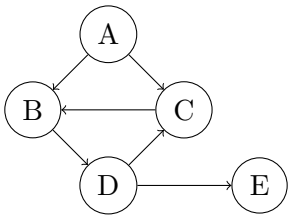
COMP285: Analysis of Algorithms
North Carolina A&T State University — Homework #4
Dr. Allison Sullivan — Due: September 21, 2018 (*in class quiz*)

Homework #4

You should try to solve these problems by yourself. I recommend that you start early and get help in office hours if needed. If you find it helpful to discuss problems with other students, go for it. **You do not need to turn in these problems. The goal is to be ready for the in class quiz that will cover the same or similar problems.**

Problem 1: DFS and BFS: The Basics

Consider the following directed graph:



(a) Draw the DFS tree for this graph, starting from node A. Assume that DFS traverses nodes in alphabetic order. (That is, if it could go to either B or C, it will always choose B first).

(b) Draw the BFS tree for this graph, starting from node A. Assume that BFS traverses nodes in alphabetic order. (That is, if it could go to either B or C, it will always choose B first).

Problem 2: Cycle Detection

Give an algorithm to detect whether a given undirected graph $G = (V, E)$ contains a cycle. Prove that your algorithm is correct (i.e., if there is a cycle it will output one and if not it will say there is no cycle). The running time of your algorithm should be $O(|V| + |E|)$.

Problem 3: Depth First Search

(a) During the execution of depth first search, we refer to an edge that connects a vertex to an ancestor in the DFS-tree as a *back edge*. Either prove the following statement or provide a counter-example: if G is an undirected, connected graph, then each of its edges is either in the depth-first search tree or is a back edge.

(b) Suppose G is a connected undirected graph. An edge whose removal disconnects the graph is called a *bridge*. Either prove the following statement or provide a counter-example: every bridge e must be an edge in a depth-first search tree of G.

Problem 4: DFS for DAGs

During Spring Break, Emilie's grandmother taught her how make lemon-meringue pie. Upon returning, Emilie can remember all the steps she needs to perform, but unfortunately, she cannot

recall the precise order in which she is supposed to do them. However, being a logical person, she figures that the precise total order cannot be important, but certain steps must precede other steps. Emilie recalls that the recipe involves the following steps:

1. Spread meringue on top of filling.
2. Cook filling until it thickens.
3. Combine flour, sugar, and cornstarch.
4. Heat oven.
5. Bake pie shell.
6. Whip sugar and egg whites to form a meringue.
7. Pour filling into baked pie shell.
8. Make pie shell.
9. Add diluted lemon juice to dry ingredients.
10. Bake pie until golden brown.
11. Add butter and egg yolks to filling.
12. Cook filling until it boils.
13. Dilute lemon juice.

And she can deduce the following common-sense constraints on the order of the steps:

- 4 must precede 5 and 10
- 6 must precede 1
- 1 must precede 10
- 13 and 3 must precede 9
- 9 must precede 12
- 11 must precede 2
- 12 must precede 11
- 2 must precede 7
- 7 must precede 1 and 10
- 8 must precede 5
- 5 must precede 6 and 7

(a) Draw a directed graph with states corresponding to the 13 steps of the recipe and a directed edge from any step to any other step that must follow it according to Emilie's constraints. A topological sort of this graph will give a feasible order for performing the steps.

(b) Emulate DFS on the graph and draw a picture of the resulting forest, plus a list of the order of events in which you discover and finish each node. Start with the lowest-numbered node that has no precedessors, and from then on, when you must make a choice, choose the lowest-numbered unvisited node. Identify edges as tree edges, forward edges, or cross edges.

(c) Demonstrate that the order of discover events in your list in part (b) does not yield a correct order for the steps, that is, performing the steps in this order does not produce a lemon-meringue pie.

(d) Now consider the reverse of the order of finish events (topological sort). List the steps. Does this satisfy all the ordering constraints? Would this correctly prepare the pie?

(e) The key to why this works is that, in performing the DFS of an acyclic directed graph, there are no situations where there is an edge in the graph from a node u to another node v, but u finishes before v finishes. Explain why no such situation arises, in your own words.

Problem 5: Discipline in Groups of Children

Your job is to arrange n rambunctious children in a straight line, facing front. You are given a list of m statements of the form "i hates j". If i hates j, then you do not want to put i somewhere behind j because then i is capable of throwing something at j.

(a) Give an algorithm that orders the line (or says it's not possible) in $O(m + n)$ time.