CS270 Spring 2020: Intro to Algorithms Exam I

Instructions:

1. This is a 2-hr exam. Closed book and notes
2. If a description to an algorithm or a proof is required please limit your description or proof to within 150 words, preferably not exceeding the space allotted for that question.
3. No space other than the pages in the exam booklet will be scanned for grading.
4. If you require an additional page for a question, you can use the extra page provided within this booklet. However please indicate clearly that you are continuing the solution on the additional page.
5. Do not detach any sheets from the booklet. Detached sheets will not be scanned.
6. If using a pencil to write the answers, make sure you apply enough pressure, so your answers are readable in the scanned copy of your exam.
7. Do not write your answers in cursive scripts.

1) 20 pts
   Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

   [TRUE/FALSE]
   In any graph we have that $|E| = \Theta(|V|^2)$

[TRUE/FALSE]
If Path P is the shortest path from u to v and w is a node on the path, then the part of the path from u to w is also the shortest path from u to w.

[TRUE/FALSE]
Dijkstra algorithm is able to find the Shortest Path in directed and undirected graphs with positive edge weights.

[TRUE/FALSE]
Kruskal's algorithm can fail in the presence of negative cost edges.

[TRUE/FALSE]
Given a graph G. If the edge e is not part of any MST of G, then it must be the maximum weight edge on some cycle in G.

[TRUE/FALSE]
In a binary max heap containing n numbers, the smallest element can be found in time O(logn).

[TRUE/FALSE]
Nodes in a binomial heap can have more than 2 children.

[TRUE/FALSE]
An array with following sequence of terms [16, 14, 10, 10, 12, 9, 3, 2, 4, 1] is a binary max-heap.

[TRUE/FALSE]
In the stable matching problem involving n men and n women, for any given set of preference lists, there will be at most two stable matchings.

2) 12 pts
Indicate if the expression of each row is Θ, O or Ω of the expressions in each column, as the example.
If an expression is both O and Ω you should write Θ.
Note 1: the solutions for the first function ($2n^2$) are given to you as an example.
Note 2: If you need to make an assumption about the base of *log* use base 2.

	n^2	$n \log(n)$	2^n
$2n^2$	Θ	Ω	O
$4n + \log^5 n$			
$\log(n^n)$			
$2^{\log n^2}$			
$100n^{1.9}$			

Solution

	n^2	$n \log(n)$	2^n
$2n^2$	Θ	Ω	O
$4n + \log^5 n$	O	O	O
$\log(n^n)$	O	Θ	O
$2^{\log n^2}$	Θ	Ω	O
$100n^{1.9}$	O	Ω	O

Rubric: 1 point for correct answer

3) 16 pts

Given a directed graph with m edges and n nodes where every edge has weight as either 1 or 2, find the shortest path from a given source vertex ‘s’ to a given destination vertex ‘t’. Expected time complexity is O(m+n).

Solution:

We can modify the graph and split all edges of weight 2 into one vertex and two edges of weight 1 each, like edge (u,v) of weight 2 to (u,u’) of weight 1 and (u’,v) of weight 1. In the modified graph, we can use BFS to find the shortest.

We can modify BFS so that it re-enqueues nodes discovered over edges of weight 2, the first time they are popped.

We can modify BFS to use 3 lists instead of a queue. One list stores the current nodes, another stores the ones that are 1 away, and another stores the ones that are 2 away. Loop through nodes in the current list and add their neighbors to the appropriate other lists. Once all current nodes are depleted, make the 1 aways the currents, the 2 aways the 1 aways, and the current the 2 aways.

We can modify Dijkstra’s to use the constant bound of 2m on path lengths to design a priority queue that you can extract min from in constant time. Make an array of lists of length 2m. As you discover new paths, insert them into the list at the index of their length. Going through the list from start to end will let you explore paths in order from shortest to longest and implement Dijkstra’s in O(m + n).

Rubric:

+2 wrote something involving applying a graph algorithm to the graph

+4 their algorithm finds the shortest path

+4 they use BFS and it runs in O(m + n)

+2 they acknowledge that weight-2 edges need to be considered the same as two weight-1 edges (but don’t necessarily explain how to implement this)

+2 they have the core idea for an algorithm that would work (with minor added details)

+2 their algorithm finds the shortest path and runs in O(m + n) (i.e. full points)

18 pts

Suppose you are the registrar for USC and you need to schedule classrooms to be used for various classes. There are n available classrooms and n classes that need rooms at a given time. Each classroom *i* has a maximum capacity c_i , and each class *j* has s_j students. A classroom can't take a class that is over its maximum capacity. Assume that there exists a perfect matching of classrooms and classes.

a) Design a greedy algorithm which returns a perfect matching of classes to appropriate classrooms (10 pts)

Solution:

Sort classrooms by capacities c_i and classes by students s_j in the same order (both ascending or both descending, either works). Schedule classrooms for classes in that order.

Rubric:

- Using heaps/sorting to arrange classes and classrooms (in same order) and matching – (10 pts)
- Any suboptimal algorithm that imitates the above property (arranging both in same order) – (7 pts)
- Minimizing c_i - s_j or finding minimum classroom for each class/largest class for each classroom or using Gale-Shapely to implement any of these previous properties – (7 pts)

Any correct algorithm with runtime complexity O(n log n) will receive 10 pts. Furthermore, any algorithm with minor errors will still be considered acceptable.

a) Give the asymptotic worst-case complexity (2 pts)

Solution:

Sorting both classes and classrooms takes O(nlog n) each.

Rubric:

- O(n log n) if part (a) is correct – (2 pts)
- O(n^2) if any suboptimal algorithm in part (a) is correct – (1 pts)

b) Prove that your algorithm produces correct results (6 pts)

Solution:

Let's consider classrooms in descending order of their capacities $\{c_1 , c_2 , \ldots . c_n\}$, then our solution schedules them with classes having students $\{s_1 , s_2 , \ldots . s_n\}$ in the

same order (descending). To prove that our algorithm works, we just need to show that it produces a perfect matching.

Assume that a perfect matching OPT exists, which might schedule some classes differently than our solution. We define an inversion to be the case when in OPT there is a class with students s_j scheduled before another class with students $s_i > s_j$ (considering classrooms in descending order of their capacities). Thus, OPT can have at most n choose 2 inversions.

Suppose there exists an inversion in the OPT. Our solution must have s_i before s_j , whereas OPT has s_j before s_i for some $i < j$. Since OPT schedules c_j with s_i , then $c_j \geq s_i$ .We also know that $c_i \geq c_j$ and $s_i \geq s_j$. Therefore, $c_i \geq c_j \geq s_i \geq s_j$ and we can swap s_i and s_j in OPT to create OPT' which also satisfies the perfect matching condition (A classroom can't take a class that is over its maximum capacity).

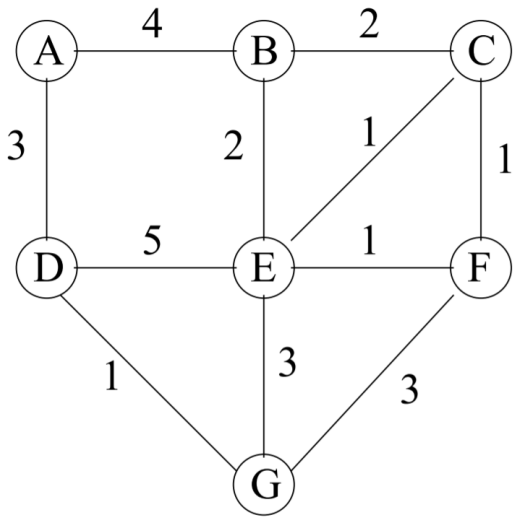
If we continue removing inversions, we eliminate all the differences between OPT and our solution without violating the perfect matching condition. In other words, our solution is a perfect matching.

Rubric:

- Inversion Argument
  - Using the argument correctly (with or without induction) – (0-4 pts)
  - Proving that inversion removal does not make OPT worse – (0-2 pts)
- Exchange Argument (OPT agrees)
  - Using the argument correctly (with induction) – (0-4 pts)
  - Proving that OPT can match our choice and not get worse – (0-2 pts)
- Contradiction
  - Using the argument correctly (with or without induction) – (0-4 pts)
  - Proving that $C_i >= S_i$ for any i – (0-2 pts)

For any other technique, points are assigned accordingly. Please note that for this section, the points are subject to your answer and grader's interpretation.

4) 16 pts



a) Execute Prim's algorithm on the above graph starting at vertex A. If there are any ties, the vertex with the lower letter comes first. List the nodes in the order in which they are connected to the MST. (8 pts)

A, D, G, E, C, F, B

b) Execute Dijkstra's algorithm on the above graph starting at vertex A. If there are any ties, the vertex with the lower letter comes first. List the nodes in the order in which they are found. (8 pts)

A, D, B, G, C, E, F

Rubric:

a)

+1 for every correct node in order, until the first pair that is wrong (afterwards we don't count)

+1 for all seven nodes in correct order

Note:

1) Lower case letters are also accepted

2) If the answer only differs by a sequence of two nodes (and must be because of tie-breaking), then a total of 2 points are deducted

3) Regardless of the type of mistake (whether forgetting to include the final node or swapping the sequence of two nodes), a MINIMUM of 2 points are taken off (1 point for final answer not correct and another 1 for the mistake)

b)

+1 for every correct vertex in order, until the first one that is wrong (afterwards we don't count)

+1 for all seven points in the correct order

Note:

1) Lower case letters are also accepted

2) If the answer only differs by a sequence of two nodes (and must be because of tie-breaking), then a total of 2 points are deducted

3) Regardless of the type of mistake (whether forgetting to include the final node or swapping the sequence of two nodes), a MINIMUM of 2 points are taken off (1 point for final answer not correct and another 1 for the mistake)

4) If the student list edges instead of nodes, the a total 2 points are deducted for this ENTIRE question ((a)+(b)), on top of other mistakes