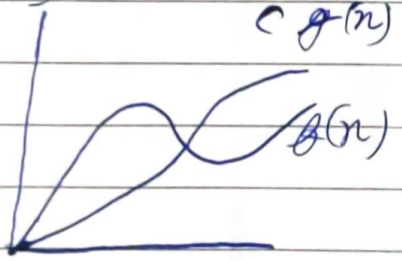
TOTORIAL - 1

Q1 What do you understand by asymptotic notation, define different asymptotic notation with examples.

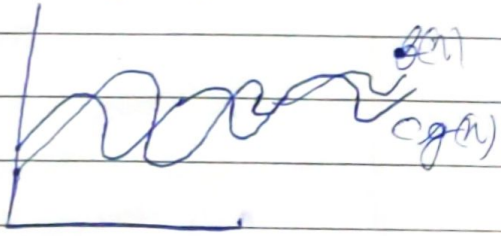
Ans1 Asymptotic notation means towards infinity. They are used to tell the complexity of an algorithm.
Having Input size very large.
It is priority analysis.
Different types of asymptotic notations are:

① Big Oh notations:
$f(n) = O(g(n))$
if $0 \leq f(n) \leq c(g(n))$ $\forall$ $n > n_0$
$g(n)$ is tight upper bound of $f(n)$.

Example:
```
for (int i=0; i<n; i++)
{
    cout << i << endl;
}
```
$T(n) = O(n)$



② Small Oh notation:
$f(n) = O(g(n))$, if $f(n) < c(g(n))$
$\forall$ $n > n_0$ $\forall c > 0$
$g(n)$ is upper bound of $f(n)$



③ Big Omega (Ω)

$f(n) = \Omega\, g(n)$ if $f(n) = C(g(n)) \cdot \forall n \geq n_0$

Some constant $C > 0$

$g(n)$ is tight lower bound of $g(n)$.

Example $f(n) = 6n^2 + n + 1$

$0 \leq g(n) \leq f(n)$

$0 \leq C \cdot g(n) \leq f(n)$

$C \leq 6 + \frac{1}{n} + \frac{1}{n^2}$ on putting $n = \infty, \frac{1}{n} = \frac{1}{\infty}$

$C \leq 6 \quad 6 \leq 6 + 1 + < 6 + 0$ True

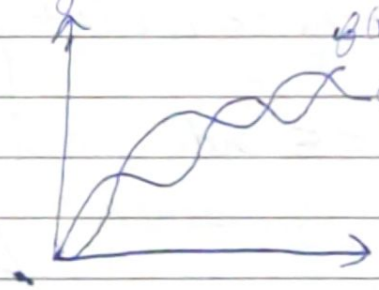
$C > 0$ and $n > n_0$ $(n = 1)$

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

④ Small Omega (ω)

$f(n) = \omega(g(n))$, if $f(n) > C(g(n)) \forall n > n_0$

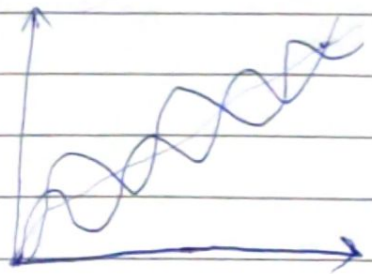
$g(n)$ is the lower bound of $f(n)$



⑤ Theta (Θ)

$f(n) = \Theta(g(n))$, if $C_1(g(n)) \leq f(n) \leq C_2(g(n))$

$\forall n \geq \max(n_1, n_2)$ and some constant $C_1, C_2 > 0$.



Q2 What should be time complexity of
for (i=1 to n) (i = i*2)

Ans 2 i would have 1, 2, 4, 8, 16, . . . n
it say there are k terms
It is a G.P with a=1, r=2
k^{th} term = $t_k = ar^{k-1}$
$n = 1(2)^{k-1}$
$= 2^{k-1}$

taking $\log_2$ on b/s
$\log_2 n = \log_2(2^{k-1})$
$\log_2 n = (k-1)\log_2 2$
$\log n = (k-1) = k = 1 + \log_2 n$
$T(n) = O(k) = O(1 + \log n) = O(\log n)$

Q3 $T(n) = [3T(n-1)$ if $n>0$, otherwise

$T(n) = 3T(n-1)$ — ①
By backward substitution
$T(n) = 3T(n-1)$
$T(n-1) = 3T(n-1-1)$
$T(n-1) = 3T(n-2)$ — ②
Put ② in ①
$T(n) = 3[3T(n-2)] \Rightarrow T(n) = 9T(n-2)$ — ③
$T(n-2) = 3T(n-3)$
$T(n) = 27T(n-3)$
Continue for k times
$T(n) = 3^k T(n-k)$
Assume $n-k = 0 \Rightarrow n = k$
$T(n) = 3^k T(0)$
$T(n) = 3^k$
$T(n) = O(3^n)$

Q 4 $T(n) = \{ 2T(n-1) - 1$ if $n > 0$, otherwise $1 \}$

Ans 4 $T(n) = 2T(n-1) - 1 \quad -①$

by using backward substitution

$T(n) = 2[2T(n-2) - 1] - 1$

$2^2 T(n-2) - 2 - 1$

$T(n) = 2T(n-1) - 1$

$T(n-1) = 2T(n-2) - 1$

$T(n-2) = 2T(n-3) - 1$

$2^2 [2T(n-3) - 1] - 2 - 1$

$2^3 T(n-3) - 4 - 2 - 1$

Continue for K times

$T(n) = 2^K T(n-K) - 2^{K-1} - 2^{K-2} \ldots 1$

Assume $n - K = 0 \Rightarrow n = K$

$2^n T(0) - 2^{n-1} - 2^{n-2} \ldots 1$

$2^n - 2^{n-1} - 2^{n-2} \ldots 1$

$2^n - [2^{n-1} + 2^{n-2} \ldots + 1]$

GP K terms

$a = 2^{n-1}, \quad r = 2^{-1} = \frac{1}{2}$

Sum of G.P

$\frac{a(1 - r^{n-1})}{1 - r} = \frac{2^{n-1}(1 - (1/2)^{n-1}/1/2)}{1/2}$

$= 2^n (1 - 2(1/2)^n)$

$= \frac{2^n (2^n - 2)}{2^n} = 2^n - 2$

$T(n) = O(1)$

Q5 What should be the complexity of

```
int i=1, s=1
while (s<=n) {
    i++
    s=s+i;
    printf ("#");
}
```

i	s
1	1
2	3
3	6
4	10
5	15
k	k times

Ans5 $s = 1, 3, 6, 10, 15 \ldots n$

Let say k terms

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

$k = 2n$ $\quad k = \sqrt{n}$ $\quad O(\sqrt{n})$

t_{k-1} would be constant

Q6 Time complexity of

```
Void function (int n){
    int i, count=0
    for (i=1; i*i<n; i++)
        count ++;
}
```

Ans6 $1^2, 2^2, 3^2, \ldots n$

let say k terms

$t_k = k^2$

$n = k^2 \Rightarrow k = \sqrt{n}$

$T(n) = O(\sqrt{n})$

Q7 Time complexity of

```
Void function (int n) {
    int i, j, k, count =0
    for (int i=n/2; i<=n; i=(i+2))
        for (k=1; k<=n; k=k*2)
            count ++;
}
```

Ans 7 $i = \frac{n}{2}, \frac{n}{2}+1, \frac{n}{2}+2 \cdots n$

$= \frac{n}{2}, \frac{n+2}{2}, \frac{n+4}{2} \cdots n$

given of form $= \frac{n-10*2}{2}, \frac{n+1*2}{2}, \frac{n+2*2}{2} \cdots n$

$\frac{n+k*2}{2}$ $\quad (k=0, 1, 2, n)$

Total terms $= k+1$

$t_{k+1} = n$

$\frac{n+(k+1)*2}{2} = n \Rightarrow 2n = n+(k+1)*2$

$n-2 = 2k$

$k = \frac{n}{2} - 1$

i	j	k
$n/2$	$\log n$ time	$(\log n)^2$
$\frac{n+2}{2}$	$\log n$ time	$(\log n)^2$
n	$\log n$ time	$(\log n)^2$

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

$\frac{n}{2} \log^2 n = \log^2 n$

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

Q8 Time complexity of

```
function (int n) {
    if (n == 1) return;
    for (i = 1 to n) {
        for (j = 1 to n)
```

```
}
}
printf("S");
function(n-3);
```

Ans8 Function call would be $n, n-3, n-6, n-9 \cdots$

Let says k terms

AP, $a = n$, $d = -3$

$a_n = a + (n-1)d$

$1 = n + (k-1)(-3)$

$1 = n - 3k + 3$

$3k = n + 2$

$k = \frac{n+2}{2}$

Function have recursive call $\frac{n+2}{2}$ times.

Time complexity for two for loop.

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

$T(n) = O(n^3)$

Q9 Time complexity of void function (int n) {

```
for (i = 1 to n)
    for (j = 1; j <= n; j = j + 1)
        print("S");
    }
}
```

Ans9 i (Outer loop)

when $i = 1 \rightarrow j = 1, 2, 3 \cdots n = n$

while $i = 2 \Rightarrow j = 1, 2, 3 \cdots n/2$

$$\sum_{j=n}^{n} n + \frac{n}{2} + \frac{n}{3} + \cdots 1$$

$$\sum_{j=n}^{n} n\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right)$$

$$O(n \log n)$$

Q/o For the function, n^n K ac C^{n^n} what is the asymptotic relationship b/w these function. Assume that $K \geq 1$ and $C > 1$ are constants. Find out the value of C and no for which relation hold.

Ans/o As given n^k and c^n

relation b/w n^k and c^n is $n^k = O(c^n)$

as $n^k \leq a c^n$ $\forall n \geq n_0$ for a constant $a > 0$

for $n = 1$

$c = 2$

$1^k < a 2^1$

$n_0 = 1$ and $c = 2$

TUTORIAL -2

Q1 What is the time complexity of below

```
Void fun (int n)
    int j=1; i=0
    while (i<n) {
        i=i+1
        j++;
    }
```

Ans1 i=0, 1, 3, 6, 10 let say K times

So given form would be

$$\frac{K(K+1)}{2}$$

K^{th} term $h = \frac{K(K+1)}{2} = n$

$$K^2 + K = 2n$$

$$K^2 = n$$

$$K = \sqrt{n}$$

Time complexity $h = O(\sqrt{n})$

Q2 Write recurrence relation for the recursion function that prints fibonacci series. Solve the recurrence relation to get the time complexity of this program and why.

Ans2

```
int fib (int n) {
    int (n<=1) <-> O(1) = i
    return n;
return fib(n-1) + fib(n-2) -> T(n-1) + T(n-2)
}
```

Recurrence relation $T(n) = T(n-1) + T(n-2) + c$

Now $T(n-1) \simeq T(n-2)$

$T_n = 2T(n-1) + c$

By backward substitution

$T(n-1) = 2T(n-1-1) + c \Rightarrow 2T(n-2) + c$

$T(n) = 2[2T(n-2) + c] + c$

$= 4T(n-2) + 3c$

Now $T(n-2) = 2T(n-2-1) + c$

$= 2T(n-3) + c$

$T(n) = 4T(n-2) + 3c$

$= 4(2T(n-3) + c) + 3c$

$T(n) = 8T(n-3) + 7c$

generalizing: $2^k T(n-k) + (2^k - 1)c$

assume $n - k = 0 \Rightarrow n = k$

$= 2^n + (0) + (2^n - 1)c$

$= 2^n + (2^n - 1)c$

$= 2^n + (2^n - 1)c$

$= 2^n$

Time complexity $= O(2^n)$

Space complexity:

For fibonaci space required is directly $\propto$ to maximum depth of recursion tree.

Since max. depth is dic to no of elements.

Q3 Write a program which have complexity $n(\log n)$

```
① for (i=1; i<=n; i++) {
      for (j=1; j<=n; j=j*2)
      {  sum = sum + i;
      }
```

② n^3

```
for (i=0; i<n; i++) {
    for (j=0; j<n; j++)
    for (k=0; k<=n; k++)
        Sum = Sum + k;
    }
}
```

③ $\log n(\log n)$

```
for (i=1; i<n; i=(i*2)
    for (k=1; k=h; k=k*2)
        Sum = Sum + j;
    }
}
```

Q4 Solve the recurrence relation

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

Ans4 $T(n/4) = T(n/2)$

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

$a \geq 1$ and $b \geq 1$

By using master's method

$T(n) = T(n/b) + f(n)$

$L = \log_2 a \Rightarrow 1$

$f(n) > n^c \Rightarrow cn^2 > n^c$

$T(n) = O(f(n))$

$O(n^2)$

Q5 What is the time complexity of following from ().

```
int fun (int n)
    for (int i=1; i<=n; i++) {
        for (int j=1; j<n; j+=6) {
            Some O(1) task
        }
    }
```

Ans5 for $i=1 \Rightarrow 1+2+3+\cdots(n+1)=n$

for $i=2 \Rightarrow 1+3+5+\cdots n \Rightarrow n/2$

for $i=3 \Rightarrow 1+4+7+\cdots n \Rightarrow n/3$

$$n+\frac{n}{2}+\frac{n}{3}+\cdots+)$$

$$\Rightarrow n\left(1+\frac{1}{2}+\frac{1}{3}+\cdots\frac{1}{n}\right)$$

Now we know

$$n\left(1+\frac{1}{2}+\cdots\frac{1}{n}\right) \leq n\left(1+\frac{1}{2}+\frac{1}{3}+\cdots\frac{1}{n}\right)$$

$$n\left(1+\frac{1}{2}+\frac{1}{3}+\cdots\frac{1}{n}\right) \leq n(1+0.5\cdots)$$

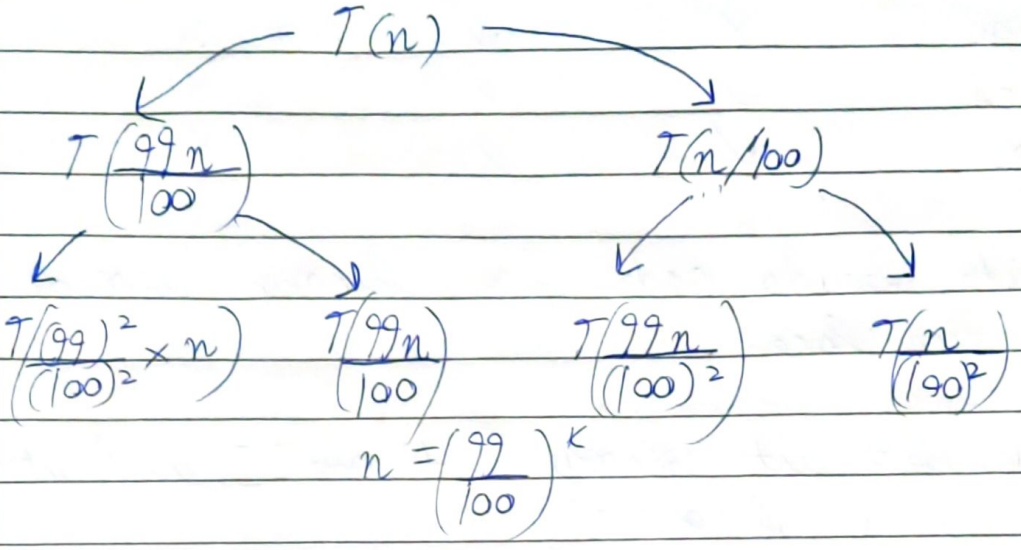
$O(n \log n)$

Q7 Write a recurrence relation color quick sort repeatedly divides the array to show the recurrence the time complexity the difference. in light of both the extreme cases what do you understand by the analysis.

Ans7 49 to 1 in quick sort

when pivot is close from front or end always.

$$T(n) = T(99/100) + T(n/100) + O(n)$$



$$n = \left(\frac{99}{100}\right)^k$$

$$\log n = k \log \frac{99}{100}$$

$$K = \log n \frac{100}{99}$$

Time complexity = $n \log$

Q8 Arrange the following in increasing order of rate of growth

Ans 8 $n!, n!, \log n, \log \log n, \sqrt{n}, \log(n!)$
$n \log n, 2^n, 2^{2n}, 4^n, n^2, 100$

Ans 8 $100 < \log(\log n) < \log n < \log^2 n < \sqrt{n} < n < \log n!$
$< n \log n < \log^{2n} < n^2 < 2^n < 4^4 < 2^{2n} < n!$

TUTORIAL-3

Q1 Write linear search pseudocode to search an element in a sorted array with min. comparisons.

Ans1
```
Void LinearSearch (int A[], int n, int key){
    int flag = 0;
    for(int i=0; i<n; i++)
        if (A[i] == key ){
            flag = 1;
            break;
    } }
    if (flag == 0)
        cout << "Not found";
    else
    cout << "found";
```

Q2 Write pseudocode for iterative algorithm for insertion sort.

```
for i=1 to n-1)
{   t = A[i]; j = i-1;
    while ( j >= 0 && A[j] > t) {
        A[j+1] = A[j]
        j--;
    }
    A[j+1] = t;
}
```

Recursive

```
Void insertionSort (int arr[], int n){
    if (n <= 1)
        return;
```

Q3 $T(n) = 3T(n/2) + n^2$

$n \log_2 3 = n^{1.5}$

$n^{1.5} > n$

Ans 3

```
insertion sort (arr, n-1)
int last = arr[n-1], j = n-2;
while( j>0 && arr[i] > last)
    arr[j+1] = arr[j]
        j--;
    arr[j+1] = last;
}
```

Insertion sort is called online sorting an input element for iteration and produces a partial solution without require access to offline algorithm.

Q4 Complexity of all sorting algorithm has been discussed.

Algorithm	Best	Average	Worst
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Count sort	$O(n)$	$O(n)$	$O(n)$
Quick Sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Heap Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

Inplace stable instance

Algorithm	Inplace	Stable	Online
Bubble	✓	✓	×
Selection	✓	×	×
Insertion	✓	✓	✓
Count	×	✓	×

Merge	X	✓	X
Quick	✓	X	X
Heap	✓	X	X

Q5 Write pseudo code for binary search the time space.

Ans Recursive = int binary (int arr[], int l, int x, int key)

```
{ if (x >= l)
  { int mid = l + (x - l)/2;
    if (arr[mid] == key)
        return mid;
    if (arr[mid] > key)
        return binary (arr, mid - 1, key)
    else
    return (arr, mid+1, x, key)
  }
return -1;
}
```

```
int binary search (int arr, int l, int x, int key)
{ while (l <= x) {
    int m = l + (x - l)/2;
    if (arr[m] == key);
        return m;
  if (arr[m] < key)
      l = m+1;
  else
    x = m - 1;
  }
return -1;
}
```

Q7 Find two indices such that A[i]+A[j]=k in minimum.

Ans7.
```
Void Sum (int A[], int K, int n)
    {
      Sort (A, A+n)
      int i=0 ; j=n-1 ;
    while (i<j)
     if (A[i]+A[j]==k)
       break;
     else if (A[i]+A[j]>k)
          j--;
     printf (i, j) ;
```

Q8 Which sorting best for practical uses? Explain.

Ans8 For practical uses, it would be best for very large data. Further, time complexity of merge sort is same in all cases, that is O(n(log n)).

TUTORIAL - 4

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

$T(n) = aT\left(\frac{n}{b}\right) + F(n)$

$a \geq 1, \quad b > 1$

Comparing

$a = 3, \quad b = 2, \quad f(n) = n^2$

Now

$c = \log_b a = \log_2 3 = 1.584$

$n^c \leq n^{1.584} < n^2$

$F(n) > n^c$

② $T(n) = 4T(n/2) + n^2$

$a > 1, \quad b > 1$

$a = 4, \quad b = 2, \quad f(n) = n^2$

$T(n) = \Theta(n^2 \log_2 n)$

$c = \log_2 4 = 2$

$n^c = n^2 = f(n) = n^2$

$T(n) = \Theta(n^2 \log_2 n)$

③ $T(n) = T(n/2) + 2^n$

let $a = 1, \quad b = 2$

$f(n) = 2^n$

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

$n^c = n^0 = 1$

$f(n) > n^c$

$T(n) = \Theta(2^n)$

④ $T(n) = 2^n T(n/2) + n^n$

$a = 2^n$

$b = 2$, $f(n) = n^n$

$c = \log_b a = \log_2 2^n$

⑤ $T(n) = 16T(n/4) + n$

$a = 16$, $b = 4$

$F(n) = n$

$c = \log_n 16 = \log_n (4)^2 = 2$

$n^c = n^2$

$F(n) < n^c$

$T(n) = \Theta(n^2)$

⑥ $T(n) = 2T(n/2) + n \log n$

$a = 2$, $b = 2$

$F(n) = n \log n$

$c = \log_2 = 1$

$n^c = n^1 = n$

Since, $n \log n > n$

$F(n) > nc$

$T(n) = \Theta(n \log n)$

⑦ $T(n) = 2T(n/2) + n/\log n$

$a = 2$, $b = 2$, $F(n) = n/\log n$

$c = \log_2 2 = 1$

$n^c = n^1 = n$

Since $\frac{n}{\log n} < n$

$F(n) < n^c$

$T(n) = \Theta(n)$

Date
Page

⑦ $T(n) = 2T\left(\frac{n}{2}\right) + n/\log n$

$a=2, b=2, f(n) = n/\log n$

$c = \log_2 2 = 1$

$n^c = n = n$

Since $\frac{n}{\log n} < n$

$F(n) < n^c$

$T(n) = \Theta(n)$

⑨ $T(n) = 0.5T(n/2) + 1/n$

$a = 0.5, b = 2$

Since here to master theorem $a \geq 1$ but here a is 0.5 so we cannot apply master theorem.

⑪ $T(n) = 4T(n/2) + \log n$

$a=4, b=2, f(n) = \log(n)$

$c = \log_b a = \log_2 4 = 2$

$n^c = n^2$

$F(n) = \log(n)$

$\log n < n^2$

$f(n) < n^2$

$T(n) = \Theta(n^2)$

$\Theta(n^2)$.

⑬ $T(n) = 3T(n/2) + n$

$a=3, b=2, f(n) = n$

$c = \log_b a = \log_2 3 = 1.53$

$n^c = n^{1.5849}$

$n < n^{1.5849}$

$f(n) < n^c$ $\quad T(n) = \Theta(n)^{1.58}$

(15) $T(n) = 4T(n/2) + n$

$a = 4, b = 2$

$c = \log_b a = \log_2 4 = 2$

$n^c = n$

$n < n^2$ (for any constant)

$f(n) < n^c$

$T(n) = \Theta(n^2)$

(17) $T(n) = 3T(n/3) + n/2$

$a = 3, \quad b = 3$

$c = \log_b a = \log_3 3 = 1$

$f(n) = n/2$

$n^c = n^1 = n$

As $n/2 < n$

$f(n) < n^c$

$T(n) = \Theta(n)$

(19) $T(n) = 4T(n/2) + n \log n$

$a = 4, b = 2, f(n) = n \log n$

$c = \log_b a = \log_2 4 = 2$

$n^c = n^2$

$\frac{n}{\log n} < n^2$

$T(n) = \Theta(n^2)$

(21) $T(n) = 7T(n/3) + n^2$

$a = 7, b = 3, f(n) = n^2$

$c = \log_b a = \log_3 7 = 1.7$

$n^c = n^{1.771}$

$n^{1.77} < n^2$

$T(n) = \Theta(n^2)$

(22) $T(n) = T(n/2) + n(2 - \cos n)$

$a = 1, \quad b = 2$

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

$n^c = n^0 = 1$

$n(2 - \cos n) > n^c$

$T(n) = \Theta(n(2 - \cos n))$

TUTORIAL -5

Q1 Differentiate b/w BFS & DFS

Ans1

BFS	DFS
• Stands for Breadth first search.	Stands for Depth first search.
• BFS uses queue to find the shortest path.	It uses stack to find shortest path.
• BFS is better when target is closer to source.	DFS is better when target is far from source.
• As BFS considers all neighbours so it is not suitable for decision.	DFS is more suitable for decision tree.
• BFS is slower than DFS.	DFS is faster than BFS.

Application of DFS:
- Using DFS we can find path between two vertices.
- We can which is used to scheduling jobs.
- We can use DFS to detect cycles.

Application of BFS:
- BFS may also used to detect cycles.
- Like Finding shortest path and minimal spanning tree.
- In network finding a route for packet transmission.

Ans 2

BFS used Queue data structure. BFS you mark any node in graph as source node, transfer all the nodes in graph and keeps as completed.
BFS visited an adjacent unvisited node, marks it as done and insert it into Queue.
DFS uses stack a graph in a depth wise, motion and uses stack to remember to get the next vertex to start to a search, when a dead an occurs in any iteration.

Ans 3

Sparse graph: A graph in which the number of edges number of edge.

Dense graph: A graph which the number of edges the maximal no of edges.
If the graph is sparse, we should store it as list of edges.

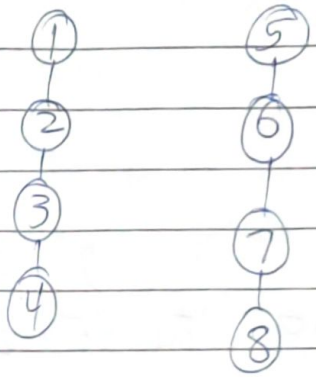
Ans 4

DFS can be used to detect cycle in a graph DFS. DFS for connected graph produces a tree. There is a cycle in graph. Only if graph. A backedge is an edge that is from a node to itself or out of its ancestor in the tree produces by DFS.

BFS can also be used cycles. Perform BFS while keeping a list of previous nodes at each node. Visited or else that is already marked by BFS. I founded a cycles.

Ans 5

Disjoint set Data structure:
- It allows to find out whether the two elements are in the same set or not efficiently.



$S_1 = \{1, 2, 3, 4\}$
$S_2 = \{5, 6, 7, 8\}$

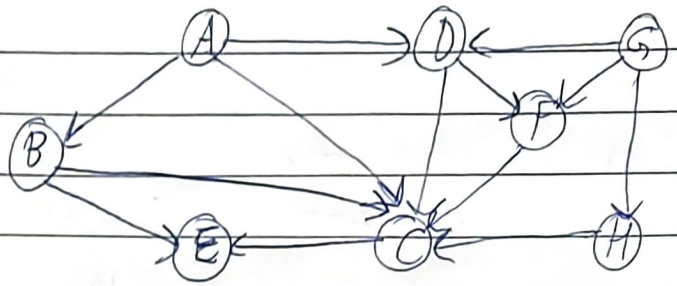
Operation performed:

① find int find (v)
```
if (v == parent[v])
    return v;
return parent[v] = find (parent[v]);
```

Union:
```
Void union (int a, int b) {
    a = find (a)
    b = find (b)
    if (a != b)
    if (size[a] < size[b])
    { Swap (a, b) }
    parent[b] = a;
```

```
Size [a] += Size [b];
}
}
```

Q6



Ans 6 BFS: Node : B E C A D E

B B E A D

Path: B → F → A → D → F

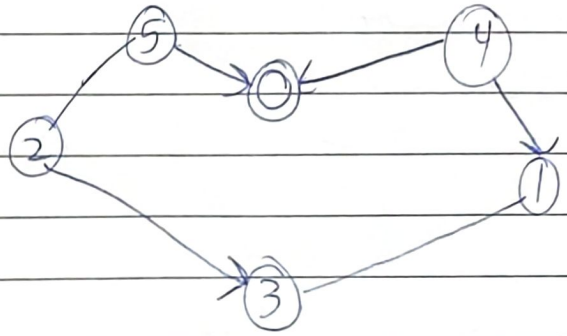
DFS:

Node: B B C E A D F

Stack: B CE EE AE DE FE E

Path B → C → E → A → D → F

Q8



V → Visited

f → false

0	1	2	3	4	5
F	F	F	F	F	F

Stack (Empty)

Ans 8 Adjacency list:

0 →

1 →

2 → 3

3 → 1

5 → 2 → 0 4 → 0 → 1

Step 1: Topological Sort[0], visited[0]=true
Stack [0]

Step 2: Topological Sort[1], visited[1]=true
Stack [0|1]

Step 3: Topological Sort(2), Visited[2]=true;
Topological Sort(3), Visited[3]=true;
Stack [0|1|3|2]

Step 4:
Stack [0|1|3|2|4]

Step 5:
Stack [0|1|3|2|4|5]

Step 6: Print all elements of stack from top to bottom
→ 5, 4, 2, 3, 1, 0

Ans 10

Min heap	Max heap
- In min heap the key present at root node must be less than or equal to among the keys present all its children.	In max heap the key present at root node must be greater or equal to the key present at all its children.
- Used to ascending priority.	Uses descending priority.

- The minimum key present at the root node.

The max- key present at the root node.

TUTORIAL-6

Q1 What do you mean by Minimum spanning tree what is the application of MST.

Ans 1 A minimum spanning tree or minimum weight spanning tree is a subset of the edges of a connected spanning tree is a subset of the edges of a connected, edge weighted undirected graph, all the vertices together, without any cycle and with the minimum possible total edge weight.

Application:

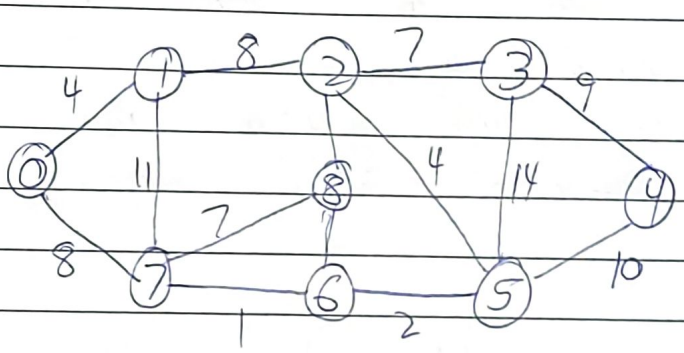
- Defining local Area Networks.
- Suppose you want to construct highway or rail road, spanning several cities, then we use the concept of MST to connect.
- To reduce cost, you use the concept of MST to connect the houses.

Q2 Please analyse the time and space complexity of prims, Kruskal, dijkstra and and Bellman food algo.

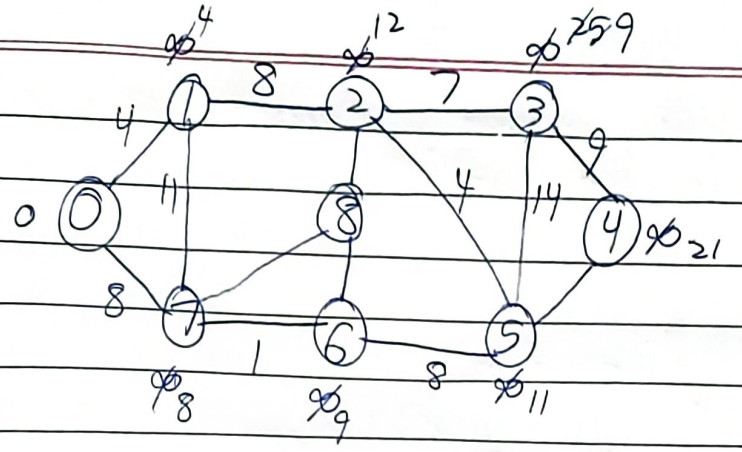
Ans 2

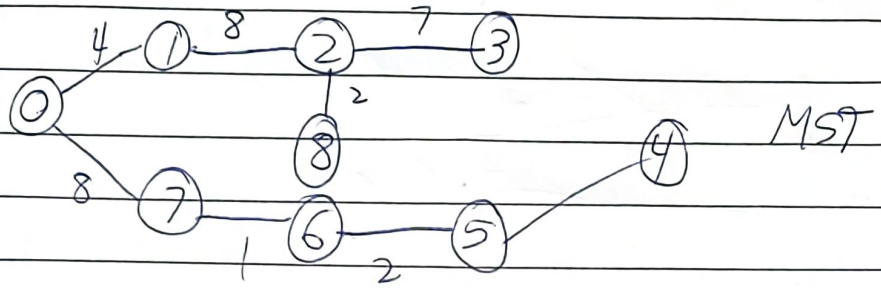
Algorithm	Time complexity	Space complexity
Prims	$O(V^2)$	$O(V+E)$
Kruskal	$O(E \log V)$	$O(\log(E))$
Dijkstra	$O(V+E)$	$O(V+E)$
Bellman	$O(V)$	$O(V)$

Q3 Apply Prims and kruskal algorithm on the graph to compute MST and its weight.



Kruskal Path	Weight
7 → 6	1
6 → 5	2
2 → 8	2
0 → 1	4
2 → 5	4
8 → 6	6
2 → 3	7
7 → 8	7
1 → 2	8
3 → 4	9
5 → 4	10
1 → 7	11
3 → 5	14





Q4 Given a weighted graph. You are also given the shortest path from a source vertex is to a given destination vertex 'F'. Does the shortest path remain same in the modified graph.

Ans/ (1) The shortest path may change. The reason is that no of edges in different paths from 'S' to 't' for example: let shortest path of weight 15 and has 5 edges, let there be another path with 2 edges and total weight is 25. The weight of the shortest is increased by 5*10 becomes 15+50, weight of other path is increased by 2*10, it becomes 25+20. So the shortest path changes to the other path whose weight is 45.

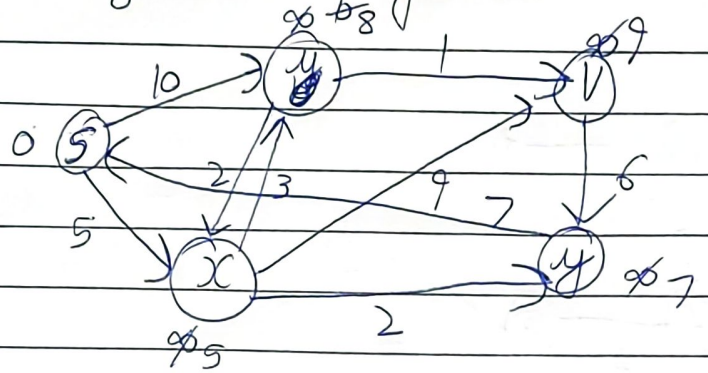
(2) If we multiply all edges weighted by 10, the shortest path doesn't change.

The reason is simple weight of all paths from s to t.

Q5

Ans 5

Dijkstra's Algorithm



node	Shortest distance from Source node
u	
x	
v	
y	

Bellman ford Algorithm