Quiz Paper

IBA Institute of Business Administration Karachi
Leadership and Ideas for Tomorrow

Full Name: _____
Student ERP I/D No: _____
Subject: _____
Quiz #: _____ Date : _____
Class/Section: _____
Teacher's Name: _____

(Q1) Correct order ($f = O(g)$) :

$1/n^3 \leq 0.0003 \leq \log\log n \leq \sqrt{n}\log n \leq \frac{n}{\log n} \leq 10^{-6}n^7 \leq 2^{n^{\frac{2}{3}}} \leq 2^{\frac{n}{\log n}} \leq (\sqrt{n})^n \leq n^n$

Working:

We will first group all functions into categories:

Constants : 0.0003

Logarithms : $\log(\log n)$

Polynomials : $10^{-6}n^7$, $\sqrt{n}\log n$, $\frac{n}{\log n}$, $\frac{1}{n^3}$

Exponents : n^n, $2^{n^{\frac{2}{3}}}$, $(\sqrt{n})^n$, $2^{\frac{n}{\log n}}$

Simplifying dominant terms of terms of polynomials for comparison:

n^7, $n^{\frac{1}{2}}$, n, n^{-3}

Arranging them:

n^{-3}, $n^{\frac{1}{2}}$, n, n^7

We already know $2^n = O(n^n)$. So using log to compare $2^{n^{\frac{2}{3}}}$ & $2^{\frac{n}{\log n}}$:

$2^{n^{\frac{2}{3}}}$		$2^{\frac{n}{\log n}}$
$\log_2(2^{n^{\frac{2}{3}}})$		$\log_2(2^{\frac{n}{\log n}})$
$n^{\frac{2}{3}}\log_2(2)$		$\frac{n}{\log n}\log_2(2)$
$n^{\frac{2}{3}}$	$\leq$	$\frac{n}{\log n}$

Arranging exponents using this knowledge:

$2^{n^{\frac{2}{3}}} \leq 2^{\frac{n}{\log n}} \leq n^{n/2} \leq n^n$

Marks _____ Signature of Instructor _____

And finally, we justify the order of $1/n^3$ and 0.0003 using limits:

$$\lim_{n\to\infty} \frac{1/n^3}{0.0003} = \lim_{n\to\infty} \frac{1}{\infty} = 0$$

so $1/n^3 \leq 0.0003$.

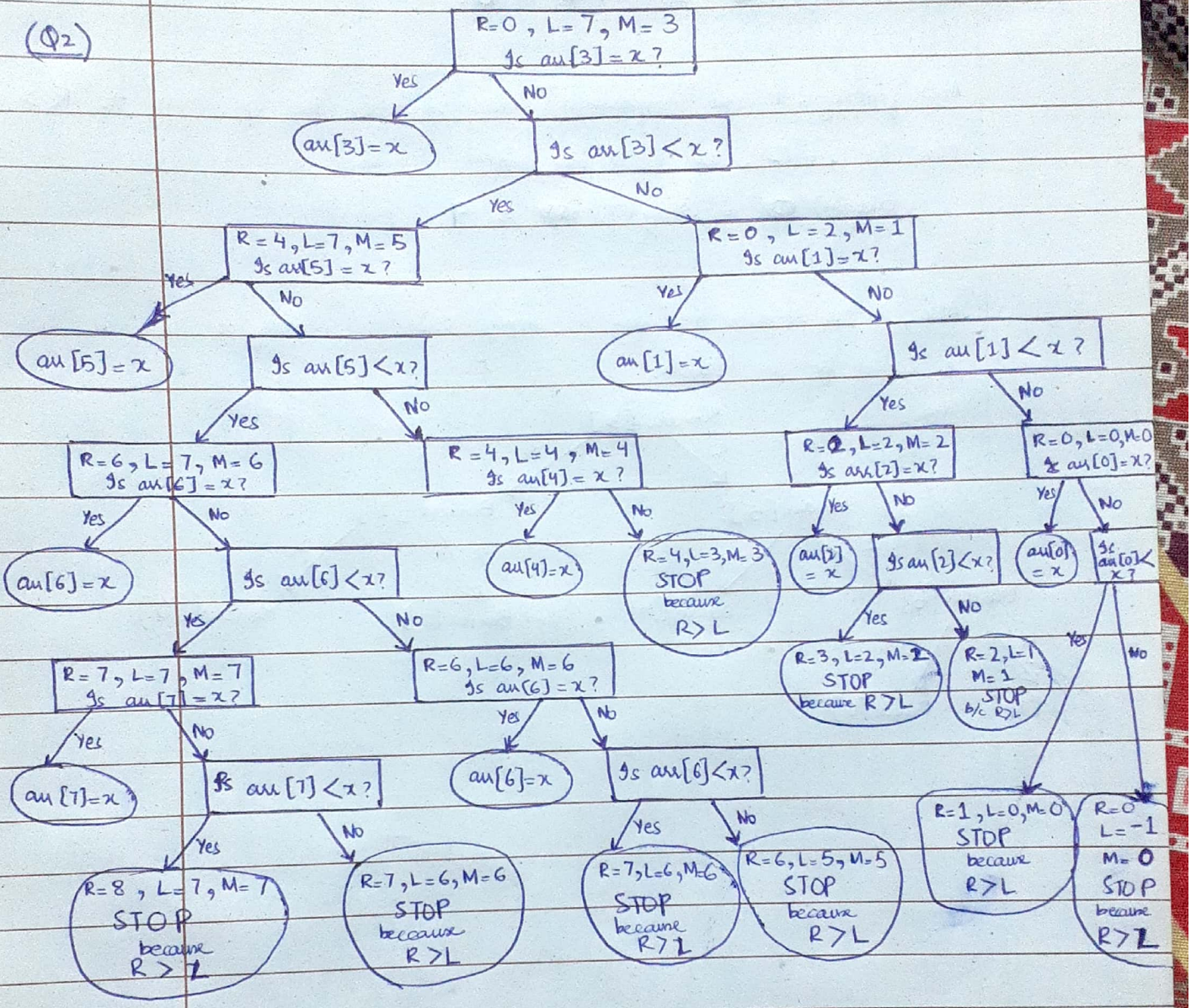
Using above ideas and constants $\leq$ logarithms $\leq$ polynomials $\leq$ exponents, we have:

$$1/n^3 \leq 0.0003 \leq \log(\log n) \leq \sqrt{n}\log n \leq \frac{n}{\log n} \leq$$
$$10^{-6}n^7 \leq 2^{n^{\frac{2}{3}}} \leq 2^{\frac{n}{\log n}} \leq n^{n/2} \leq n^n.$$

IBA Institute of Business Administration Karachi
Leadership and Ideas for Tomorrow

Full Name: _____
Student ERP I/D No: _____
Subject: _____
Quiz #: _____ Date : _____
Class/Section: _____
Teacher's Name: _____

(Q2)



At each level, if we consider the number of "compairsons" to be just the dominant operation "Is arr[i]<x?", then for x=12, our total compairsons become: $\log(n) = \log(7) \approx 3$.

However, if ~~at each level~~ we consider "compairsons" to include the operation "Is arr[i]==x?" then for x=12, our total comparisons

Marks _____ Signature of Instructor _____

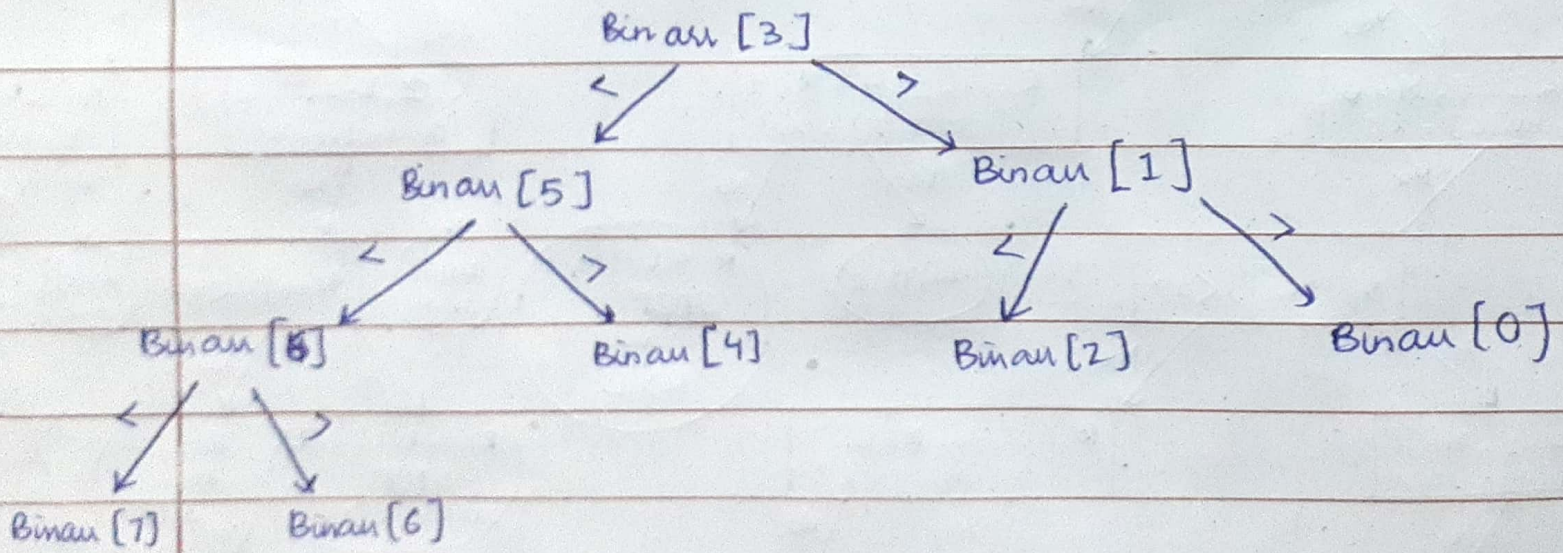
become : $2\log(n) = 2\log(7) \approx 6$.

And finally, if our "compaisons" also includes the operation "Is R >= L?"

then for $x = 12$, our total compaisons becomes:

$$3\log(n) = 3\log(7) \approx 9.$$

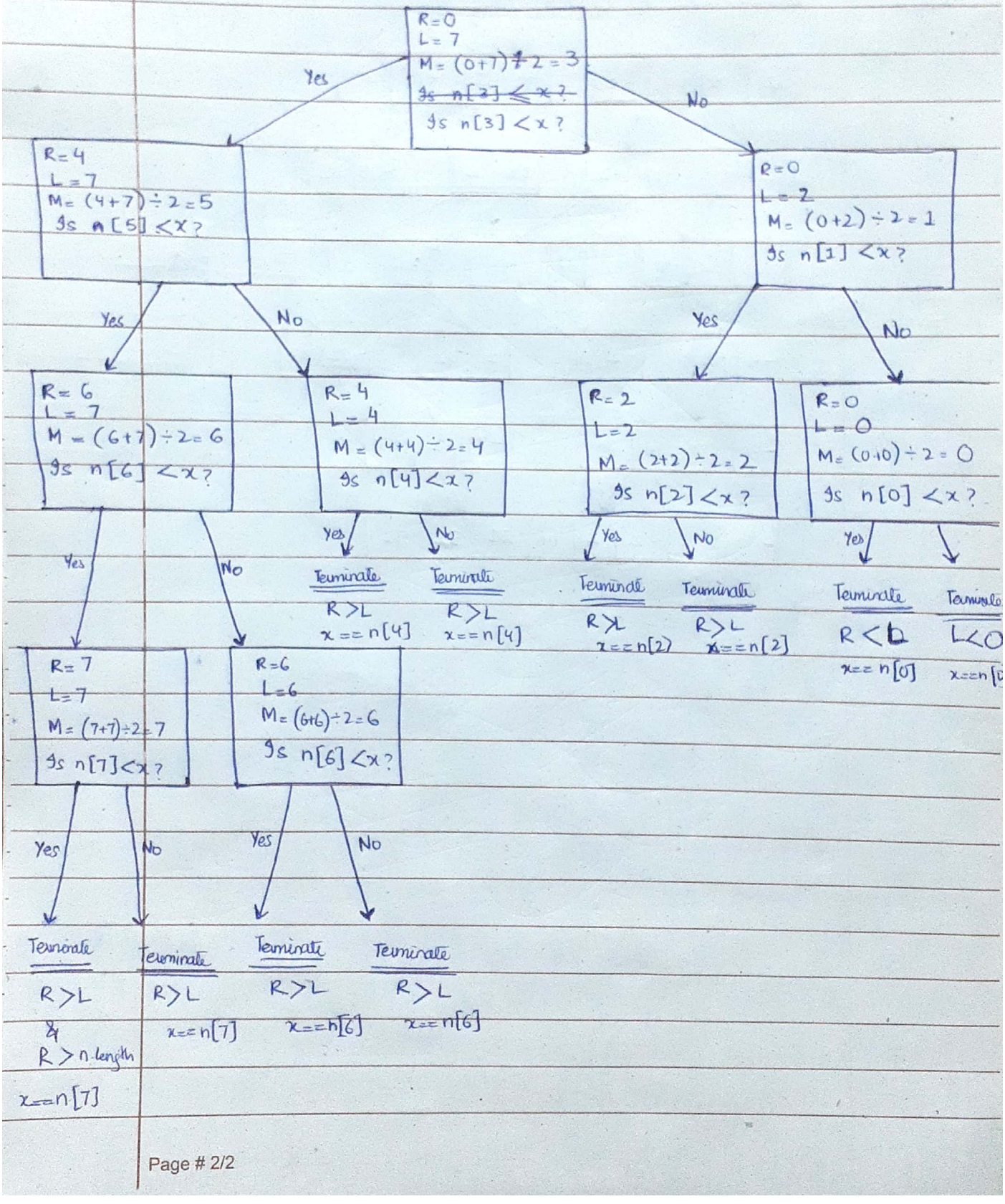
Also, if the decision tree was to simply show the binary seach part then it would be as follows:



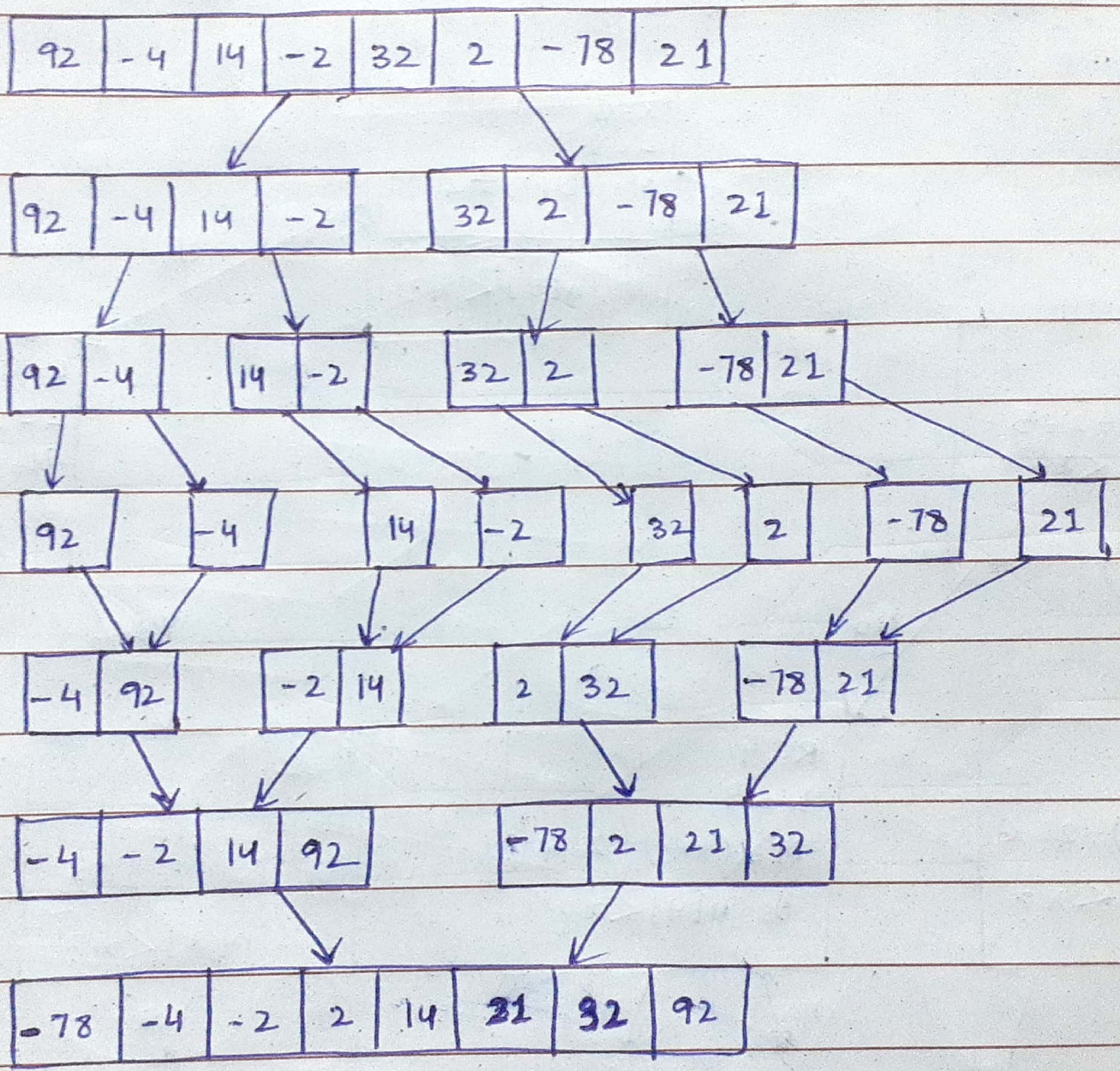
(Q2)

0	1	2	3	4	5	6	7
1	10	12	32	57	89	122	200

x = 12



(Q3)



(Q4)

1. If $f(n) = O(g(n))$ then :

$\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq \infty$, and

$|f(n)| \leq c \cdot g(n)$ for a constant $c > 0$ for all $n > n_0$.

$f(n)$ & $g(n)$ are also known to be positive functions.

If $\log_a(f(n)) = O(\log_a(g(n)))$ then :

$\lim_{n\to\infty} \frac{\log_a(f(n))}{\log_a(g(n))} \neq \infty$ and

~~$|f(n)| \leq c \cdot g($~~

$|\log_a(f(n))| \leq c \cdot \log_a(g(n))$ for $a, c > 0$ for all $n > n_0$

To prove false by contradiction, we set the following functions:

$f(n) = a$ and $g(n) = 1$

Since $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{a}{1} \neq \infty$ so $f(n) = O(g(n))$ holds.

Now see

$\log_a(f(n)) = O(\log_a(g(n))$ holds or not :

$\log_a(a) \leq c \cdot \log_a(1)$

$1 \leq c \cdot 0 \longrightarrow$ does not hold!

Since $c \cdot 0$ will always be equal to 0 and $\log_a a$ will always be 1 and $1 > 0$ so $\log_a f(n) \neq O \log_a g(n)$

& because we have a contradiction, this is :

False Ans

Marks _____ Signature of Instructor _____

2. $2^{f(n)} = \Omega(2^{g(n)})$ implies $\lim_{n\to\infty} \frac{2^{f(n)}}{2^{g(n)}} \neq 0$.

To prove false by contradiction,

we use the following counterexamples:

$f(n) = n \qquad g(n) = n^n$

Since $f(n) \leq c \cdot g(n)$ for $c > 0$, $n > n_0$.

$\therefore n \leq c \cdot n^n$

so $f(n) = O(g(n))$ holds.

Now applying limits to $2^{(f)}$ & $2^{g(n)}$:

$$\lim_{n\to\infty} \frac{2^n}{2^{n^n}}$$

$$\lim_{n\to\infty} \frac{1}{2^{n^n - n}} = 0 \longrightarrow \text{contradiction}$$

So it is False Ans

3. Since $f(n) = O(g(n))$ so for $c > 0$, $n > n_0$ this exists:

$f(n) \leq c \cdot g(n)$

Take square root on both sides.

$\sqrt{f(n)} \leq \sqrt{c \cdot g(n)}$

$\sqrt{f(n)} \leq \sqrt{c} \cdot \sqrt{g(n)}$

↓

Since $c > 0$ so $\sqrt{c} > 0$ and lets call it c_0

$\sqrt{f(n)} \leq c_0 \cdot \sqrt{g(n)}$

and this can be written as $\sqrt{f(n)} = O(\sqrt{g(n)})$ so it is true.

True Ans

Institute of Business Administration Karachi
d Ideas for Tomorrow

Full Name: _____
Student ERP I/D No: _____
Subject: _____
Quiz #: _____ Date : _____
Class/Section: _____
Teacher's Name: _____

4. If $10^{100} f(n) = O(10^{-100} \cdot g(n))$ then $\lim_{n\to\infty} \frac{10^{100} f(n)}{10^{-100} g(n)} \neq \infty$

Proof:

$$\lim_{n\to\infty} \frac{10^{100} f(n)}{10^{-100} g(n)}$$

$$\lim_{n\to\infty} \frac{10^{100-(-100)} f(n)}{g(n)}$$

$$\lim_{n\to\infty} \frac{10^{200} f(n)}{g(n)}$$

$$\lim_{n\to\infty} 10^{200} \times \lim_{n\to\infty} \frac{f(n)}{g(n)}$$

$\lim_{n\to\infty} 10^{200} \neq \infty$ and $\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq \infty$ because $f(n) = O(g(n))$

so $\lim_{n\to\infty} \frac{10^{100} f(n)}{10^{-100} g(n)} \neq \infty$.

so True Ans

5. If $2^{\frac{n}{1+\log(f(n))}} = O\left(2^{\frac{n}{1+\log(g(n))}}\right)$ then since our entire functions seem identical except the denominators of our exponents powers, we just need to show $\frac{n}{1+\log f(n)}$ does not grow at a faster rate than $\frac{n}{1+\log(g(n))}$ or that $\lim_{n\to\infty} \frac{n/1+\log(f(n))}{n/1+\log(g(n))} \neq \infty$.

Using part 1 we can argue that $\log(f(n))$ will not grow faster than $\log(g(n))$ and the slower our denominator grows, the larger the value of our power of 2 meaning

Marks _____ Signature of Instructor _____

at some point $2^{\frac{n}{1+\log(f(n))}}$ will start to grow faster than $2^{\frac{n}{1+\log(g(n))}}$

so lets use a counter example to prove this:

$f(n) = n$

$g(n) = n^n$

$$\lim_{n\to\infty} \frac{1+\log(n)}{1+\log(n^n)}$$

$$\lim_{n\to\infty} \frac{1+\log(n)}{1+n\log(n)}$$

$$\lim_{n\to\infty} \frac{\frac{1}{1+\log(n)}}{\frac{1}{1+\log(n^n)}}$$

$$\lim_{n\to\infty} \frac{1+n\log n}{1+\log n}$$

$$\lim_{n\to\infty} \frac{\frac{1}{\log n}+n}{\frac{1}{\log n}+1}$$

$$\lim_{n\to\infty} \frac{0+\infty}{0+1} = \infty$$

so $\frac{1}{1+\log(f(n))}$ grows faster than $\frac{1}{1+\log(g(n))}$ so

even $2^{\frac{n}{1+\log f(n)}}$ grows faster than $2^{\frac{n}{1+\log(g(n)}}$ so

this part is false.

False Ans

(Q5)

Points to consider:

- This is not a binary seach tree, so nodes are not arranged in sorted order.
- This is a complete binary tree so number of leaf nodes is maximum up to the second last level of the tree. On the last level, the leaf nodes are filled from left to right.
- Probing is assumed to take $O(1)$ time.
- By 'edge' we mean all nodes connected to a specific node which can be max 3 for any node in a binary tree.
- We need to find just a local minimum in T.

An algorithm for this would be:

1. Start at root.
2. Probe the node you are on.
3. Probe the left child (if any) of the node.
4. Probe the right child (if any) of the node.
5. If the x_v first node is smaller than both its children, you have found a local minimum.
6. If any of the children nodes, x_v are smaller than the first node's, come down to the smaller of the two (it is now your first node and go back to step 2.

At each level of T, a maximum of 3 nodes are checked. The total number of levels in T ~~are~~ for n nodes is: ~~log n~~

$$n = 2^d - 1$$

$$d = \frac{\log(n+1)}{\log(2)}$$

Institute of
Business Administration
Karachi
Ideas for Tomorrow

Full Name: _____
Student ERP I/D No: _____
Subject: _____
Quiz #: _____ Date: _____
Class/Section: _____
Teacher's Name: _____

So if $\max_h 3$ probes are made at each level and the maximum number of levels we have in T are $\frac{\log(n+1)}{\log(2)}$ and we know that our algorithm ~~changes its~~ travels to the next depth after making these probes (~~basically~~ not checking any other nodes than these) then our total probes become maximum $\frac{3\log(n+1)}{\log(2)}$ which is $O(\log n)$.

Lets visualize it this way: we are probing any vertex in $O(1)$ time and at each depth, we are choosing only one of the child nodes (in case at least one of them is smaller than our current node) which means reducing our problem size by half. We can formulate it as this and then apply Master Theorem like this on it to get the $O(\log n)$ complexity:

Runtime: $T(n) = 1\,T(n/2) + O(1)$

$a = 1$, $b = 2$, $d = 0$

$\log_b a = \log_2 1 = 0 = d$

if $\log_b a = d$ then $O(n^d \log n)$

$O(n^d \log n) = O(n^0 \log n)$

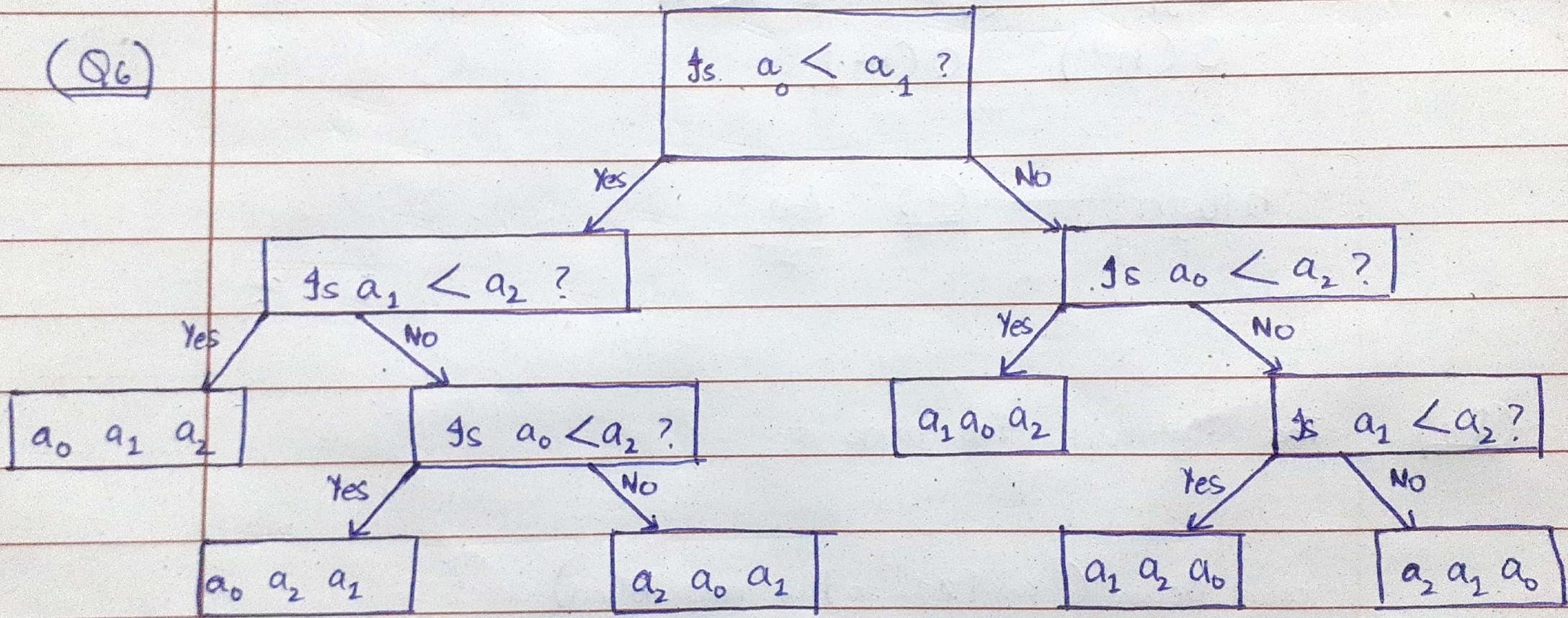
$= O(\log n)$ Ans

This looks more simple to prove!

Marks _____ Signature of Instructor _____

Teacher's Name: _____

(Q6)



$a_0\ a_2\ a_1$ | $a_2\ a_0\ a_1$

(Q7) (a) $T(n) = 7T(n/7) + 3n + 20$

$a = 7$, $b = 7$, $d = 1$

$\log_b a = \log_7 7 = 1 = d$

$O(n^d \log n) = O(n \log n)$

(b) $T(n) = 16T(n/4) + 100$

$a = 16$, $b = 4$, $d = 0$

$\log_b a = \log_4 16 = 2 > d$

$O(n^{\log_b a}) = O(n^2)$

Marks

(c) $T(n) = 2T(n/2) + 5n^2 + 2n + 3$

$a = 2, \quad b = 2, \quad d = 2$

$\log_b a = \log_2 2 = 1 < d$

$O(n^d) = O(n^2)$

$n \log n \longrightarrow$ (a) Ans

(Q8) The algorithm I've made is:

```
countOccur ( arr[0...n] , numberToFind)
{
    if ( arr.length == 1)
    {     if ( arr[0] == numberToFind)
                return 1;
          else
                return 0;
    }
    else
    {
        return countOccur ( arr[1...n/2] , numberToFind ) + countOccur (arr[n/2+1...n, numberToFind)
    }
}
```

Logic:
At each step, the array or size of problem is being halved and both these parts are being solved separately. The result at base case is being added in O(1) times.

Time complexity:

$T(n) = 2T(n/2) + O(1)$ $\quad a = 2, b = 2, d = 0$

$\log_b a = \log_2 2 = 1 > 0$

$O(n^{\log_b a}) = O(n^1) = O(n)$ Ans