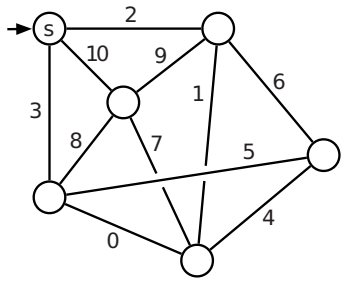
CS 6363.005 Midterm 2—Problems and Instructions

April 16, 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 problem sheets, your answer sheets, and scratch paper at the end of the exam period.
- Feel free to ask for clarification on any of the problems.

1. Clearly indicate the following spanning trees for the weighted graph shown below.



   (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. Suppose we have n skiers with heights given in an array $P[1 .. n]$, and n skis with heights given in an array $S[1 .. n]$. We wish to assign a ski to each skier, so that the average difference between the height of a skier and her assigned ski is as small as possible.

   (Formally, we want to find a permutation σ such that the expression $\frac{1}{n}\sum_{i=1}^{n} |P[i] - S[\sigma(i)]|$ is as small as possible.)

   (a) **(4 out of 10)** The following greedy strategy ***does not*** lead to an optimal set of assignments.

   Let skier i and ski j have the smallest height difference between any ski and skier. Assign ski j to skier i and recursively find the best assignment for the remaining skis and skiers.

   Describe two ***small*** arrays $P[1 .. n]$ and $S[1 .. n]$ where the proposed strategy does not lead to an optimal set of assignments.

   (b) **(6 out of 10)** The following greedy strategy ***does*** lead to an optimal set of assignments.

   Assign the lowest ski to the lowest skier and recursively find the best assignment for the remaining skis and skiers.

   Prove that the proposed strategy leads to an optimal set of assignments.

3. (a) **(5 out of 10)** Recall Dijkstra's algorithm for computing single source shortest paths from a given vertex s.

```
DIJKSTRA(s):
  INITSSSP(s)
  INSERT(s, 0)
  while the priority queue is not empty
      u ← EXTRACTMIN()
      for all edges u→v
          if u→v is tense
              RELAX(u→v)
              if v is in priority queue
                  DECREASEKEY(v, dist(v))
              else
                  INSERT(v, dist(v))
```

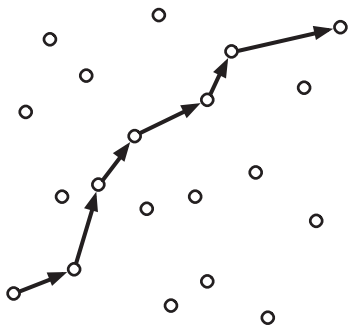
Suppose we implement DIJKSTRA using a priority queue that performs INSERT in α time units, EXTRACTMIN in β time units, and DECREASEKEY in γ time units. What is the total running time of DIJKSTRA(s) when run on a directed graph $G = (V, E)$ with non-negative edge lengths? Briefly justify your answer.

(b) **(5 out of 10)** Suppose are you given an edge-weighted directed graph $G = (V, E)$ where edge weights could be positive, zero, or negative along with a source vertex $s \in V$ and target vertex $t \in V$. You are guaranteed two facts about G:

- There are no negative cycles, and
- the shortest path from s to t uses at most k edges.

Using big-Oh notation in terms of one or more of k, V, and E, how quickly can you compute the shortest path from s to t? You must briefly justify your answer, but you do not need to give complete details on the relevant algorithm.

4. A ***polygonal path*** is a sequence of line segments joined end-to-end; the endpoints of these line segments are called ***vertices*** of the path. The ***length*** of a polygonal path is the sum of the lengths of its segments. A polygonal path with vertices $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$ is ***monotonically increasing*** if $x_i < x_{i+1}$ and $y_i < y_{i+1}$ for every index i—informally, each vertex of the path is above and to the right of its predecessor. See below for an example of a monotonically increasing polygonal path.



Suppose are you given a set S of n points in the plane, represented as two arrays $X[1 .. n]$ and $Y[1 .. n]$. Describe and analyze an algorithm to compute the length of the longest monotonically increasing path with vertices in S. Assume you have a subroutine LENGTH(x, y, x', y') that returns the length of the segment from (x, y) to (x', y'). You ***do***

not need to justify correctness of your algorithm (time is short), but you ***do*** need to explain your running time analysis.

Any correct algorithm running in $O(n^2)$ time is worth full credit. *[Hint: Reduce to a graph problem from class or do dynamic programming.]*