Michał Więcek
248956
05.05.2020

Projektowanie Algorytmów i Metody Sztucznej Inteligencji
Projekt 2

Mgr inż. Marcin Ochman
Grupa: Wtorek 15^{15}-16^{55}

1. Wstęp

Celem projektu jest porównanie w jaki sposób reprezentacja grafu w programie wpływa na wydajność algorytmów grafowych. Badany jest problem najkrótszej ścieżki. Polega on na znalezieniu w grafie ważonym najkrótszej możliwej ścieżki między dwoma wierzchołkami. Wykorzystane reprezentacje: lista sąsiedztwa oraz macierz sąsiedztwa. Do rozwiązania tego problemu użyłem algorytmu Dijkstry.

2. Lista sąsiedztwa

Reprezentacja ta polega na stworzeniu tablicy list o wymiarze $\boldsymbol{V}$. Gdzie $\boldsymbol{V}$ oznacza liczbę wierzchołków. Na każdej liście przechowywane są indeksy wierzchołków końcowych, czyli sąsiadów wierzchołka początkowego. Zaletą tej reprezentacji jest mały wymóg pamięci. Wadą jest duża złożoność obliczeniowa w momencie kiedy chcemy dostać się do konkretnej krawędzi.

3. Macierz sąsiedztwa

Reprezentacja ta polega na stworzeniu macierzy o wymiarze $\boldsymbol{V}$ x $\boldsymbol{V}$. Wadą tej reprezentacji jest duży wymóg pamięci. Zaletą jest natychmiastowy dostęp do określonej krawędzi.

4. Algorytm Dijkstry

Algorytm znajduje najkrótsze ścieżki między źródłem jakie ustalimy a wszystkimi wierzchołkami w grafie. Przy okazji wylicza koszt dotarcia do wierzchołków. Można go łatwo zmodyfikować żeby wyszukiwał tylko drogę do jednego wierzchołka. Można zaimplementować ten algorytm na trzy podane sposoby:
Przy użyciu zwykłej **tablicy**. Złożoność – $O(V^2)$
Przy użyciu **kolejki priorytetowej**. Złożoność – $O(E * \log V)$
Przy użyciu **kopca Fibonacciego**. Złożoność – $O(E + \log V)$
W projekcie wykorzystany został algorytm działający przy użyciu kolejki priorytetowej.

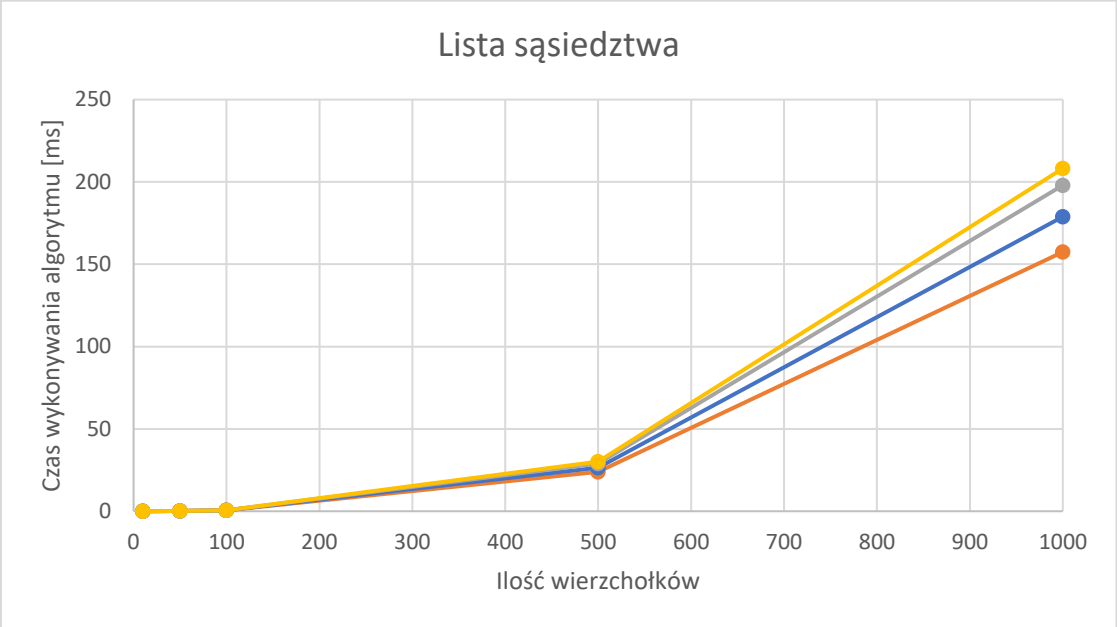
5. Przebieg eksperymentów

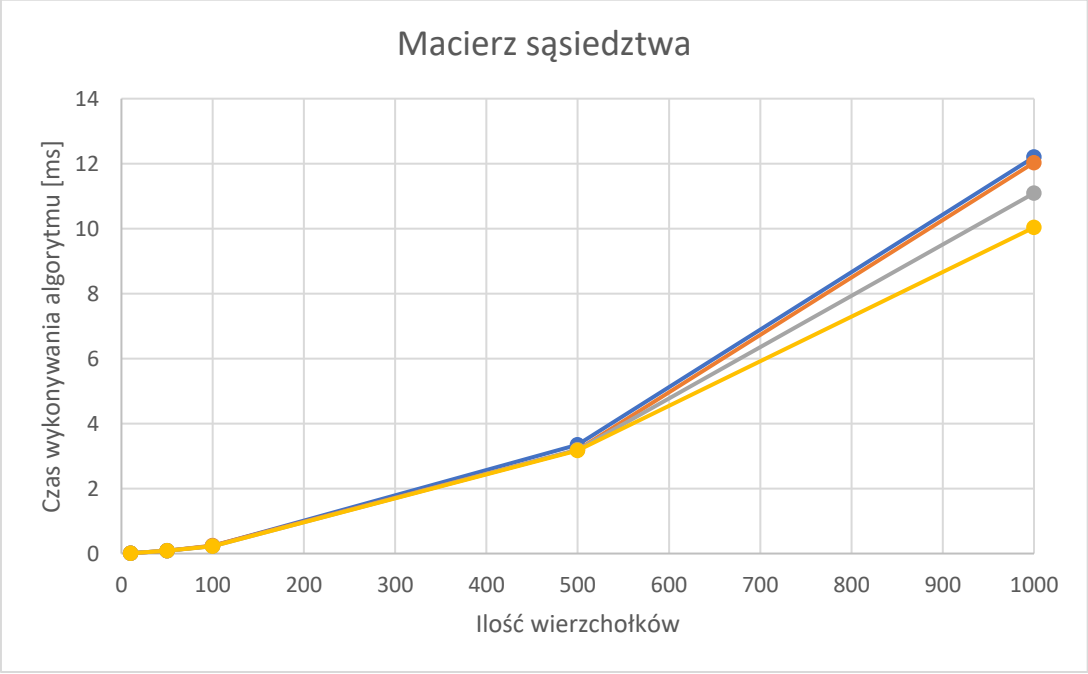
Lista sąsiedztwa				
Ilość wierzchołków	Gęstość			
	0,25	0,5	0,75	1
	Czas wykonywania algorytmu [ms]			
10	0,010	0,009	0,009	0,009
50	0,115	0,121	0,129	0,130
100	0,507	0,543	0,662	0,601
500	23,825	26,448	28,936	30,207
1000	157,454	178,858	197,958	208,156

Michał Więcek
248956

05.05.2020

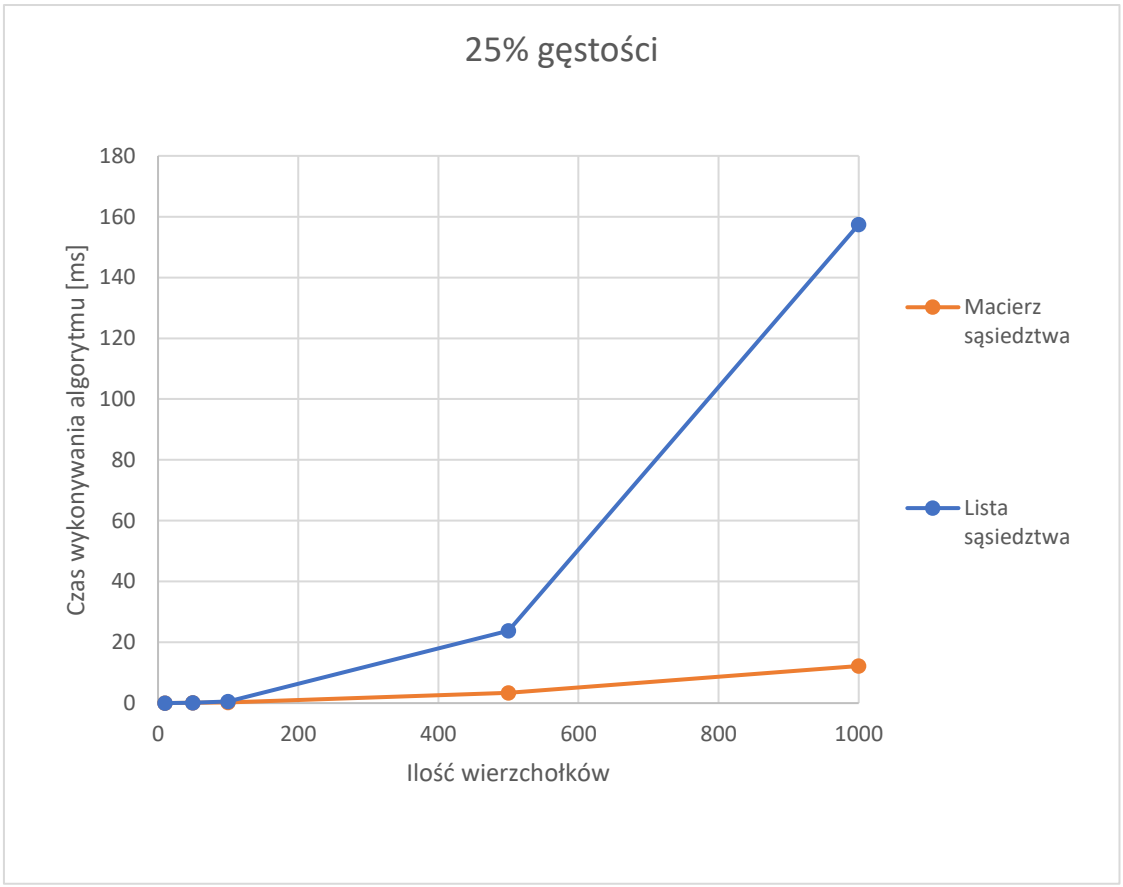
Macierz sąsiedztwa				
Ilość wierzchołków	Gęstość			
	0,25	0,5	0,75	1
	Czas wykonywania algorytmu [ms]			
10	0,011	0,010	0,010	0,010
50	0,088	0,085	0,084	0,083
100	0,232	0,245	0,228	0,226
500	3,346	3,194	3,195	3,170
1000	12,201	12,031	11,091	10,037

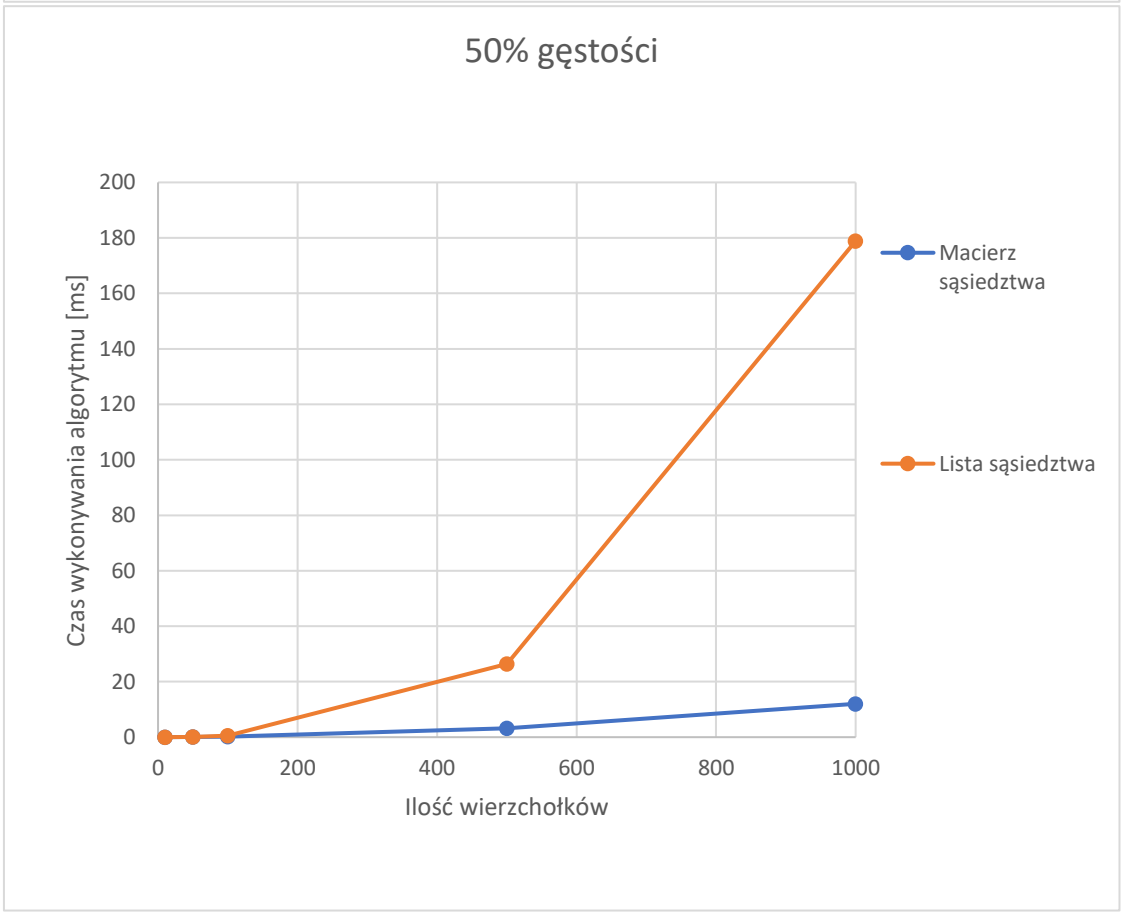




Michał Więcek
248956

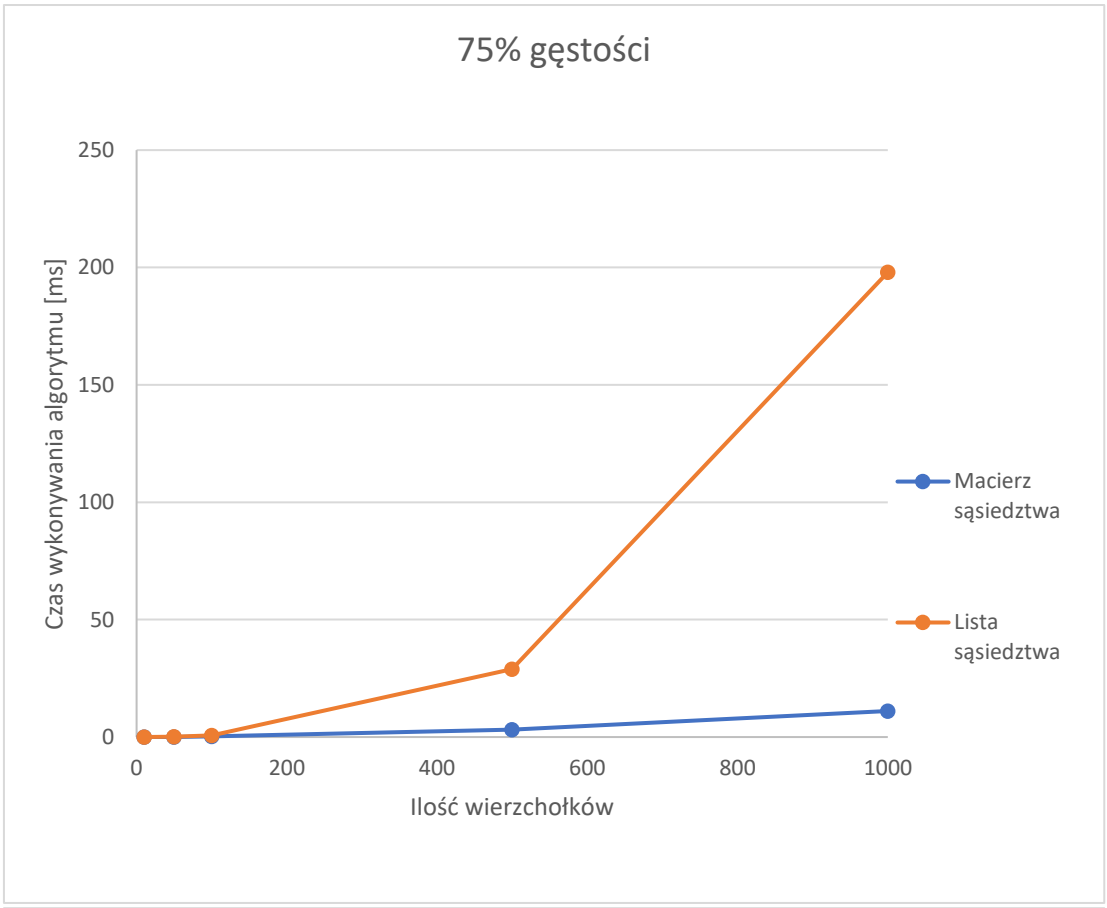
05.05.2020

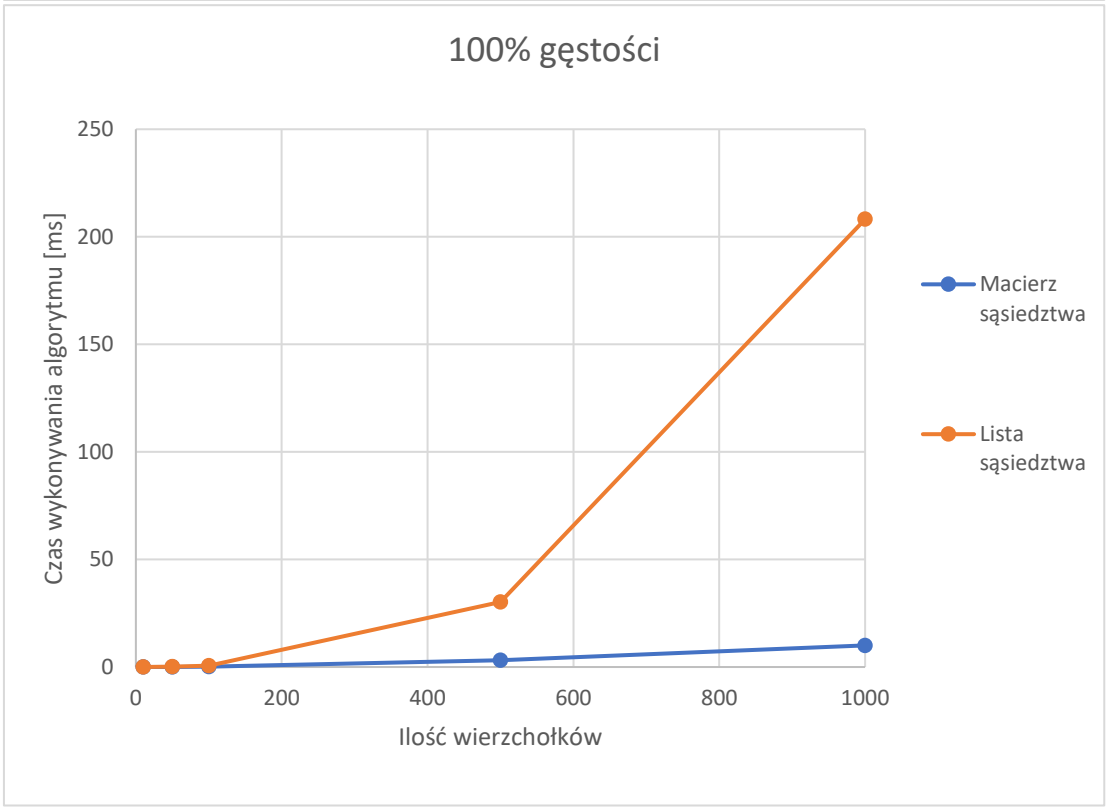




Michał Więcek
248956

05.05.2020





6. Wnioski:

Na podstawie wykonanych badań można stwierdzić, że przy badaniu problemu optymalna okazała się macierz sąsiedztwa. Również zmiana liczby krawędzi grafu nie miała większego wpływu na macierz. W przypadku listy można zauważyć duże zmiany podczas zwiększania liczby krawędzi.

7. Literatura:

https://pl.wikipedia.org/wiki/Algorytm_Dijkstry
https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
http://algorytmy.ency.pl/tutorial/algorytm_dijkstry
https://pl.wikipedia.org/wiki/Graf_(matematyka)
https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)
https://graphonline.ru/en/

8. Dodatkowo

Sprawdziłem czas wykonywania się tego algorytmu na różnych konfiguracjach sprzętowych. I mniej więcej porównałem je z tymi zaprezentowanymi w sprawozdaniu.

1. Intel Core i7-5600U 2.6GHz, Win10, Laptop – Opisane wyniki.
2. Intel Core i5-4460 3.2GHz, Win10, PC – Około o 8% wolniej.
3. Intel Core i5-8250U 1.6 GHz, Win10, Laptop - Około 40% wolniej.
4. AMD Athlon II X4 645 3.10GHz, Win7, PC – Około 100% wolniej.
5. Ryzen 5 3600X 3.8GHz ,Win10, PC – Dla wszystkich wyników około 40% szybciej **ale** dla największego rozmiaru listy około 50% wolniej.