

PROBLEM 1 (GRAPH ALTERATIONS) We've seen these algorithms work on normal graphs, but graphs often change or have unpredictable properties! In this problem, we'll explore the implications of those changes. Say you have a graph G with n vertices and m edges.

1. You add one edge to the graph.

(a) What is the most that the distance from s to some other vertex v can increase? decrease?

Answers: 0: the old path still exists. $n - 2$: longest path was $n - 1$, could now be 1.

(b) The (undirected) graph consists of some connected components. What is the most that the number of connected components can increase? decrease?

Answers: 0: adding an edge can not create a new component. 1: an edge can turn two components into one.

(c) Say the graph is directed. Can the number of topological orders increase? decrease?

Answer: It can only decrease – adding an edge only adds a constraint to topological orders.

(d) Say the graph is directed, and you've split it into SCCs. What is the most that the number of SCCs can increase? decrease?

Answers: 0: can't make a new component or split one. $n - 1$. A one-directional path graph can become a cycle, turning n SCCs into 1.

(e) (Bonus) How do you find the edge that decreases the number of SCCs the most?

Answer: Run Kosaraju's to find SCC's, run BFS to find farthest sink-source pair, take edge from sink to source. All SCCs on the path become part of one SCC.

2. The graph is directed. You reverse every edge in the graph.

(a) Say the original graph was a DAG. How is a topological order of the original graph related to a topological order of this new one?

Answer: Reversed.

(b) Say the original graph had c SCCs. How many can it have now?

Answer: Exactly c .

3. You take every edge in G and insert a vertex in between them, so it becomes two edges through a midpoint vertex. Each edge gets a different midpoint vertex.

(a) If the original graph was bipartite, is the new graph still bipartite? What if it wasn't?

Answers: Yes either way. Color midpoints blue, other vertices red.

(b) Say the distance from s to v was d . What is the new distance?

Answer: $2d$ – just twice as many points.

4. The graph was weighted.

(a) You increase every edge weight by 1. Is the shortest path the same?

Answer: No. A path with one long edge will be penalized less than a path with many short edges.

(b) You multiply every edge weight by 2. Is the shortest path the same?

Answer: Yes. Multiplies all paths' lengths by 2, so shortest original is shortest on new graph.

(c) (Bonus) You multiply every edge weight by -1 . Does running Dijkstra's algorithm on this graph find you the longest path on the original graph?

Answer: No. Consider a graph where there is an edge of weight 2 from A to B and a path $A \rightarrow C \rightarrow D \rightarrow B$ where each step has length 1. Then Dijkstra's on the negative graph finds -2 .

PROBLEM 2 (GRAPH MODELING PRACTICE) For each of the following problems, provide a solution that involves modeling it as a graph and running one of our known algorithms some number of times. You'll need to state the following:

- What are the vertices in your graph?
- What are the edges in your graph?
- Which algorithm would you use here? How would you use it?
- How do you use the result of the algorithm to return the answer?

1. You want to split the class into two teams to run a debate, but you know some students are in messy situationships and want to make sure they're split up. Determine whether it's possible to split the students up.

Answer: Vertices: Students. Edges: Situationships. Algorithm: 2-coloring. Coloring = Teams.

2. (Bonus) What if you also know some students are best friends and want to be together?

Answer: Treat each set of best friends as one vertex. This vertex has an edge for each situationship that any of the friends is in.

3. You've made a new social networking app called "Convrs". Users on Convrs can have asymmetric following. A group chat can only be formed if, for every two people a, b in the chat, there is a "follow chain" from a to b (a follows b , or a follows someone who follows b , etc). For any group chat, determine quickly whether this is possible.

Answer: Vertices: Users. Edges: Directed, from a to b if a follows b . Algorithm: Kosaraju's. If everyone is in the same SCC, then the group chat is possible. Otherwise, impossible.

4. You want to go from school to your friend's house, but you need to stop by the library to pick up a book first. Find the shortest path from school to your friend's house that passes through the library.

Answer: Vertices: Locations. Edges: Roads. Algorithm: Dijkstra's. Run twice, once from school and once from the library. Return distance from school to library + distance from library to friend's house.

5. (Bonus) Try it with only one run of Dijkstra's!

Answer: Make two graphs, G and G^R , where G^R has all the same vertices of G but with the edges reversed. These graphs should not share any vertices. Now, join the vertices at the library (so they have the same library vertex but otherwise different edges and vertices), and run Dijkstra's algorithm from there. Take the shortest path from library to friend's house on G , and the shortest path from library to school on G^R , and return their sum.

6. A person's "Erdős number" is the distance they are from collaborating with prolific mathematician Paul Erdős. For example, if a scholar has co-authored a paper with Paul Erdős, they have a Erdős number of 1. If they've co-authored a paper with someone who co-authored with Erdős, they have a Erdős number of 2. Given a database of all papers and authors, find everyone's Erdős number.

Answer: Vertices: Scholars. Edges: If collaborated on a paper. Algorithm: BFS. Return BFS distances from the vertex representing Erdős.

7. (Very Bonus) You want to go on a roadtrip from A to B , but you're driving a new futuristic electric car that replaces its battery at every battery station. Furthermore, different batteries can go different distances! You're starting at battery station A , and you're given a map that contains all the different battery stations, roads that connect them (with distances), and the distance that the battery at each station can take you. Find the shortest achievable path from A to B , and the batteries you'd need to take. Remember you can get to a station but skip its battery if your current battery can take you to the next station as well.

Answer: The map you're given is actually not the one you want. You want to start by constructing a new graph:

- *Vertices: Battery Stations*
- *Edges: Directed, from a battery station to any station that can be reached with the battery there. Weighted with distance.*
- *To construct this graph: Run Dijkstra's algorithm from each battery station on the original map, stopping your loop when you start visiting vertices outside of your battery's capacity. Use the visited vertices to draw edges on your new graph.*

Once you have your new graph, you can just run Dijkstra's algorithm starting from A.