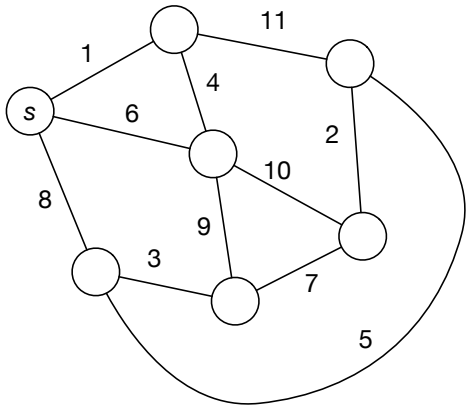
CS 4349.003 Midterm 2—Questions and Instructions

November 13, 2019

Please read the following instructions carefully before you begin.

- Write your name and Net ID on the *answer sheets* cover page and your Net ID on each additional page. Answer each of the ***four questions*** on the answer sheets provided. One sheet was intentionally left blank to provide you with scratch paper.
- Questions are not necessary given in order of difficulty, so read through them all before you begin writing!
- This exam is closed book. No notes or calculators are permitted.
- You have one hour and 15 minutes to take the exam.
- Please turn in these question sheets, your answer sheets, and scratch paper at the end of the exam period.
- Feel free to ask for clarification on any of the questions.
- Just breathe. You can do this.

1. Clearly indicate (draw) the following spanning trees for the weighted graph shown below. There may be more than one correct answer.



   (a) **(2.5 out of 10)** A depth-first spanning tree rooted at s.

   (b) **(2.5 out of 10)** A breadth-first spanning tree rooted at s.

   (c) **(2.5 out of 10)** A shortest path tree rooted at s.

   (d) **(2.5 out of 10)** A minimum spanning tree.

2. For each of the following graph problems, do the following.

   - Either *name* or *briefly describe* the fastest algorithm seen in class to solve the problem.
   - State the running time of the algorithm in terms of V and E.

   If an algorithm involves priority queues, feel free to state the running time assuming it uses a standard min-heap with $O(\log n)$ time operations. If there are multiple choices for the fastest algorithm, name any one of them. Naming or describing a correct algorithm that is slower than the best one we discussed for the problem is worth partial credit. You may assume the graph is connected for each part.

   (a) **(2 out of 10)** Given a directed graph $G = (V, E)$ with non-negative edge weights $w : E \to \mathbb{R}_{\geq 0}$ and a vertex $s \in V$, compute a shortest path tree out of s.

   (b) **(2 out of 10)** Given an undirected graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$, compute a spanning tree of G with minimum total edge weight.

   (c) **(2 out of 10)** Given a directed graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$, compute $dist(u, v)$, the shortest path distance from u to v, for all pairs of vertices u and v.

   (d) **(2 out of 10)** Given a directed graph $G = (V, E)$ and two vertices s and t in V, compute the *minimum number of edges* along any path from s to t.

   (e) **(2 out of 10)** Given an undirected graph $G = (V, E)$, compute a spanning tree of G with a *minimum number of edges*. (We did not directly discuss this problem in class, but one or more algorithms we did discuss will solve it anyway.)

3. **(10 points)** Let X be a set of n intervals on the real line. Assume the input consists of two arrays $L[1 .. n]$ and $R[1 .. n]$, representing the left and right endpoints of the intervals in X. We say that a set P of points ***stabs*** X if every interval in X contains at least one point in P.

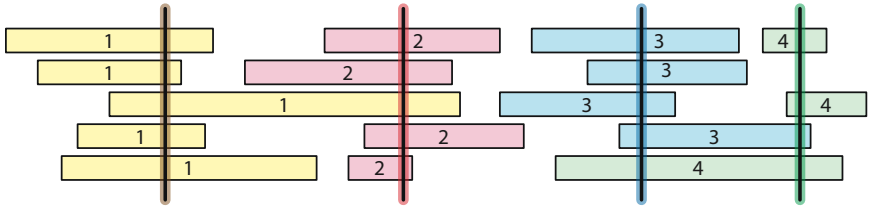


Figure 1. A set of intervals stabbed by four points (shown here as vertical segments)

Our goal is to compute a ***minimum stabbing set***, the *smallest* set of points that stabs X. For each of the following greedy algorithms for the minimum stabbing set problem, write ***Correct*** if the algorithm always constructs a minimum stabbing set or write ***Wrong*** if there is some input for which the algorithm does not produce a minimum stabbing set. *You must clearly write exactly one of* **Correct** *or* **Wrong** *to get credit for each part.* ***Proofs/counterexamples are not required***, but you may want to do them on the back of the page or your scratch paper to convince yourself that your answers are correct. Each wrong algorithm has a small counterexample.

(a) **(2 out of 10)** Choose the *left* endpoint p of the interval that *starts first*, discard all intervals containing p, and recurse. (In other words, p is the minimum value in $L[1 .. n]$.)

(b) **(2 out of 10)** Choose the *right* endpoint p of the interval that *ends first*, discard all intervals containing p, and recurse. (In other words, p is the minimum value in $R[1 .. n]$.)

(c) **(2 out of 10)** Choose a point p that is included in the *maximum number of intervals*, discard all intervals containing p, and recurse.

(d) **(2 out of 10)** Choose the *left* endpoint p of the interval that *starts last*, discard all intervals containing p, and recurse. (In other words, p is the maximum value in $L[1 .. n]$.)

(e) **(2 out of 10)** Choose a point p that is included in at least one interval, but is otherwise in the minimum number of intervals, discard all intervals containing p, and recurse.

4. It's time for the semi-annual CS 4349 Battleship Tournament. Battleship is a two-player game in which we learn which of two players is better at sinking tiny plastic ships with tiny plastic pegs. For the purposes of this problem, let's assume there are no ties in Battleship, so one of the two players wins and the other loses.

   Battleship is a game of skill and certainly not a game of luck, so we want to create an ordered list of students from best Battleship player to worst after all the games have been played. The list should be ***consistent*** with the results of the games, meaning if student i won at least one game against student j, then student i should be placed before student j in the list.

   Suppose there are n students participating in the tournament and they play m games total. We want to make a consistent list of students (if possible) by topologically sorting the vertices of a graph.

   (a) **(2 out of 10)** What set should we use for the vertices of the graph? (Either define the set plainly in English, or provide a short algorithm for creating a useful vertex set.)

   (b) **(2 out of 10)** What set should we use for the edges of the graph? (Either define the set of edges plainly in English, or provide a short algorithm for creating a useful edge set.) Don't forget to specify the direction of the edges.

   (c) **(2 out of 10)** What must be true about the graph for a consistent list of students to exist?

   (d) **(2 out of 10)** Assuming a consistent list of students does exist, what will the topological sort return, and how will that help us order the students according to their ability to play Battleship?

   (e) **(2 out of 10)** *In terms of n and m*, how long will it take to build the graph, run topological sort, and build the consistent list of students if a consistent list exists?