

**Note:** when asked to “describe and analyze” an algorithm, you should describe the algorithm using pseudocode, prove its correctness, and analyze its worst-case running time.

Hints are provided in QR codes in the margin, numbered by their corresponding question. Some questions have multiple hints; generally the hints are in order of hintiness. If you do not have a device capable of reading QR codes, you can find the hints in the `.tex` source for this document.



### *Dijkstra’s Algorithm*

**Question 1.** Recall *Dijkstra’s Algorithm* for finding shortest paths in a directed, weighted graph.

1. Why doesn’t Dijkstra’s algorithm work if edges in the graph can have negative weights? Give an example of a graph where Dijkstra’s algorithm fails to find the shortest path between a pair of vertices.
2. What happens if we replace “smallest” in Dijkstra’s algorithm with “biggest”—that is, we use a max priority queue so we pull out the *biggest* edge on each iteration, and when we add a vertex  $u$  to the set  $S$ , for each edge  $(u, v)$  we update  $\pi[v]$  with the *maximum* of  $\pi[v]$  and the sum  $d[u] + l_{uv}$ . Can we use Dijkstra’s algorithm to find *longest* paths between nodes in a graph?

### *Some Problems from Jeff Erickson*

**Question 2.** A *feedback edge set* of an undirected, connected graph  $G$  is a subset  $F$  of the edges such that every cycle in  $G$  contains at least one edge in  $F$ . In other words, removing every edge in  $F$  makes the graph  $G$  acyclic. Describe and analyze a fast algorithm to compute the minimum weight feedback edge set of a given edge-weighted graph.

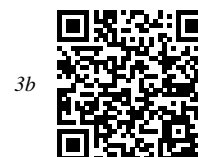
**Question 3.** Suppose you are given a graph  $G$  with weighted edges and a minimum spanning tree  $T$  of  $G$ .

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

**Question 4.** An *Euler tour* or *Euler circuit* of a graph  $G$  is a cycle that traverses every edge of  $G$  exactly once.

- (a) Prove that a connected graph  $G$  has an Euler tour if and only if every vertex has even degree.



(b) Describe and analyze a linear time algorithm to compute an Euler tour in a given graph, or correctly report that no such tour exists.

*Extra Credit*

**Question 5. (Extra credit.)** One can use MST algorithms to generate mazes. There's a nice Wikipedia entry on the topic:

[http://en.wikipedia.org/wiki/Maze\\_generation\\_algorithm](http://en.wikipedia.org/wiki/Maze_generation_algorithm)

Read the sections about randomized Kruskal's and randomized Prim's algorithms and implement a maze generation program using one of the algorithms. Your program should take three parameters, an integer width, an integer height, and a filename, and output a maze in PNG format. For example,

```
genmaze 30 40 mymaze.png
```

should generate a  $30 \times 40$  maze in the file `mymaze.png`. Hand in a printout of a sample maze generated from your program, along with a printout of your source code.

