Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt 2

Grafy

Data oddania sprawozdania	08.05.2019
Imię i nazwisko	Kinga Tokarska
Nr indeksu	241621
Termin zajęć	Środa, 11.15-13.00
Prowadzący kurs	Dr inż. Łukasz Jeleń
Kod kursu	E02-47f

1. Wstęp teoretyczny

Graf jest powszechnie stosowaną we współczesnej informatyce strukturą danych składającą się z dwóch zbiorów: zbioru wierzchołków oznaczanych jako V i zbioru krawędzi łączących te wierzchołki oznaczanych jako E. Matematycznie zapisuje się je w postaci uporządkowanej pary $G = (V, E)$.

Zbiór wierzchołków może być skończony lub nieskończony. Każdy element zbioru krawędzi to para składająca się z dwóch elementów ze zbioru wierzchołków.

Grafy są często przedstawiane wizualnie, poprzez rysowanie elementów wierzchołków ustawionych jako ramki lub okręgi i rysowanie elementów krawędzi ustawionych jako linie lub łuki między polami lub okręgami. Pomiędzy v_1 a v_2 znajduje się łuk, jeśli (v_1, v_2) jest elementem zbioru krawędzi.

Wśród grafów wyróżnia się skierowane i nieskierowane. Grafem skierowanym jest graf, którego wszystkie krawędzie są skierowane, to znaczy przebywają tylko w określoną stronę. Nadany jest kierunek poruszania się pomiędzy dwoma wierzchołkami. Grafem nieskierowanym jest graf, w którym relacja sąsiedztwa jest symetryczna, połączenie między wierzchołkami jest dwukierunkowe.

Z krawędziami grafu mogą być związane dodatkowe wartości liczbowe. Wartości te nazywane są wagami i są przypisywane każdej krawędzi grafu. Mogę one oznaczać na przykład długość krawędzi lub jej przepustowość. Graf ważony posiada zbiór krawędzi zbudowany z uporządkowanych trójek, gdzie dwa pierwsze elementy określają wierzchołki połączone daną krawędzią (w grafie skierowanym wierzchołki te są parą uporządkowaną), a trzeci element określa wagę tej krawędzi.

Gęstość grafu to stosunek liczby krawędzi do największej możliwej liczby krawędzi. Określona jest wzorem $d = \frac{2E}{V(V-1)}$, gdzie E to liczba krawędzi, a V to liczba wierzchołków w danym grafie.

Istnieje kilka sposobów reprezentacji grafów w pamięci komputera. Najczęściej przechowywane są one w postaci listy lub macierzy sąsiedztwa. Wybór reprezentacji grafu może mieć wpływ na czas obliczeń lub ilość wykorzystanej pamięci. Cechą charakterystyczną tych implementacji jest wykorzystanie tablic do przechowywania danych na temat wierzchołków lub łączących je krawędzi.

W przypadku macierzy sąsiedztwa w pamięci komputera przechowywana jest macierz kwadratowa A o stopniu n, gdzie n oznacza liczbę wierzchołków w grafie. Odwzorowuje ona połączenia wierzchołków krawędziami. Każdy wiersz i każda kolumna odpowiadają innemu wierzchołkowi. Wiersze macierzy odwzorowują zawsze wierzchołki startowe krawędzi, a kolumny odwzorowują wierzchołki końcowe krawędzi. W przypadku grafu nieskierowanego macierz zawsze jest symetryczna.

W przypadku listy sąsiedztwa w pamięci komputera przechowywana jest jednowymiarowa tablica n-elementowa A, gdzie n oznacza liczbę wierzchołków. Każdy element tej tablicy jest jednokierunkową listą. Lista reprezentuje wierzchołek startowy. Na liście przechowywane są numery wierzchołków końcowych, czyli sąsiadów wierzchołka startowego, z którymi jest on połączony krawędzią [1].

2. Opis badanego algorytmu

Algorytm Dijkstry pozwala rozwiązać problem wyznaczenia najkrótszych ścieżek pomiędzy wybranym wierzchołkiem (zwanym wierzchołkiem źródłowym) a wszystkimi pozostałymi wierzchołkami w grafie. Algorytm wymaga, aby wagi krawędzi grafu były nieujemne.

W trakcie wykonywania algorytmu dla każdego wierzchołka zostają wyznaczone dwie wartości: koszt dotarcia do tego wierzchołka oraz poprzedni wierzchołek na ścieżce. Na początku działania algorytmu dla wierzchołka źródłowego koszt dotarcia wynosi 0, a dla każdego innego wierzchołka. Wszystkie wierzchołki na początku znajdują się w zbiorze wierzchołków nieprzejrzanych Q. Dopóki zbiór Q nie jest pusty algorytm pobiera z tego zbioru wierzchołek o najmniejszym koszcie dotarcia, oznacza go jako v i usuwa ze zbioru. Dla każdej krawędzi wychodzącej z wierzchołka v oznaczonej jako e oznacza wierzchołek znajdujący się na drugim jej końcu jako u. Jeśli koszt dotarcia do wierzchołka u z wierzchołka v poprzez krawędź e jest mniejszy od aktualnego kosztu dotarcia do wierzchołka u, wówczas algorytm przypisuje kosztowi dotarcia do wierzchołka u koszt dotarcia do wierzchołka v powiększony o wagę krawędzi e. Następnie ustawa wierzchołek v jako poprzednik wierzchołka u.

Algorytm realizuje podejście zachłanne. W każdej iteracji wybierany jest ten spośród nieodwiedzonych wierzchołków, do którego można dotrzeć najmniejszym kosztem. Po wyznaczeniu ścieżki do konkretnego wierzchołka nie zostanie ona zmodyfikowana w trakcie wykonywania dalszej części algorytmu.

Oznaczmy liczbę wierzchołków jako v, a liczbę krawędzi przez e. Każda krawędź jest analizowana jeden raz w przypadku grafu skierowanego lub dwa razy w przypadku grafu nieskierowanego. Oprócz analizy krawędzi w trakcie wykonywania algorytmu v razy przeszukany zostaje zbiór Q. W przypadku przechowywania zbioru Q w zwykłej tablicy, podczas szukania wierzchołków o najmniejszym koszcie dojścia wykorzystane jest wyszukiwanie liniowe, a algorytm na złożoność czasową $O(v^2)$. Jeśli zbiór Q przechowywany jest w postaci kopca, zarówno wyszukiwanie elementu jak i przebudowa kopca po każdej zmianie kosztu dotarcia do wierzchołka będą się odbywało w czasie $O(e \log v)$. Złożoność obliczeniowa algorytmu wyniesie wówczas $O(e \log v)$. Jeśli do przechowywania zbioru Q zastosowany zostanie kopiec Fibonacciego, złożoność obliczeniowa algorytmu wynosi $O(v \log v + e)$ [2].

3. Przebieg ćwiczenia

Zadanie polegało na zbadaniu efektywności wybranego algorytmu w zależności od sposobu reprezentacji grafu (reprezentacja w postaci macierzy *tab.1* i listy *tab.2*) oraz gęstości grafu (gęstości równe 25%, 50%, 75% i 100%). Badania przeprowadzono dla pięciu różnych liczb wierzchołków: 10, 50, 100, 500 I 1000. Dla każdego zestawu: reprezentacja grafu, liczba wierzchołków i gęstość wygenerowano po 100 losowych instancji.

Program wczytuje graf z pliku tekstowego o odpowiednim formacie danych (ilość krawędzi, ilość wierzchołków, wierzchołek startowy, wierzchołek początkowy, wierzchołek końcowy, waga). Wynik działania algorytmu także zapisywany jest do pliku tekstowego (koszt drogi, ciąg wierzchołków od wierzchołka startowego).

Eksperymenty przeprowadzano, korzystając z komputera wyposażonego w procesor Intel Core i5-7200U (2.50 GHz, 2712 MHz, 2 rdzenie) i 8 GB pamięci RAM SO-DIMM DDR4.

Wyniki przeprowadzanych eksperymentów (uśrednione wyniki badań ze 100 wygenerowanych losowych instancji) przedstawiono w poniższych tabelach oraz na poniższych wykresach.

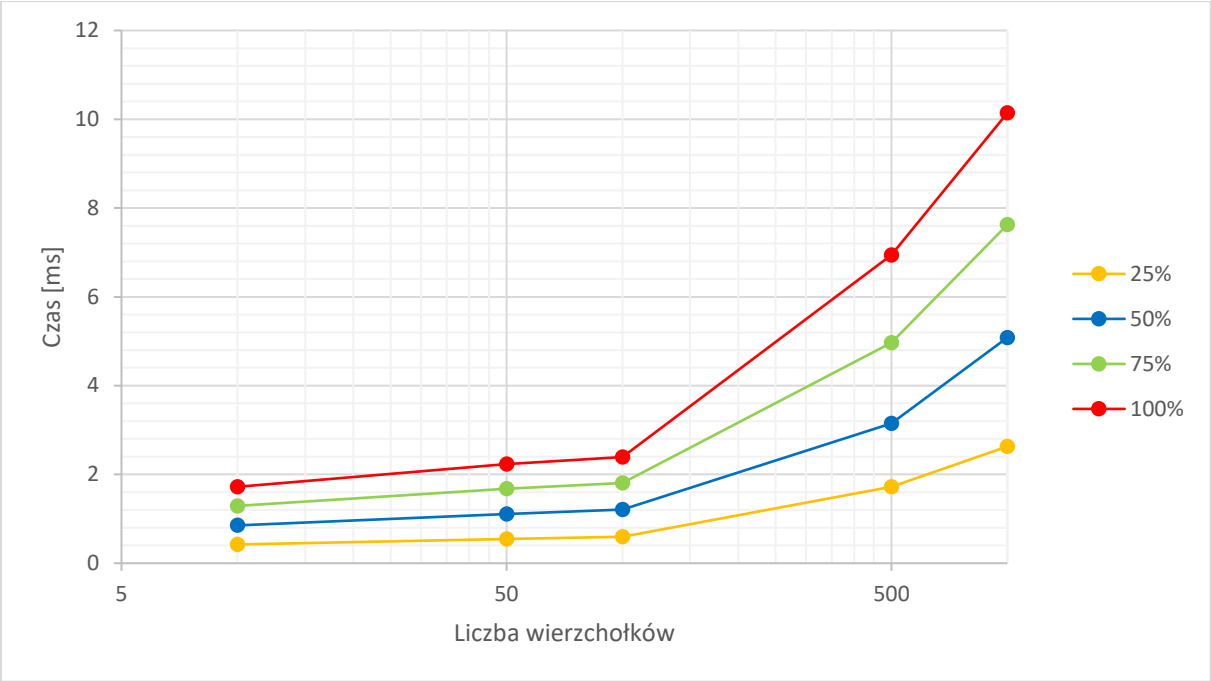
4. Wyniki pomiarów

Tab.1. Wyniki pomiarów dla reprezentacji grafu w postaci macierzy

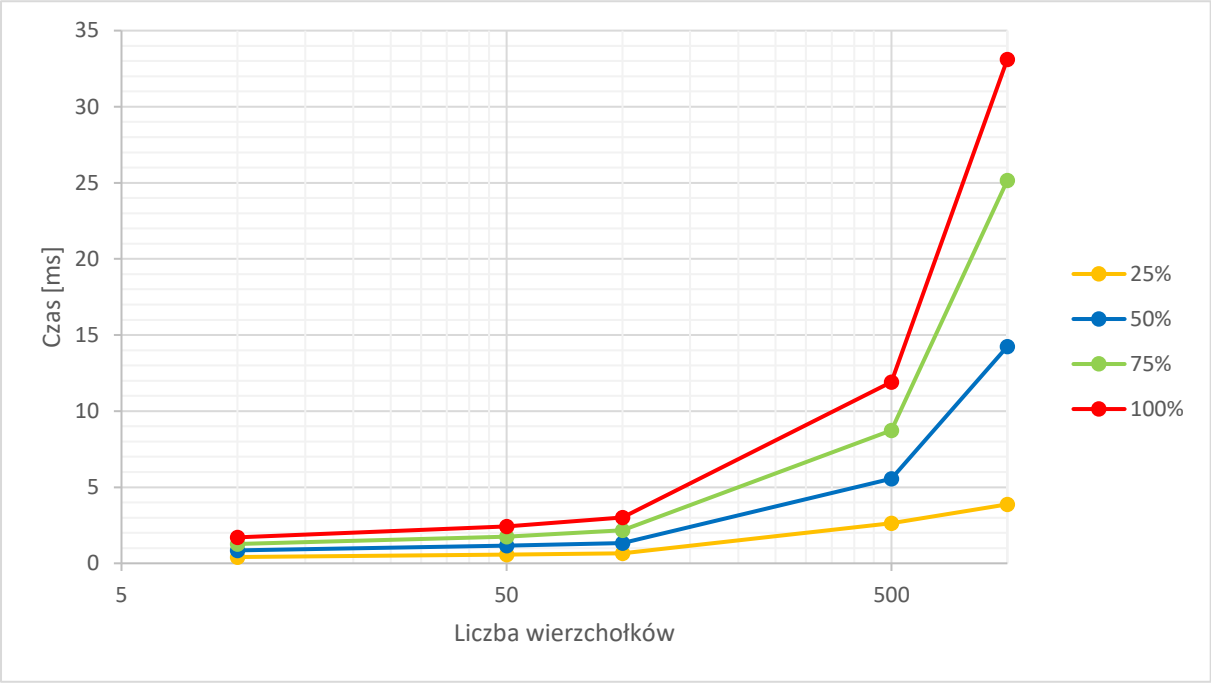
		Liczba wierzchołków				
		10	50	100	500	1000
Gęstość	25%	0,422 ms	0,548 ms	0,599 ms	1,723 ms	2,629 ms
	50%	0,852 ms	1,107 ms	1,206 ms	3,146 ms	5,083 ms
	75%	1,291 ms	1,674 ms	1,806 ms	4,963 ms	7,624 ms
	100%	1,721 ms	2,23 ms	2,393 ms	6,942 ms	10,141 ms

Tab.2. Wyniki pomiarów dla reprezentacji grafu w postaci listy

		Liczba wierzchołków				
		10	50	100	500	1000
Gęstość	25%	0,414 ms	0,577 ms	0,65 ms	2,626 ms	3,876 ms
	50%	0,851 ms	1,165 ms	1,324 ms	5,554 ms	14,243 ms
	75%	1,271 ms	1,748 ms	2,162 ms	8,737 ms	25,154 ms
	100%	1,705 ms	2,427 ms	3,003 ms	11,905 ms	33,117 ms

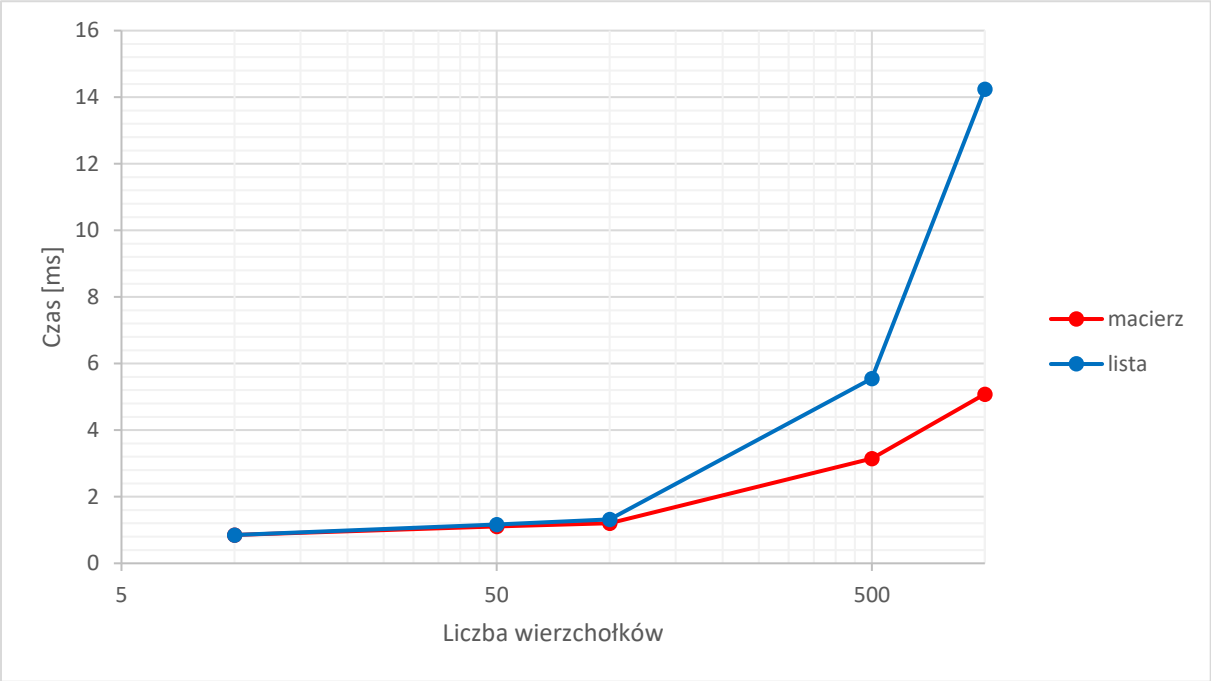


Rys. 1. Wykres zależności czasu wykonania algorytmu od ilości wierzchołków dla reprezentacji grafu w postaci macierzy



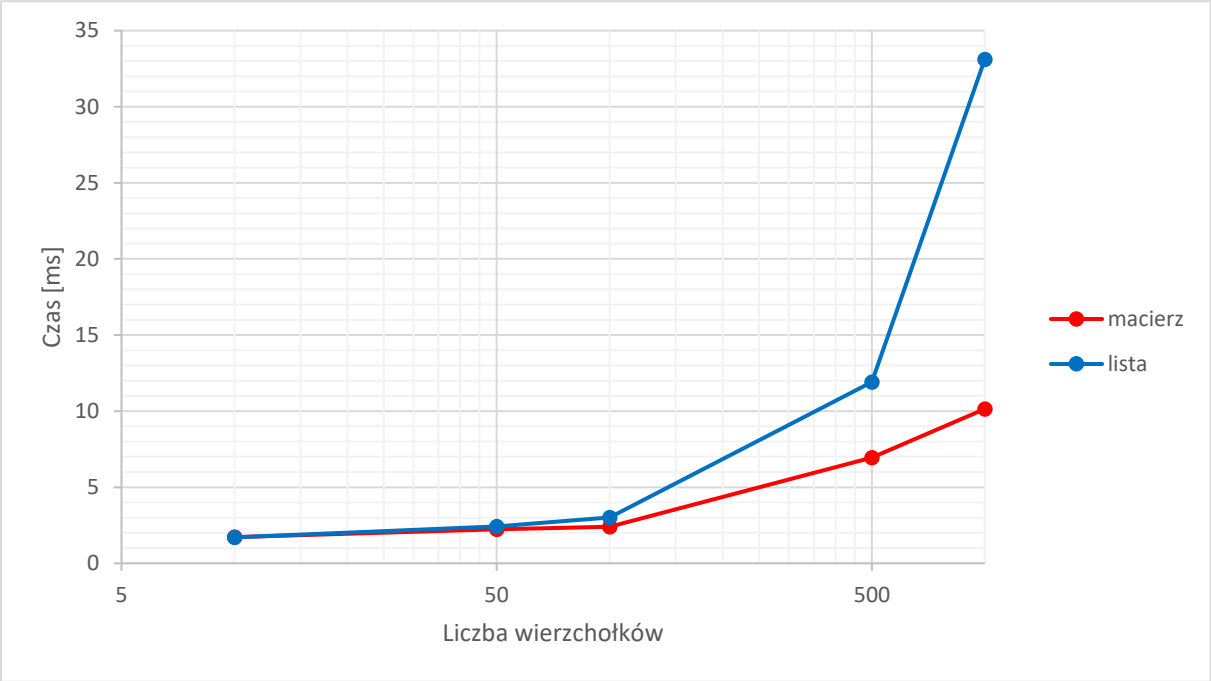
Rys. 2. Wykres zależności czasu wykonania algorytmu od ilości wierzchołków dla reprezentacji grafu w postaci listy

Rys. 3. Wykres zależności czasu wykonania algorytmu od ilości wierzchołków dla gęstości grafu równej 25%



Rys. 4. Wykres zależności czasu wykonania algorytmu od ilości wierzchołków dla gęstości grafu równej 50%

Rys. 5. Wykres zależności czasu wykonania algorytmu od ilości wierzchołków dla gęstości grafu równej 75%



Rys. 6. Wykres zależności czasu wykonania algorytmu od ilości wierzchołków dla gęstości grafu równej 100%

5. Wnioski

Istotne różnice pomiędzy przedstawionymi reprezentacjami grafów są widoczne już przy wykonywaniu prostych operacji.

Po przeprowadzeniu eksperymentu na podstawie zbiorczych danych oraz wygenerowania wykresów można zauważyć, iż czasy podczas wykonywania algorytmu w zależności od ilości wierzchołków różnią się między reprezentacją macierzy i listy. Im większa ilość zadanych wierzchołków tym więcej algorytm potrzebuje czasu na wykonanie w przypadku listy. Przy mniejszej ilości wierzchołków czasy są porównywalne do siebie.

Macierze sąsiedztwa są preferowanym sposobem reprezentacji grafów, wówczas gdy grafy są gęste, to znaczy kiedy liczba krawędzi jest bliska maksymalnej możliwej ich liczbie. Jeżeli graf jest rzadki, reprezentacja oparta na listach sąsiedztwa może pozwolić na zaoszczędzenie czasu i pamięci.

Dijkstra wybiera krawędzie z najmniejszym kosztem na każdym kroku, co zwykle obejmuje duży obszar grafu. Jest to szczególnie przydatne, gdy mamy wiele węzłów docelowych, ale nie wiadomo, który jest najbliższy.

6. Literatura

[1] https://eduinf.waw.pl/inf/alg/001_search/0123.php (dostęp 07.06.2010)
[2] http://algorytmy.ency.pl/artykul/algorytm_dijkstry (dostęp 07.06.2010)