Graph algorithms - practical work no. 4

Due: week 11-12.

Solve the assigned problem from those below. Additionaly, you may solve one or more of the bonus problems at the end of the page. Use the abstract data type created for lab. 1; modify it if necessary.

Assigned problems

1. Write a program that, given a list of activities with duration and list of prerequisites for each activity, does the following:

- verify if the corresponding graph is a DAG and performs a topological sorting of the activities using the algorithm based on depth-first traversal (Tarjan's algorithm);
- prints the earliest and the latest starting time for each activity and the total time of the project.
- prints the critical activities.

2. Write a program that, given a list of activities with duration and list of prerequisites for each activity, does the following:

- verify if the corresponding graph is a DAG and performs a topological sorting of the activities using the algorithm based on predecessor counters;
- prints the earliest and the latest starting time for each activity and the total time of the project.
- prints the critical activities.

3. Write a program that, given a graph with costs, does the following:

- verify if the corresponding graph is a DAG and performs a topological sorting of the activities using the algorithm based on depth-first traversal (Tarjan's algorithm);
- if it is a DAG, finds a highest cost path between two given vertices, in O(m+n).

4. Write a program that, given a graph with costs, does the following:

- verify if the corresponding graph is a DAG and performs a topological sorting of the activities using the algorithm based on predecessor counters;
- if it is a DAG, finds a highest cost path between two given vertices, in O(m+n).

5. Write a program that, given an undirected connected graph, constructs a minumal spanning tree using the Kruskal's algorithm.

6. Write a program that, given an undirected connected graph, constructs a minumal spanning tree using the Prim's algorithm.

Optional, "bonus" problems

1B. For an unknown tree, we are given two of the three lists representing the vertices parsed in pre-order, post-order an in-order. Reconstruct the tree.

2B. Write a program that, given a graph, does the following:

- verify if the corresponding graph is a DAG and performs a topological sorting of the activities;
- if it is a DAG, finds the number of distinct paths between two given vertices, in O(m+n).

3B. Write a program that, given a graph with costs, does the following:

- verify if the corresponding graph is a DAG and performs a topological sorting of the activities;
- if it is a DAG, finds the number of distinct lowest cost paths between two given vertices, in O(m+n).