CS 5633 Analysis of Algorithms – Spring 23

Exam 1

NAME: Md. Sohanur Rahman

- 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 (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

(7)

1 Big-Oh Notation (10 Points)

Use the definition of Big-Oh and Omega (not the limit theorem) to show that $4n^2 - 2n + 1 \in \Theta(n^2)$.

If $f(n) \leq c \cdot g(n)$ for all $n > n_0$ and $c > 0$, then $f(n) \in O(g(n))$

$4n^2 - 2n + 1 \leq 4n^2$ ~~+ 1 + 2 + 2~~ ($-2n + 1 \leq 0$, $n_0 \geq \frac{1}{2}$)

for $c = 4$ and $n \geq \frac{1}{2}$

$\therefore f(n) \in O(n^2)$ ✓

If $f(n) \geq c \cdot g(n)$ for all $n > n_0$ and $c > 0$ then $f(n) \in \Omega(g(n))$

$4n^2 - 2n + 1 > 4n^2 - 2n \geq 4n^2$ ($-2n \geq 0$, $n \geq 0$)

(✗ negative)

where $c = 4$, $n \geq 0$

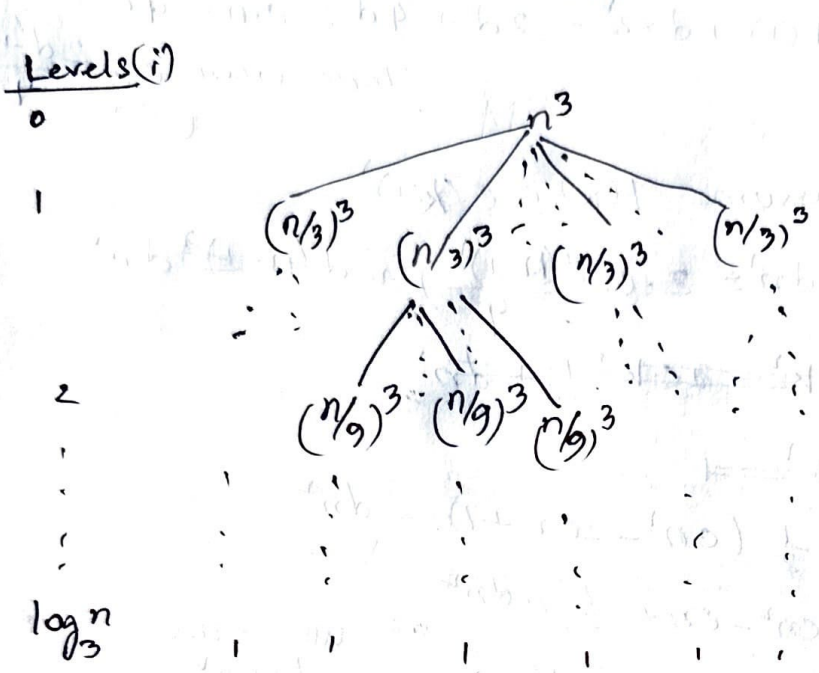
$\therefore f(n) \in \Omega(n^2)$

So $4n^2 - 2n + 1 \in \Theta(n^2)$

2 Recursion Trees (15 Points)

(13)

Use a recursion tree to generate a guess of the asymptotic complexity of a divide and conquer algorithm with the running time $T(n) = 9T(n/3) + n^3$.



Total cost $= \sum_{i=0}^{\log_3 n} 9^i \times \left(\frac{n}{3^i}\right)^3 = n^3 \sum_{i=0}^{\log_3 n} \frac{9^i}{3^{3i}}$

$= n^3 \sum_{i=0}^{\log_3 n} \frac{1}{3^i} = \sum_{i=0}^{\infty} \frac{1}{3^i}$ ✓

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

Then $T(n) \in \Theta(n^3)$

(18)

3 Induction (20 Points)

Let $T(n) = 2T(n/2) + dn^2$ when $n > 2$ and $T(n) = d'$ when $n \leq 3$ for constants d and d'. Use induction to prove that $T(n) \in O(n^2)$.

Base Case: $T(2) = 2T(1) + d \times 2^2 = 2d' + 4d$; $c \cdot n^2 = 4c$

True when $c > \frac{2d' + 4d}{4}$ ✓

Inductive Step: Lets assume $T(k) \leq c(k-1)^2$

$\therefore T(n) = 2T(n/2) + dn^2 \leq 2 \cdot c \cdot \left(\frac{(k-1)^2}{4}\right) + d(n+1)^2 dn^2$

~~$= \frac{1}{2}c \cdot k^2 = 2c \cdot k + 1/2 + dn^2$~~

~~$= ck^2 = +$~~

$= 2 \cdot \frac{1}{2}(cn^2 - 2cn + 1) + dn^2$

$= \frac{1}{2}cn^2 - cn + \frac{1}{2} + dn^2$

$\leq cn^2 - 2cn + 1 + \frac{1}{2}cn^2 + cn + \frac{1}{2} + dn^2$ all positive residuals

$\leq c(n-1)^2$

$\frac{1}{2}cn^2 + cn + \frac{1}{2} + dn^2 \leq 0$

$\frac{1}{2}cn^2 + cn \leq -\frac{1}{2} - dn^2$

$cn\left(\frac{1}{2}n + 1\right) \leq -\left(\frac{1}{2} + dn^2\right)$

$c \geq$

$\therefore T(n) \in O(n^2)$

4 Master Method (15 Points)

15

Solve the following recurrences using the master theorem (or if MT does not apply then say so). Justify your answers shortly (i.e. specify ϵ and check the regularity condition if necessary).

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

$a = 3,\ b = 3,\ n^{\log_b a} = n^{\log_3 3} = n$

$f(n) \in \Omega(n^{\log_b a + \epsilon}) \Rightarrow n^2 \in \Omega(n^{1+\epsilon})$ for $\epsilon = 1$ ✓

So. $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$

$\Rightarrow 3 \cdot \frac{n^2}{9} \leq c \cdot n^2$

$\Rightarrow \frac{1}{3} n^2 \leq c \cdot n^2 \quad \therefore c \geq 1/3$ $c < 1$

Case 3 holds. So $T(n) \in \Theta(n^2)$

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

$a = 2,\ b = 2,\ n^{\log_b a} = n^{\log_2 2} = n,\ n \in \Omega(n^{1+\epsilon})$, for $\epsilon = 1$ ✓

$\therefore a \cdot f(n/b) \leq c \cdot f(n)$

$\Rightarrow 2 \times \frac{n^2}{4} \leq c \cdot n^2$

$\Rightarrow \frac{1}{2} n^2 \leq c \cdot n^2$ for $c \geq \frac{1}{2}$ $c < 1$

Case 3 holds $T(n) \in \Theta(n^2)$

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

$a = 4,\ b = 4.\ n^{\log_b a} = n^1$, $\log n \in O(n^{\log_b a - \epsilon}) \Rightarrow \log n \in O(n^{1-\epsilon})$ ✓

for some $\alpha\epsilon < 1$ (any), $\log n \in O(n^{1-\epsilon})$

Case 1 holds $T(n) \in \Theta(n)$

(15)

5 Randomized Analysis (15 Points)

Suppose someone offers to let you play a game. You pay $1 to play the game. You roll two dice. If the sum of the dice is 7 *or* if you roll doubles (same number on both dice), then you lose. Otherwise you will get some money back. To determine how much money you get back, you roll the one die again. The amount you are paid is the number on the final roll paid in dollars. What is the expected value of this game from your perspective?

Examples: If we roll two 4s on the first roll, our result is -$1. If we roll a 5 and a 2 on the first roll, our result is -$1. If we roll a 1 and a 4 on the first roll, and our second roll is a 1 then our result is $0 (we get $1 back but we paid $1 to play in the first place). If we roll a 2 and a 3 on the first roll and then roll a 6 on the second roll, our result is $5.

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

The probability to lose in first chance $P_{lose} = \frac{6}{36} + \frac{6}{36} = \frac{12}{36} = \frac{1}{3}$

to win in first case $P_{win} = \frac{2}{3}$

For the 2nd chance each number have same probability $= \frac{1}{6}$

for gain we need successive win then $p(win) = \frac{1}{6} \times \frac{2}{3} = \frac{2}{18} = \frac{1}{9}$

Let x to be our gain $\begin{cases} -1 \text{ for lose.} \\ 0, 1, 2, 3, 4, 5, 6 \text{ for 2nd die to } (1,2,3,4,5,6) \end{cases}$

Expected gain $E[x] = P(P_{lose}) \times (-1) + P(P_{win}) \times (0+1+2+3+4+5)$

$= \frac{-1}{3} + \left(\frac{1}{9}\right) \times 15 = -\frac{1}{3} + \frac{15}{9 \cdot 3} = \frac{5}{3} - \frac{1}{3}$

$= \frac{4}{3}$

$= \frac{6}{3} = 2$

6 Divide and Conquer (25 Points)

25

Let A be a *sorted* array of n integers. Give a divide and conquer algorithm that finds an index i such that $A[i] = 2i$ (you can use indices from 0..n-1 or 1..n, just state which one you are using). If there is no such index, then your algorithm should return -1. Your algorithm should run in $o(n)$ time. Argue why your algorithm is correct. State the running time of your algorithm as a recurrence relation, and use the master method to show that the running time is indeed $o(n)$.

Example using 0-indexing: $A = [1, 3, 4]$. Then we want to return 2 since $A[2] = 2$ and $2 \cdot 2 = 4$. We cannot return 0 or 1 as $0 \neq 2 \cdot 1$ and $1 \neq 2 \cdot 3$.

Note: That example can show that this problem is a bit more complicated than just using a straightforward binary search approach. So essentially running straightforward binary search will not give you the correct answer all of the time.

A = [1 ... n].

i = 1 2 3 4 5 6
n = 2 4 5 6 10 12

Good example here!
Shows problem is harder than intended.

FindtheIndex (A, l, r)
  if (l > r) then return -1
  m = ⌊(l + r)/2⌋
  if (A[m] == 2·m) then return m
  if (A[l] ≤ 2·m and A[m] ≤ 2·m)
    return FindtheIndex (A, l, m-1)
  if (A[h] ≥ 2·m and A[h] ≤ 2·h)
    return FindtheIndex (A, m+1, h)

l m
i = 2 3 4
A[i] = need to range in (1 to 8)

(m) (h)
i = 2 3 4
5 6 9

need to be (2·m ≤ 2·h) to search in this range.

The recurrence relation $T(n) = 2T(n/2) + O(1)$ OK

$a = 2$, $b = 2$, $n^{\log_b a} = n^1$, $f(n) \in O(n^{\log_b a - \varepsilon})$