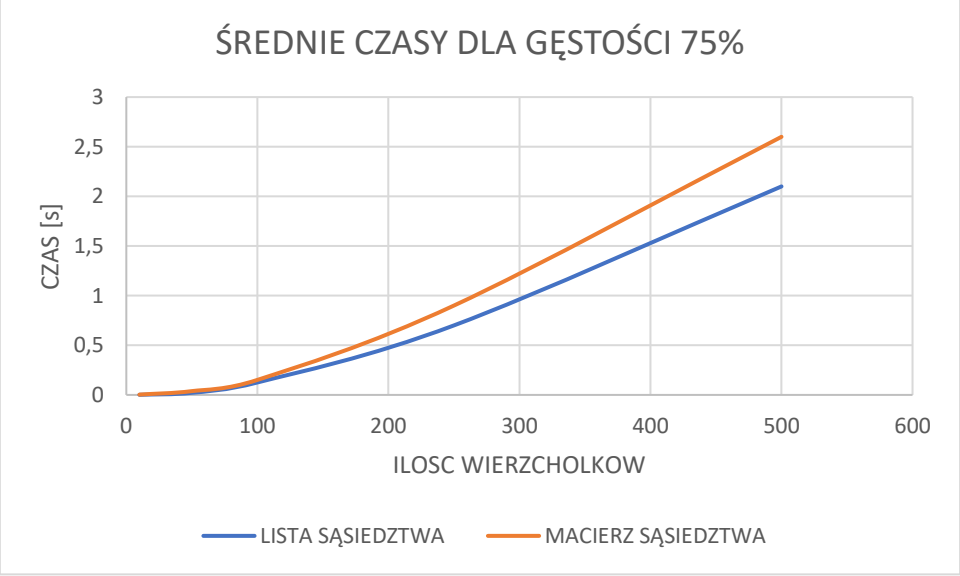
MACIERZ	10	50	100	250	500
25%	0,001	0,013	0,076	0,3	0,8
50%	0,002	0,021	0,113	0,6	1,7
75%	0,003	0,038	0,151	0,9	2,6
100%	0,003	0,048	0,186	1,3	3,6

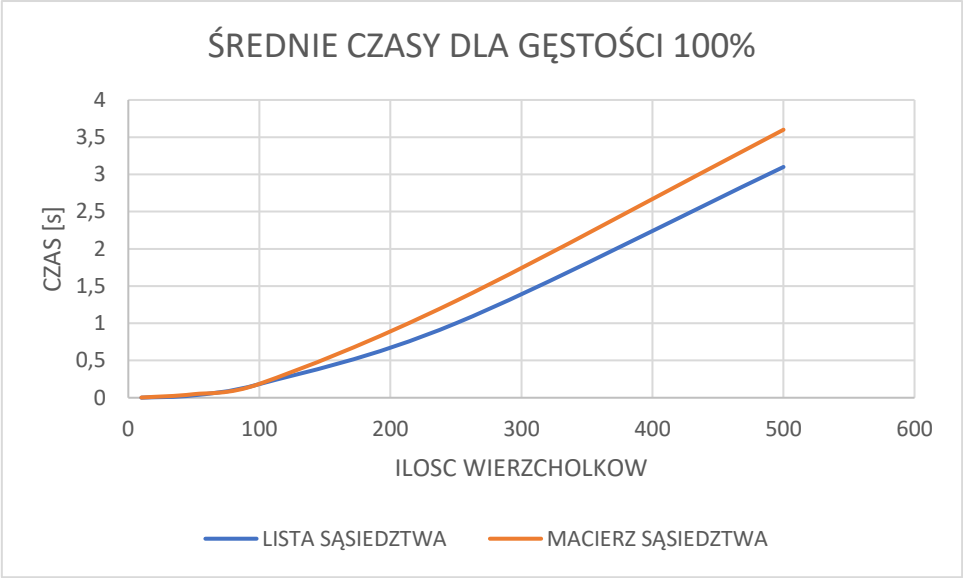





ŚREDNIE CZASY DLA GĘSTOŚCI 25%
CZAS [s]
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
0
100
200
300
400
500
600
ILOSC WIERZCHOLKOW
LISTA SĄSIEDZTWA
MACIERZ SĄSIEDZTWA


ŚREDNIE CZASY DLA GĘSTOŚCI 50%
CZAS [s]
1,8
1,6
1,4
1,2
1
0,8
0,6
0,4
0,2
0
0
100
200
300
400
500
600
ILOSC WIERZCHOLKOW
LISTA SĄSIEDZTWA
MACIERZ SĄSIEDZTWA


ŚREDNIE CZASY DLA GĘSTOŚCI 75%
CZAS [s]
3
2,5
2
1,5
1
0,5
0
0
100
200
300
400
500
600
ILOSC WIERZCHOLKOW
LISTA SĄSIEDZTWA
MACIERZ SĄSIEDZTWA



4. Podsumowanie

Z otrzymanych wykresów wynika, że czas działania algorytmu Dijkstry jest mniejszy jeśli zastosowany jest on na grafie reprezentowanym jako lista sąsiedztwa. Różnica ta rośnie wraz z liczbą wierzchołków i jest szczególnie widoczna dla dużych grafów. Algorytm ten nie wymaga odpowiedniego uporządkowania krawędzi przed rozpoczęciem wyznaczania drzewa rozpinającego, dlatego też jest bardziej ogólne w porównaniu do algorytmu Kruskala czy Jarnika-Prima. Algorytm Dijkstry nie działa, jeśli w grafie występują krawędzie z wagami ujemnymi. W tym przypadku używa się wolniejszego, lecz bardziej ogólnego algorytmu Bellmana-Forda.

5. Bibliografia

https://pl.wikipedia.org/wiki/Algorytm_Dijkstry
https://www.youtube.com/watch?v=5GT5hYzjNoo&t=172s
Drozdek A., C++. Algorytmy i struktury danych, Helion