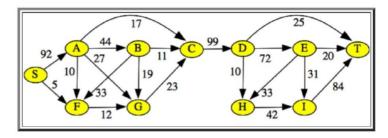
Suppose you are determining the maximum flow through the graph to the right. Tell me your first augmenting path using each of the following path determination algorithms:

- Question 1: Greedy DFS
- Question 2: Modified Dijkstra
- Question 3: Edmonds-Karp



The maximum flow through this graph is 51. You've asked your friend Skippy to tell you what the minimum cut of this graph is. Below are four potential answers that Skippy gives you. For each of these answers, tell me whether the answer is correct or not. If incorrect, tell me why.

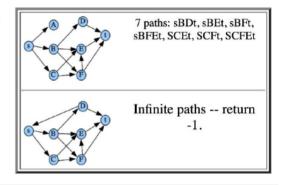
Question 4: 51

Question 5: CD

Question 6: AC, BC, GC Question 7: ET EI

Question 8: You are given a directed graph with a starting node s and an ending node t. All nodes are reachable from s, and there are no edges from t to any other node in the graph. Your job is to tell me how many distinct paths there are from s to t. If there are an infinite number of paths, return negative one.

I want you to tell me the steps that you would take to solve this problem in terms of an algorithm or mulitple algorithms that you have learned in this class. Be precise about how you are using the algorithm/s. Tell me the running time. You may use V for the number of vertices, and E for the number of edges.



There are two examples to the right.

Question 1 -- Do a DFS where you travel the edges from high to low. You'll get: SABFGCDEHIT.

Question 2 -- Take a look at the edges into C. The biggest of these is 23, so any path that includes GC and has a flow of 23 will do. One example is SAGCDT. There are a bunch of other paths that will work, like SABFGCDEHIT.

Question 3 -- The path from S to T with the fewest nodes: SACDT.

Question 4 -- No. 51 is the total weight of the cut, but it's not a cut. A cut is a collection of edges, not a number.

Question 5 -- No. CD is a cut, but its total weight is 99.

Question 6 -- Yes -- it is a cut and the sum of the weights is 51.

Question 7 -- No -- their sum is 51, but they are not a cut -- when you remove them, S is still connected to T.

Question 8: The most obvious thing here is that cycles can lead to an infinite number of paths. So a first step might be to do cycle detection and return -1 if you discover a cycle. That would be using DFS, and its running time complexity is O(E). You don't need to say O(N+E), because the problem says that every node is reachable from s.

There's a flaw with this answer, though. Look at the following ASCII-Art graph:



You'll note that this one has a cycle, but since it's not on a path from S to T, it doesn't matter.

So you need to do something else.

I think the best answer is the following. First, do a DFS to see if there is a cycle that includes S. If so, return -1.

Now, you need to find all of the nodes that are not on paths from S to T, and remove them from the graph. To do that, I would reverse all of the edges, remove S from the graph, and do a DFS from T, marking all nodes reachable from T. May as well detect cycles in this DFS, and return -1 if you find one. The marked nodes are all of the nodes on paths from S to T.

Delete the unmarked nodes, put S back on the graph, and restore the edges to their normal direction. Now do a topological sort to count the paths. T. This is just like the IncreasingSubsequences Topcoder Problem that we went over in class.

All parts are O(IEI).

I gave this one as an exam question, many years ago and no one came close to getting it right. It's an example of an exam question that is simply too hard for an exam. I didn't punish the students that year - I simply reduced the value of the question.

Regardless, you should understand the answer.