Sprawozdanie algorytmy grafowe

Algorytmy i struktury danych

Kacper Majorkowski
Adam Mikołajczak

I. Metodologia

Utworzone zostały trzy spójne, acykliczne, skierowane reprezentacje grafu, z nasyceniem wierzchołków równym 50%, $G(n,m)$, gdzie n - ilość wierzchołków, m - ilość krawędzi

- Macierz sąsiedztwa - $adjMatrix$
- Lista następników - $adjList$
- Tabela krawędzi - $edgeList$

Dla każdej z nich wykonano testy złożoności czasowej sortowań topologicznych BFS oraz BFS. Wykonano dziesięć testów, dla grafów z ilością wierzchołków n w przedziale 100 - 1000 z krokiem co 100. Podany czas jest średnią wykonywania z pięciu prób.

II. Grafy

1. Macierz sąsiedztwa, złożoność:

a) pamięciowa $O(n^2)$ - trzeba utworzyć tabelę o rozmiarach $n\ x\ n$ wierzchołków
b) znalezienia pojedynczej krawędzi $O(1)$ - wystarczy podać wierzchołek wyjścia i wejścia oraz odwrotnie jako parametry tablicy, i sprawdzić czy jest tam 1, która oznacza że istnieje taka krawędź
c) znalezienie wszystkich następników wierzchołka $O(n)$ - należy przeszukać cały wiersz tablicy, aby mieć pewność że sprawdziliśmy wszystkie wierzchołki

2. Lista następników, złożoność:

a) pamięciowa $O(n + m)$ - utworzenie wierszy w tabeli dla wszystkich wierzchołków, następnie dla wszystkich wierszy dopisanie następników (łącznie ilość krawędzi)
b) znalezienia pojedynczej krawędzi średnio $O(\frac{n}{m})$, jednak w najgorszym przypadku gdy wierzchołek będzie na końcu wiersza, będzie to $O(n)$
c) znalezienie wszystkich następników wierzchołka średnio $O(\frac{n}{m})$, w najgorszym przypadku tak jak powyżej, wierzchołek jako następników może posiadać wszystkie inne wierzchołki, przez co złożoność będzie wynosić $O(n)$

3. Tabela krawędzi, złożoność:

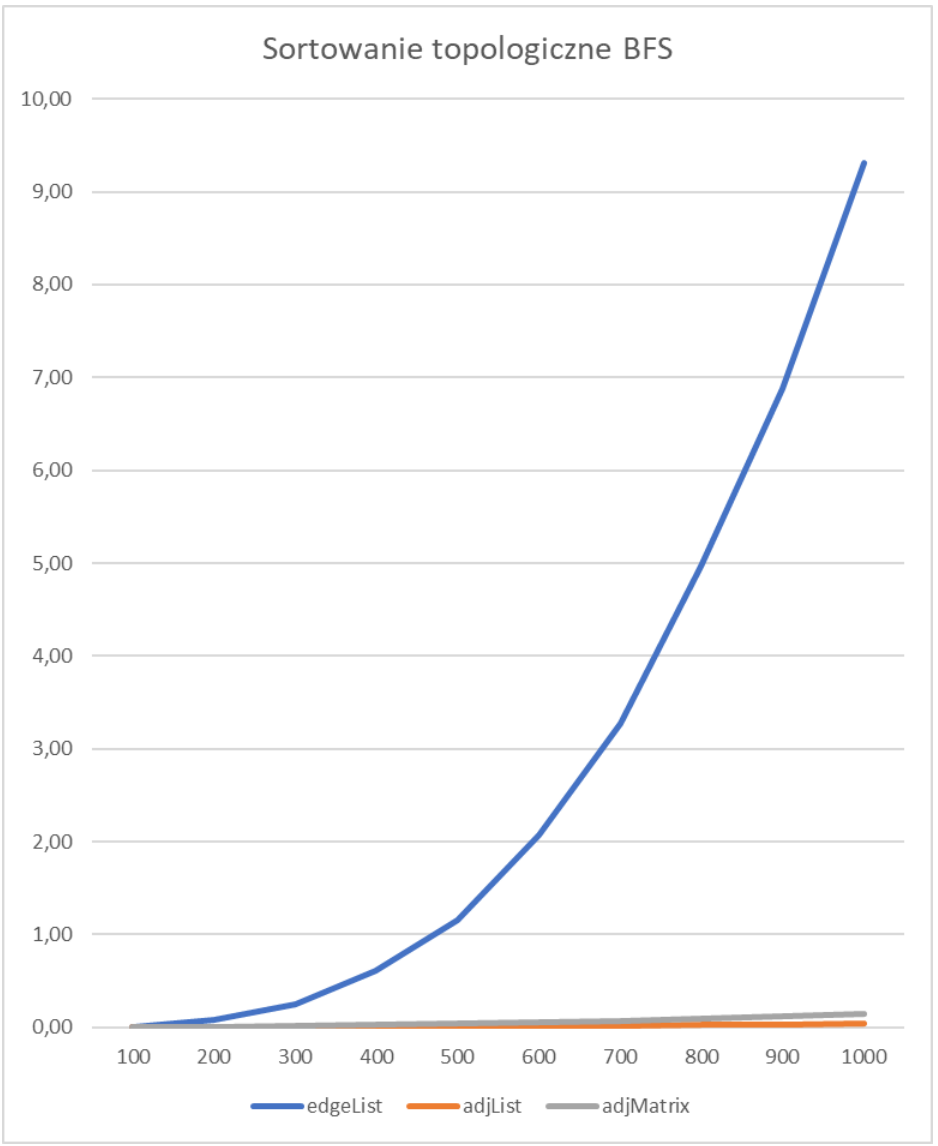
a) pamięciowa $O(m)$ - tworzymy dwie kolumny, o długości m, do pierwszej wpisujemy wierzchołek wyjściowy, do drugiej wejściowy
b) znalezienia pojedynczej krawędzi $O(m)$ - trzeba przeszukać wszystkie wiersze w poszukiwaniu krawędzi
c) znalezienie wszystkich następników $O(m)$ - tak samo jak powyżej, musimy przejść przez wszystkie krawędzie aby być pewnym, iż nic się nie pominęło

III. Złożoność sortowania topologicznego

1. BFS

Tworzymy tablicę pomocniczą in_degree, która reprezentuje stopnie wejściowe wierzchołków w grafie. Najpierw inicjalizujemy ją z samymi zerami dla każdego wierzchołka, więc złożoność tej operacji to $O(n)$. Następnie kalkujemy faktyczne stopnie wierzchołków, złożoność tej operacji jest zależna od reprezentacji grafu: musimy przejść w niej przez każdą krawędź (wyszukiwanie następników danych wierzchołków). Następnie przechodzimy przez tablicę in_degree w poszukiwaniu wierzchołka z zerowym stopniem ($O(n)$). W trakcie działania algorytmu zmniejszamy stopnie wejściowe następników wierzchołków z zerowym stopniem wejściowym (operacja zależna od reprezentacji). Tą operacje wykonujemy dla każdego wierzchołka ($O(n)$).

a) Macierz sąsiedztwa - $O(n + n + n + (n * n)) = O(n^2 + 3n) = O(n^2)$
b) Lista następników:
   i) Średnio: $O(n + \frac{n}{m} + n + (n * \frac{n}{m})) = O(\frac{n^2}{m} + \frac{n}{m} + 2n) = O(\frac{n^2}{m})$
   ii) Najgorszy: $O(n + n + n + (n * n)) = O(n^2 + 3n) = O(n^2)$
c) Tabela krawędzi - $O(n + m + n + (n * m)) = O(n * m + 2n + m) = O(n * m)$



2. DFS

Tworzymy tablicę pomocniczą visited, która mówi o tym czy dany wierzchołek został już odwiedzony. Najpierw inicjalizujemy ją z wartościami False dla każdego wierzchołka, więc złożoność tej operacji to $O(n)$. Algorytm przechodzi przez wszystkie wierzchołki ($O(n)$). Dla każdego z nich wyszukujemy następników (zależnie od reprezentacji). Następnie sprawdzamy czy dany następnik został już odwiedzony (średnio ($O(\frac{m}{n})$).

a) Macierz sąsiedztwa - $O(n + n * n * \frac{m}{n}) = O(n * m)$

b) Lista następników:

i) Średnio: $O(n + n * \frac{n}{m} * \frac{m}{n}) = O(n)$

ii) Najgorszy: $O(n + n * n * \frac{m}{n}) = O(n * m)$

c) Tabela krawędzi - $O(n + n * m * \frac{m}{n}) = O(m^2)$