CECS 528, Exam 2 Version a Solutions, March 24th, Spring 2023, Dr. Ebert

Rules for Completing the Problems

NO NOTES, BOOKS, ELECTRONIC DEVICES, OR INTERPERSONAL COMMUNICATION allowed when solving these problems. Make sure all these items are put away BEFORE looking at the problems. FAILURE TO ABIDE BY THESE RULES MAY RESULT IN A FINAL COURSE GRADE OF F.

Directions

Choose up to **six problems** to solve. Clearly mark each problem you want graded by placing an 'X' or check mark in the appropriate box. **If you don't mark any problems or mark more than six, then we will record grades for the six attempted problems that received the *fewest* points.**

Problem	1	2	3	4	5	6	LO1	LO2	LO3	LO4
Grade?										
Result										

Your Full Name:

Class ID Number:

1. Note: correctly solving parts a and b for this problem counts for passing LO5. Recall that the `Minimum Positive Subsequence Sum (MPSS)` problem admits a divide-and-conquer algorithm that, on input integer array a, requires computing the mpss of any subarray of a that contains both $a[n/2-1]$ and $a[n/2]$ (the end of a_{left} and the beginning of a_{right}). For

$$a = -28, 12, -19, 33, -14, -29, 39, 42, -18, 21$$

   a. Provide the two sorted arrays b and c from which the minimum positive sum $b[i] + c[j]$ represents the desired mpss, for some i in the index range of b and some j within the index range of c. (8 pts)

   Solution.

   Left sums: -14, 19, 0, 12, 19

   Right sums: -29, 10, 52, 34, 55

$$b = -16, -14, 0, 12, 19.$$

$$c = -29, 10, 34, 52, 55.$$

   b. For the b and c in part a, demonstrate how the minimum positive sum $b[i] + c[j]$ may be computed in $\mathrm{O}(n)$ steps. (10 pts)

   Solution.

i	j	$b[i] + c[j]$	**Action**
0	4	39	$j--$
0	3	36	$j--$
0	2	18	$j--$
0	1	-6	$i++$
1	1	-4	$i++$
2	1	10	$j--$
2	0	-29	$i++$
3	0	-17	$i++$
4	0	-10	None

   MPSS Middle equals 10.

   c. State the running time $T(n)$ for the above MPSS divide-and-conquer algorithm. (7 pts)

   Solution. $T(n)$ satisfies $T(n) = 2T(n/2) + n \log n$. Master Theorem does not apply, but using the Master Equation one gets $T(n) = \Theta(n \log^2 n)$.

2. Recall that the `find_statistic` algorithm makes use of Quicksort's partitioning algorithm and uses a pivot that is guaranteed to have at least

$$3(\lfloor \frac{1}{2} \lceil \frac{n}{5} \rceil \rfloor - 2) \geq 3(\frac{1}{2} \cdot \frac{n}{5} - 3) = \frac{3n}{10} - 9.$$

members of a on both its left and right sides. Note: correctly solving parts a and b for this problem counts for passing LO6.

a. Rewrite both the inequality and equality with updated constants, assuming that the algorithm now uses groups of 3 instead of groups of 5. (10 pts)

Solution.

$$2(\lfloor \frac{1}{2} \lceil \frac{n}{3} \rceil \rfloor - 2) \geq 2(\frac{1}{2} \cdot \frac{n}{3} - 3) = \frac{n}{3} - 6.$$

b. Assuming groups of 3 instead of groups of 5, provide a new worst-case divide-and-conquer recurrence for the running time $T(n)$ of the `find_statistic` algorithm, and show that $T(n)$ is no longer worst-case linear. (15 pts)

Solution. The worst-case recurrence is now $T(n) = T(n/3) + T(2n/3 + 6) \geq T(n/3) + T(2n/3) = \Omega(n \log n)$ (verify using the Substitution method).

3. Answer the following. Note: correctly solving this problem counts for passing LO7.

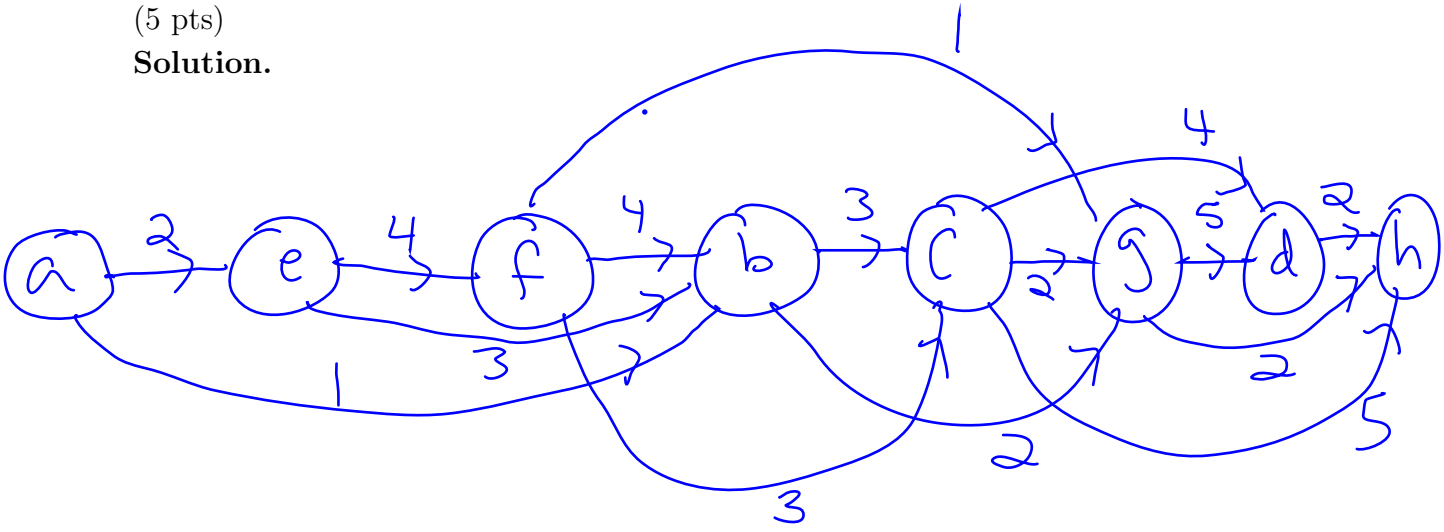
   a. Provide the dynamic-programming recurrence for computing $\mathrm{mc}(u, v)$ the cost of the maximum-cost path from vertex u to vertex v in a directed acyclic graph (DAG) $G = (V, E, c)$, where $c(e)$ gives the cost of edge e, for each $e \in E$. (10 pts)
   Solution.

$$\mathrm{mc}(u, v) = \begin{cases} 0 & \text{if } u = v \\ \infty & \text{if } u \neq v \text{ and } \deg^-(u) = 0 \\ \max_{(u,w)\in E}(c(u, w) + \mathrm{mc}(w, v)) & \text{otherwise} \end{cases}$$

   b. Draw the vertices of the following DAG G in a linear left-to-right manner so that the vertices are topologically sorted, meaning, if (u, v) is an edge of G, then u appears to the left of v. The vertices of G are a-h, while the weighted edges of G are

$$(a, b, 1), (a, e, 2), (a, f, 3), (b, c, 3), (b, g, 2), (c, d, 4), (c, g, 2), (c, h, 5), (d, h, 2), (e, b, 3), (e, f, 4),$$
$$(f, b, 4), (f, c, 3), (f, g, 1), (g, d, 5), (g, h, 2).$$

   (5 pts)
   Solution.



   c. Starting with $u = h$, and working backwards (from right to left), use the recurrence from part a to compute each of $\mathrm{mc}(u, h)$, where the ultimate goal is to compute $\mathrm{mc}(a, h)$. (10 pts)

      i. $\mathrm{mc}(h, h) = 0$
      ii. $\mathrm{mc}(d, h) = 2 + \mathrm{mc}(h, h) = 2$;
      iii. $\mathrm{mc}(g, h) = \max(5 + \mathrm{mc}(d, h), 2 + \mathrm{mc}(h, h)) = \max(5 + 2, 2) = 7$
      iv. $\mathrm{mc}(c, h) = \max(4 + \mathrm{mc}(d, h), 2 + \mathrm{mc}(g, h), 5 + \mathrm{mc}(h, h)) = 9$.
      v. $\mathrm{mc}(b, h) = \max(3 + \mathrm{mc}(c, h), 2 + \mathrm{mc}(g, h)) = 12$.
      vi. $\mathrm{mc}(f, h) = \max(1 + \mathrm{mc}(g, h), 4 + \mathrm{mc}(b, h), 3 + \mathrm{mc}(c, h)) = 16$.
      vii. $\mathrm{mc}(e, h) = \max(3 + \mathrm{mc}(b, h), 4 + \mathrm{mc}(f, h)) = 20$.
      viii. $\mathrm{mc}(a, h) = \max(2 + \mathrm{mc}(e, h), 1 + \mathrm{mc}(b, h)) = 22$.

4. Given an array of n strictly increasing integers, describe an optimal (with respect to big-O running-time) algorithm for determining whether or not a has a **fixed point**, meaning the existence of an index $i \in \{0, \ldots, n-1\}$ for which $a[i] = i$. Please do *not* use pseudocode. Rather, number each step of your algorithm and clearly explain each step. Before beginning the first step clearly define all variables used by the algorithm and indicate their initial values. Provide the running time of your algorithm and justify your answer. Suboptimal and/or incompletely/vaguely written algorithms will be awarded 0 points. (25 pts)

   Solution.

   a. Initialize variables $l = 0$ and $u = n - 1$.

   b. If a has a size of of 3 or less, then check, one by one, if some array member is a fixed point. If yes, then return 1. Otherwise, return false.

   c. Set variable $m = (l + u)/2$, and consider $a[m]$. If $a[m] = m$, then return 1. Else if $a[m] > m$, then a can only have a fixed point if it occurs at index $i < m$. In this case assign $r = m - 1$ and go to Step 2. Finally, if $a[m] < m$, a can only have a fixed point if it occurs at index $i > m$. In this case assign $l = m + 1$ and go to Step 2.

   Running time is $T(n) = \mathrm{O}(\log n)$ since $T(n)$ satisfies $T(n) \leq T(n/2) + 1$.

5. Consider the following algorithm that finds multiple statistics for the input array a, where the different k values are provided in a set K.

Name: `find_statistics`

Inputs:

Array a of n integers.

Set $K = \{k_1, \ldots, k_j\}$ of statistic indices, where each $\texttt{lower} \leq k_i \leq \texttt{upper}$.

Indices `lower` and `upper`, where $0 \leq \texttt{lower} \leq \texttt{upper} < n$.

Output: subset $S \subseteq a$ consisting of the j desired statistics.

If $K = \emptyset$, then Return $\emptyset$.

If `lower` = `upper`, then Return $\{a[\texttt{lower}]\}$.

If $K = \{k\}$, then Return $\{\texttt{find\_statistic}(a, k, \texttt{lower}, \texttt{upper})\}$. //Use `find_statistic` from lecture

`index` = `index_of_median(a,lower,upper)`. //This takes linear time

`median` = $a[\texttt{index}]$.

$S = \emptyset$. //Initialize the set of statistics to return.

If `index` $\in K$, then $S = S \cup \{\texttt{median}\}$. //The median is one of the sought statistics

$K_{\text{left}} = K \cap \{\text{lower}, \ldots, \text{index} - 1\}$.

$S = S \cup \texttt{find\_statistics}(a, K_{\text{left}}, \text{lower}, \text{index-1})$.

$K_{\text{right}} = K \cap \{\text{index} + 1, \ldots, \text{upper}\}$.

$S = S \cup \texttt{find\_statistics}(a, K_{\text{right}}, \text{index} + 1, \text{upper})$.

Return S.

Suppose for some large n we run `find_statistics` on inputs $K = \{\frac{n}{\lfloor \log n \rfloor}, \frac{2n}{\lfloor \log n \rfloor}, \ldots, \frac{(\lfloor \log n \rfloor - 1)n}{\lfloor \log n \rfloor}\}$ with lower $= 0$ and upper $= n - 1$. Determine the big-Θ running time on this input. Defend your answer. Hint: at what level of the recursion does the algorithm revert to the orginal `find_statistic` algorithm from lecture?

Solution.

The algorithm's recurrence obeys $T(n) = 2T(n/2) + n$ until the array size becomes small enough to where there is at most one statistic index in the array. Since each statistic index is a distance of $n/\log n$ from either of its neighbors, the above recurrence becomes invalid at depth d where $n/2^d < n/\log n$ which implies $d > \log(\log n)$. Thus the total work of the algorithm in the first $d-1$ levels of the recursion tree equals $\Theta(n \log(\log n))$. In addition, the original `find_statistic` algorithm must be run $\log n - 1$ times on inputs of size $n/\log n$ for an additional running time of $O(n)$. Therefore, $T(n) = \Theta(n \log(\log n))$. (25 pts)

6. The `Subset Sum` decision problem consists of a sequence of integers $S = x_1, \ldots, x_n$ and target integer t. The problem is to decide if there is a (possibly non-contiguous) subsequence A of S whose members sum to t. For example, `Subset Sum` instance $S = 3, 7, 13, 19, 22, 26, 35, 38, 41$ and $t = 102$ is a positive instance since $A = 3, 7, 13, 38, 41$ is a subsequence of S and

$$3 + 7 + 13 + 38 + 41 = 102.$$

   a. Provide a dynamic-programming recurrence for solving the `Subset Sum` problem. Do this by defining the predicate function (i.e. one that returns either 0 or 1) $\texttt{has\_sum}(i, c)$, for appropriate variables i and c which you should define. (15 pts)

   b. Apply your recurrence to $S = 2, 4, 5, 6, 9$ and $t = 10$. Provide the matrix of subproblem solutions. And circle the numbers that are used to make up the sum. (10 pts)

LO1. Which function grows faster $2^{\sqrt{n}}$ or $(\log n)^{\sqrt[3]{n}}$? Defend your answer.

LO2. Recall the use of the disjoint-set data structure for the purpose of improving the running time of the UTS algorithm. For the set of tasks

Task	a	b	c	d	e	f
Deadline	1	2	2	3	3	0
Profit	60	50	40	30	20	10

For each task τ, show the M-Tree forest after τ has been inserted (or at least has attempted to be inserted in case the scheduling array is full). Notice that the earliest deadline is 0, meaning that the earliest slot in the schedule array has index 0. Hint: to receive credit, your solution should show six different snapshots of the M-Tree forest.

LO3. Answer the following with regards to a correctness-proof outline for Dijkstra's algorithm. Note: correctly solvng this problem counts for passing LO3.

(a) Complete the following sentence. "In relation to Dijkstra's algorithm, an **i-neighboring path** from source s to a vertex v that is external to DDT_i is ..."

(b) Complete the following sentence. "Furthermore, the **i-neighboring distance** $d_i(s, v)$ from source s to v is *defined as* ..."

(c) Complete the following sentence. "The greedy choice made by Dijkstra's algorithm at round $i + 1$ is to select the external vertex v^* which has ..."

(d) Complete the following sentence. "If P is any path from s to v^* (from part c), $\text{cost}(P) \geq d_i(s, v^*)$ since any path P has an i-neighboring subpath, and ..."

LO4. Use the substitution method to prove that, if $T(n) = T(n/3) + T(2n/3) + n$, then $T(n) = \Omega(n \log n)$. Remember to clearly state the inductive assumption.