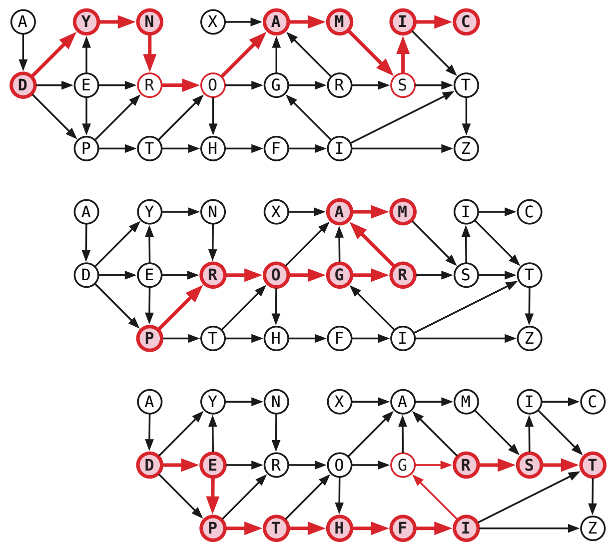
1. Let G be a directed acyclic graph where each node has a label from some finite alphabet; different nodes may have the same label. Any directed path in G has a ***signature***, which is the string defined by concatenating the labels of its vertices. A ***subsequence*** of G is a subsequence of the signature of some directed path in G. For example, the strings `DYNAMIC`, `PROGRAM`, and `DETPHFIRST` are all subsequences of the dag shown below; in fact, `PROGRAM` is the signature of a path.



Describe and analyze an algorithm to compute the length of the longest common subsequence of two given directed acyclic graphs, that is, the longest string that is a subsequence of both dags.

2. Suppose you are given a graph G with weighted edges and a minimum spanning tree T of G.

   (a) Describe an algorithm to update the minimum spanning tree when the weight of a single edge e is decreased.

   (b) Describe an algorithm to update the minimum spanning tree when the weight of a single edge e is increased.

   In both cases, the input to your algorithm is the edge e and its new weight; your algorithms should modify T so that it is still a minimum spanning tree. *[Hint: Consider the cases $e \in T$ and $e \notin T$ separately.]*

3. When there is more than one shortest paths from one node s to another node t, it is often most convenient to choose the shortest path with the fewest edges; call this the ***best path*** from s to t. For instance, if nodes represent cities and edge lengths represent costs of flying between cities, there could be many ways to fly from city s to city t for the same cost; the most desirable these schedules is the one with the fewest flights.

   Suppose we are given a directed graph G with positive edge weights and a source vertex s in G. Describe and analyze an algorithm to compute *best* paths in G from s to every other vertex. *[Hint: What is the actual output of your algorithm? If possible, use one of the standard shortest-path algorithms as a black box.]*