Question 01

Original Array:

9, 19, 14, 29, 13, 4, 23, 10, 37

```
Algo(A, n)
    for(j=1; j<n; j++)
      A[j] = key
      i = j-1
    for(i>0 && A[i]>key)
      A[i+1] = A[i]
      i = i-1
      A[i+1] = key
```

First iteration:- — swap count = 0

Key = 19

since 19 > 9

then,

Second:- 19, 9, 14, 29, 13, 4, 23, 10, 37 — swap count = 1

14 > 9 swap

19, 14, 9

14 ≯ 19

Third:- 19, 14, 9, 29, 13, 4, 23, 10, 37 swap count = 2

29 > 9, 29 > 14, 29 > 14 swap

29, 19, 14, 9, 13, 4, 23, 10, 37

Fourth:-

13 > 9, 13 ≯ 14

29, 19, 14, 13, 9, 4, 23, 10, 37 swap count = 3

Fifth:- 4 ≯ 9

29, 19, 14, 13, 9, 4, 23, 10, 37 = 4

Sixth:-

23 > 4, 23 > 9, 23 > 13, 23 > 14, 23 > 19, 23 ≯ 29 = 4

29, 23, 19, 14, 13, 9, 4, 10, 37

Seventh:- 10 > 4, 10 > 9, 10 ≯ 13

29, 23, 19, 14, 13, 10, 9, 4, 37 = 6

Page No.

Teacher's Signature

Date

Eighth :-

key = 7

29, 23, 19, 14, 13, 10, 9, 4, 37

37 > 4, 37 > 9, 37 > 10, 37 > 13, 37 > 14, 37 > 19, 37 > 23, 37

37, 29, 23, 19, 14, 13, 10, 9, 4.

Swap count = 0, 1, 2, 3 ..... n-1, n

Sum of n natual numbers

$$T(n) = \frac{n(n-1)}{2}$$

$$= \frac{n^2}{2} - \frac{n}{2}$$ dominating term is n^2

$$= O(n^2)$$

Question: 02 Date

Merge Sort:-

- If the array is of length 0 or 1, it is already sorted
- divide unsorted array into three - sub arrays
- Use merge sort algorithm recursively to sort each sub array
- Merge three - sorted sub - array to form a single sorted list

Original array

9, 19, 14, 29, 13, 4, 23, 10, 37 - 9 1

l_1 l_2 l_3

9 19 14, 29, 13, 4, 23, 10, 37 - 3

9, 19, 14, 29, 13, 4, 23, 10, 37 - 9

19, 14, 9, 29, 13, 4, 37, 23, 10 - 3

37, 29, 23, 19, 14, 13, 10, 9, 4 - 1

$= 3^0, 3^1, 3^2 \ldots 3^n$

$= O(\log_3 n)$

Date

Question: 03

- Select an element pivot from array elements
- Re arrange the elements in the array in such a way that are greater than the pivot appear before pivot and less than the pivot element come after it
- After such partitioning, the pivot is placed in its final position. This is called partition operation
- Recursively sort the two sub-array thus obtained

i j

9, 19, 14, 29, 13, 4, 23, 10, 37

Acc to condition, (#

pivot = 9

19 > 9 ✓ · swap(A[i], A[j]) i++

First 19, 9, 14, 29, 13, 4, 23, 10, 37

i j

14 > 9 ✓ swap(A[i], A[j]) i++

Second 19, 14, 9, 29, 13, 4, 23, 10, 37

i j

third since 29 > 9 shift to left

fourth j = 13 13 > 9 shift to left

fifth j = 4 4 ✗ 9 shift to right

sixth j = 23 23 > 9 shift to left

seventh j = 10 10 > 9 shift to left

eighth j = 37 37 > 9 shift to left

array becomes

19, 14, 29, 13, 23, 10, 37, 9, 4

left [array] > pivot left [array] < pivot

sort L_1 and L_2

37, 29, 23, 19, 14, 13, 9, 4

- $O(n \log n)$

Date

Question: 04

i)

```
void divideTeam(arr[] t1, t2)
for (i=0; i<2n; j++)
  { for (j=0; j<2n-1; j++) {
    if (arr[j] > arr[j])
    arr[i] = temp   temp = arr[j]
    temp = arr[i]   arr[i] = arr[j]
    arr[j] = temp;
  }
  } l=0
for (K=0; K<2n; K+=2)
  {
    if (K%4 == 0)
    t1[l] = arr[K]
    t2[l++] arr[K+1]
  }
  else
    {
    team2[i] = arr[K]
    team[l++] = arr[K+1]
    }
}
```

```
Quicksort ( A, m+1, q)
}

void Sort(arr[], e, s)
if(e<s)
m = e + (s-1)/2
    Sor ( arr, s, m)
    Sort ( arr, m+1, e)
     merge( arr, s, m, e)
     }
void divideTeam ( arr, team[], team2[], n) {
sort(arr, 2n);
  u=0;
for(i=0; i< 2n; i+=2)
    if ( k % 4 == 0)
    team[e] = arr[i]
    team2[u++] = arr[i+1]
    }
else
    team2[u] = arr[i]
    team[ u++] = arr[i++]
}
}
```

Date

ii)

```
void merge ( arr[ ], l, m, r)
    s1 = m - l + 1
    s2 = r - m;
l[ ] = new int [s1]
r[ ] = new int [s2]
for (i = 0 ; i < s1 ; i++)
    l [i] = arr [i + l]
for (j = 0 ; j < s2 ; j++)
    R[j] = arr [j + m + 1]
k = l , u = 0 , v = 0
while (u < n1 && v < n2)
    if (L[0] > R[v]) {
        arr[k++] = l[u++]
    Else
        arr[k++] = R[u++]
    }
while (u < s1)
    arr[k++] = l[u++];
while [v < s2]
    arr[k++] = R[v++]
}
```

Date

Question 05

```
void merge ( arr[ ], l, m, r)
    s1 = m - l + 1
    s2 = r - m;
l[ ] = new int [s1]
r[ ] = new int [s2]
for (i = 0 ; i < s1 ; i++)
    l [i] = arr [i + l]
for (j = 0 j < s2 ; j++)
    R[j] = arr [j + m + 1]
k = 1 , u = 0, v = 0
while(u < n1 && v < n2)
    if (L[0] > R[v]) {
        arr[k++] = L[u++]
    else
        arr[k++] = R[u++]
    }
while ( u < s1 )
    arr[k++] = l[u++];
while [v < s2]
    arr[k++] = R[v++]
}
}
```

Date

```
void sort(A, e, s)
if(e<s)
{
m = e + (s-1)/2
sort(a, s, m)
Sort(a, m+1, e)
merge(a, e, m, s)
}

void Pairs(A, n)
{
Arr[2n)
i=0, j=(2n-1), k=0
Sort(Arr, 0, 2n);
while(i<j){
tempArr[k++] = arr[i++];
tempArr[k++] = arr[j--];
}
arr = tempArr
}
```

$f(n) \in \Theta(n^2)$

Question : 06

$f(n) = n^2 + 3n + 15$

let $g(n) = 10n^2 + 3n + 15$ $\because c = 10, n_0 \geq 1$

so $f(n) \leq g(n)$

$f(n) \in \Theta(n^2)$

2) $5n^2 \log_2 n + 2n^2$

let $g(n) = 5n^2 \log_2 n + 4n^2$ $\because c = 5, n_0 \geq 1$

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

$f(n) \in \Theta(n^2 \log n)$

~~Question~~

Question: 09

For an algorithm we must make sure ~~two~~ important measure before applying it

1. Measure Effeciency:-

The algorithm we are applying must give relatively better speed and resource usage on all the size of input data sets.

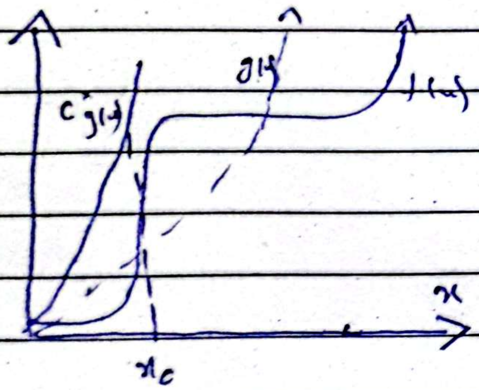
2. For calculating speed, our algorithm must work the n-inputs effeciently

We measure speed in three types of Bound

- Big O, Big Ω, Big Θ

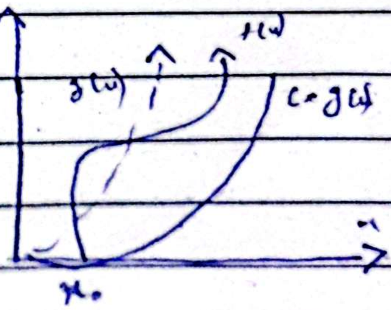
Big O:

$f(x) \in O(g(x))$ if there exist $c, x_0 > 0$ such that $f(x) \le c \cdot g(x)$ for all $x \geq x_0$



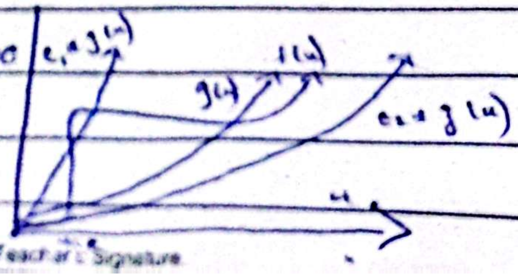
Big Ω:

$f(x) \in \Omega(g(x))$ if there exist $c, x_0 > 0$ such that $f(x) \geq c \cdot g(x)$ for all $x \geq x_0$



Big (Θ)

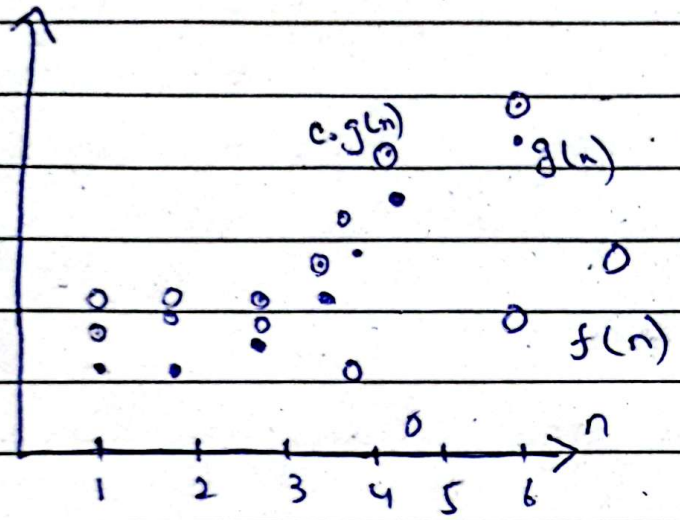
$f(x) \in \Theta\, g(x)$ if there exist $c_1, c_2, x_0 > 0$ such that $c_1 \cdot g(x) \geq f(x) \geq c_2 \cdot g(x)$ for all $x \geq x_0$



Page No. Teacher's Signature

Date

for the better understanding of graphs we will assign integer values to the function, so instead of curve lines, we will use ~~integers~~ single dot lines. i)



so this how the integer values work for the algorithm speed bound, this is general representation of it ii)

iii)

Question : 08

i) $T(n) = \sqrt{2}\, T(n/2) + \log n$

Applying master theorem

$a = 2^{1/2}$, $b = 2^0$, $f(n) = \log n$, $d = 0$

Since

$a > b^d$

hence

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

$= \Theta(n^{\log_2 \sqrt{2}})$

$= \Theta(n^{1/2})$

$= \Theta(\sqrt{n})$

ii) $T(n) = 64\, T(n/8) - n^2 \log n$

It cannot be computed since $f(n) < 0$

iii) $16T(n/4) + n!$

$a = 16$, $b = 4$, $f(n) = n!$

since it doesn't have upper bound we cannot apply master theorem in this equation.

Date:

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

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

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

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

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

$= 2^K T\left(\frac{n}{2^K}\right) + n\log\frac{n}{2^{K-1}} + n\log\frac{n}{2^{K-2}} \ldots + n\log n$

$= n + n\left[\log\frac{n}{2^{K-1}} + \log\frac{n}{2^{K-2}} + \ldots\ldots \log n\right]$ $\quad \because \frac{n}{2^K} = 1$

$= n + n\left[\log 2 + \log 4 + \ldots \log n\right]$ $\quad \because n = 2^K$

$= n + n\left[\log 2^1 + \log 2^2 + \log 2^3 + \ldots\ldots \log n\right]$ $\quad \because K = \log n$

$= n + n\left[1\log 2 + 2\log 2 + 3\log 2 \ldots\ldots \log n\right]$

$= n + n\left[1 + 2 + 3 + \ldots\ldots \log\frac{n}{2} + \log n\right]$

$= n + n\left[1 + 2 + 3 + \ldots\ldots \log n - \log 2 + \log n\right]$

$= n + n\left[1 + 2 + 3 + \ldots\ldots \log n - 1 + \log n\right]$

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

$= n + n(\log n \times \log n + \log n)$

$= n + n(\log n)^2 + n\log n$ $= O(n(\log^2 n))$

$\log\frac{n}{2^{K-1}}$ $\quad \frac{n}{2^K}$ $\quad \frac{n}{2}$

$\because \frac{2n}{2^K}$ $\quad \frac{2^K}{K}$

$\frac{n}{2^{K-2}}$ $\quad \frac{n}{2^K} = \frac{4^K}{K}$ $\quad \frac{4^K}{K}$

$\frac{n}{4}$

$\log m^n = n\log m$

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

$T\left(\frac{n}{2^1}\right) = 2T\left(\frac{n}{2^2}\right) + \frac{n}{2^1}\log\frac{n}{2^1}$

Date:

A) $T(n) = 8T(n/2) + n^2$; $T(n/2) = 8T\left(\frac{n}{2^2}\right) + \left(\frac{n^2}{2^2}\right)$

$= 8\left[8T\left(\frac{n}{2^2}\right) + \frac{n^2}{2^2}\right] + n^2$

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

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

$= 8^3 T\left(\frac{n}{2^3}\right) + \frac{4n^2}{2} + 2n^2 + n^2$; $T\left(\frac{n}{2^3}\right) = 8T\left(\frac{n}{2^4}\right) + \left(\frac{n}{2^3}\right)^2$

$= 8^3\left[8T\left(\frac{n}{2^4}\right) + \left(\frac{n}{2^3}\right)^2\right] + 4n^2 + 2n^2 + n^2$

$= 8^4 T\left(\frac{n}{2^4}\right) + 8n^2 + 4n^2 + 2n^2 + n^2$

$= 8^4 T\left(\frac{n}{2^4}\right) + 2^3 n^2 + 2^2 n^2 + 2^1 n^2 + 2^0 n^2$

$= 8^K T\left(\frac{n}{2^K}\right) + 2^{K-1} n^2 + 2^{K-2} n^2 + 2^{K-3} n^2 \ldots\ldots 2^1 n^2 + 2^0 n^2$

$= (2^K)^3 + n^2\left[2^0 + 2^1 + 2^2 \ldots\ldots 2^{K-1}\right]$ $\quad \because \frac{n}{2^K} = 1$

$a = 1, r = 2, n = K.$ $\quad S = a\frac{(r^n - 1)}{r-1}$ $\quad n = 2^K$

$n^3 + n^2\left[\frac{(2^K - 1)}{1}\right]$ $\quad K = \log_2 n$

$n^3 + n^2\left[2^{K} - 1\right]$

$n^3 + n^2[n-1]$

$n^3 + n^3 - n^2$

$= O(n^3)$

Date

Question 10.

i) for all positive $f(n)$ T

$\omega(f(n)) = n^2$

$O(g(n)) = n!$

$n^2 + n! = O(n^2)$

It is true

ii) For all positive $f(n)$, $f(n) + O(f(n)) = O(f(n))$ F

Sol.

$$O(f(n)) = \lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$$

$1 \neq \infty$

It is false

iii) for all positive $f(n)$, $g(n)$ and $h(n)$. T

$f(n) = n$, $g(n) = n^2$, $h(n) = n!$

$f(n) = O(g(n))$ and $f(n) = \Omega(h(n))$

Let $g(n) + h(n) \Rightarrow n^2 + n! \Rightarrow \Omega(n)$

It is true