CMPE 343 HW 3 Report

4865115490,
Fatih Mehmet Yiğitel,
Section 02,
HW3

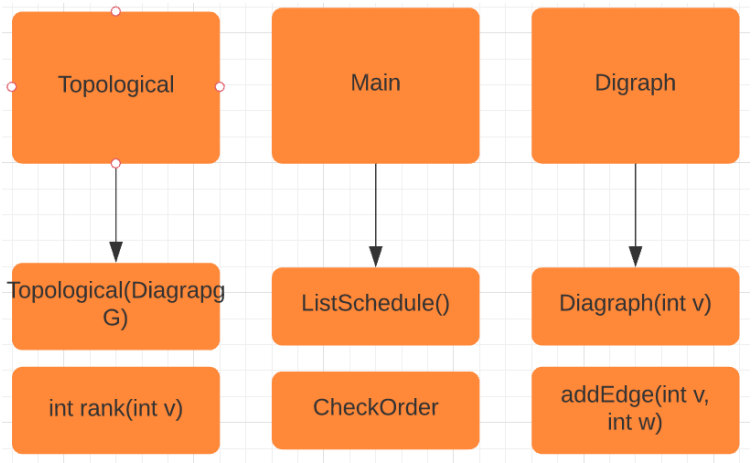
Problem Statement and Code Design:

We do task order with using topological sort. We order them with considering their prerequisites. To do that I used diagraph and topological sort class. Also, other classes for digraph and topological classes data structure implementations.

At first to get input from the text file I use scanner object and I keep the data in arrays. After the getting input and topological sort I rearrange my array according to their prerequisites and placed them into the week arrays.

Secondly, to check and giving information about order I write CheckOrder method and it gives to the user which task is before/after which tasks.

These are the important classes and their important constructor or methods.



Implementation and Functionality:

I use so many classes to implement Topological.java and Digraph.java class such as Bag, DepthFirstOrder, Digraph, DirectedCycle, DirectedEdge, EdgeWeightedDigraph, Queue

Diagraph class:

```
public Digraph(int V)
```

In the Diagraph constructor we create Bag datatype to keep the vertex's adjacent. And initialize indegree array to keep degrees of the vertex that includes only the number of edges directed into a vertex in a directed graph.

```
public int V()
```

It returns the number of vertices in this digraph**.**

```
public int E()
```
It returns the number of edges in this digraph**.**

```
private void validateVertex(int v)
```
It is private methods to check validation of vertex. If the value of vertex bigger than V or less than 0 that vertex can not pass the validation test.

```
public void addEdge(int v, int w)
```
After the vertex validation it connects to vertex each other. First parameters outdegree is incremented and second parameter's indegree's incremented.

```
public Iterable<Integer> adj(int v)
```
It returns the vertices adjacent from vertex v in this digraph.

```
public int outdegree(int v)
```
It returns the number of edges directed out of a vertex in a directed graph.

```
public int indegree(int v)
```
It returns the number of edges directed into a vertex in a directed graph.

```
public Digraph reverse()
```
It reverses whole edges directions and returns reversed diagraph.

```
public String toString()
```
It prints the whole digraph using StringBuilder library.

Topological class:

```
public Topological(Digraph G)
```
In the constructor we determine we have a DAG(directed acyclic graph) or not. I it is DAG we have topological order we find the topological order. To do that we use hasCycle method to check any cycle and we find reverse post order dfs. In addition, we have int array that called rank and it keeps the rank of the vertex in the array.

```
public Iterable<Integer> order()
```
It returns a topological order of the vertices (as an iterable) if the digraph has a topological order (or equivalently, if the digraph is a DAG), otherwise it returns null

```
public boolean hasOrder()
```
It checks does the digraph have a topological order? If it has it returns true otherwise false.

```
public boolean isDAG()
```
It checks does the digraph have a topological order. If it has it returns true otherwise false.

```
public int rank(int v)
```
It returns the position of vertex v in a topological order of the digraph. It returns -1 if the digraph is not a DAG.

```
private void validateVertex(int v)
```
It is private methods to check validation of vertex. If the value of vertex bigger than V or less than 0 that vertex cannot pass the validation test.

Main class:

There are two void methods in the main class and also we have while loop to display the outputs while we are not pressing 0 to exit.

```
public static void ListSchedule(){
```

In the ListSchedule class we take the inputs from text file, and we separate them into arrays. We use diagraph's addEdge method and we create given edges in pdf. After that we implement topological sort and according to the sort, we have an new order for the tasks. We rearrange the task arrays according to this order and we placed it into weeks. (Every week can take 3 task at most) In the main we call this function when the command is equal to 1.

```
public static void CheckOrder(){
```

In the CheckOrder method we have some Boolean methods that determine where the first task and second task place is. Using that Boolean methods and with support of the if loops we return an output that says which task is comes before/after which tasks. In the main we call this function when the command is equal to 2.

We check they are in the same week

```
if(firstweek1 && secondweek1 )
```

We check if they are not in the same week which one is comes before

```
if(firstweek1){
if(secondweek2||secondweek3||secondweek4){
```

Testing:

- In testing phase if there is string input that came from user it shows input mismatch exception for the main iterated while loop

  Enter choice (0: Exit, 1: List schedule, 2: Check order):hi

  Exception in thread "main" java.util.InputMismatchException

- For the wrong input integer type input the while loop iterated.

- For the check order method for the wrong input integer type input the while loop iterated. It does not return any exception

Final Assessments:

Hardest part for me to think about finding correct topological sort order.

I understand that for the big task designing works in future I will use topological sort

It is easy to find that kind of small graphs which one has prerequisite but in the big graph it is impossible

Also, to do correct order I want to order the task using indegree method but I couldn't implement it correctly.