Computer Science Foundation Exam

May 21, 2022

Section C

ALGORITHM ANALYSIS

SOLUTION

Question #	Max Pts	Category	Score
1	10	ANL	
2	5	ANL	
3	10	ANL	
TOTAL	25		

You must do all 3 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and <u>not</u> graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all <u>be neat</u>. For each coding question, assume that all of the necessary includes (stdlib.h, stdio.h, math.h, string.h) for that particular question have been made.

1) (10 pts) ANL (Algorithm Analysis)

What is the worst case Big-Oh runtime for the function **f,** in terms of its input parameter n? You may assume that the array pointed to by arr is of length n. (Grading note: 2 pts will be awarded for the answer, 8 pts for the proof of the answer. Your proof must include either summations or recurrence relations related to the code below.)

```
int f(int* arr, int n, int minVal) {
    return fHelp(arr, 0, n-1, minVal);
}

int fHelp(int* arr, int low, int high, int minVal) {
    if (low > high) return 0;
    if (low == high) return arr[low] >= minVal;

    int mid = (low+high)/2;
    int left = fHelp(arr, low, mid, minVal);
    int right = fHelp(arr, mid+1, high, minVal);
    int res = left;
    if (right > left)
        res = right;

    int alt = 0, i;
    for (i=mid; i>=low; i--) {
        if (arr[i] < minVal) break;
        alt++;
    }
    for (i=mid+1; i<=high; i++) {
        if (arr[i] < minVal) break;
        alt++;
    }

    if (alt > res) res = alt;
    return res;
}
```

There are two recursive calls in fHelp, both to arrays of half the size of the original array. Let T(n) be the run time of function f. Effectively, T(n) breaks down into two function calls, each of which take time T(n/2), plus the work after the recursive calls. There are two loops, each which run n/2 times at most, so in total the loops run n times with constant time operations inside the loops. Thus, the total amount of work beyond the recursive calls is O(n). It follows that T(n) satisfies the following recurrence relation:

$T(n) = 2T(n/2) + O(n)$.

We can solve this recurrence relation via the Master Theorem, getting a solution of **O(nlgn).** ($A = 2$, $B = 2$ and $k = 1$. Since $B^k = 2$ and $A = 2$, the solution follows.)

Grading: 2 pts for correct answer (give pts even if no work)
2 pts for ANY recurrence relation
2 pts (additional) if recurrence has T(n/2) on RHS
2 pt (additional) if term is 2T(n/2)
2 pt for O(n) added extra work