CS 5633 Analysis of Algorithms – Fall 16

Exam 1

NAME:

- 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) Big-Oh Notation		10
2) Recursion Tree		15
3) Induction		20
4) Master Method		15
5) Randomized Analysis		15
6) Divide and Conquer		25
		100

1 Big-Oh Notation (10 Points)

$\Theta(n^2)$. Sorry for typo.

Use the definition of Big-Oh and Omega to show that $4n^2 - 8n + 6 \in \cancel{\Theta(n^3)}$.

O: $4n^2 - 8n + 6 \leq 4n^2 + 6 \leq 4n^2 + 6n^2 = 10n^2$.

$f(n) \leq c \cdot g(n)$ for $c = 10$ for all $n \geq n_0 = 1$.

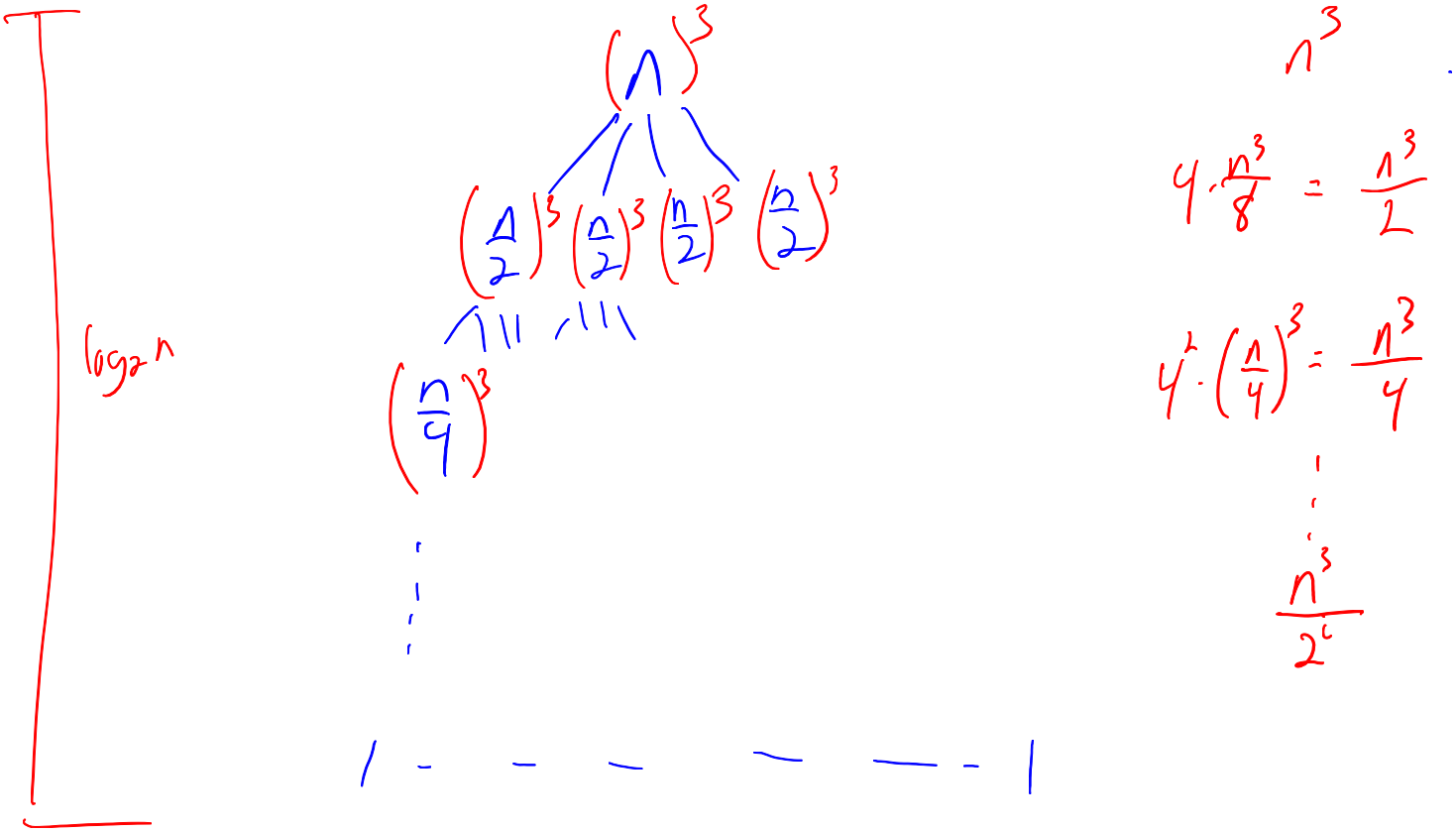
Ω: $4n^2 - 8n + 6 \geq 4n^2 - 8n = n^2 + (3n^2 - 8n) \geq n^2$

when $3n^2 - 8n \geq 0 \Rightarrow n \geq \frac{8}{3}$

$f(n) \geq c \cdot g(n)$ for $c = 1$ for all $n \geq n_0 = 3$.

2 Recursion Tree (15 Points)

Use a recursion tree to generate a guess of the asymptotic complexity of a divide and conquer algorithm with the running time $T(n) = 4T(n/2) + O(n^3)$.



$$\sum_{i=0}^{\log n} \frac{n^3}{2^i} = n^3 \sum_{i=0}^{\log n} \left(\frac{1}{2}\right)^i < n^3 \underbrace{\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i}_{<2} < 2n^3.$$

3 Induction (20 Points)

Let $T(n) = 3T(n/3) + dn$ with $T(1) = a$ for constants d and a. Use induction to prove that $T(n) \in O(n \log_3 n)$.

Base Case: $n = 3$

$$T(3) = 3T(1) + 3d = 3a + 3d = 3(a+d)$$

Inductive Step:

Assume $T(k) \leq c \cdot k \log_3 k$ for all $k < n$.

$$T(n) = 3T\left(\frac{n}{3}\right) + dn \leq 3\left[c \frac{n}{3} \log_3 \frac{n}{3}\right] + dn$$

$$= cn\left[\log_3 n - \log_3 3\right] + dn$$

$$= cn \log_3 n - cn + dn$$

$$\leq cn \log_3 n$$

↑ true when $c \geq d$.

$\Rightarrow T(n) \leq c \cdot n \log_3 n$ for $c = a+d$ for all $n \geq 3$.

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

$a=2,\ b=2,\ f(n)=n^2,\ n^{\log_b a} = n^{\log_2 2} = n^1.$

$n^2 \in \Omega(n^{1+\varepsilon})$ for $\varepsilon = 1.$

$af(\frac{n}{b}) = 2(\frac{n}{2})^2 = \frac{n}{2} = cn$ for $c = \frac{1}{2}.$ Case 3 holds.

$T(n) \in \Theta(n^2).$

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

$a=27,\ b=3,\ f(n)=n^3,\ n^{\log_3 27} = n^3.$

Case 2 holds.

$\Rightarrow T(n) \in \Theta(n^3 \log n).$

3. $T(n) = 9T(n/3) + n \log n$

$a=9,\ b=3,\ f(n)=n\log n,\ n^{\log_3 9} = n^2.$

$n \log n \in O(n^{2-\varepsilon})$ for $\varepsilon = \frac{1}{2}.$ (Note any $\varepsilon \in (0,1)$ works. $\varepsilon = 1$ does not work).

Case 1 holds.

$T(n) \in \Theta(n^2).$

5 Randomized Analysis (15 Points)

Suppose someone offers to let you play a game. They will randomly (and independently) roll three dice: a red die, a green die, and a blue die. If you choose to play, you will pay $5 to enter the game, and you will try to guess the number that is rolled for each of the three dice (note you submit a guess for each color, e.g. red = 5, green = 2, and blue = 1). If you guess exactly one of the three correctly, then they will give you $1 back. If you guess exactly two out of three correctly, they will give you $5 back. If you guess all three correctly, they will give you $100 back. What is the expected value of this game from your perspective?

$X(S) = \{-4, 0, 95, -5\}$

miss 2, miss 1, miss 0, miss 3

$E[X] = P(X=-4)\cdot -4 + \underbrace{P(X=0)\cdot 0}_{=0} + P(X=95)\,95 + P(X=5)\cdot -5$

$P(X=95) = \frac{1}{6}\cdot\frac{1}{6}\cdot\frac{1}{6} = \left(\frac{1}{6}\right)^3$

$P(X=5) = \frac{5}{6}\cdot\frac{5}{6}\cdot\frac{5}{6} = \left(\frac{5}{6}\right)^3$

$P(X=-4) = \underbrace{\frac{1}{6}\cdot\frac{5}{6}\cdot\frac{5}{6}}_{\text{Prob of getting first 1 and missing last two}} \times \underbrace{3}_{\text{3 ways to pick the die you got correct}} = \frac{75}{6^3}$.

$E[X] = 95\left(\frac{1}{6}\right)^3 + -5\cdot\left(\frac{5}{6}\right)^3 + -4\cdot\left(\frac{75}{6^3}\right)$.

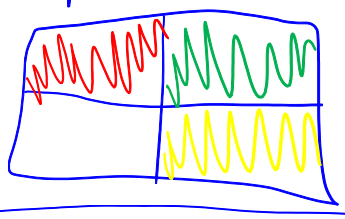
dont need to simplify on an exam. I only care that you know how to compute it.

6 Divide and Conquer (25 Points)

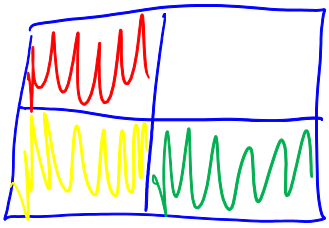
Suppose we have a two-dimensional array of (positive or negative) integers $A[][]$ such that each each row and column is monotonically increasing. That is, if you consider a row and scan from left to right, the numbers increase. Similarly, if you scan a column from bottom to top, the numbers increase. We are interested in finding the largest number in the array that is less than 0. Give a divide and conquer algorithm which runs in $o(n)$ time (i.e. strictly faster than linear time), where n is the total number of integers in A.

Idea: Check # in middle. If <0, then answer cannot be in lower left quadrant (all numbers here are less than middle value). Similarly, if it is ≥ 0 then answer cannot be in upper right (all numbers here are greater than middle).

If middle <0, we call 3 subproblems:



If middle ≥ 0, we call 3 subproblems:



```
RecSearch(A, rb, rt, cl, cr)  ← where rt & rb are the top & bottom rows of our subproblem & cl & cr are the left and right columns of our subproblem.
{
  if(rt-rb==1 OR cr-cl==1){
      binary search that column or row for 0 and then return the largest # < 0. If all are ≥0, return -∞.
  }
  midRow = l/2; midCol = n/2;
  middle = A[midRow][midCol];
  if(middle <0){
      upLeft = RecSearch(A, midRow+1, rt, cl, midCol);
      downRight = RecSearch(A, rb, midRow, midCol+1, cr);
      upRight = RecSearch(A, midRow, cl, midCol, cr);
      Return max(upLeft, downRight, upRight);
  }
  else{
      upLeft = RecSearch(A, midRow, rt, cl, midCol-1);
      downRight = RecSearch(A, rb, midRow-1, midCol, cr);
      bottomLeft = RecSearch(A, rb, midRow, cl, midCol);
      Return max(upLeft, downRight, bottomLeft);
  }
}
```

$T(n) = 3T(\frac{n}{4}) + O(1)$.

$\Rightarrow T(n) \in \Theta(n^{\log_4 3})$.