Midterm Exam Solution CS 600 Fall 2018

100 Points

Answer the following questions in this document or a word document and email it in Moodle to me and Ruiyang Sun.

1. (5 Points) Using the very definition of Big-omega notation, prove that n3 logn is Ω(n3). You must use the definition and finding the constants in the definition to receive credit.
   1. Therefore for n0 = 2 and c = 1 This holds and is therefore true
2. (5 Points) Given that T(n) = 1 if n=0 and T(n) = T(n-1)+ 2n otherwise; show, by induction, that T(n) = 2n+1 -1. Show all three steps of your induction explicitly.
3. (6 Points) Given T(n) = T(n-1) + 1/n, show that The Master Theorem is not applicable to this recurrence equation, hence show that T(n)=O(logn) using algebraic substitution given T(1) =1
   1. Masters Theorem: T(n) above clearly doesn’t fit the equation for the master’s theorem.  
      Using substitution,  
      I believe this is a harmonic series and therefore converges at logn
4. (14 Points) Design an O(n lg n)-time algorithm that, given an array A of n integers and another integer x, determines whether or not there exist two (not necessarily distinct) elements in A whose sum is exactly x.
   1. Sort the array in increasing order, this will be an O(nlogn) operation. Then create a pointer at the beginning and a pointer at the end of the array. Create a loop that checks the summation of the values at the two pointers. If the summation is less than x increment the pointer at the beginning of the list, if the summation is greater than x decrement the pointer towards the end of the list. Once the check equals x return the two pointers. The comparison would be an O(n) operation making the whole operation O(nlogn).
5. (14 Points) Recall the Extendable Array Implementation from Sections 1.4.1 and 1.4.2. Now cconsider an extendable table that supports both add and remove methods. Suppose we grow the underlying array implementing the table by doubling its capacity any time we need to increase the size of this array, and we shrink the underlying array by half any time the number of (actual) elements in the table dips below N/4, where N is the current capacity of the array. Use amortization method with cyber dollars to show that a sequence of n add and remove methods, starting from an array with capacity N = 1, takes O (n) time. You must use amortization Technique with cyber dollars to receive any credit.
   1. For adding, simply charge 3 cyber dollars: 1 to cover adding the item to the list and a surplus of 2 to cover doubling the list as it is.  
        
      For removing, the number of cyber dollars charged should depend on the length of the list. If the list length is less than or equal to half capacity charge 3 cyber dollars: 1 to remove the item from the list and 2 surplus dollars to shrink the list by two places. If the list length is greater than half capacity but less the ¾ capacity charge 2 cyber dollars: 1 to remove the item from the list and 1 to shrink the list by one space and using one of the surplus dollars from an add operation to shrink the list by one more space. If the list length is greater than ¾ charge one cyber dollar to remove the item from the list and use 2 of the surplus dollars from the add operation to shrink the list by two spaces. Checking the length has been shown to be an O(1) operation. Using this method will ensure that the list always has enough surplus

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| List Length |  | 1 Cyber $ \* # of Operations | | |  |  |
| Current Surplus | Remove | Shrink | Surplus | Number of Operations | $/Operation |
| 8 | 0 | 4 | 8 | 8 | 4 | 5 |
| 9 | 2 | 5 | 8 | 8 | 5 | 4.2 |
| 12 | 8 | 8 | 8 | 8 | 8 | 3 |
| 16 | 16 | 12 | 8 | 8 | 12 | 2.33333 |

1. (14 Points) Develop an algorithm that computes the kth smallest element of a set of n distinct integers in O (n + k log n) time.
   1. Using a priority queue implemented by a min-heap, simply insert all the elements of the set into the queue (O(n)), and then perform k removeMin operations (O(klogn)). O(n+klogn).
2. (14 Points). Suppose a social network, N, contains n people, m edges, and c connected components. What is the exact number of times that each of the methods, makeSet, union, and find, are called in computing the connected components for N using Algorithm 7.2?
   1. makeSet is called n times. Union is called logn times. Find is called 2m+n times.
3. (14 Points) Consider a single machine scheduling problem, where we are given a set, T, of tasks specified by their start times and finish times, as in the task scheduling problem, except now we have only one machine and we wish to maximize the number of tasks that this single machine performs. Design a greedy algorithm for this single machine scheduling problem.
   1. Order the tasks by their running time (start time – end time) and place priority on executing the smaller run times. Calculating the run times runs in O(n) and sorting them is done in O(nlogn) time, therefore, the algorithm runs in O(nlogn) time.
4. (14 Points) Show that we can solve the telescope schedulingproblem in O(n) time even if the list of n observation requests is not given to us in sorted order, provided that start and finish times are given as integer indices in the range from 1 to n2 .
   1. Simply apply a radix sort on the finish times and apply the telescope scheduling method both would be O(n) time, making for a total of O(n) time.