CS 5633 Analysis of Algorithms – Fall 15

Exam 1

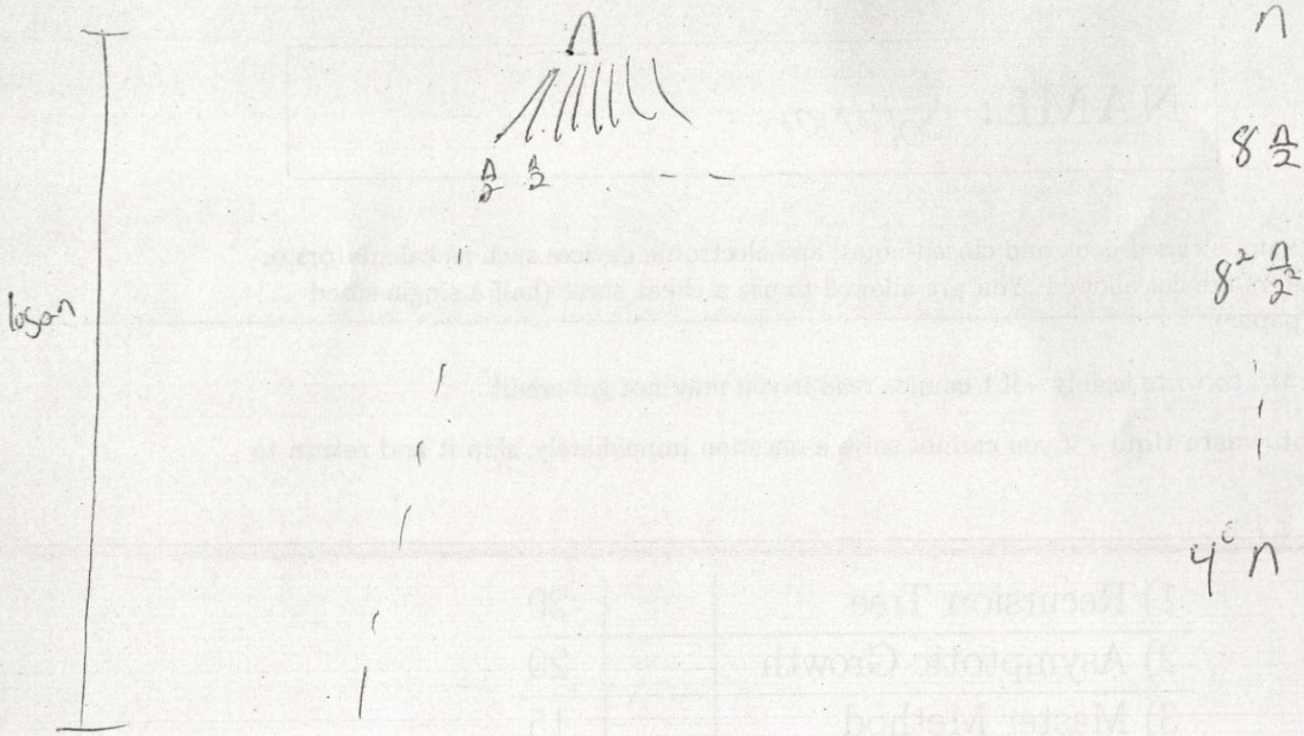
NAME: Solution

- This exam is closed-book and closed-notes, and electronic devices such as calculators or computers are not allowed. You are allowed to use a cheat sheet (half a single-sided letter paper).
- Please try to write legibly – if I cannot read it you may not get credit.
- **Do not waste time** – if you cannot solve a question immediately, skip it and return to it later.

1) Recursion Tree		20
2) Asymptotic Growth		20
3) Master Method		15
4) Randomized Analysis		15
5) Divide and Conquer		30
		100

1 Recursion Tree (20 Points)

Let $T(n) = 8T(n/2) + n$ with $T(1) = 1$. Use a recursion tree to generate a guess of what $T(n)$ solves to.



$$\sum_{i=0}^{\log n} 4^i n = n \sum_{i=0}^{\log n} 4^i = n\left[\frac{4^{\log n + 1} - 1}{4 - 1}\right]$$

$$= n\left[\frac{(2^2)^{\log n + 1} - 1}{3}\right]$$

$$= n\left[\frac{(2^{\log n + 1})^2 - 1}{3}\right]$$

$$= n\left[\frac{(2n)^2 - 1}{3}\right]$$

$$\approx O(n^3)$$

2 Asymptotic Growth (20 Points)

6

1. Show that $3n^3 + 3n^2 - 6n - 6 \in \Theta(n^3)$.

$3n^3 + 3n^2 - 6n - 6 < 3n^3 + 3n^2 < 3n^3 + 3n^3 = 6n^3$, $c = 6$, $n_0 = 1$

$n^3 < n^3 + (n^3 - 6n - 6) < 3n^3 + 3n^2 - 6n - 6$, $c = 1$, $n_0 = 3$

$n = 3$

14

2. Let $T(n) = 4T(n/2) + n^2$ with $T(1) = 1$. Use big-Oh induction to prove that $T(n) \in O(n^2 \log n)$. Prove the inductive step (substitution method) as well as the base case.

Assume $T(k) \leq c \cdot k^2 \log k$ for all $k < n$.

$T(n) = 4T(n/2) + n^2 \leq 4c \left(\frac{n}{2}\right)^2 \log\left(\frac{n}{2}\right) + n^2$

$= 4c \frac{n^2}{4} (\log n - \log 2) + n^2$

$= cn^2 \log n - cn^2 + n^2$

$\leq cn^2 \log n$ when $c \geq 1$.

Base case: $n = 2$

$T(2) = 4T(1) + 4 = 4 + 4$,

$c 2^2 \log 2 = 4c$

$4 + 4 < 4c$ if $c > 2$.

3 Master Method (15 Points)

Solve the following recurrences using the master theorem. Justify your answers shortly (i.e. specify ϵ and check the regularity condition if necessary).

1. $T(n) = 27T(n/3) + n^3$

$a=27 \quad b=3 \quad n^{\log_b a} = n^3$

$f(n) \in \Theta(n^3)$

Case 2 applies

$\Theta(n^3 \log n)$

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

$a=1 \quad b=2 \quad n^{\log_b a} = n^0 = 1$

$f(n) = n \quad n \in \Omega(n^{0+\varepsilon})$ for $\varepsilon = 1$.

$a \cdot f(\frac{n}{b}) = \frac{n}{2} = c \cdot n$ for $c = \frac{1}{2}$.

Case 3 applies. $\Theta(n)$

3. $T(n) = 4T(n/2) + n \log n$

$a=4 \quad b=2. \quad n^{\log_b a} = n^2$

$n \log n \in O(n^{2-\varepsilon})$ for $\varepsilon = 0.5$

Case 1

$\Theta(n^2)$

4 Randomized Analysis (15 Points)

Suppose someone offers to let you play a game. They will randomly (and independently) pick three numbers x_1, x_2, x_3 in the range 1-30. If x_1 is odd, you will be paid \$5 and if it is even you will pay \$2. If x_2 is in the range 1-10 you will be paid \$3, and if it is any other value you will pay \$1. If $x_3 = 1$ then you must pay \$20 (and will be paid nothing nothing otherwise). What is the expected value of this game from your perspective?

X_1 = outcome for x_1. X_1 will be 5 or -2

X_2 = outcome for x_2 X_2 will be 3 or -1.

X_3 = outcome for x_3. X_3 will be 0 or -20,

$$E[X_1] = Pr(X_1 = 5) \cdot 5 + Pr(X_1 = -2) \cdot -2$$

$$= \frac{5}{2} + -\frac{2}{2} = \frac{3}{2}$$

$$E[X_2] = Pr(X_2 = 3) \cdot 3 + Pr(X_2 = -1) \cdot -1$$

$$= \frac{1}{3} \cdot 3 + \frac{2}{3} \cdot -1 = \frac{1}{3}$$

$$E[X_3] = Pr(X_3 = -20) \cdot -20$$

$$= \frac{-20}{30} = -\frac{2}{3}$$

$$\frac{3}{2} + \frac{1}{3} - \frac{2}{3} = \frac{9}{6} + \frac{2}{6} - \frac{4}{6} = \frac{7}{6}$$

5 Divide and Conquer (30 Points)

Let $A[1..n]$ be an array of $n \geq 2$ integers which consist of the prices of some stock over the previous n days. Suppose we were allowed to purchase the stock on some day p and then later sell the stock on some day s (note here we must have $s > p$ as we can only sell after we have purchased). We want to determine the the buy and sell days that maximize our profit $A[s] - A[p]$.
For example, if $A = [8, 10, 3, 7, 1]$, then the optimal solution is $p = 3$ and $s = 4$ as $A[s] - A[p] = 4$ and any other option will result in a smaller profit. This problem can easily be solved in $O(n^2)$ time by checking all possibilities with two nested for-loops. This problem is about designing a divide and conquer algorithm for this problem with running time $o(n^2)$.

1. Suppose we design a divide and conquer algorithm with running time $T(n) = 2T(n/2) + f(n)$ by dividing the problem in half, recursively solve both halves, and then spend $f(n)$ time "combining". Since we are trying for an overall running time of $o(n^2)$, what must be true of $f(n)$ (i.e., how much time are we allowed to use on this combine step)?

$n^{\log_2 2} = n^1$. $f(n) \in o(n^2)$ then $T(n) \in o(n^2)$.

2. Suppose A is an array of size n where n is an even number, and let B denote the first $n/2$ elements of A and let C denote the last $n/2$ elements of A. Suppose we know an optimal solution for B and C (we know the correct days to purchase and sell if both days must be made in their respective halves). Note that the optimal solution for A could be one of the optimal solutions for one of the halves. If it isn't, then what can we say about the structure of the optimal solution for A? That is, what can we say about p and s?

Then p is in B and s is in C.

Moreover, p is index of smallest element in B
and s is index of largest in C.

3. Given the optimal solutions for B and C, give a $O(n)$ time algorithm that computes the optimal solution for A.

Let p_1, s_1 be optimal for B and p_2, s_2 be optimal for C.

Then let p_3 be the index of min element in B, and let s_3 be index of maximum element in C.

return $\max\{A[s_1] - A[p_1],\ A[s_2] - A[p_2],\ A[s_3] - A[p_3]\}$.