Class Mini Project

Assigned: Friday 11/20/2015; Due: Thursday 12/11/2015 via Oncourse.

(total: 100 points)

Consider a directed acyclic graph (DAG) $G = (V_G, E_G)$, where V_G is a set of vertices and E_G is a set of directed edges. A sub-DAG $H = (V_H, E_G)$ of G is called consistent if it has the following property: $V_H \subseteq V_G$ and for any $v \in V_H$ all of the parents of v must also be in V_H.

Develop and implement an algorithm that enumerates all consistent sub-DAGs for a given DAG G. You can assume the DAG has a single root node (node without any parents). Your algorithm must be sub-exponential in the number of nodes V_G; that is, it is not allowed to generate all subsets of the set of nodes V_G and then check if each such subset is consistent. However, you can use this brute-force algorithm to test your new algorithm on various small problems.

a) (30 points) Special case: The DAG G is a tree. That is, each node can have at most one parent, but it can have any number of children.

b) (70 points) General case: The DAG G is not a tree. That is, each node can have any number of parents and any number of children, as provided by the data.

Develop a small example in which you can count the sub-DAGs manually. Visualize this DAG and use it to explain your algorithm. Demonstrate (or better, prove) correctness of your approach and then implement your algorithm and apply on the data sets provided to achieve counts. Your program should be able to accept an input of the type we provided in the data sets so that it can be tested independently. You can use any programming language you are comfortable with.

The data provided contains four graphs with their parent-child pairs. Each term represents a node in the graph. The root node corresponds to the term that has an underscore in its term name. For example, in bpo_names.txt, the term GO:0008150 is the root of the DAG.

Note that some of these graphs are probably too large, so the best way to start is to use mini-mfo.txt and then larger DAGs later.

Policies:

All code (if applicable) should be turned in when you submit your assignment. Use Matlab, Python, R, or C/C++.

Policy for late submission assignments: Unless there are legitimate circumstances, late assignments will be accepted up to 5 days after the due date and graded using the following rule:

on time:	your score $\times$ 1
1 day late:	your score $\times$ 0.9
2 days late:	your score $\times$ 0.7
3 days late:	your score $\times$ 0.5
4 days late:	your score $\times$ 0.3
5 days late:	your score $\times$ 0.1

For example, this means that if you submit 3 days late and get 80 points for your answers, your total number of points will be $80 \times 0.5 = 40$ points.

You can form teams of up to three of your classmates. Your submitted report needs to discuss contributions of each member of the group. All members of the group need to submit the same report before submission deadline. All the sources used for problem solution must be acknowledged, e.g. web sites, books, research papers, personal communication with people, etc. Academic honesty is taken seriously; for detailed information see Indiana University Code of Student Rights, Responsibilities, and Conduct.

Good luck!