CS 5006 Final

Northeastern University— Seattle

Summer 2018

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

Name: ______________________________

1. Consider a recursive divide and conquer algorithm that satisfies the following recurrence on its running time: $T(n) = 4T(n/3) + n$, with $T(1) = 1$. In the following you may assume that n is a power of 3.

   (a) How many subproblems are there at depth k in the recursion tree? (The number of subproblems at depth 0, namely at the root of the tree, is 1.) (4)

   (b) What is the size of each subproblem at depth k of the recursion tree? (The size of the subproblem at depth 0, namely at the root of the tree, is n). (4)

   (c) What is the running time $T(n)$ of this algorithm? (4)

2. Solve the following recurrences. Show your work.

(a) (4)

$$T(n) = \begin{cases} 3T(n/3) + n^2 & \text{if } n > 0 \\ 1 & \text{if } n = 0 \end{cases}$$

(b) (4)

$$T(n) = \begin{cases} 4T(n/3) + n & \text{if } n > 0 \\ 1 & \text{if } n = 0 \end{cases}$$

(c) (4)

$$T(n) = \begin{cases} T(n/3) + 1 & \text{if } n > 1 \\ 0 & \text{if } n = 0 \end{cases}$$

3. Answer Yes or No to the following questions. Show your work for partial credit.

   (a) A divide and conquer algorithm for $n \times n$ matrix multiplication that solves the problem using 24 matrix multiplication subproblems of size $n/3$ and $O(n^2)$ additional work for recombining would be more efficient than the usual $\Theta(n^3)$matrix multiplication algorithm. (5)

   (b) A divide and conquer algorithm for the selection problem that reduced the problem on lists of size n to one selection problem of size $3n/20$ and another selection problem of size $9n/10$ plus a linear amount of extra work would yield a linear time algorithm. (5)

4. In World War I, the army was testing recruits for syphilis, which was rare, but required a time-consuming though accurate blood test. They realized that they could pool the blood from several recruits at once and save time by eliminating large groups of recruits who didn't have syphilis.

   But, today, you are given a sequence of n bits $x_1, ..., x_n$, where each $x_i \in \{0, 1\}$. I want you to design an algorithm that outputs either

   - Any i such that $x_i = 1$
   - 0 if the input is all 0s.

   The only operation you are allowed to use to access the inputs is a function GROUP-TEST where:

$$\text{Group-Test}(i, j) = \begin{cases} 1 & \text{if some bit in } x_i, \ldots, x_j \text{ has value } 1 \\ 0 & \text{if all bits in } x_i, \ldots, x_j \text{ have value } 0 \end{cases}$$

   (a) Design a divide and conquer algorithm to solve the problem that uses only $O(\log n)$ calls to GROUP-TEST in the worst case. Your algorithm should never access the x_i directly. (10)

(b) Show that your algorithm above produces the correct result. (10)

(c) Briefly justify your bound on the number of calls. (5)

5. You are given a directed acyclic graph $G = (V, E)$ and a node $t \in V$. Design a linear time algorithm to compute for each vertex $v \in V$ the number of different paths from v to t in G. Analyze its running time in terms of $n = |V|$ and $m = |E|$. (25)

   Be sure to address correctness and runtime.

   HINT: Start with a topological sort.

6. The two processor interval scheduling problem takes as input a sequence of request intervals $(s_1, f_1), ..., (s_n, f_n)$ **(10)** just like the unweighted interval scheduling problem except that it produces two disjoint subsets $A_1, A_2 \subset [n]$ such that all requests in A_1 are compatible with each other and all requests in A_2 are compatible with each other and $|A_1 \cup A_2|$ is as large as possible.

   (A_1 might contain requests that are incompatible with requests in A_2.) Does the following greedy algorithm produce optimal results? If yes argue why it does; if no produce a counterexample.

```
DoThing($\{(s_1, f_1), \ldots, (s_n, f_n)\}$)
Sort requests by increasing finish time
$A_1 = \emptyset$
$A_2 = \emptyset$
while there is any request $(s_i, f_i)$ compatible with either $A_1$ or $A_2$:
      Add the first unused request, if any, compatible with $A_1$ to $A_1$.
      Add the first unused request, if any, compatible with $A_2$ to $A_2$.
```

7. Given sorted array of n distinct integers, arranged in increasing order $A[1, n]$, you want to find out whether there is an index i for which $A[i] = i$. Give an algorithm that runs in time $O(\log n)$ for this problem. (15)

Question	Points	Score
1	12	
2	12	
3	10	
4	25	
5	25	
6	10	
7	15	
Total:	109	

Scratch Paper

Scratch Paper