Algo Assignment

k200353 Mohsin Ali Mirza

Q1 6, 16, 12, 27, 9, 1, 18, 5, 31

1st Pass 6, 16, 12, 27, 9, 1, 18, 5, 31 key = 16

2nd Pass 16, 6, 12, 27, 9, 1, 18, 5, 31 key = 12

3rd Pass 16, 12, 6, 27, 9, 1, 18, 5, 31 key = 27

4th Pass 27, 16, 12, 6, 9, 1, 18, 5, 31 key = 9

5th Pass 27, 16, 12, 9, 6, 1, 18, 5, 31 key = 1

6th Pass 27, 16, 12, 9, 6, 1, 18, 5, 31 key = 18

7th Pass 27, 18, 16, 12, 9, 6, 1, 5, 31 key = 5

8th Pass 27, 18, 16, 12, 9, 6, 5, 1, 31 key = 31

31, 27, 18, 16, 12, 9, 6, 5, 1

		Time Complexity	
`for(int i=1; i<n; i++)`	c_1	n	
`{`			
`key = arr[i];`	c_2	$n-1$	
`j = i-1;`	c_3	$n-1$	
`while (j>=0 && arr[j]<key)`	c_4	$n-1$	$\frac{(n-1)n}{2}$
`{ arr[j+1] = arr[j];`	c_5	0	$\frac{n(n-1)}{2}$
`j--; }`	c_6	0	$\frac{n(n-1)}{2}$
`arr[j+1] = key; }`	c_7	$n-1$	

K20-0353 Mohsin Ali Nizam

$nc_1 + (n-1)c_2 + (n-1)c_3 + (n-1)c_7$

$nc_1 + nc_2 - c_2 + nc_3 - c_3 + nc_7 - c_7$

$n(c_1+c_2+c_3+c_7) + (-c_2-c_3-c_7)$

For Best:-

$n(c_1+c_2+c_3+c_7) - (c_2+c_3+c_7) + c_4(n-1) + c_5(0) + c_6(0)$

$n(c_1+c_2+c_3+c_4+c_7) - (c_2+c_3+c_4+c_7) = f(n)$

$O(f(n)) \equiv O(n)$

For Worst:-

$n(c_1+c_2+c_3+c_7) - (c_2+c_3+c_7) + \frac{n(n-1)}{2}(c_4+c_5+c_6)$

$n(c_1+c_2+c_3+c_7) - (c_2+c_3+c_7) + \left[\frac{n^2-n}{2}\right]^2 (c_4+c_5+c_6)$

$\frac{n^2}{2}\left(\frac{c_4}{2}+\frac{c_5}{2}+\frac{c_6}{2}\right) + n\left(c_1+c_2+c_3-\frac{c_4}{2}-\frac{c_5}{2}-\frac{c_6}{2}+c_7\right) - (c_2+c_3$

$O(f(n)) \equiv O(n^2)$

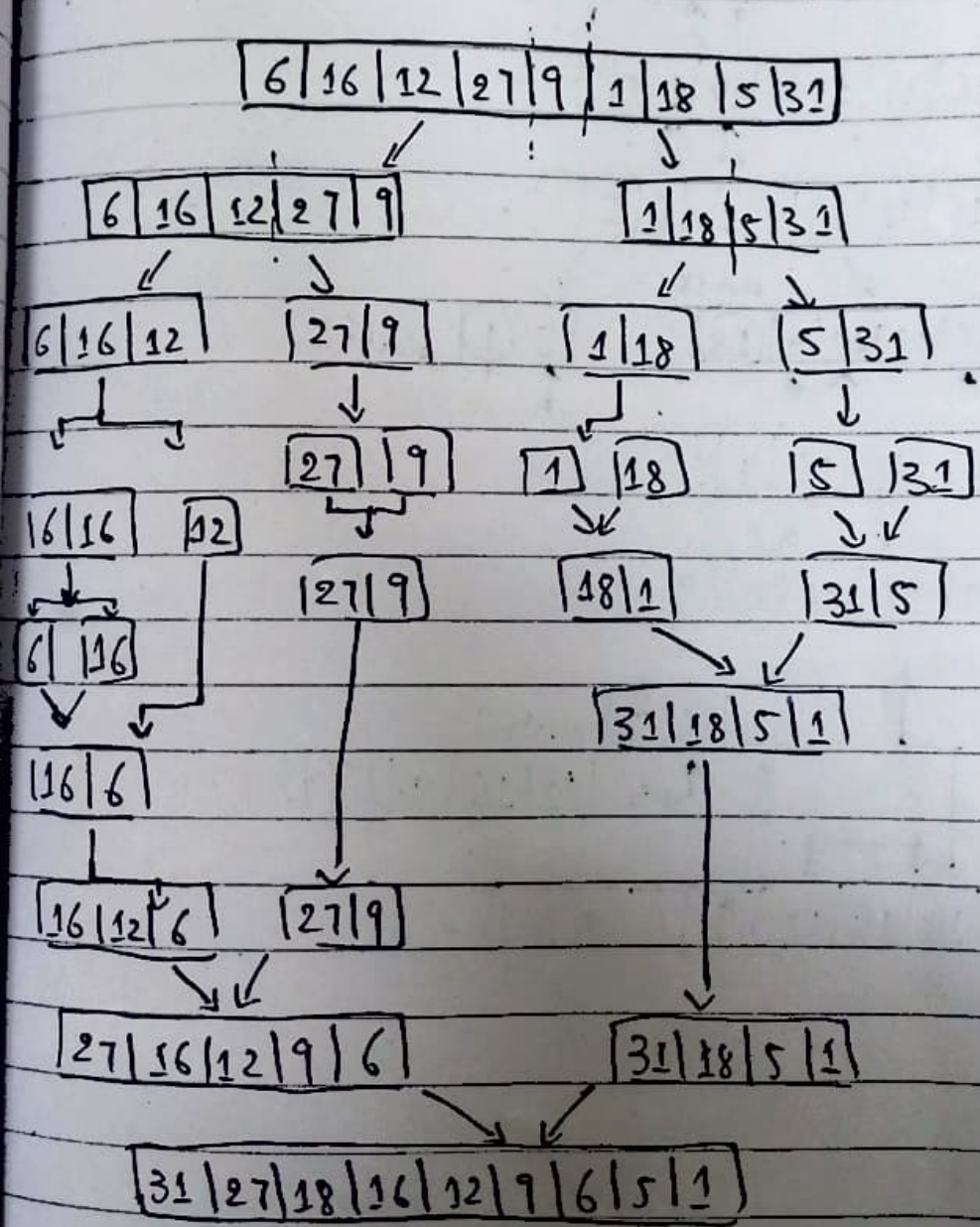
Loop invariant

1. Initialization:- If i=1 and the loops range is from 1 to i-1 then at arr[i] or arr[1] the sub array is already sorted because it is only a single element therefore the loop invariant holds prior to the first iteration of the loop.

2. Maintainence:- During the for loop we make sure that the elements on the left of key are greater than the key at any iteration of the loop. i.e for arr[1 to i] is sorted and sorts the arr for arr[1 to i+1] as well by comparing & swapping making sure that the loop invariant holds.

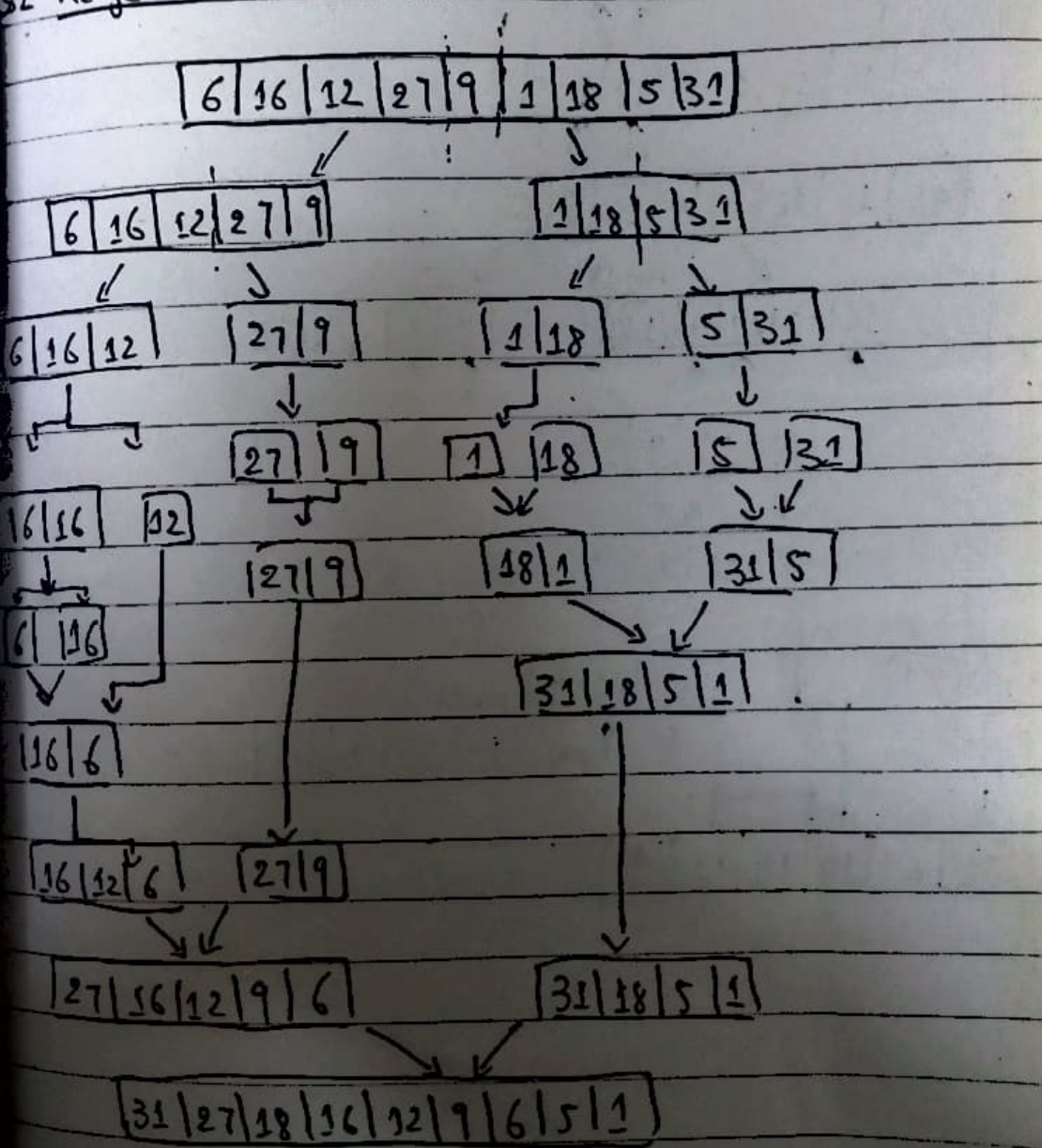
3. Termination:- When the loop reaches arr[n-1], all of the elements from arr[1 to n-1] are sorted and the length of arr has not changed. Hence the algorithm is correct and the elements in the arr has been successfully sorted in the descending order. (The loop terminates at i=n, making sure the length doesn't change).

Q2 Merge Sort:-



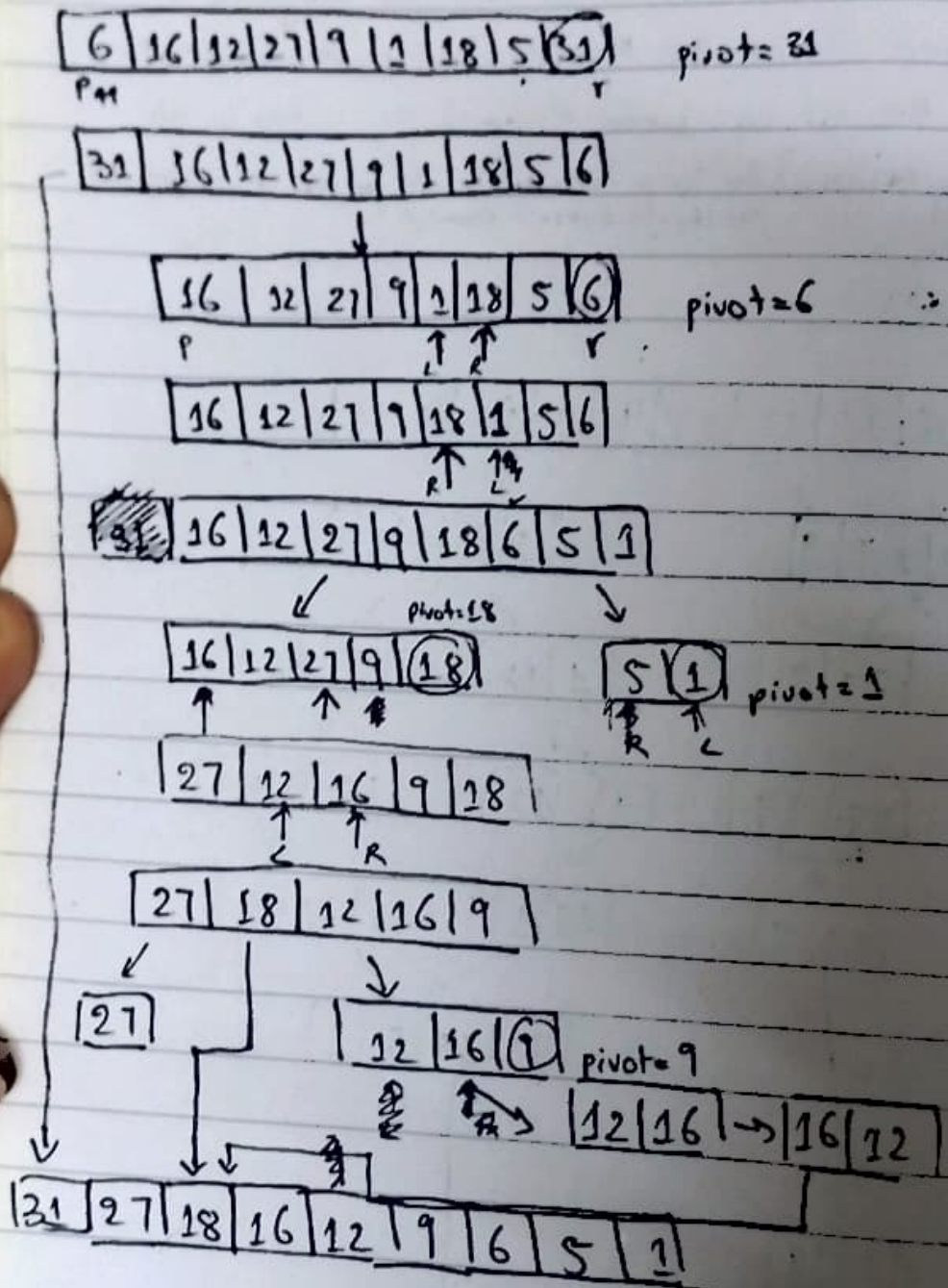
3. Termination:- When the loop reaches arr[n-1], all of the elements from arr[1 to n-1] are sorted and the length of arr has not changed. Hence the algorithm is correct and the elements in the arr has been successfully sorted in the descending order. (The loop terminates at i=n, making sure the length doesn't change).

2 Merge Sort:-



1k20c353 Mohsin Ali Mirza

Q3 Quicksort



k220353 Mohsin Ali Mirza

Loop Invariant:

Initialization:- Before the loop starts, all of the conditions of loop invariant are satisfied because r is the pivot and the subarrays arr[p....i] and arr[i+1.... j-1] are empty.

Maintainence:- While the loop is running, if A[j] ≥ pivot, then A[j] and A[i+1] are swapped and then i & j are incremed. If A[j] < pivot then increment only j.

Termination:- When the loop terminates j=r, all of the elements in A are partitioned into 1 of the 3 cases

1. Arr[p...i] ≥ pivot
2. Arr[i+1....r] < pivot
3. Arr[r] = pivot

1C200353 Mohsin Ali Mirza

Q4: a Using Merge sort the array

arr = sort (arr) — $n\log_2 n$

Compare if the values are same

```
i=0;
for (int j=n/2; j<n;j++)
{ if (arr[i]=arr[j]) return true; }
  i++;
}
return false;
```

$T(n) = n\log_2 n + \frac{n}{2} \Rightarrow O(n\log_2 n)$

b

```
map<int, int> m;
for (int i=0; i<n;i++)
{ m[arr[i]]++;
   if (m[arr[i]]>=n/2) retun true;
}
return false;
```

Thus time complexity is O(n) because a single loop is used

$T(n) = (n+1)c_1 + nc_2 \Rightarrow O(n)$

Q5. To minimize sum of ~~pair, of~~ we can pair largest number with the pair of smallest number with smallest

To minimize sum of pair, we must pair the largest number & the smallest number, and pair the 2nd largest number with the 2nd smallest number.

To achieve this we first try to sort the array so that we can easily access the largest & smallest values.

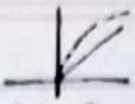
```
Step 1:- Sort(Arr[ ]);   // size of 2n
Step 2:- // partition the array into subarrays

    for(int i=0; i<n; i++)
{
    pointer1 = i;
    pointer2 = n-i;

    make_pair(Arr[pointer1], Arr[pointer2]);   -> Max sum pairs
}
```

Step 3:- find the minimum of those pairs, compare & print the result.

k200353 Mohsin Ali Mirza

Q6) $n^3 - 2n + 1 = O(n^3)$

$f(n) = n^3 - 2n + 1$



$f(n) \leq c \cdot g(n)$ for $n \& c \geq 0$ and are positive integers.

$g(n) = n^3 + 2n^3 + n^3$

$= 3n^3 \quad c = 3$

$= 4n^3 \quad c = 4$

$n^3 - 2n + 1 \leq 4n^3$

$O(4n^3) \equiv O(n^3)$ therefore proved

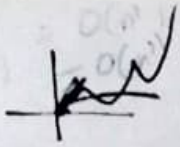
ii) $5n^2 \log_2 n + 2n^2 = O(n^2 \log_2 n)$

$5n^2 \log_2 n + 2n^2 \log_2 n = O(n^2 \log_2 4)$

$c * g(n) \geq f(n)$

$7n^2 \log_2 n \geq 5n^2 \log_2 n + 2n^2$ for $n \geq 3$

$O(g(n)) \equiv O(n^2 \log_2 n)$ hence proved to be a good estimate

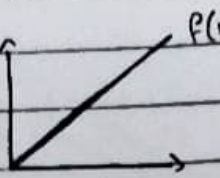


Ω $n! = n \times n \ldots n$

$\log n \log n$

Date:

Asymptotic Bounding



$T(n) <= c * f(n)$ $n >= n_0$

$f(n) <= c * g(n)$ $n >= n_0$ → for Big O (worst case) Upper Bound $\forall n \geq R$ (+ve integers)

$f(n) >= c * g(n)$ $n >= n_0$ → for Big Ω (Best case) Lower Bound

Best case $\leq \Theta \leq$ Worst case

↓

Average case

* $c_1 * g(n) \leq \Theta \leq c_2 * g(n)$

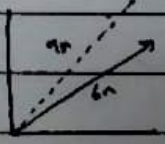
should belong to same class

$f(n) = 6n+3$ → For Big O

$O(1) < O(\log n) < O(\sqrt{n}) < O(n) < O(n \log n)$
$< O(n^2) < O(n^{10}) < O(3^n) < O(n^n) < O(2^{2^n})$

$6n+3 \leq 6n+3n$

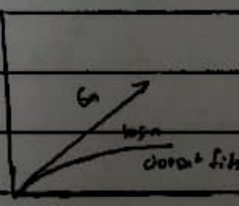
$6n+3 \leq 9n$

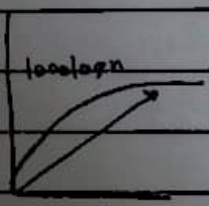


true for $\forall n \geq 1$

however lets take $O(\log n)$

$6n+3 \leq \log n$





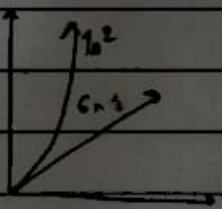
fails at large values of n

∴ you can't use $\log n$.

$6n \leq 9n^2$

↑

Also true but you should try to take the tightest bound therefore you use linear $O(n)$.



$6n \leq 10n$ also true but our main concern over here is $O(n)$, so the tightness within a class is not an issue

$n! \rightarrow$ Lower Bound $1 * 1 * 1 * \ldots = 1$

$n! \rightarrow$ Upper Bound $n * n * n-1 * \ldots n = n^n$

$1 \leq n! \leq n^n$

different class therefore no Ω, Θ.

$6n \leq$

1c200353 Mohsin Ali Mecca

Q8

a) $T(n) = 2T\left(\frac{n}{3}\right) + cn^2$ $\quad a=2, b=3, k=2$

$\log_3 2 = 0.630$ ~~0.176~~ | Case 3 | $\log_b a < k$

$0.630 < 2$ ~~0.176~~ | $T(n) = O(n^2)$ |

b) $T(n) = 4T\left(\frac{n}{3}\right) + cn$ $\quad a=4, b=3$ $k=1$

$\log_3 4 = 1.261$

| Case 1 | $\log_b a > k$

$1.261 > 1$

$T(n) = n^{\log_3 4}$ | $T(n) = O(n^{\log_3 4})$ |

c) $T(n) = 8T\left(\frac{n}{2}\right) + cn^3$

$a=8, b=2, k=3$ | Case 2 | $\log_b a = k$

$\log_2 8 = 3$

$3 = 3$

log | $T(n) = O(n^3 \log n)$ |

[illegible] [illegible]

Q1:

$T(n) = 2T\left(\frac{n}{3}\right) + n^2$

$T(n/3) = 2T\left(\frac{n}{3^2}\right) + \left(\frac{n}{3}\right)^2$

$T(n/9) = 2T(n/3^3) + (n/3^3)^2$

$T(n) = 2\left[2T\left(\frac{n}{9}\right) + \left(\frac{n}{3}\right)^2\right] + n^2 = 2^2T\left(\frac{n}{9}\right) + 2\left(\frac{n}{3}\right)^2 + n^2$

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

$= 2^3T\left(\frac{n}{3^3}\right) + 2^2\left(\frac{n}{3^3}\right)^2 + 2\left(\frac{n}{3}\right)^2 + n^2$

$= 2^kT\left(\frac{n}{3^k}\right) + n^2\left(\frac{2^2}{3^{6}} + \frac{2^2}{3^2} + 1 \ldots\right)$

$T(1) = 1$

$n = 3^k$

$\log_3 n = k$

$= 2^{\log_3 n}\, T(1) + n^2 * c$

↑ Some constant.

$O(n^{\log_3 2} + n^2 * c) \approx O(n^2)$ because

$O(n^{\log_3 2}) < O(n^2)$

K200353 Mohsin Ali Mirza

2. $T(n) = 4T\left(\frac{n}{3}\right) + n \qquad T(1) = 1$

$T(n_3) = 4T\left(\frac{n}{9}\right) + \frac{n}{3}$

$T\left(\frac{n}{9}\right) = 4T\left(\frac{n}{81}\right) + \frac{n}{27}$

$T\left(\frac{n}{81}\right) = 4T($

$T(n) = 4\left[4T\left(\frac{n}{9}\right) + \frac{n}{3}\right] = 4^2 T\left(\frac{n}{9}\right) + \frac{4n}{3}$

$T(n) = 4^2\left[4T\left(\frac{n}{81}\right) + \frac{n}{27}\right] + \frac{4n}{3}$

$= 4^3 T\left(\frac{n}{81}\right) + \frac{4n}{3} + \frac{16n}{27} \ldots$ Le

$T(n) = 4^3 T\left(\frac{n}{3^3}\right) + n\underbrace{\left[\frac{4}{3} + \frac{16}{27} \ldots\right]}_{\text{Constant } c}$ Let

$T(n) = 4^k T\left(\frac{n}{3^k}\right) + n * c \qquad 3^k = n$

$k = \log_3 n = k$

$T(n) = 4^{\log_3 n} T(1) + cn$

$T(n) = n^{\log_3 4} + cn \qquad O(T(n)) = \underline{O(n^{\log_3 4})}$

K200353 Mohsin Ali Mirza

Q10

i $f(n) = O(g(n))$ $\quad f(n) = \Omega(h(n))$

then $g(n) + h(n) = \Omega(h(n))$

$f(n) \le c * g(n_1)$ $\quad f(n) \ge c * h(n_2)$

for $n \ge n_1$ $\quad n \ge n_2$

$\frac{f(n)}{c} \le g(n)$ $\quad \frac{f(n)}{c} \ge h(n)$

$g(n) + h(n) = \frac{f(n)}{c} + \frac{f(n)}{c} = \frac{2f(n)}{c}$

Hence, $g(n) + h(n) \ge f(n)$ $\quad g(n) \le \frac{f(n)}{c} \le h(n)$

$g(n) + h(n) = \Omega(f(n))$ True

ii $cg(n) \le f(n) \le Cg(n)$ $\quad n \ge n_0$

$\max\{f(n), g(n)\} \le f(n) + g(n) \le 2\max\{f(n), g(n)\}$

for all n $\max\{f(n), g(n)\} \in \Theta(f(n) + g(n))$ True

i If $f(n) \in O(g(n))$ there exists $c > 0$, $n \ge n_0$

such that

$f(n) \le c * g(n)$

$[f(n)]^2 \le [c * g(n)]^2$

$(f(n))^2 \le (O(g(n)))^2$ True