CSE 421 : Sample Midterm Exam 2 Solutions

Name:	NetID: @uw.edu

Instructions

- This Sample Midterm was assembled from problems given in previous 421 exams.
- The exam here is approximately the length of an "in-class" (50 minute) exam. Yours may be of somewhat different length.
- No electronics are allowed at the exam. These include laptops, tablets, calculators, smartphones, smart watches, etc. If it has an on/off switch, it needs to be off. If it does not, it needs to be stored in your bag.
- You will receive a 1-page two-sided reference sheet, a sample of which is part of this sample exam.
- This is a *closed book* exam with the following exception:
  - Other than your writing implements, you are permitted one piece of 8.5x11 inch paper with handwritten notes (notes are allowed on both sides of the paper).
  - Your handwritten page must be clearly labelled with your name and Student number or UW Netid.
  - You must hand in both your handwritten page and your copy of the reference sheet along with your completed exam.

Advice

- Write your solutions in the appropriate spaces. The backs of pages are also available for solutions. if you use them to extend an answer to a question, please put a pointer from that question to the place on the back that you use.
- Move around the exam; if you get stuck on a problem, save it until the end.
- Proofs are not required unless otherwise stated.
- Remember to take deep breaths.

Question	**Max points**
Short Answer	0
k-wise merge	0
Product-Sum	0
Path in a DAG	0
Total	**???**

1. Short Answer

(a) True or False: Let $G = (V, E)$ be a weighted undirected graph. Suppose all edges in E have distinct weights except for two edges which share the same weight. The graph has exactly two minimum spanning trees.

Solution:

False. Consider a graph in which all edges are of cost 2 except for a single edge of cost 1. If that is not connects to s (start node for shortest paths) then the MST will use that edge but the SPT won't.

(b) True or False: Let $G = (V, E)$ be a graph where its BFS tree contains at least 3 layers. Consider the cut $(L_0 \cup L_1, L_2)$ (i.e. all nodes that are 2 edges away from s form one side of the cut). All edges which cross the cut are tree edges.

Solution:

False. You could have one node in layer L_2 which shares an edge with two nodes in L_1. In this case only one of those edges will be a tree edge.

(c) True or False: If $T(n) = 12T(n/6) + n^2, T(1) = 1$, then $T(n) = O(n^2 \log n)$. Briefly justify your answer.

Solution:

False. By master theorem, since $6^2 > 12, T(n) = O(n^2)$.

(d) True or False: If $T(n) \leq 2T(n/2) + n \log n$, and $T(1) = 1$, then $T(n) = O(n^2)$. Briefly justify your answer.

Solution:

True. Since $T(n) \leq 2T(n/2) + n^2$, and the master theorem says that recurrences solves to $O(n^2)$

(e) True or False: If we're given preference lists for there is a stable matching that gives no agent their first choice then there exists at least 3 stable matchings.

Solution:

True. A proposer-optimal matching will always match at least one proposer with their first choice. A receiver-optimal will match at least one receiver with their first choice. Since this matching is neither, it must be a third table matching.

2. Celebrity

There is a party of n guests $A = [a_1, ..., a_n]$, with one celebrity in attendance. We define a "celebrity" as a guest whom all other guests know, yet who knows no other guests. Specifically, if guest a_i is the celebrity, $\forall a_j \neq a_i$, $knows(a_j, a_i) == true$ and $knows(a_i, a_j) == false$. Other guests may or may not know each other, as "normal" parties go.

(a) Describe an efficient divide and conquer algorithm CELEBRITY(A) which returns the celebrity. Suppose you have access to the $knows$ function above, and that it runs in constant time.

Solution:

We begin by pairing up guests such that each even-indexed a_i is paired with a_{i+1}. If a_i knows a_{i+1}, then a_i is not the celebrity. If a_i does not know a_{i+1} then a_{i+1}. As such, at most one member of each pair can be ruled out as the celebrity. We can therefore conquer on the remaining guests. For a base case, if there are only 2 celebrities left then the one who is known by the other must be the celebrity.

(b) What is the run time of your algorithm in terms of the number of guests, n?

Solution:

After each round we divide the number of remainin guests in half. There are $n/2$ pairs to consider in the divide step, and so it requires linear time. Overall, the running time is given by $T(n) = T(n/2) + n$. We can apply the master theorem where $a = 1, b = 2, k = 1$, giving a running time of $\Theta(n)$.

3. Present prank

I want to play a prank on my brother for his birthday by putting his tiny gift in a bunch of progressively larger boxes, so that when he opens the large box there's a smaller box inside, which contains a smaller box, etc. until he's finally gotten to the tiny gift inside. Write a dynamic programming algorithm which, given a list of dimensions (length, width, and height) of n boxes, returns the maximum number of boxes I can nest (i.e. gives the count of the maximum number of boxes my brother must open). You may assume you have access to a method $\texttt{FITS}(b_1, b_2)$ which indicates wether the box $b_1 = (x_1, y_1, z_1)$ fits within box $b_2 = (x_2, y_2, z_2)$.

(a) First, show that this greedy algorithm is not correct: First select the box with the smallest volume. Next select the box with the smallest volume which fits inside of it. Repeat.

Solution:

We could have a box that is very long an narrow compared to some other nearly cubic boxes. For example, the boxes $(3, 0.5, 0.5)$, $(1, 1, 1)$, $(2, 2, 2)$. The first box has the smallest volume 0.75, but it doesn't fit in either of the other two boxes. The greedy algorithm will return 1 box, whereas a 2 box solution exists.

(b) Give the optimization formula for computing the maximum depth nesting of boxes if box j is the outer-most box.

Solution:

$$OPT[j] = \begin{cases} 1 & \text{if no other box fits} \\ \max_{k \text{ where } \texttt{FITS}(k,j)} (OPT[k] + 1) & \text{otherwise} \end{cases}$$

4. Stunt Planning

Jackie Chan is trying to plan a stunt to rapidly descend a tall building. Jackie will leap from balcony to balcony on the building until he reaches the ground. The building is n meters tall, and you have a list of the heights of the m balconies $h = [h_1, \dots h_m]$ where h is sorted from highest to lowest balcony. The ground is at height 0. Jackie knows that any fall greater than k feet will cause injury.

(a) Describe an efficient greedy algorithm `SafeDescent`(n, h, k) that finds the shortest sub sequence of balconies which safely get Jackie to the ground.

Solution:

idea: repeatedly select the lowest balcony that is within distance k.

function SafeDescent(n, h, k)
  $i = 1$
  $ans =$ empty list
  while $n > 0$ **do**:
    while $n - h_i > k$ **do**
      $i+ = 1$
    add h_i to ans
    $n = h_i$
  return ans

(b) Argue the correctness of your algorithm using either a greedy stays ahead argument or else an exchange argument.

Solution:

We procede by exchange argument. Let $G = [h_{g_1}, ...h_{g_x}]$ be our greedy solution which contains x balconies. and $O = [h_{o_1}, ...h_{o_y}]$ be an alternative solution which contains y balconies. Assume G and O match for the first j balconies. We will do an exchange so that O will be a valid solution which matches G for $j + 1$ balconies.

Because G always selects the lowest balcony that is still within k, we know that $h_{g_{j+1}} \leq h_{o_{j+1}}$ and is still within k of h_{g_j}. Because $h_{g_{j+1}}$ is lower than $h_{o_{j+1}}$ it must be closer to $h_{o_{j+2}}$, meaning it is certainly within k. The sequence $[h_{g_1}, ..., h_{g_j}, h_{g_{j+1}} ..., h_{o_y}]$ is a valid sequence that agrees with G for $j + 1$ balconies.