Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Filip Kaśkos 249009

05.05.2020

Prowadzący: Mgr inż. Marcin Ochman

Termin: wtorek 15.15

1.Wprowadzenie

Zadanie polegało na zaimplementowaniu grafu za pomocą listy sąsiedztwa i macierzy sąsiedztwa oraz implementacji i sprawdzeniu algorytmu rozwiązującego problem najkrótszej ścieżki pomiędzy wierzchołkami grafu.

Graf to struktura składający się z wierzchołków oraz krawędzi które łączą owe wierzchołki. W zadaniu wykorzystany został graf ważony, w którym każde połączenie(krawędź) ma swoją wagę. Poruszanie się pomiędzy wierzchołkami takiego grafu nie jest oczywiste, gdyż bezpośrednia droga do najbliższego wierzchołka nie zawsze jest drogą najkrótsza(najszybszą).

Problem najkrótszej drogi można rozwiązać na różne sposoby. Wykorzystałem do tego algorytm Dijkstry. Algorytm ten służy do znajdowania najkrótszej ścieżki od wybranego wierzchołka do wszystkich pozostałych. Znajduje ścieżki pomiędzy wierzchołkami oraz oblicza koszt ich przejścia. Aby algorytm ten mógł działać wagi krawędzi w grafie muszą być nieujemne. Istnieją różne sposoby na implementację algorytmu Dijkstry. Każdy z nich ma inną złożoność obliczeniową. Wykorzystując tablicę możemy uzyskać złożoność $O(V^2)$, kopiec O(ElogV) lub kopiec Fibonacciego O(E+VlogV). Moja implementacja opiera się o działanie kolejki priorytetowej z STL.

Testowanie opierało się na zbadaniu czasu w jakim wykonywał się algorytm Dijkstry zarówno dla grafu reprezentowanego przez listę sąsiedztwa jak i macierz sąsiedztwa. Testom zostały poddane różne grafy:

- o ilości wierzchołków (10,50,100,250),

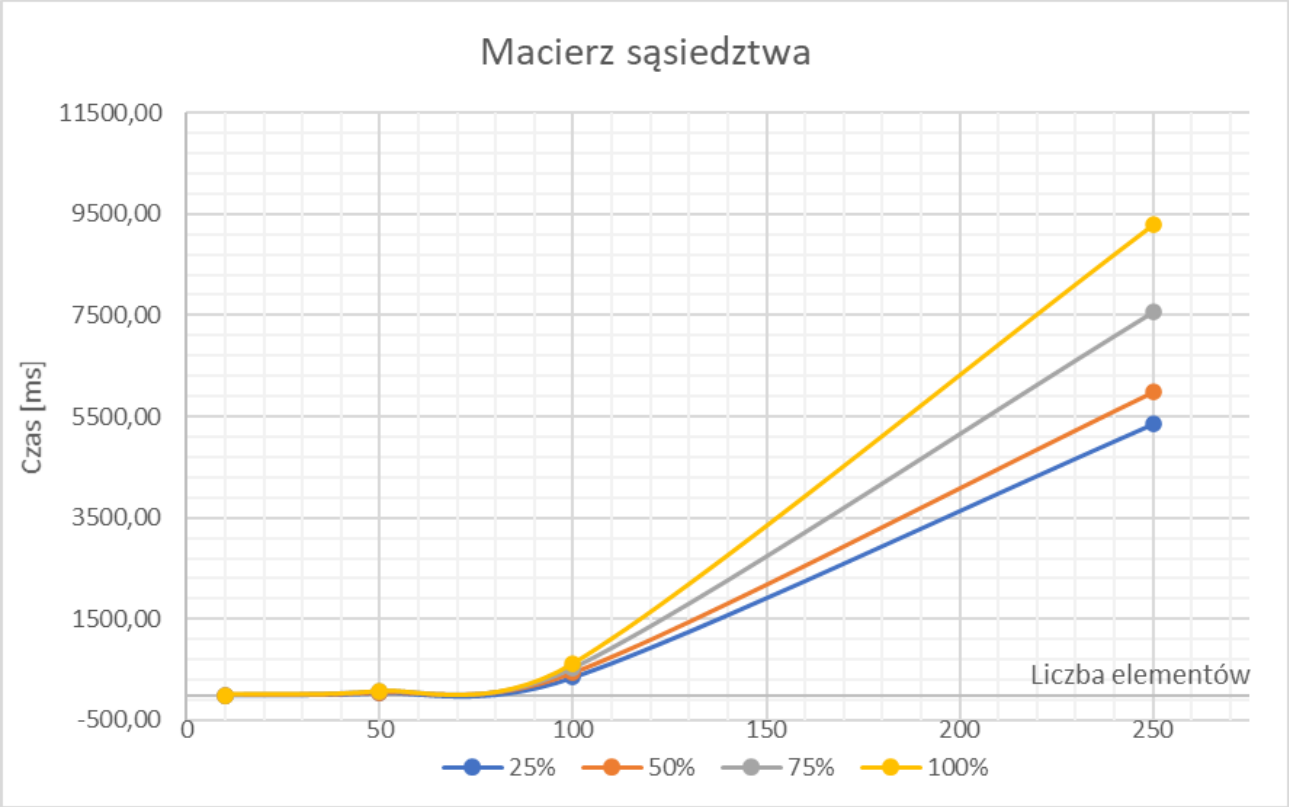
- o zadanej gęstości grafu (25%,50%,75%,100%).

Z próbki 100 grafów o zadanych właściwościach został obliczony czas maksymalny, minimalny oraz średni.

2. Macierz sąsiedztwa

Gestość	Elementy	10	50	100	250
25%	Min [ms]	0,00	30,00	166,00	2039,00
	Max [ms]	1,00	72,00	805,00	13139,00
	Średni [ms]	0,01	45,82	348,21	5347,30
50%	Min [ms]	0,00	37,00	221,00	3086,00
	Max [ms]	1,00	89,00	890,00	15115,00
	Średni [ms]	0,06	53,98	434,31	5985,73
75%	Min [ms]	0,00	44,00	337,00	4213,00
	Max [ms]	1,00	113,00	1000,00	13893,00
	Średni [ms]	0,16	68,14	520,50	7556,28
100%	Min [ms]	0,00	47,00	441,00	6931,00
	Max [ms]	0,00	101,00	1010,00	14313,00
	Średni [ms]	0,14	66,78	610,12	9290,81

Tabela 1. Wyniki testów przeprowadzonych dla macierzy sąsiedztwa

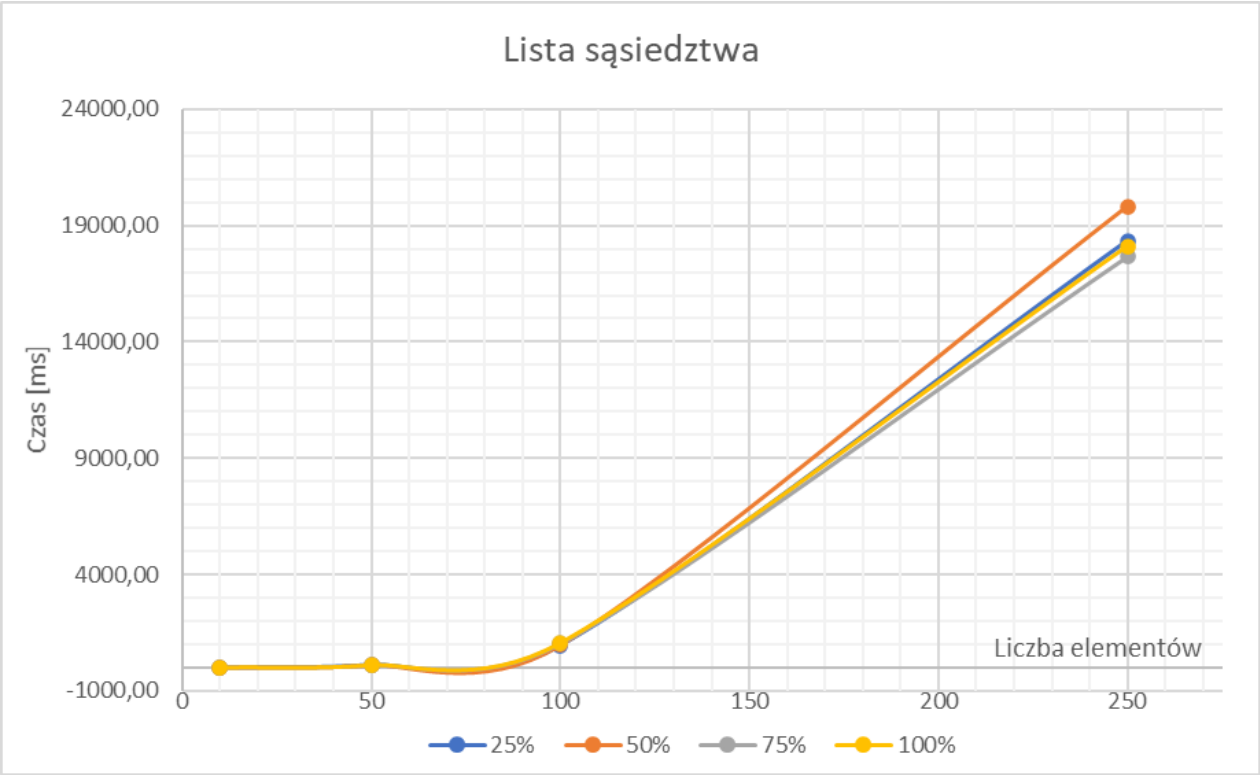


Wykres 1. Zależność czasu działania od ilości wierzchołków oraz gęstości

3.Lista sąsiedztwa

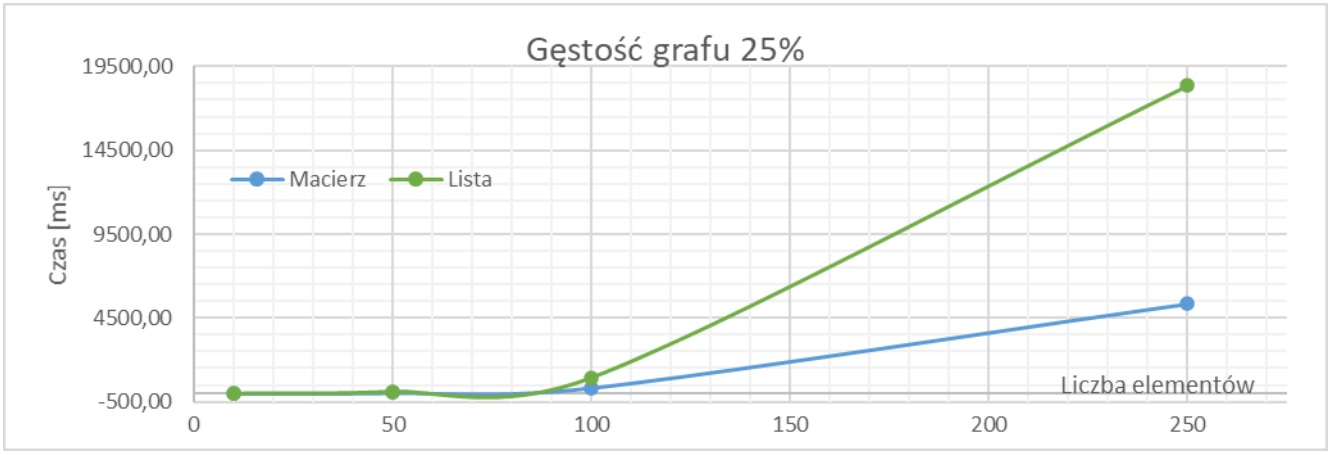
Gestość	Elementy	10	50	100	250
25%	Min [ms]	0,00	64,00	510,00	13755,00
	Max [ms]	1,00	378,00	1768,00	22777,00
	Średni [ms]	0,11	117,36	949,60	18339,00
50%	Min [ms]	0,00	62,00	579,00	16152,00
	Max [ms]	12,00	203,00	1532,00	24212,00
	Średni [ms]	0,26	106,21	967,17	19819,20
75%	Min [ms]	0,00	68,00	650,00	13763,00
	Max [ms]	13,00	200,00	1749,00	22202,00
	Średni [ms]	0,66	108,83	1026,39	17659,50
100%	Min [ms]	0,00	65,00	693,00	14310,00
	Max [ms]	12,00	215,00	1831,00	21839,00
	Średni [ms]	0,63	109,06	1061,31	18103,80

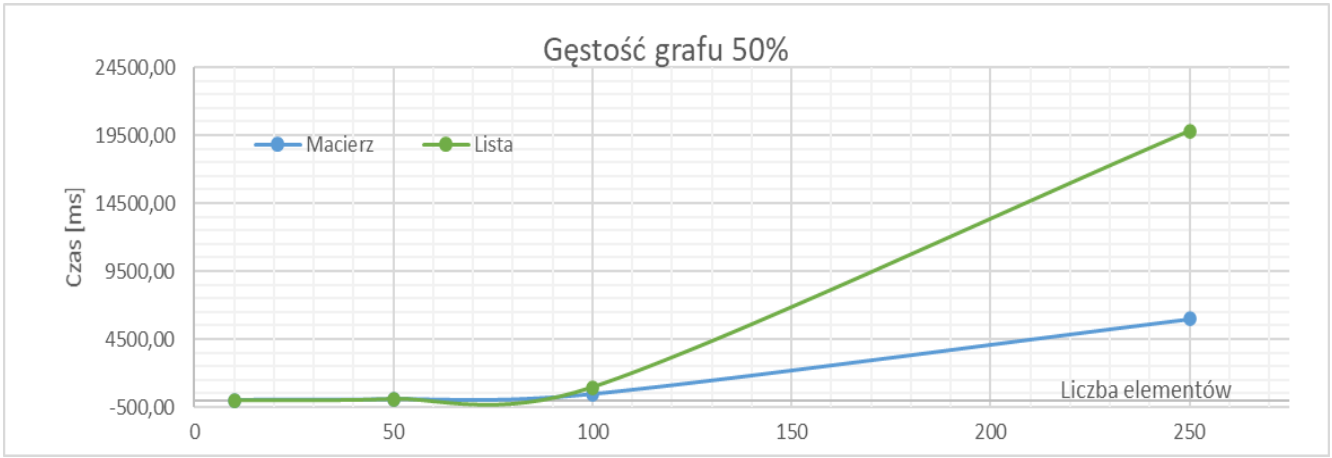
Tabela 2. Wyniki testów przeprowadzonych dla listy sąsiedztwa

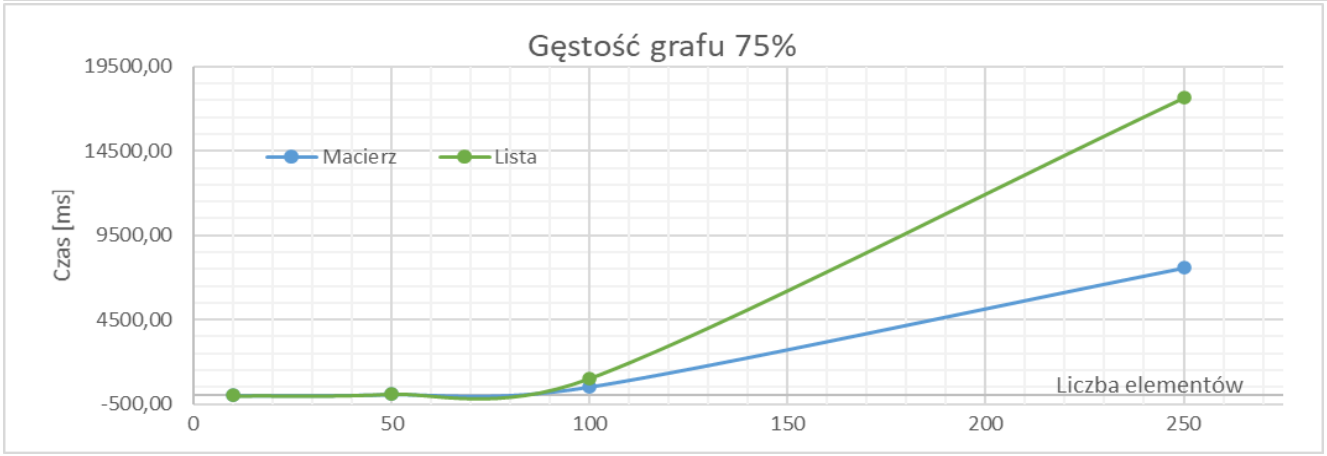


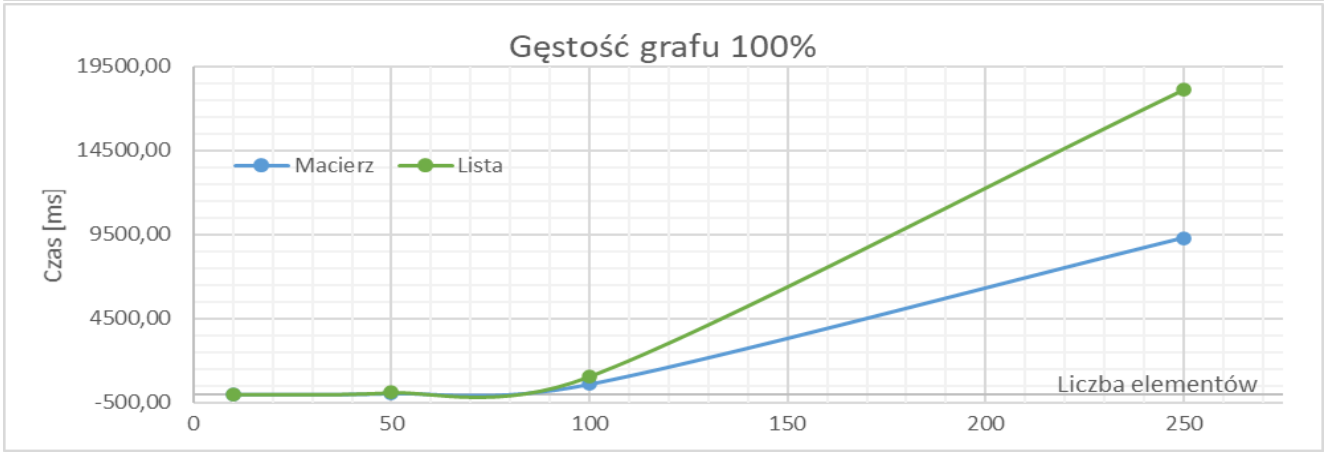
Wykres 2.Zaleśność czasu działania od ilości wierzchołków oraz gęstości

4.Porównanie









5.Wnioski

Na wykresach widać zależność pomiedzy gestością grafu a czasem działania algorytmu Dijkstry. Dla macierzy sąsiedztwa im większa gęstość grafu tym dłuższy czas działania. Wiąże się to z większa ilościa krawędzi, które algorytm musi sprawdzać. Dla listy sąsiedztwa różnice nie są aż tak widoczne a czasmi nawet mniejsza gęstość grafu powoduje dłuższy czas. Może to wynikać z losowości wypełenienia grafu. Widać także, że czas działania algorytmu jest wyraźnie mniejszy dla grafu zaimplementowanego przez macierz sąsiedztwa niż listę sąsiedztwa.