University of Washington — January 24, 2024
Department of Computer Science and Engineering
CSE 421, Winter 2024

Homework 4, Due Wednesday, January 31, 11:59 pm, 2024

Turnin instructions: Electronic submission on gradescope using the CSE 421 gradescope site. Submit the assignment as a PDF, with separate pages for different numbered problems. Problems consisting of multiple parts (e.g., 2a, 2b) can be submitted on the same page.

Problem 1 (10 points):

(From the text book, page 192, problem 8) Suppose you are given a connected graph G, with edge costs that are all distinct. Prove that G has a unique minumum spanning tree.

Problem 2 (10 points):

(From the text book, page 193, problem 11) Suppose you are given a connected graph $G = (V, E)$, with cost c_e on each edge e. In the previous problem, you proved that if all edges have distinct costs, the minimum spanning tree is unique. However G may have many minimum spanning trees when the edge costs are not all distinct. Can Kruskal's Algorithm be made to find any particular minimum spanning tree of G?

Kruskal's Algorithm sorted the edges in order of increasing cost, then greedily processed edges one by one, adding an edge e as long as it did not form a cycle. When some edges have the same cost, the phrase "in order of increasing cost" has be specified more carefully: we will say that an ordering is *valid* if the corresponding sequence of the edge costs is nondecreasing. We will say that a *valid execution* of Kruskal' Algorithm is one that begins with a valid ordering of the edges of G.

For any graph G, and any minimum spanning tree T of G, is there a valid execution of Kruskal's Algorithm on G that produces T as output? Give a proof or counter example.

Problem 3 (10 points):

Solve the following recurrences by unrolling the recurrence. Do not apply the Master Theorem:

a) $T(n) = 4T(n/3) + n^{3/2}$ for $n \geq 2$; $T(1) = 1$;

b) $T(n) = T(3n/4) + n$ for $n \geq 2$; $T(1) = 1$;

c) $T(n) = 16T(n/4) + n^2$ for $n \geq 2$; $T(1) = 1$;

d) $T(n) = 7T(n/3) + n^2$ for $n \geq 2$; $T(1) = 1$;

Problem 4 (10 points):

Solve the following recurrences:

a)

$$T(n) = \begin{cases} T(\frac{n}{2}) * T(\frac{n}{2}) & \\ 2 & \text{if } n \leq 1 \end{cases}$$

b)

$$T(n) = \begin{cases} T(n-1) * T(n-1) & \\ 2 & \text{if } n \leq 1 \end{cases}$$

Problem 5 (10 points):

Given an array of elements $A[1, ..., n]$, give an $O(n \log n)$ time divide and conquer algorithm to find all of the *thirdary* elements, where a *thirdary* element is an element that is stored in more than $n/3$ locations. The array can have 0, 1, or 2 *thirdary* elements. For this problem, you can only test if two elements are the same by using an *Equivalent(x,y)* method, which returns true if the elements are the same, and false if they are different. You do not have access to a method that will order the elements or hash the elements (since that would make the problem too easy). Give an explanation why your algorithm correctly find the *thirdary* elements. Use divide and conquer for this problem, and justify why your algorithm finds a correct solution.