Anthony Rusignuolo

CS 600WS – Advanced Algorithms

Reza Peyrovian

Homework 1

I pledge my honor that I have abided by the Stevens Honor System.

1. R-1.7 Order the following list of functions by the big-Oh notation. Group together (for example, by underlining) those functions that are big-Theta of one another. \* check for big theta
   1. 1. 1/n
      2. 2100
      3. loglogn
      4. log.5n
      5. log2n
      6. n.01
      7. n.5, 3n.5
      8. 2logn, 5n
      9. nlog4n, 6nlogn
      10. 2nlog2n
      11. 4n3/2
      12. 4logn
      13. n2logn
      14. n3
      15. 2n
      16. 4n
      17. 22^n
2. R-1.9 Bill has an algorithm, ﬁnd2D, to ﬁnd an element x in an n × n array A. The algorithm ﬁnd2D iterates over the rows of A and calls the algorithm arrayFind, of Algorithm 1.12, on each one, until x is found or it has searched all rows of A. What is the worst-case running time of ﬁnd2D in terms of n? Is this a linear-time algorithm? Why or why not?
   1. The worst-case running time is n2 and, therefore, it is not a linear time algorithm.
3. R-1.22 Show that n is o(nlog n).
   1. f(n) = n  
      Let c > 0 be any constant  
      if n0 = 1/c, then 1 <= cn for n > n0Therefore, if n >= n0, f(n) = n <= nlogn <= cnlogn
4. R-1.23 Show that n2 is ω(n).
   1. f(n) = n2Let c > 0 be  
      if n0 = c, then for n > n0, n > c  
      Therefore, if n >= n0, f(n) = n2 >= cn
5. R-1.24 Show that n3logn is Ω(n3).
   1. n3logn >= n3 for n >= 2
6. R-1.32 Suppose we have a set of n balls and we choose each one independently with probability 1/n1/2 to go into a basket. Derive an upper bound on the probability that there are more than 3n1/2 balls in the basket.
7. C-1.4 What is the total running time of counting from 1 to n in binary if the time needed to add 1 to the current number i is proportional to the number of bits in the binary expansion of i that must change in going from i to i +1?
   1. Not sure, but after using excel, the equation appears to be > 2n, therefore it is O(n)
8. C-1.7 Consider the following recurrence equation, deﬁning a function T(n):  
   Show, by induction, that T(n) = 2n
9. C-1.22 Show that the summation is O(n). You may assume that n is a power of 2.  
   ***Hint***: Use induction to reduce the problem to that for n/2.
   1. Therefore O(n)
10. C-1.30 Consider an implementation of the extendable table, but instead of copying the elements of the table into an array of double the size (that is, from N to 2N) when its capacity is reached, we copy the elements into an array with additional cells, going from capacity N to N + . Show that performing a sequence of n add operations (that is, insertions at the end) runs in Θ(n3/2) time in this case.
    1. Copying the original table of N items takes n operations, and extending it by takes operations  
       Therefore O(n3/2)
11. A-1.8 Given an array, A, describe an efﬁcient algorithm for reversing A. For example, if A =[3, 4, 1, 5], then its reversal is A =[5, 1, 4, 3]. You can only use O(1) memory in addition to that used by A itself. What is the running time of your algorithm?
    1. Have a For loop run from i = 0 to A.length/2. Create a temp variable to hold the value of A[i]. Set the value of A[i] to A[A.length - 1 – i]. Set the value of A[A.length - 1 – i] to the temp variable. The loop terminates and the array is reversed. O(1) memory, O(n) time
12. A-1.15 Given an integer k>0 and an array, A, of n bits, describe an efﬁcient algorithm for ﬁnding the shortest subarray of A that contains k 1’s. What is the running time of your method?
    1. Set a temp variable equal to 0. Loop that iterates from 0 to the length of the array. Checks for temp == k, if so, break. Check for A[i] = 1, if so increment temp, else temp = 0. Return [i-k, i-1]. Run time is O(n).