**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 dwa algorytmy; BFS(breadth first search) oraz DFS(depth first search). Algorytmy oraz reprezentacje grafów zostały zaimplementowane w języku python 3. W celu zmierzenia czasów działań algorytmów wykorzystaliśmy 10 grafów o wielkości od 300 do 1200 wierzchołków.

**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.

**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, 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 pamięciowa**

* Macierz sąsiedztwa O(V2), gdzie V to liczba wierzchołków
* Lista następników O(V+E), gdzie V to liczba wierzchołków, a E to liczba krawędzi
* Tabela krawędzi O(E), gdzie E to liczba krawędzi

**Złożoność obliczeniowa**

* Macierz sąsiedztwa O(V2), gdzie V to liczba wierzchołków
* Lista następników O(V+E), gdzie V to liczba wierzchołków, a E to liczba krawędzi
* Tabela krawędzi

**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 pamięciowa**

* Macierz sąsiedztwa
* Lista następników
* Tabela krawędzi

**Złożoność obliczeniowa**

* Macierz sąsiedztwa
* Lista następników
* Tabela krawędzi