Projekt 3 – sortowanie topologiczne

Wiktor Zmiendak gr. 13

1. **Opis problemu:**
   Naszym celem jest posortować topologicznie zadany graf. Oznacza to, że listę wierzchołków w grafie należy ustawić tak, aby te posiadające swoich sąsiadów znalazły się na liście przed nimi. Sortowanie będzie możliwe pod warunkiem, że nasz graf będzie acykliczny, skierowany. Nieraz istnieje wiele różnych sposobów na posortowanie tego samego grafu, na przykład wtedy gdy występują w nim wierzchołki nie połączone z innymi.

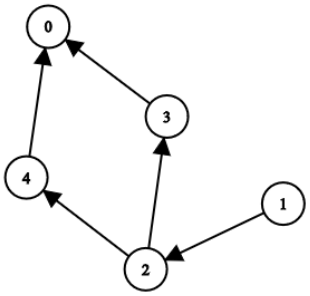
2. **Sposoby rozwiązania problemu:**
   - Algorytm DFS - rekurencyjne przeszukiwanie w głąb grafu, chodzimy od punktu od punktu aż nie odwiedzimy ich wszystkich,
   - Algorytm BFS - przeszukiwanie wszerz grafu, tworzymy kolejkę w myśl zasady FIFO (first in first out) i dodajemy do niej po kolei elementy grafu o stopniu 0

3. **Zastosowania algorytmu:**
   - w budowaniu projektu lub zarządzaniu zadaniami, sortowanie topologiczne może pomóc w ustaleniu kolejności, w jakiej należy wykonywać zadania, uwzględniając ich zależności,
   - w dziedzinach takich jak analiza sieci społecznych, analiza sieci transportowych lub analiza zależności w projekcie informatycznym, sortowanie topologiczne może pomóc w zrozumieniu zależności między różnymi elementami,
   - w przypadku kompilatorów, sortowanie topologiczne jest używane do określania kolejności kompilacji plików źródłowych tzn. jeśli plik A zależy od pliku B, to plik B musi zostać skompilowany przed plikiem A

4. **Zawartość programu:**
   Program sortujący składa się z pliku Topology_sort.c zawierającym cały algorytm stworzony na podstawie algorytmu BFS oraz Topology_sort.h będącym headerem z deklaracjami funkcji.
   Spis funkcji:
   - CreateGraph – w interakcji z użytkownikiem tworzy graf o podanej liczbie wierzchołków, ścieżki pomiędzy nimi oraz macierz sąsiedztwa wierzchołków,
   - PrintGraph – wypisuje macierz sąsiedztwa wierzchołków,
   - VerticesDegree – sprawdza czy dany wierzchołek ma stopień 0 i jeśli tak to wywołuje Insert(),
   - Degree – zwraca stopień danego wierzchołka,
   - Insert – dodaje wierzchołek o stopniu 0 do kolejki,
   - IsEmpty – sprawdza czy kolejka nie jest pusta,
   - DeleteArr – zwraca kolejny element do wstawienia do listy posortowanej,
   - CheckCycle – sprawdza czy graf zawiera cykl,
   - PrintSorted – Wypisuje posortowany topologicznie spis wierzchołków

5. Opis algorytmu na przykładzie:

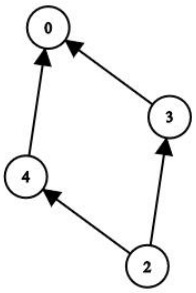
1) Zaczynami od utworzenia grafu, załóżmy że ma on postać:



2) Wyznaczamy stopnie wierzchołków i tworzymy macierz sąsiedztwa,

3) Wierzchołek nr 1 jako jedyny ma stopień 0 dlatego tylko jego dodajemy do kolejki,

4) Usuwamy wierzchołek z kolejki i dodajemy go do posortowanej listy, graf wygląda:



5) Graf uległ zmianie dlatego aktualizujemy stopnie wierzchołków oraz macierz sąsiedztwa, a następnie powtarzamy to co robiliśmy do tej pory, w kolejnych krokach graf będzie wyglądać: