Projektowanie algorytmów i metody sztucznej inteligencji

Prowadzący	Wykonawca	Nr. Indeksu	Data
dr inż. Łukasz Jeleń	Patryk Marciniak	248978	30.04.2020

1 Wstęp

Projekt polegał na zaimplementowaniu i przetestowaniu jednego z algorytmów wyznaczających najkrótszą drogę z danego wierzchołka grafu ważonego i skierowanego do każdego innego wierzchołka. Algorytmy, które były do wyboru to:

- Algorytm Dijkstry
- Algorytm Bellmana-Forda

Ponadto grafy, na których wykonane powinny być testy powinny także różnić się od siebie sposobem reprezentacji, gęstością oraz liczbą wierzchołków. Testowane reprezentacje grafów to:

- Macierz sąsiedztwa
- Lista sąsiedztwa

Natomiast testowane gęstości to:

- 25%
- 50%
- 75%
- 100%

Ilości wierzchołków wykorzystywane do testów mogły być dowolnie dobrane. Do poniższych testów wykorzystano następujące ilości:

- 20
- 50
- 100
- 250
- 500

Dla każdego zestawu parametrów wejściowych zostało wygenerowane po 100 losowych instancji, na których przeprowadzono testy i zapisano wyniki uśrednione.

2 Opis algorytmu

Algorytm Bellmana-Forda w odróżnieniu od algorytmu Dijkstry obsługuje grafy, w których występują wagi ujemne, jednak nie może w nim wystąpić cykl ujemny. Mimo tego jest algorytmem wolniejszym od algorymtu Dijkstry.
Złożoność obliczeniowa algorytmu Bellman-Forda:

$$\begin{aligned} &\text{Dla macierzy sąsiedztwa} &&\longrightarrow \quad \mathcal{O}(|V|^3) \\ &\text{Dla listy sąsiedztwa} &&\longrightarrow \quad \mathcal{O}(|V| \times |E|) \end{aligned}$$

Złożoność pamięciowa algorytmu:

$$\begin{aligned} &\text{Dla macierzy sąsiedztwa} &&\longrightarrow \quad \mathcal{O}(|V|^2) \\ &\text{Dla listy sąsiedztwa} &&\longrightarrow \quad \mathcal{O}(|V| + |E|) \end{aligned}$$

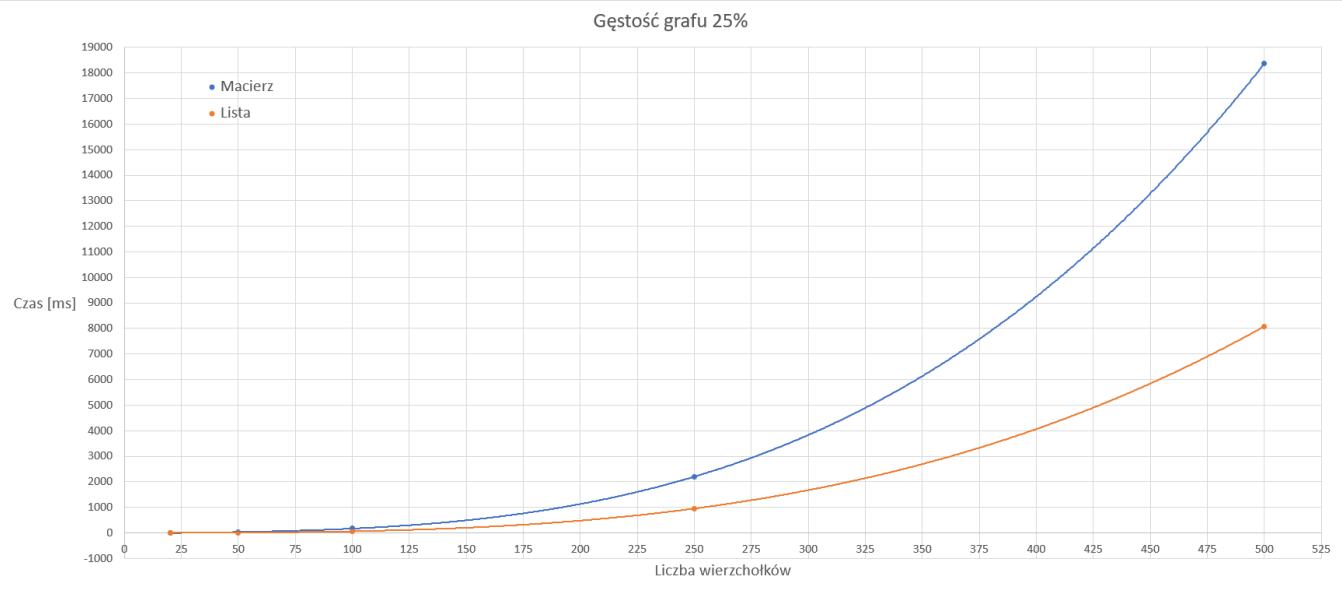
Z powyższych danych wynika, że na złożoność obliczeniową w przypadku macierzy sąsiedztwa wpływa jedynie liczba wierzchołków, co sugeruje, że zmiana gęstości grafu nie powinna wpływać na czas realizacji algorytmu. Z kolei w przypadku listy sąsiedztwa, dla której złożoność obliczeniowa zależy zarówno od liczby wierzchołków jak i krawędzi, czas wykonania algorytmu będzie zależał od gęstości. Co więcej zauważyć można, że dla mniejszych gęstości, ten sposób reprezentacji grafu będzie efektywniejszy.

3 Otrzymane wyniki

3.1 Gęstość grafu 25%

L. wierzchołków / Sposób reprezentacji	20	50	100	250	500
Macierz sąsiedztwa [ms]	1,12	18,18	176,33	2200,07	18364,30
Lista sąsiedztwa [ms]	0,42	7,31	63,64	952,20	8085,86

Tabela 1: Tabela czasów realizacji algorytmu dla gęstości grafu 25%

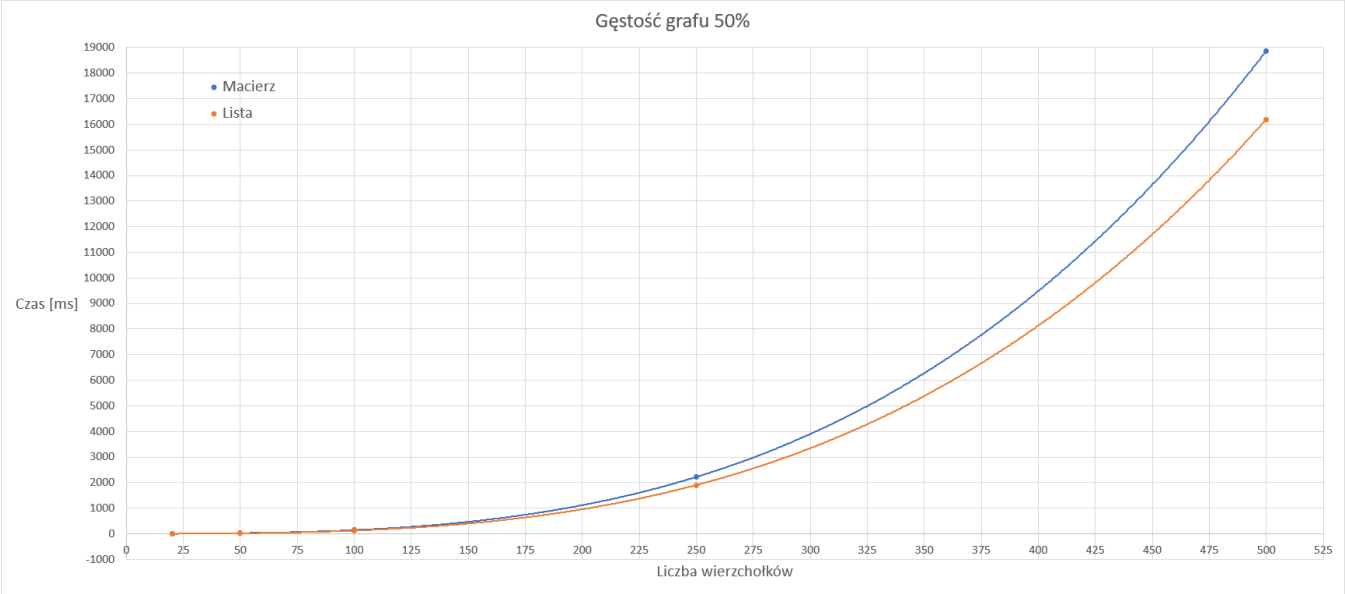


Rysunek 1: Wykres dla gęstości grafu 25%

3.2 Gęstość grafu 50%

L. wierzchołków / Sposób reprezentacji	20	50	100	250	500
Macierz sąsiedztwa [ms]	1,02	17,35	148,96	2220,30	18876,90
Lista sąsiedztwa [ms]	0,83	14,59	126,81	1903,76	16190,50

Tabela 2: Tabela czasów realizacji algorytmu dla gęstości grafu 50%

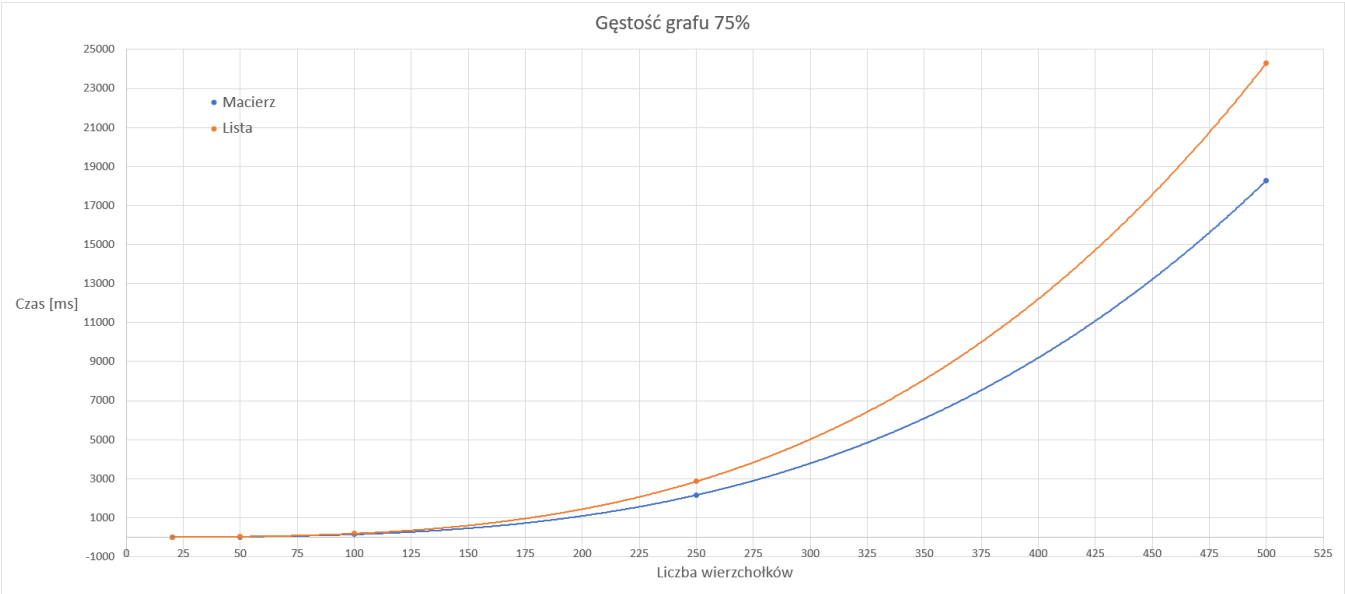


Rysunek 2: Wykres dla gęstości grafu 50%

3.3 Gęstość grafu 75%

L. wierzchołków / Sposób reprezentacji	20	50	100	250	500
Macierz sąsiedztwa [ms]	1,00	16,93	145,10	2158,09	18282,00
Lista sąsiedztwa [ms]	1,25	21,87	190,28	2855,77	24299,90

Tabela 3: Tabela czasów realizacji algorytmu dla gęstości grafu 75%

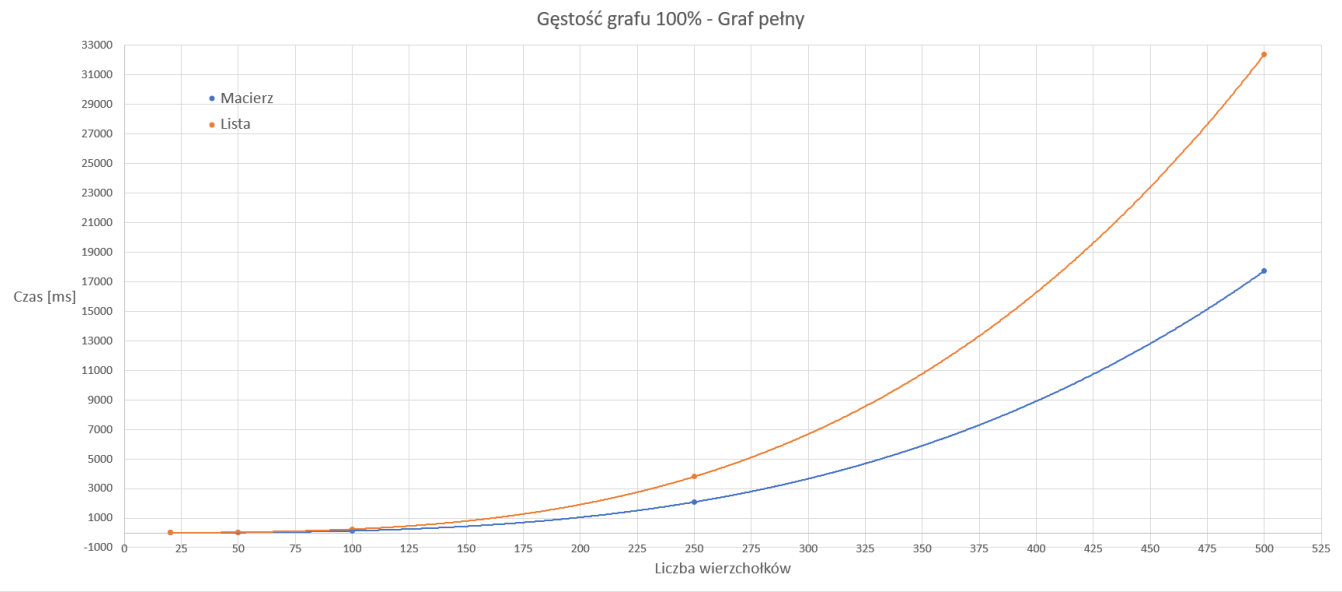


Rysunek 3: Wykres dla gęstości grafu 75%

3.4 Gęstość grafu 100% (graf pełny)

L. wierzchołków / Sposób reprezentacji	20	50	100	250	500
Macierz sąsiedztwa [ms]	0,96	16,34	140,52	2087,66	17735,10
Lista sąsiedztwa [ms]	1,66	29,17	253,67	3807,33	32393,70

Tabela 4: Tabela czasów realizacji algorytmu dla grafu pełnego%

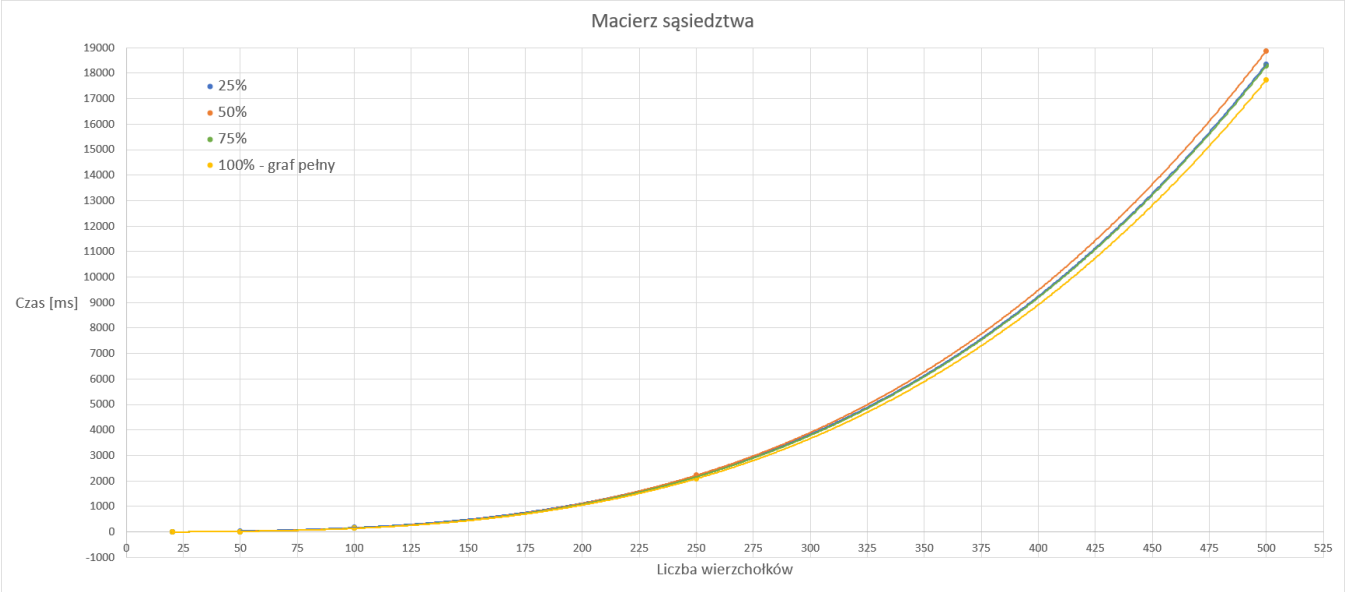


Rysunek 4: Wykres dla grafu pełnego%

3.5 Sposób reprezentacji grafu - macierz sąsiedztwa

L. wierzchołków / Gęstość	20	50	100	250	500
25% [ms]	1,12	18,18	176,33	2200,07	18364,30
50% [ms]	1,02	17,35	148,96	2220,30	18876,90
75% [ms]	1,00	16,93	145,10	2158,09	18282,00
100% [ms]	0,96	16,34	140,52	2087,66	17735,10

Tabela 5: Tabela czasów realizacji algorytmu dla grafu reprezentowanego za pomocą macierzy sąsiedztwa

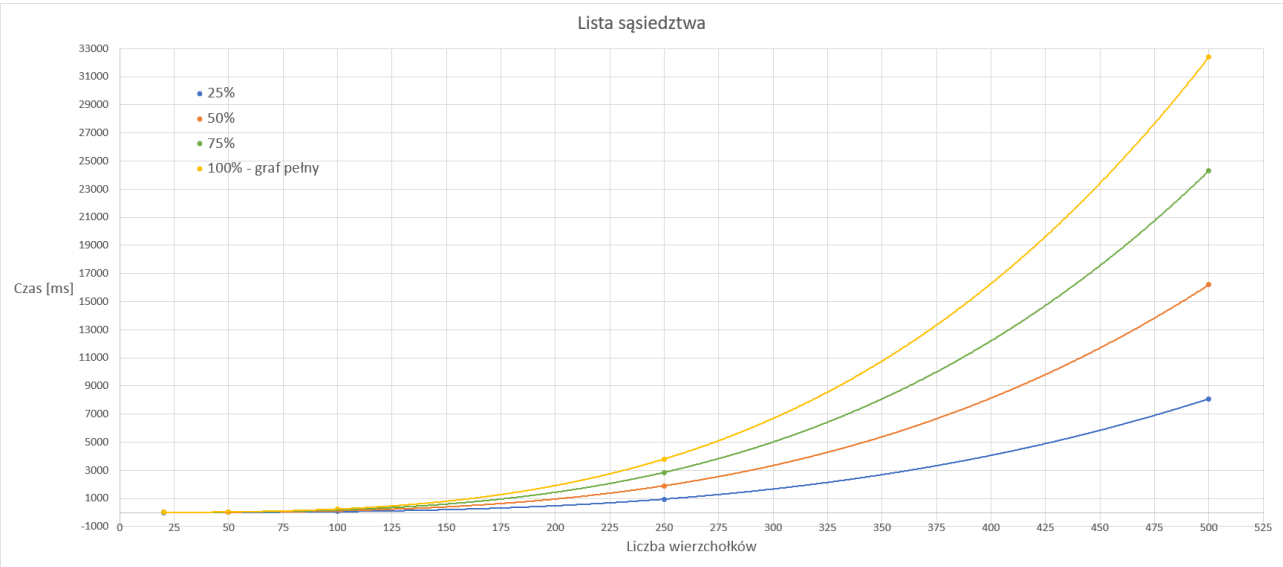


Rysunek 5: Wykres dla grafu reprezentowanego za pomocą macierzy sąsiedztwa

3.6 Sposób reprezentacji grafu - lista sąsiedztwa

L. wierzchołków / Gęstość	20	50	100	250	500
25% [ms]	0,42	7,31	63,64	952,20	8085,86
50% [ms]	0,83	14,59	126,81	1903,76	16190,50
75% [ms]	1,25	21,87	190,28	2855,77	24299,90
100% [ms]	1,66	29,17	253,67	3807,33	32393,70

Tabela 6: Tabela czasów realizacji algorytmu dla grafu reprezentowanego za pomocą listy sąsiedztwa



Rysunek 6: Wykres dla grafu reprezentowanego za pomocą listy sąsiedztwa

4 Podsumowanie i wnioski

- Zgodnie z przypuszczeniami, gęstość grafu nie miała wpływu na czas wykonywania algorytmu, przy korzystaniu z macierzy, jako metody reprezentacji grafu. Co ciekawe, dla gęstości o większych wartościach (powyżej 50%), czas realizacji był nawet mniejszy od czasu realizacji przy mniejszych gęstościach. Powyższe wnioski ilustruje wykres 5.
- Zgadza się również przypuszczenie, że wraz ze wzrostem gęstości grafu, czyli ilości krawędzi, czas realizacji algorytmu, używając listy sąsiedztwa, jako metody reprezentacji grafu, rośnie. Da się zauważyć, że przyrost czasu, jest prawie wprost proporcjonalny do gęstości. Powyższe wnioski widać na wykresie 6.
- Algorytm potrzebuje więcej czasu, przy implementacji grafu jako macierzy, lecz tylko dla mniejszych gęstości (poniżej 50%). Dla pozostałych gęstości przez przyrost krawędzi, implementacja za pomocą listy staje się wolniejsza. Wnioskować z tego można fakt, że dla grafów o małych gęstościach, opłacalną metodą reprezentacji jest lista sąsiedztwa. Powyższe wnioski wyciągnąc możemy poprzez analizę wykresów 1-4.

5 Literatura

- https://www.programiz.com/dsa/graph-adjacency-matrix
- https://www.geeksforgeeks.org/graph-and-its-representations/
- https://www.programiz.com/dsa/graph-adjacency-list
- https://algorithms.tutorialhorizon.com/graph-representation-adjacency-matrix-and-adjacency-list/
- https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/
- https://www.programiz.com/dsa/bellman-ford-algorithm