Dawid Grajoszek 249021
Termin zajęć: piątek 9:15 - 11:00
Prowadzący: mgr inż. Marta Emirsajłow

Projektowanie algorytmów i metody sztucznej inteligencji

Projekt II
Grafy

1 Opis projektu

Celem niniejszego projektu było rozwiązanie problemu znalezienia w grafie ważonym najkrótszej ścieżki pomiędzy dwoma wierzchołkami. Do rozwiązania tej kwestii wykorzystano algorytm Dijkstry. W eksperymencie należało przeanalizować 100 losowo wybranych grafów o różnym stopniu gęstości z zakresu 25%, 50%, 75% oraz 100%, a także o różnej liczbie wierzchołków - tutaj 160, 240, 320, 480, 560. Grafy te były reprezentowane przed macierz sąsiedztwa oraz listę sąsiedztwa. Zmierzono czas działania algorytmu Dijkstry w każdej sytuacji oraz dokonano na tej podstawie analizy i porównania pomiędzy poszczególnymi parametrami.

2 Algorytm Dijkstry

Algorytm ten znajduje najkrótszą ścieżkę pomiędzy dwoma wierzchołkami w grafie ważonym. Najbardziej popularne implementacje, na bazie których pracuje ten algorytm to przeszukiwanie liniowe lub kolejka w formie kopca. Na potrzeby tego eksperymentu powyższy algorytm został zaimplementowany przy użyciu wyszukiwania liniowego. Na początku tworzymy dwie tablice pomocnicze: w jednej (cost[]) zapisywany jest koszt (odległość) dojścia do danego wierzchołka od źródła, w drugiej (prev[]) z kolei zapisywani są poprzednicy mijani w drodze do wierzchołka docelowego. Koszt dojścia do każdego wierzchołka początkowo ustawiamy na $+\infty$ za wyjątkiem źródła, dla

którego odległość jest równa 0. Tabelę poprzedników wypełniamy liczbami niebędącymi numerami wierzchołków, np -1. Dodatkowo należy utworzyć zbiór przechowujący wierzchołki już odwiedzone visited[]. Ze zbioru, który reprezentuje nasz graf wybieramy wierzchołek o najmniejszym koszcie dojścia od źródła, który nie znajduje się jednocześnie w zbiorze visited[]. Wybrany wierzchołek zostaje przeniesiony do zbioru visited[]. Dla każdego wierzchołka odwiedzonego w od pewnego wierzchołka u, jeśli koszt dojścia do niego równy wadze krawędzi łączącej wierzchołki w i u plus koszt dojścia do wierzchołka w jest mniejszy od aktualnego kosztu, wówczas aktualizujemy odległości cost[w] = cost[u] + waga krawędzi między (w,u). W kolejnym kroku wierzchołek u czynimy poprzednikiem w, czyli prev[w] = u. Taką procedurę przeprowadzamy aż do momentu kiedy wszystkie wierzchołki grafu zostaną odwiedzone. Algorytm Dijkstry posiada jednak pewne ograniczenie - w grafie nie mogą występować krawędzie o wagach ujemnych.

3 Złożoność obliczeniowa

Złożoność obliczeniowa zależy przede wszystkim od liczby krawędzi E oraz wierzchołków V w badanym grafie ważonym. Ważna rolę w tej kwestii odgrywa również sposób, w jaki został zaimplementowany algorytm Dijsktry: przeszukiwanie liniowe - $O(V^2)$ lub wykorzystując kolejkę priorytetową na bazie kopca - $O(E + VlogV)$.
Miejsce, jakie zajmuje w pamięci macierz sąsiedztwa ma złożoność asymptotyczną $O(V^2)$, natomiast lista sąsiedztwa już tylko $O(V + E)$. Na podstawie tych złożoności można przypuszczać, że algorytm Dijkstry bazujący na liście sąsiedztwa będzie działał znacznie szybciej.

4 Wyniki i wykresy

Macierz sąsiedztwa				
N / gęstość grafu	25%	50%	75%	100%
160	0,052197s	0,079012s	0,114476s	0,16459s
240	0,117931s	0,199841s	0,26919s	0,351549s
320	0,224358s	0,39813s	0,786973s	1,35315s
480	0,548664s	0,842217s	1,18125s	2,12812s
560	0,839686s	1,44795s	1,80147s	3,01593s

Table 1: Czasy dla macierzy

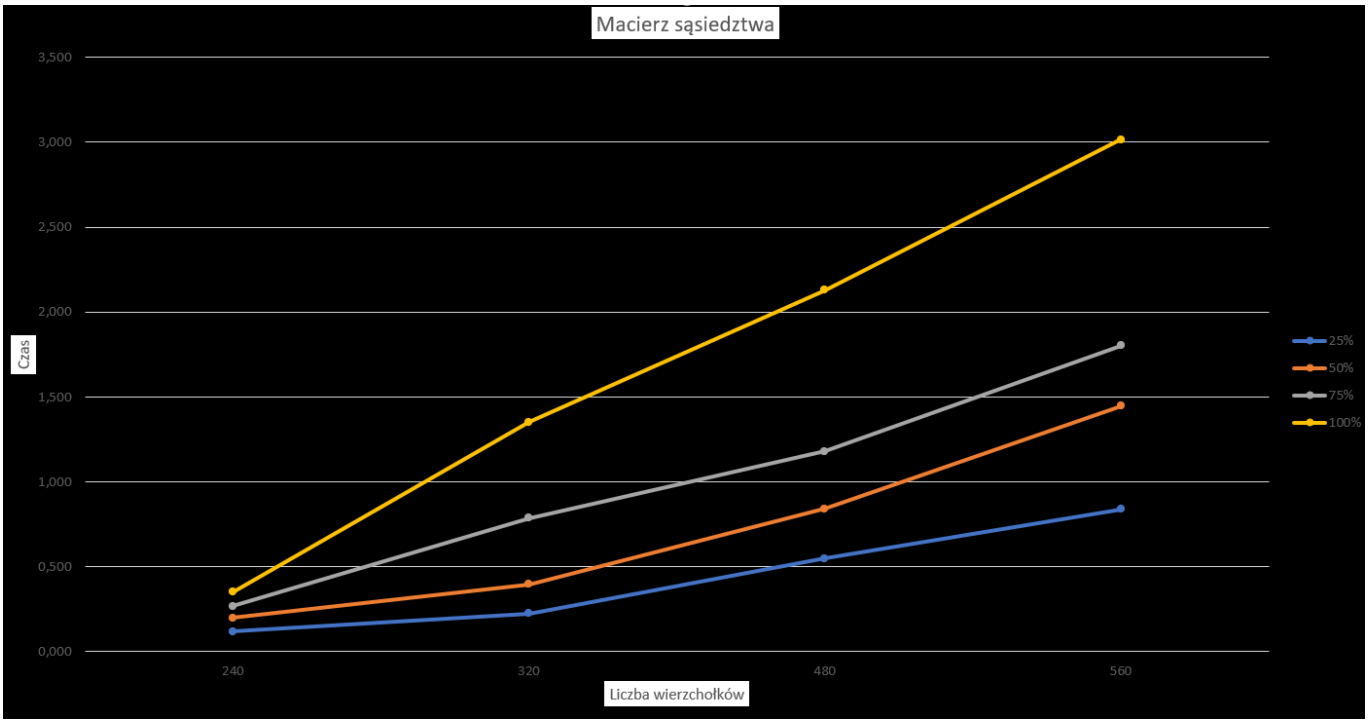

Figure 1: Czasy dla macierzy sąsiedztwa

Lista sąsiedztwa				
N / gęstość grafu	25%	50%	75%	100%
160	0,029722s	0,038577s	0,046566s	0,061427s
240	0,051269s	0,065172s	0,081026s	0,111682s
320	0,088763s	0,119225s	0,16577s	0,194937s
480	0,21057s	0,276719s	0,368871s	0,426039s
560	0,296165s	0,355198s	0,485009s	1,59576s

Table 2: Czasy dla listy

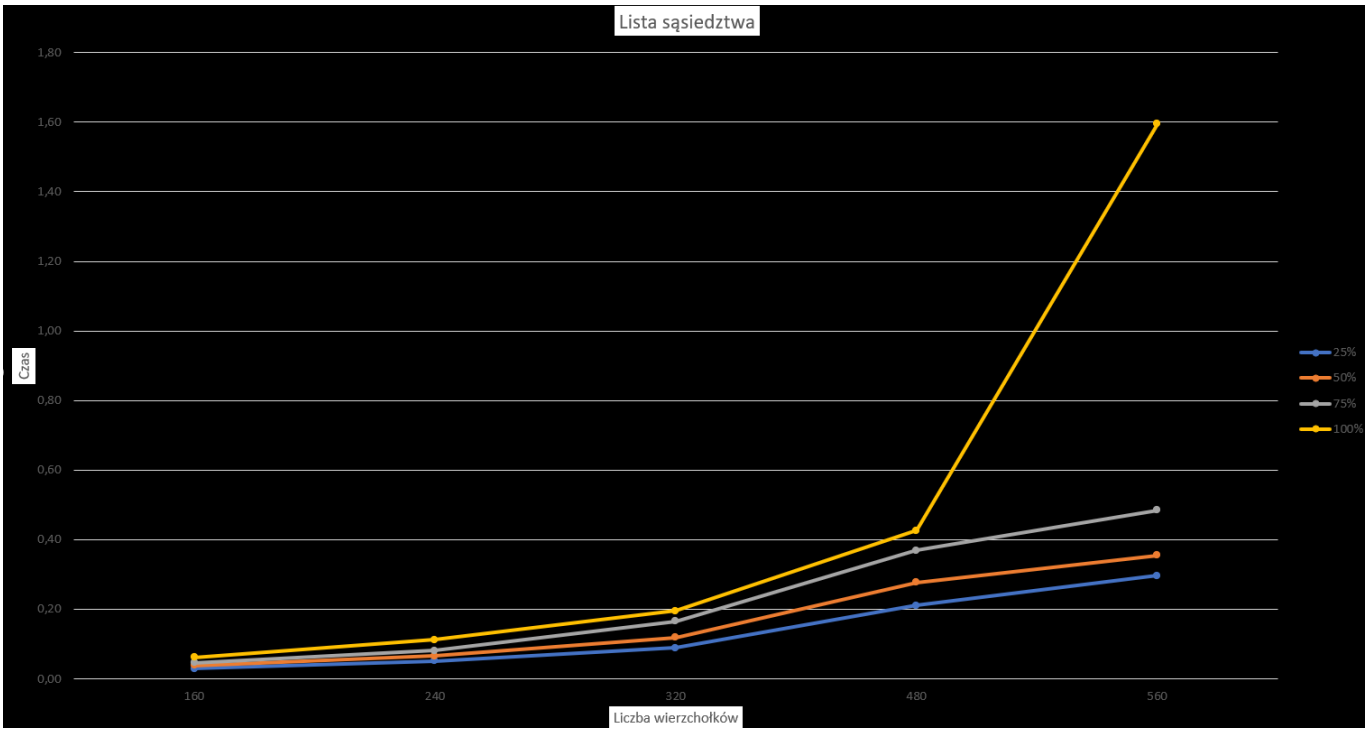

Figure 2: Czasy dla listy sąsiedztwa

5 Podsumowanie

- Jak zostało napisane w rozdziale "Złożoność obliczeniowa" algorytm Dijsktry oparty na liście sąsiedztwa działa zdecydowanie szybciej niż w przypadku macierzy sąsiedztwa i mimo zwiększania się liczby wierzchołków czas potrzebny na obliczenia niewiele się różni dla trzech pierwszych gęstości.
- Jak można zauważyć wykresy dla obu form reprezentacji grafu w przypadku wyszukiwania liniowego przypominają wykresy funkcji kwadratowej co jest zgodne z wcześniej przyjętą złożonością obliczeniową. A więc eksperyment potwierdza postawione wcześniej hipotezy.
- Biorąc pod uwagę graf pełny o licznie wierzchołków równej 560 lista jest około $\frac{3{,}01593s}{1{,}59576s} = 1,89 \approx 2$ razy szybsza niż macierz sąsiedztwa.
- Algorytm Dijkstry znajduje szerokie zastosowanie, np. przy wyznaczaniu najkrótszej drogi między dwoma miejscowościami czy też przy transportach pakietów w sieciach komputerowych.

6 Literatura

- https://pl.wikipedia.org/wiki/Algorytm_Dijkstry
- http://algorytmy.ency.pl/artykul/algorytm_dijkstry