**Wstęp**

Sprawozdanie dotyczy algorytmów grafowych oraz sortowań topologicznych przy ich użyciu w trzech różnych reprezentacjach, tj. macierzy sąsiedztwa, listy następników, oraz tabeli krawędzi. Badaniu poddane zostały cztery algorytmy: BFS (breadth first search) oraz DFS (depth first search), a także sortowania topologiczne oparte na BFS i DFS. W przypadku algorytmu DFS i opartego na nim sortowania topologicznego zaimplementowano jego iteracyjną postać. Algorytmy oraz reprezentacje grafów zostały zaimplementowane w języku python 3. W celu zmierzenia czasów działań algorytmów wykorzystano 10 grafów o wielkości od 300 do 1200 wierzchołków. Dla każdego algorytmu przedstawiono wykresy zarówno bez (po lewej) jak i z zastosowaniem skali logarytmicznej (po prawej). W opisach złożoności zastosowano oznaczenia: n – liczba wierzchołków, m – liczba krawędzi.

**Tworzenie grafu**

Jednym z celów zadania było topologiczne sortowanie, które odbywać się może tylko na grafach, które są zarówno acykliczne jak i skierowane. W celu uzyskania takiego grafu tworzyliśmy najpierw macierz górno-trójkątną z nasyceniem łuków 50%. Następnie wykorzystaliśmy ją w celu utworzenia pozostałych reprezentacji.

**Złożoność pamięciowa**

* Macierz sąsiedztwa O(n2)
* Lista następników O(n+m)
* Tabela krawędzi O(m)

**Złożoność obliczeniowa znalezienia krawędzi między danymi wierzchołkami (sprawdzenie czy istnieje)**

* Macierz sąsiedztwa O(1)
* Lista następników O(m/n)
* Tabela krawędzi O(m)

**Złożoność obliczeniowa znalezienia dowolnego następnika wierzchołka**

* Macierz sąsiedztwa O(n)
* Lista następników O(1)
* Tabela krawędzi O(m)

**Złożoność obliczeniowa znalezienia wszystkich następników wierzchołka**

* Macierz sąsiedztwa O(n)
* Lista następników O(m/n)
* Tabela krawędzi O(m)

**Przechodzenie przez graf metodą BFS**

Algorytm BFS zaczyna się od wybrania wierzchołka, z którego zaczniemy badanie grafu, z którego następnie przechodzimy do wszystkich osiągalnych z niego wierzchołków (w takiej kolejności że najpierw wypisujemy wszystkich następników danego wierzchołka, po czym powtarzamy to dla każdego następnika), jeżeli nie uda nam się przejść przez wszystkie, wybieramy nowy wierzchołek i powtarzamy proces. Wynikiem przeszukiwań jest drzewo, o korzeniu będącym wybranym przez nas wierzchołkiem, zawierające wszystkie wierzchołki osiągalne z korzenia, do każdego z nich prowadzi dokładnie jedna oraz najkrótsza ścieżka z wybranego wierzchołka.

**Złożoność obliczeniowa**

* Macierz sąsiedztwa O(n+m)
* Lista następników O(n+m)
* Tabela krawędzi O(nm)

**Przechodzenie przez graf metodą DFS**

Podobnie jak przy algorytmie BFS wybieramy początkowy wierzchołek, z którego nasz algorytm rozpoczyna działanie, wierzchołek oznaczamy jako odwiedzony i przechodzimy dalej wzdłuż dostępnej krawędzi do sąsiada tego wierzchołka, który nie został jeszcze odwiedzony, przechodzimy w ten sposób tak długo aż nie trafimy na wierzchołek bez nieodwiedzonych sąsiadów, wtedy algorytm cofa się do wierzchołka, który takich sąsiadów posiada, i powtarza czynność aż odwiedzi wszystkie wierzchołki.

**Złożoność obliczeniowa**

* Macierz sąsiedztwa O(n+m)
* Lista następników O(n+m)
* Tabela krawędzi O(nm)

**Sortowanie topologiczne oparte na BFS**

Sortowanie topologiczne polega na uszeregowaniu wierzchołków w taki sposób, że każdy z nich znajduje się za swoimi poprzednikami, a przed następnikami. Dla wielu grafów istnieje więcej niż jeden sposób posortowania topologicznego. Algorytm oparty na BFS (algorytm Kahna) korzysta z pomocniczej tabeli, która zawiera stopień wejściowy każdego wierzchołka. Na początku wybieramy wierzchołek o zerowym stopniu wejściowym, wypisujemy go, a wszystkim jego następnikom zmniejszamy stopień wejściowy w tabeli pomocniczej o jeden. Wtedy wypisujemy kolejny niewypisany jeszcze wierzchołek o zerowym stopniu (znowu odejmując 1 jego następnikom w tabeli). Operację powtarzamy aż wypiszemy wszystkie wierzchołki.

**Złożoność obliczeniowa**

Różnice w złożonościach między poszczególnymi reprezentacjami wynikają z różnicy w sposobie wyszukiwania następników danego wierzchołka

* Macierz sąsiedztwa O(n+m)
* Lista następników O(n+m)
* Tabela krawędzi O(nm)

**Sortowanie topologiczne oparte na DFS**

Algorytm rozpoczynamy od wybranego wierzchołka, oznaczamy go jako odwiedzonego i przeszukujemy graf w głąb (jak w DFS), oznaczając wierzchołki jako odwiedzone. Gdy dojdziemy do wierzchołka, z którego nie da się dojść dalej do nieodwiedzonego wierzchołka, wtedy wypisujemy ten wierzchołek i oznaczamy jako wypisany. Cofamy się do jego poprzednika i powtarzamy proces (znajdujemy nieodwiedzonego następnika i do niego wchodzimy, a gdy takich nie ma – wypisujemy wierzchołek i oznaczamy jako odwiedzony). Dopóki nie wypisano wszystkich wierzchołków, wywołujemy algorytm na nieodwiedzonych wierzchołkach, aż zostaną wypisane wszystkie.

**Złożoność obliczeniowa**

Różnice w złożonościach między poszczególnymi reprezentacjami wynikają z różnicy w sposobie wyszukiwania następników danego wierzchołka

* Macierz sąsiedztwa O(n+m)
* Lista następników O(n+m)
* Tabela krawędzi O(nm)