Politechnika Wrocławska

Projektowanie algorytmów i metody sztucznej inteligencji
Autor: Piotr Kuboń 252871
Kod grupy: E12-99c
Prowadzący: Mgr inż. Marta Emirsajłow
Data zajęć: 10.05.2021

1. Wprowadzenie

Pomiary wykonano dla mniejszych liczb wierzchołków. Spowodowane jest to faktem pojawiania się błędów systemu przy alokacji pamięci o znacznym rozmiarze.

Celem projektu była zaimplementowanie grafów, czyli zbioru wierzchołków, które mogą być połączone krawędziami oraz implementacja algorytmu szukania najkrótszej ścieżki w grafie. W celu realizacji projektu zostały zmierzone czasy szukania ścieżek w 100 grafach, a następnie wyniki zostały uśrednione. Rozpatrzone zostały przypadki grafów:

- O różnym rozmiarze (liczbie wierzchołków): 10, 50, 100, 200, 300
- O różnej gęstości: 25%, 50%, 75%, 100%
- O różnych reprezentacjach: macierz sąsiedztwa oraz lista sąsiedztwa

Badania przeprowadzono przy użyciu algorytmu Dijkstra.

2. Opis algorytmu Dijkstry

Pierwszym krokiem algorytmu jest ustawienie wartości etykiety długości ścieżki dla każdego wierzchołka na nieskończoność (czyli w programie wybrać liczbę bardzo dużą), natomiast etykieta długości ścieżki wierzchołka startowego powinna zostać ustawiona na 0. Następnym krokiem jest wyciągnięcie ze zbioru wierzchołka o najmniejszej wartości dystansu (etykiecie ścieżki), czyli przy pierwszym wykonaniu jest to wierzchołek startowy. W informatyce do przeprowadzenia sortowania dystansu wierzchołków zaleca się korzystanie z kolejki priorytetowej o budowie kopca, która jest w stanie znacznie zwiększyć efektywność algorytmu. Kolejnym krokiem jest porównanie ścieżki dla wszystkich sąsiadów danego wierzchołka oraz ewentualne przypisanie nowych wartości dystansu, gdy znaleziono krótszą ścieżkę. Zaleca się stosowanie kopca, jednak z przyczyn dostępności opracowań, algorytm zaimplementowano bez jego użycia.

W przypadku zastosowania przeszukiwania liniowego złożoność może sięgnąć $O(|V|\ 2\ )$, natomiast w przypadku zastosowania kolejki priorytetowej o budowie kopca, zmniejsza się ona do $O(|E| + |V|log|V|)$.

3. Prezentacja wyników

Uśredniony czas podano w milisekundach.

Algorytm z grafem zaimplementowanym w postaci macierzy.

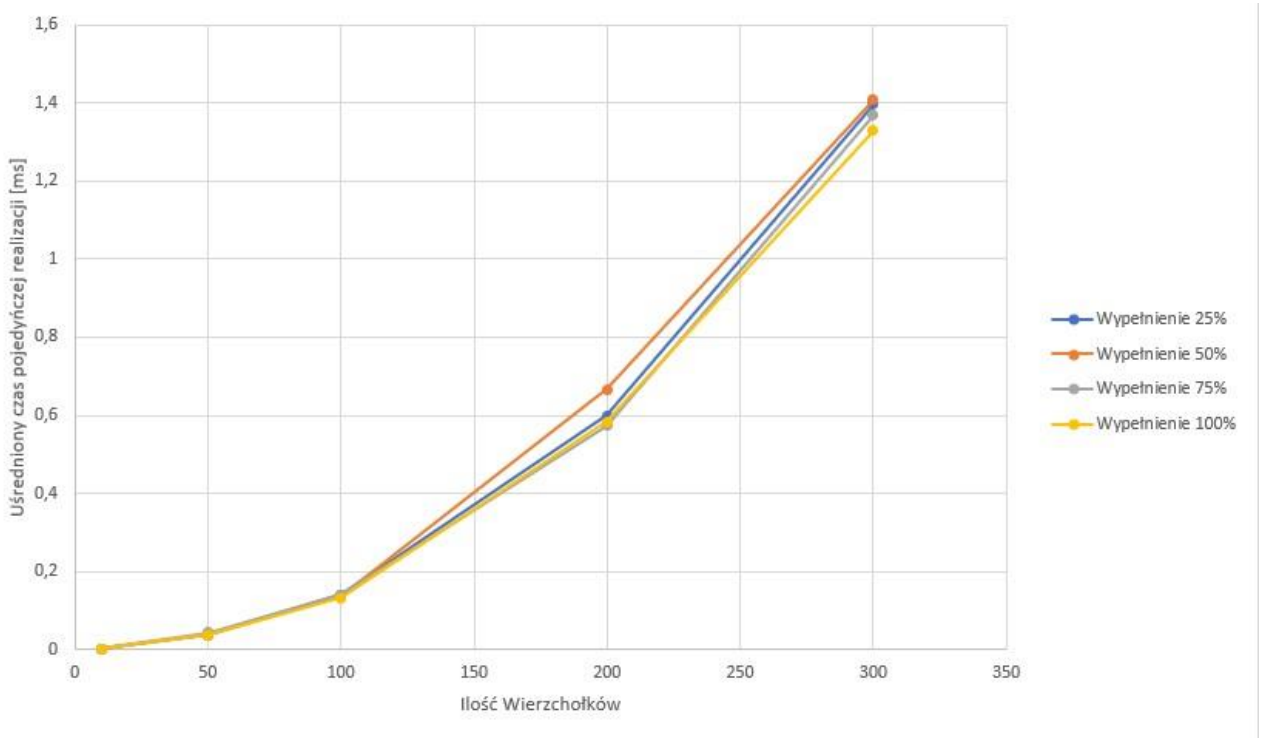
Gęstość grafu	Liczba wierzchołków				
	10	50	100	200	300
25%	0,002326	0,038014	0,140635	0,600173	1,39678
50%	0,002524	0,038144	0,136837	0,666	1,40733
75%	0,0022	0,042973	0,140602	0,572854	1,36856
100%	0,002614	0,037338	0,13346	0,5832	1,32832

Algorytm z grafem zaimplementowanym w postaci listy.

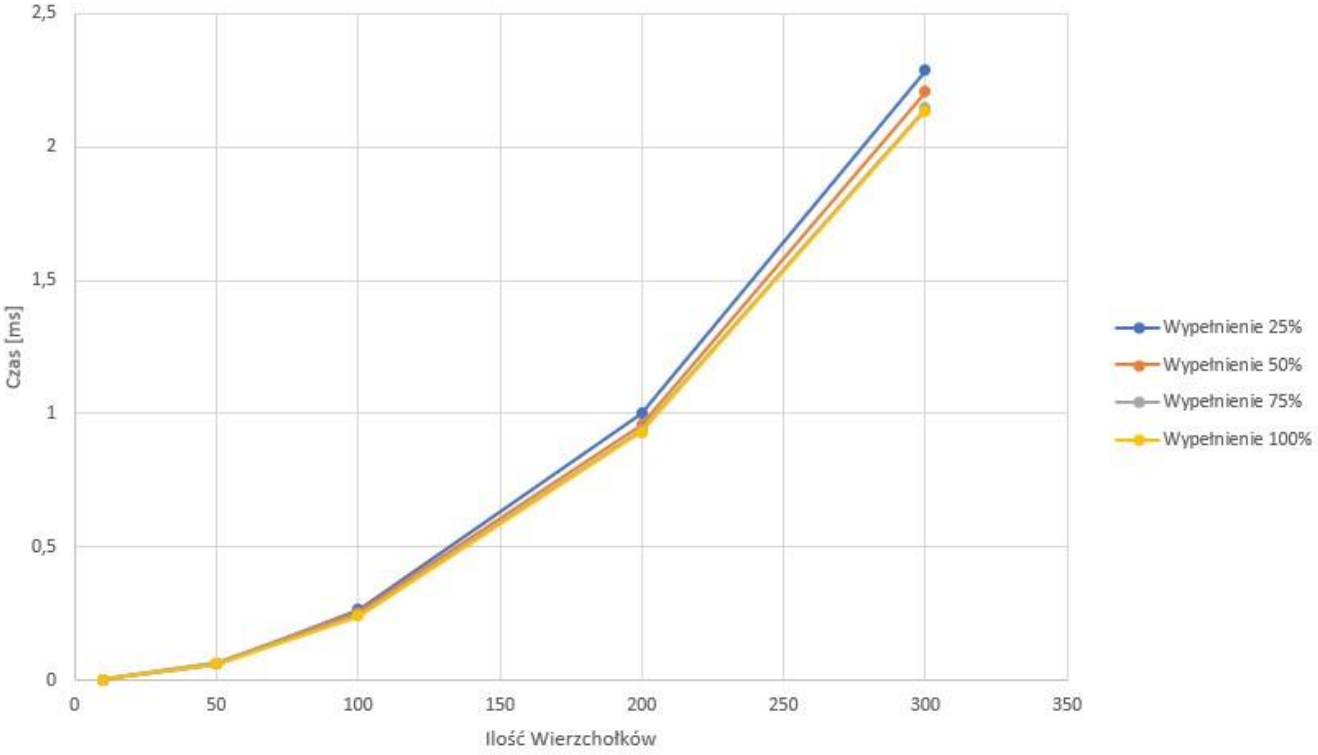
Gęstość grafu	Liczba wierzchołków				
	10	50	100	200	300
25%	0,00337	0,061562	0,266461	1,00216	2,28488
50%	0,003201	0,064863	0,258	0,958443	2,20667
75%	0,003963	0,066	0,2462	0,939866	2,144
100%	0,003442	0,062635	0,239824	0,93065	2,13514

3.1. Wykresy

Wykres algorytmu z grafem zaimplementowanym w postaci macierzy.

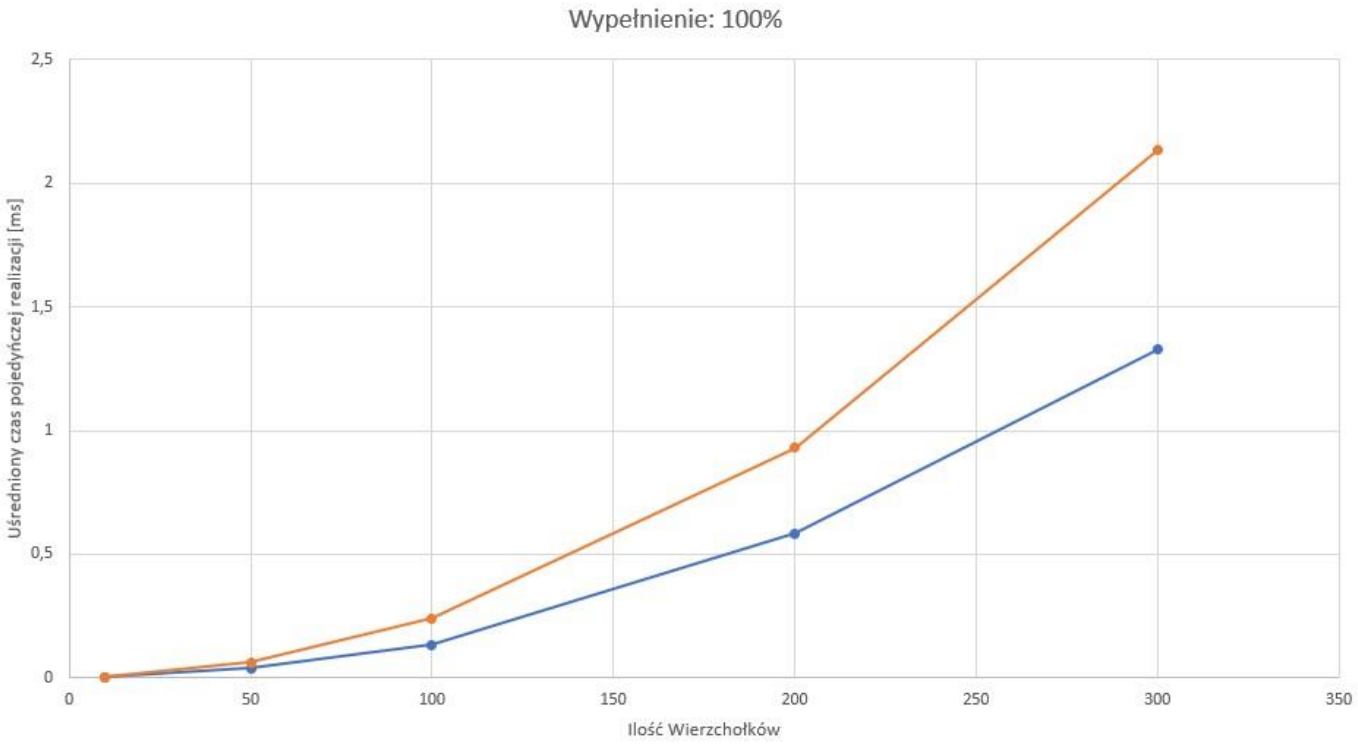


Wykres algorytmu z grafem zaimplementowanym w postaci listy.

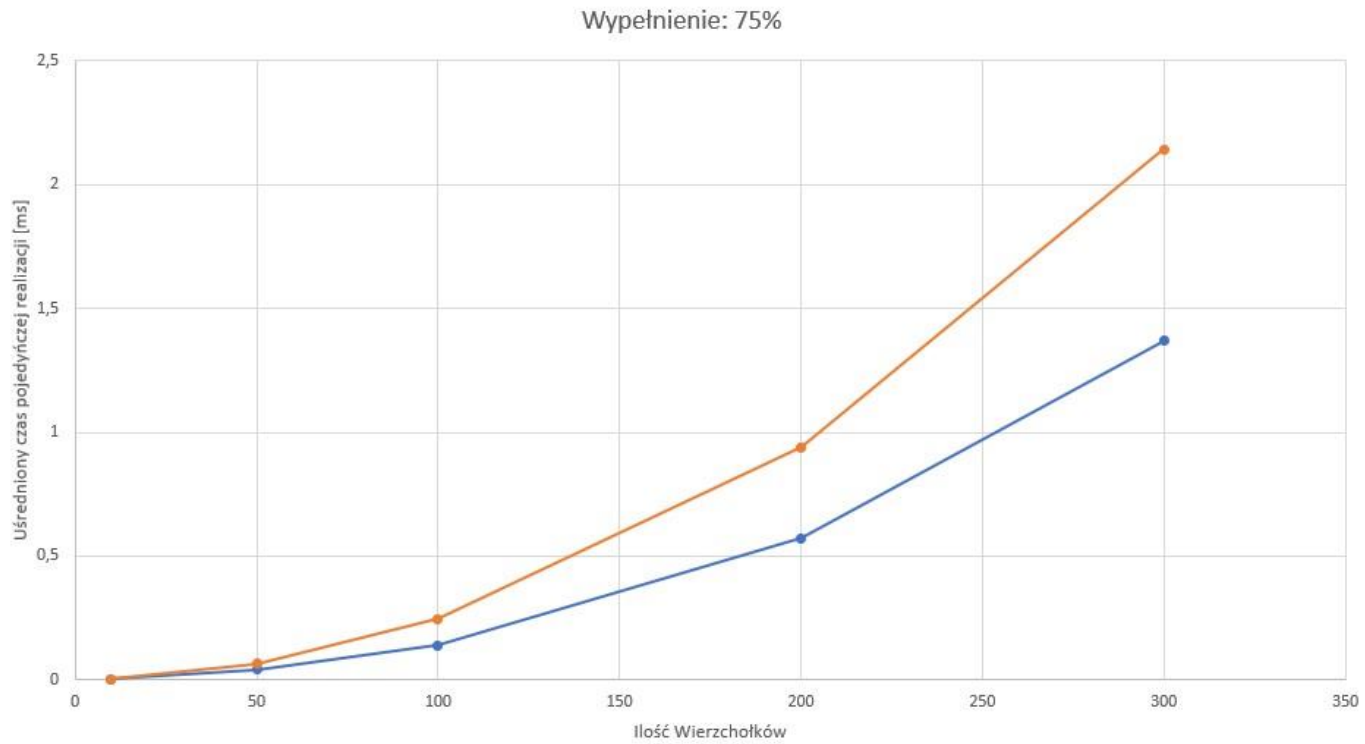


Kolorem czerwonym oznaczono implementacje w postaci listy, natomiast kolorem niebieskim w postaci macierzy.

Wykres algorytmów dla gęstości grafu wynoszącej 100%

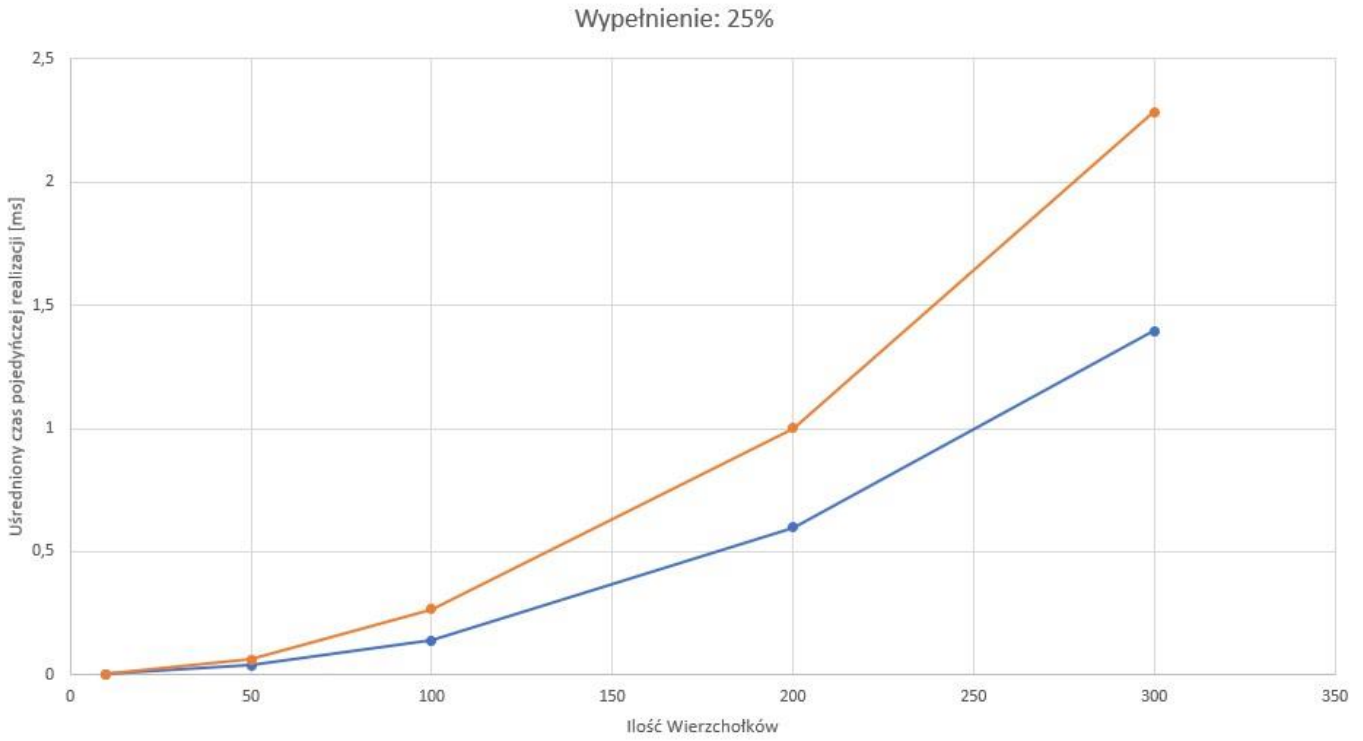


Wykres algorytmów dla gęstości grafu wynoszącej 75%



Wykres algorytmów dla gęstości grafu wynoszącej 50%

Wykres algorytmów dla gęstości grafu wynoszącej 25%



4. Wnioski

- Algorytm poprawnie znajduje najkrótsze ścieżki do wierzchołków końcowych.
- Wpływ na czas realizacji zadania ma zarówno ilość wierzchołków, jak i wypełnienie, przy czym istotnym parametrem jest ilość wierzchołków.
- Implementacja grafu w postaci listy znacząco, bo prawie dwukrotnie wydłuża czas pracy programu. Spowodowane to może być działaniem na masywniejszych obiektach od tablic typu int, jak również licznymi iteracjami listy w celu dojścia do odpowiedniego elementu.

5. Biografia

- **https://www.youtube.com/watch?v=EFg3u_E6eHU**
- **https://pl.wikipedia.org/wiki/Algorytm_Dijkstry**
- **https://stackoverflow.com/**