Name → Mond Nasir
Section → F
Roll No → 14

Student Id → 20021595
University Roll → 2016855

DESIGN AND ANALYSIS OF ALGORITHM
TUTORIAL-01.

Ques→1 Asymptotic Notation :→ Asymptotic notations are the mathematical notations used to describe the complexity (runing time of an algorithm. Different types of asymptotic Notation.

(i) Big-O :→ Big O notation specifically describes worst case scenario. It represents the tight upper bound running time complexity of an algorithm.

$$f(n) \leq c \cdot g(n). \quad \forall \; n \geq n_0, \; c \geq 0$$

Ex :→ O(1), O(n), O(nlogn).

```
for (i=0; i<n; i++)
    sum = sum + i;
```

The time complexity of above example is O(n).

(ii) Omega (Ω) :- omega Notation specifically describe best case scenario. It represents the tight lower bound running time of an algorithm.

$$f(n) \geq c \cdot g(n) \quad \forall \; n \geq n_0, \; c \geq 0$$

Example :- $\Omega(1)$, $\Omega(\log n)$.

```
void search (int size) {
    int element, i;
    if (& size == 1) {
        Printf ("found");
        return;
    }.
    for (i=0; i< size; i++)
    {
        if (arr [i] = element) {
            Printf ("found");
            return;
        }
    }
    Printf ("Not found");
}.
```

The Best Complexity of above Example is :- $\Omega(1)$.

(iii) Theta (Θ) :- This Notation describes both tight upper bound and tight lower bound of an algorithm. In real scenario the algorithm not always run on best and worst cases, the avg running lies between best and worst and can be represent by 'Θ' Notation.

$$c_1 g(n) \leq f(n) \leq c_2 g(n). \qquad \forall\ n \geq \max(n_1, n_2)$$
$$c_1 \geq 0 \text{ and } c_2 \geq 0$$

Ques. 10:- For the function $n^{-1}R$ and c^n, what is the asymptotic Relationship b/w these function ?

assume that $k >= 1$ of $C > 1$ are constants find out the value of C. and no. of which relationship hold.?

↳ as given n^k and c^n

Relationship between n^k and c^n is.

$$n^k = O(c^n)$$
$$n^k \leq a(c^n)$$
$$\forall \quad n \geq n_0 \quad \text{constant} \quad a > 0$$

for $n_0 = 1$: $C = 2$

$1^k < a^2$

$\Rightarrow \quad n_0 = 1$ and $C = 2$

Date : Page :
Topic :

Ques→2 For (i = 1 to n)
{ i = i * 2 }
i → 1, 2, 2^2, 2^3, 2^4, ..., 2^K.

$2^K = n$

take $\log_2$ both side.

$\log_2 2^K = \log n$

$K \log_2 2 = \log n$

$\boxed{K = O(\log n)}$

Ques→3 $T(n) = 3T(n-1)$ $n>0$, otherwise 1.

$T(n) = 3T(n-1)$ — (1)

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

From equ (1) :→

$T(n) = 3[3T(n-2)]$

$= 3^2 T(n-2)$ — (2)

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

From equ (2) :→

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

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

⋮

$T(n) = 3^K T(n-K)$. — (3)

$n - K = 0$

$n = K$

STUDENT STYLE

Signature.....

Date _____ Page _____
Topic _____

From equ (3) :-

$T(n) = 3^k T(n-n)$

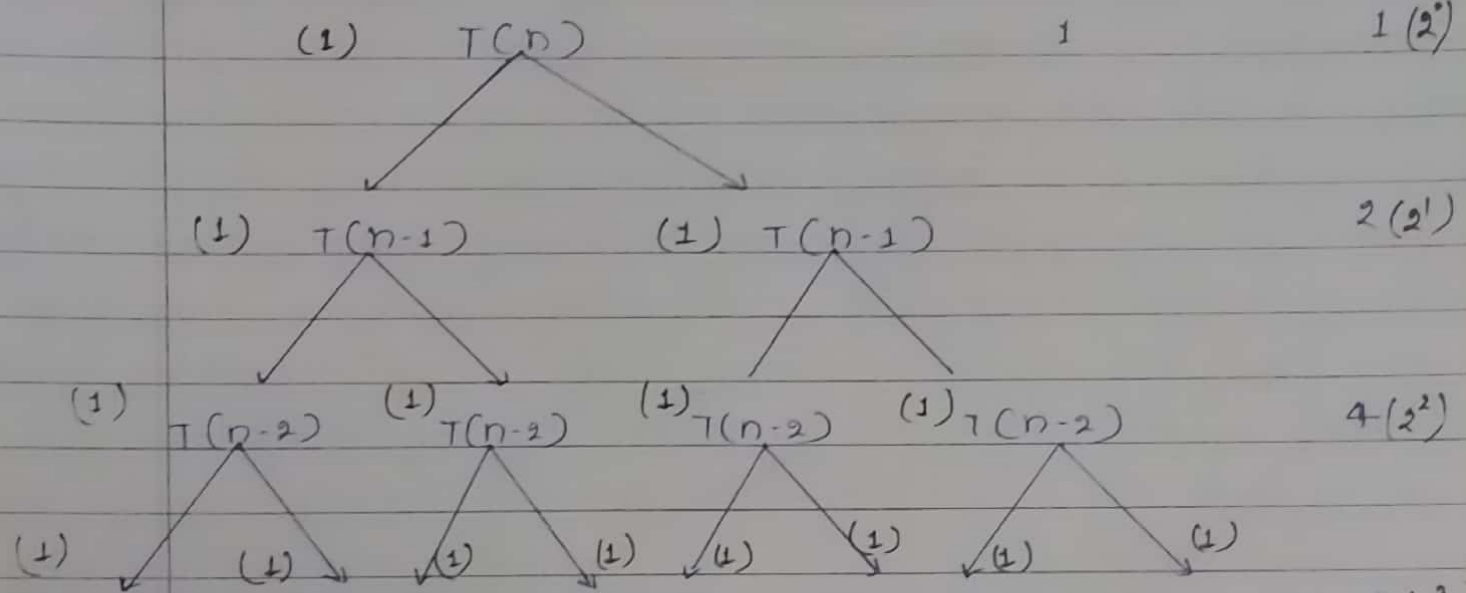
$= 3^k T(0)$

$T(0) = 1.$

$T(n) = 3^k$

$\boxed{T(n) = O(3^k)}$

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



$2^0, 2^1, 2^2, 2^3 \ldots\ldots, 2^k$

$T(n) = T(n-K) * 2^k$

$n - K = 0$

$n = K$

$T(n) = T(0) \cdot 2^k$

$\boxed{T(n) = O(2^n)}$

STUDENT STYLE

Signature _____

Date Page
Topic

Ques→5
```
int i=1, S=1;
while (S<=n){
    i++;
    S=S+i;
    Print+("#");
}.
```

S (Value of S):→

1, 3, 6, 10, ... n

(1) (1+2) (1+2+3) (1+2+3+4) (1+2+3+ ... +K)

Termination Condition.

$(1+2+3+\cdots+K) = n$

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

$K^2 + K = 2n$

$K^2 = n$

$K = \sqrt{n}$

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

Ques→6.
```
void Function (int n){
  int i, count=0;
  for (i=1; i*i<=n; i++)
        count++;
}.
```

i→ 1, 2, 3, ..., K^2

$K^2 = n$ (Terminate the program)

$K = \sqrt{n}$ $K^2 > n$

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

STUDENT STYLE

Signature

Date : Page :
Topic :

Ques-7

```
void fun(int n) {
int i, j, k, count = 0;
for(i = n/2; i <= n; i++) {          loop 1.
    for(j = 1; j <= n; j = j * 2;) {   loop 2
        for(k = 1; k <= n; k = k * 2) {   loop 3
            count++;
        }
}.
```

loop 1 :→

$i \to \frac{n}{2}, \frac{n}{2}+1, \frac{n}{2}+2, \frac{n}{2}+3 \cdots \frac{n}{2}+k$

↳ $k = \frac{n}{2}$

$k > n$ (terminate pro).

$k = n$

$T(n) = O(n)$

For lop 1 and loop 2 we know time complexity.

$T(n) = \log n$.

total time complexity.

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

$$\boxed{T(n) = O(n \log^2 n)}$$

STUDENT STYLE

Signature.....

Date : Page :
Topic :

Ques→8

```
fun (int n){
  if(n==0)
      return;
  for (i=1 to n)
      for (j=1 to n)
        print("#");

  fun (n-3)
}.
```

$T(n) = (n-3) + O(n^2)$

$= T(n-3) + n^2$ —— (1)

$T(n-3) = T(n-6) + (n-3)^2$

From equ (1) :→

$T(n) = T(n-6) + (n-3)^2 + n^2$

$T(n-6) = T(n-9) + (n-6)^2$

$T(n) = T(n-9) + (n-6)^2 + (n-3)^2 + n^2$

$T(n) = T(n-3K) + (n-(3K-3))^2 + [n-(3K-6)]^2 + n^2$

$n - 3K = 0$

$n = 3K$

$K = n/3$

$T(n) = T(n-n) + 1^2 + 2^2 + 3^2 \cdots + n^2$

$= T(0) + \frac{n(n+1)(n+2)}{6}$

$T(n) = n * n * n$

$\boxed{T(n) = O(n^3)}$

STUDENT STYLE

Signature.....

Date : Page :
Topic :

Ques 9. Void Fun(int n) {
  For (i=1 to n)
    for (j=1 ; j<=n ; j=j+i)
      printf("*");
}

i=01, j → 1, 2, 3, 4, ... , n
i=2, j → 1, 3, 5, 7, ... , n/2
i=3, j → 1, 4, 7, 19, ... , n/3
⋮
i=n, j → 1, ... , n/n

$$T(n) = n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \cdots + \frac{n}{n}$$

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

→ logn.

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

$$\boxed{T(n) = O(\log n)}$$

STUDENT STYLE

Signature.....

Name → Mohd Nasir
Section → F
Roll No → 14
Student [illegible]
University Roll → 2016855

TUTORIAL-02

Page :

Topic :

DESIGN AND ANALYSIS OF ALGORITHM

Ques 1. Void Fun(int n){
int i=0, j=1;
while (i<n){
i = i+j;
j++;
}.

i → 0, 1, 3, 6, 10, ..., k

i → 0, 1, 3, 6, ..., k

0 1 1+2 1+2+3 1+2+3+4+ ... +k

(1+2+3+...+k) = n (terminate the program)

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

$k^2 + k = 2n$

$k^2 = n$

$k = \sqrt{n}$

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

Ques 2 $T(n) = T(n-1) + T(n-2) + 1$ if $(n>0)$ otherwise 1.

$T(n-1) \approx T(n-2)$

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

$T(n) = 2T(n-1) + 1$ —— (1)

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

From equ (1) :→

STUDENT STYLE

Signature

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

$= 2^2T(n-2)+2$ — (2)

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

From equ (1):

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

$T(n) = 2^kT(n-k)+k$ — (3)

$n-k=0$

$n=k$

From equ (3):

$T(n) = 2^nT(n-n)+n$

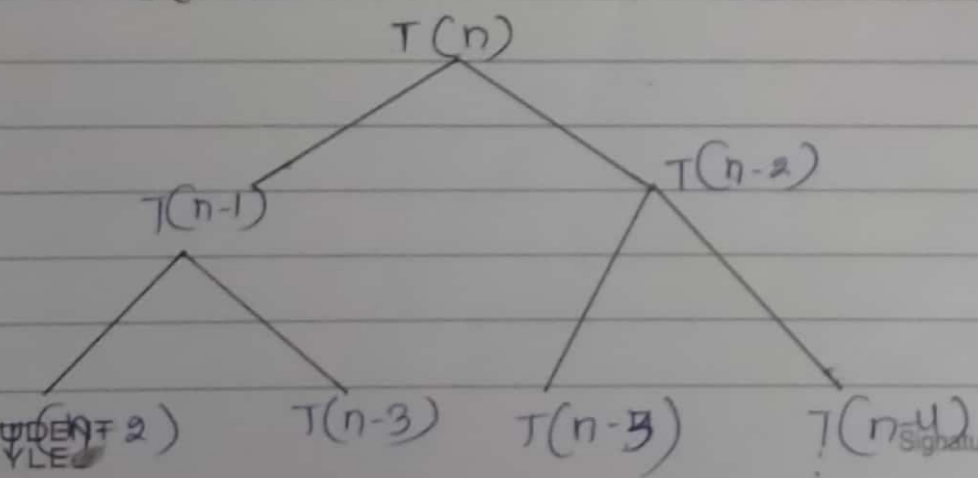
$= 2^nT(0)+n$

$T(n) = 2^n \times 1+n$

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

$\boxed{T(n) = O(2^n)}$

Space Complexity → If we draw the recursion tree of the fibonacci recursion then we found the maximum height of tree will be n and hence the space complexity of fibonacci recursion will be O(n).



Date Page
Topic

Ques→3 $O(n \log n)$, $O(n^3)$, $O(\log(\log n))$

(i) $n \log n$.

```
void fun(int a){
  int i, j, sum=0;  → if(n==0) return
  for(i=0; i<a; i=i+2)
      sum = sum + i;
  fun(n-1);
}
```

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

(ii)
```
void fun(int n){
  int i,j; sum=0;   if(n==0) return;
  for(i=0; i<n; i++)
    for(j=0; j<n; j++)
      sum = i+j;
  fun(n-1)
}.
```

$T(n) = (n-1) + n^2$ if $n>0$ otherwise 1.

(iii) $\log(\log n)$.

```
for(int i=2; i<=n; i=pow(i,k))
{
    print("i", i);
}.
```

$k \geq 2$

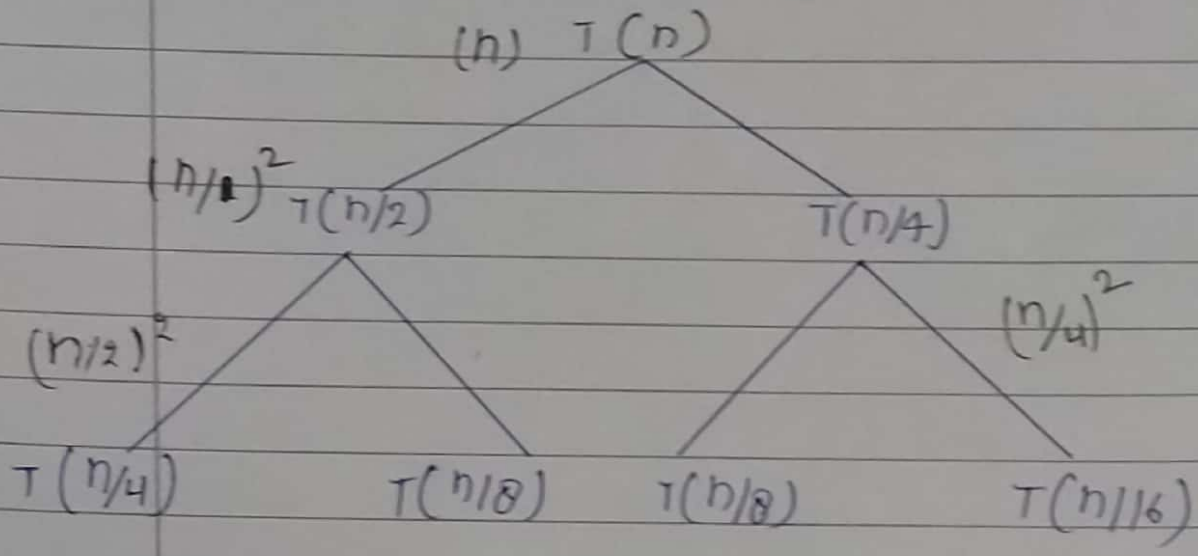
STUDENT STYLE

Signature

Date : _____ Page : _____

Topic : _____

Ques 4 $T(n) = T(n/2) + T(\frac{n}{4}) + n^2$



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

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

$\rightarrow 1$

$$T(n) = n^2 \times 1$$

$$\boxed{T(n) = O(n^2)}$$

Or

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

$\alpha = \frac{1}{2}$, $\beta = \frac{1}{4}$, $f(n) = n^2$

$\alpha + \beta = \frac{1}{2} + \frac{1}{4} = \frac{3}{4} < 1$

$T(n) = f(n) = \underline{O(n^2)}$

STUDENT STYLE

Signature _____

Date : Page :
Topic :

Ques→5
```
int fun(int n){
    for (int i=1; i<=n; i++){
        for (int j=1; j<n; j=i+j) → j=i+j;
            some O(1) task
```

3.

i→01 j→ 1, 2, 3, 4, 5, ..., n

i→2 j→ 1, 3, 5, 7, ..., n/2

i→3 j→ 1, 4, 7, 11, ..., n/3

⋮

i→n j→ 1 ..... , n/n

$$T(n) = n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \cdots + \frac{n}{n}$$

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

→ logn

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

$$\boxed{T(n) = O(n\log n)}$$

STUDENT STYLE

Signature.....

Date : Page :
Topic :

Ques → 6. for (int i = 2 ; i <= n ; Pow(i, k))
Some O(1) Expression.

$i = 2, 2^k, (2^k)^k, (2^{k^2})^k, (2^{k^3})^k \ldots\ldots, 2^{k^k}$

$2^{k^2} \quad 2^{k^3} \quad 2^{k^4} \quad 2^{k^k}$

$2^{k^k} = n$

$\log_2 2^{k^k} = \log_2 n$

$k^k \log_2 2 = \log n$

$k^k = \log n$

$\log k^k = \log(\log n)$

$k \log_2 k = \log(\log n)$.

$k = O(\log \log n)$.

$$\boxed{T(n) = O[\log(\log n)]}$$

STUDENT STYLE

Signature.....

Date: Page:
Topic:

Ques 8 Arrange the following in increasing order of rate of growth.

(a) $n, n!, \log n, \log\log n, \text{root}(n), \log(n!), n\log n, \log^2 n, 2^n, 2^{2^n}, 4^n, n^2, 100$

Ans $100 < \log\log n < \log n < \log^2 n < \text{root}(n) < n < n\log n < \log(n!) < n^2 < 2^n < 4^n < 2^{2^n} < n!$

(b) $2(2^n), 4n, 2n, 1, \log(n), \log\log(n), \sqrt{\log n}, \log_2 n, 2\log n, n, \log(n!), n!, n^2, n\log(n)$.

Ans $1 < \log\log n < \sqrt{\log n} < \log n < \log_2 n < 2\log n < n < 2n < 4n < n\log n < \log n! < n^2 < 2^{2^n} < n!$

(c) $8^{2n}, \log_2(n), n\log_6(n), n\log_2(n), \log(n!), n!, \log_8(n), 96, 8n^2, 7n^3, 5n$.

Ans $96 < \log_8 n < \log_2 n < 5n < n\log_6 n < n\log_2 n < \log(n!) < 8n^2 < 7n^3 < n! < 8^{2n} < n!$

STUDENT STYLE

Signature

Tutorial→03

Name → Mohd Nasir
Sect → 'F'
Roll → 14
Uni Roll → 2016855

Ans:-01

```
while (low ≤ high)
{
    mid = low+(high-low)/2;
    if (arr[mid] == key)
    {
        Print("Found");
        Print (Count);
    }
    else if (arr[mid] > key)
            high = mid-1;
    else
        low = mid+1;
    Count++;
}
Print ("Not found");
```

Ans:-02 Iterative Insertion Sort:-

```
for (int i=1; i<n; i++)
{   j= i-1;
    x=arr[i];
  while (j>-1 and arr[j]>x)
    {
      A[j+1] = A[j];
       j--;
    }
    arr[j+1]= x;
}
```

Recursive:-

```
Void InsertionSort (int arr[], int n)
{
    if (n<=1)
      return;
  insertionSort(arr, n-1);
  int last = arr[n-1];
     j=n-2;
  while(j>=0 and arr[j]>last)
     {
       arr[j+1] = arr[j];
           j--;
     }
     arr[j+1] = last.
}.
```

Insertion sort is online sorting because whenever a new element come, insertion sort define its right place.

Ans :-03

Bubble Sort → $O(n^2)$
Insertion Sort → $O(n^2)$
Selection Sort → $O(n^2)$
Merge Sort → $O(n \log n)$
Quick Sort → $O(n \log n)$
Count Sort → $O(n)$
Bucket Sort → $O(n)$.

Ans :-04

Online Sorting :→ Insertion Sort.
Stable Sorting :→ Merge Sort, Bubble Sort.
Inplace Sorting → Bubble Sort, Selection Sort.

Ans :-05

Iterative Binary Search.

```
while(low <= high)
{
    mid = low + (high + low)/2;
    if (arr[mid] == key)
        return true;
    else if (arr[mid] > key)
        high = mid-1;
    else
        low = mid+1;
}
return false;
```

$O(n \log n)$

Recursive Binary Search.

```
if (low <= high)
{
    int mid = low + (high+low)/2;
    if (arr[mid] == key)
        return true;
    else if (arr[mid] > key)
        Binarysearch (arr, low, mid-1);
    else
        Binarysearch (arr, mid+1, high);
}
return false;
```

Ans:→6 $T(n) = T(n/2) + T(n/2) + C$

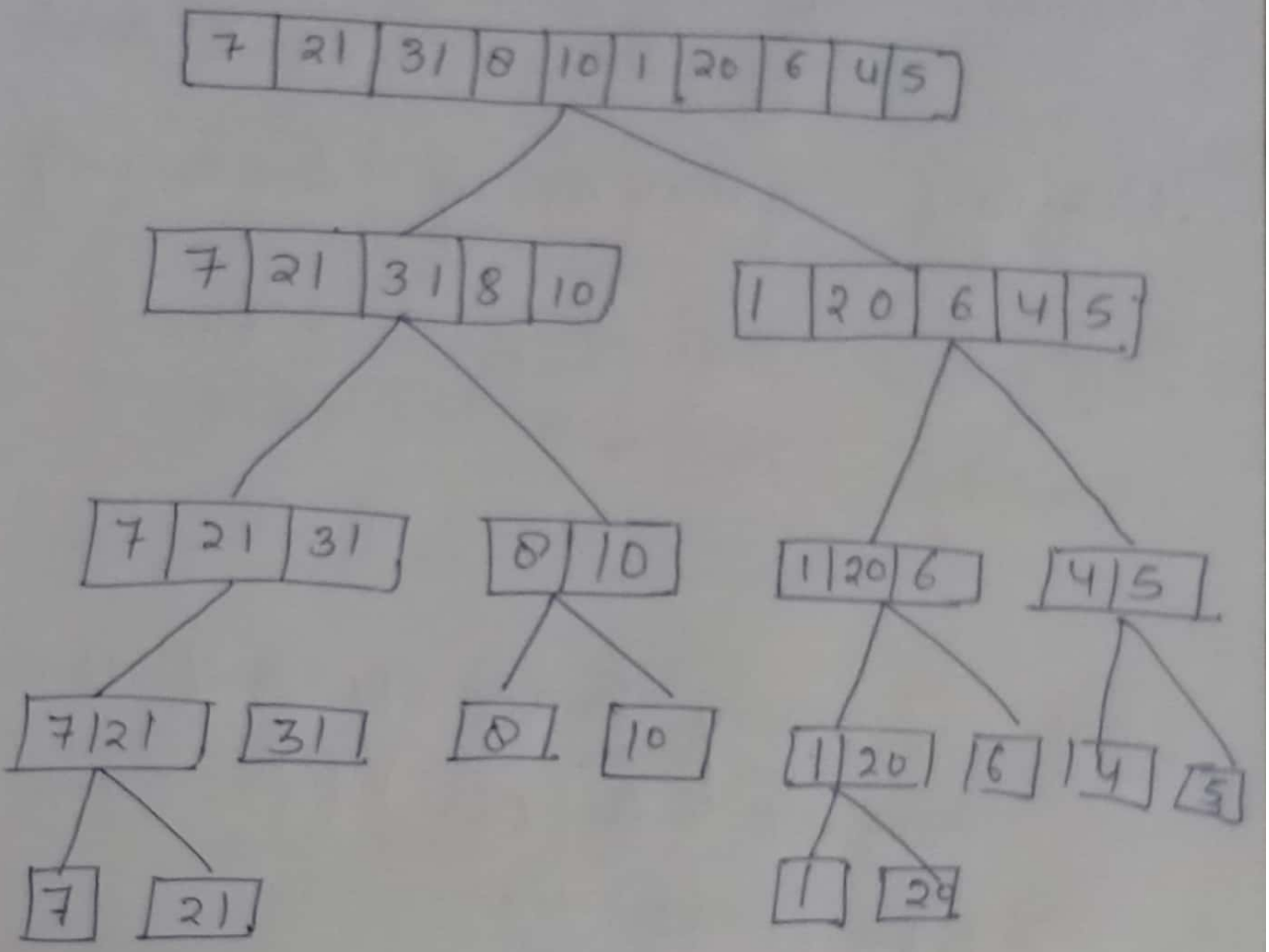
$$\boxed{T(n) = T(n/2) + 1}$$

Ans:→7

```
void (vector<int> arr, int key)
{
    sort (arr.begin(), arr.end());
    int low = 0, high = arr.size() - 1;
    while (low < high)
    {
        if (arr[low] + arr[high] == key){
            print (low, high)
            Break;
        }
        else if (arr[low] + arr[high] > key)
            high--;
        else
            low++;
    } print ("No found");
}
```

Ans: 08 Quick Sort is the fastest general purpose Sort. In most practical solution, quicksort is the method of choice if stability is important and space is available, merge sort might be best.

Ans: 09 Inversion indicates - how far are close the array is from being sorted.



Inversion → 31

Ques→11 Merge sort

Best Case → $2T(n/2)+O(n)$

Worst Case → $2T(n/2)+n$

Quick sort.

Best Case → $2T(n/2)+n$

Worst Case → $T(n-1)+n$.

Basis	Quick sort	Mergesort.
• Position	Splitting done in any ratio.	Array divided in two equal parts.
• Works well on	Smaller array.	fine on any size of array.
• Sorting Method	Internal	External.
• Stability	Not Stable	Stable.

Ques→10 if array is already sort than Quick Sort will take $O(n^2)$ time. Otherwise it takes $(n \log n)$ time.

Ans 12:-

```
Void StableSelectionSort (int arr[], int n)
{
    for (int i=0; i<n-1; i++)
    { int Pos=i;
      int min = i;
      for (int j= i+1; j<n; j++)
      { if (arr[j]<min)
        {
          min = arr[j];
          Pos = j;
        }
      }
    if(Pos!=i)
      while (Pos>i)
      {
        arr[Pos]=arr[Pos-1];
        Pos--;
      }
      arr[i] = min;
    }
}
```

Ques → 13 We will use the merge sort because we can divide the 4 GB data into 4 parts of 1 GB and sort them separately and combine them ladter.

Internal sorting → All the data is sorted in memory at all times sorting is in progress.

External sorting → all the data is stored outside memory and only load in small parts.

Tutorial → 04.

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

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

$n^{\log_b a} = n^{\log_2 3}$

comparing $n^{\log_2 3}$ and n^2

$n^{\log_2 3} < n^2$ Case 3.

∴ according to Master's theorem.

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

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

$a = 4, b = 2$

$n^{\log_b a} = n^{\log_2 4} = n^2 = f(n)$ Case 2.

∴ according Master's theorem.

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

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

$a = 1, b = 2$

$n^{\log_2 1} = n^0 = 1$

$1 < 2^n$ Case 3.

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

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

Master theorem is not applicable.

(5) $T(n) = 16T(n/4) + n$

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

$n^{\log_b a} = n^{\log_4 16} \Rightarrow n^2, \; f(n) < n^2$

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

(6) $T(n) = 2T(n/2) + n\log n$.

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

$n^{\log_b a} = n^{\log_2 2} \Rightarrow n$

$f(n) > n$

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

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

$a = 2, \; b = 2, \; K = 1, \; P = -1$

$a = b^K$, $P \neq -1$

$(2 = 2^1)$

$T(n) = \Theta(n^{\log_b a} \log\log n)$

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

(8) $T(n) = 2T(n/4) + n^{0.51}$

$a = 2, b = 4, f(n) = n^{0.51}$

$n^{\log_b a} = n^{\log_4 2} = n^{0.5}$

$n^{0.5} < f(n)$

$T(n) = \Theta(n^{0.51})$

(9) $T(n) = 0.5T(n/2) + \frac{1}{n}$

$a < 1$, not applicable Master theorem.

(10) $T(n) = 16T(n/4) + n!$

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

$n^{\log_b a} = n^{\log_4 16} = n^2$

$n^2 < n!$

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

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

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

$n^{\log_b a} = n^{\log_2 4} = n^2$

$n^2 > f(n)$

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

(12) $T(n) = \text{sqrt}(n) + \frac{n}{2} + \log n$

Master's theorem not applicable

(13). $T(n) = 3T(n/2) + n$

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

$n^{\log_b a} = n^{\log_2 3} = n^{1.58}$

$n^{1.58} > f(n)$

$T(n) = \Theta(n^{\log_2 3})$

(14) $T(n) = 3T(n/3) + \sqrt{n}$

$a = 3, b = 3, f(n) = \sqrt{n}$

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

$n > \sqrt{n}$

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

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

$a = 4, b = 2, f(n) = c * n$

$n^{\log_b a} = n^{\log_2 4} = n^2$

$n^2 > c * n$

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

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

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

$n^{\log_b a} = n^{\log_2 4} = n^2$

$n^2 > n\log n$

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

(20). $T(n) = 64T(n/8) - n^2\log n.$

Master's theorem is not applicable as $f(n)$ is not increasing function.

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

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

$n^{\log_b a} = n^{\log_3 7} = n^{1.7}$

$n^{1.7} < n^2$

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

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

Master's theorem is not applicable.

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

$a = 3, b = 4, f(n) = n\log n$

$n^{\log_b a} = n^{\log_4 3} = n^{0.79}$

$n^{0.79} < n\log n.$

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

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

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

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

$O(n) \cong O(n/2)$

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

(18). $T(n) = 6T(n/3) + n^2\log n$

$a = 6, b = 3, f(n) = n^2\log n.$

$n^{\log_b a} = n^{\log_3 6} = n^{1.63}$

$n^{1.63} < n^2\log n$

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

TUTORIAL:→05

Name → Mohd Nasir
Sec → 'F'
Roll → '14'
Uni Roll → 2016055
Student → 20021595
Subject → Design And Analysis of Algorithms.

Ques→1. What is difference between DFS and BFS. Write application both the algorithm.

BFS	DFS
(1). It's stands for Breath First Search.	(1) It's stand for Depth first search.
(2). In this searching we use Queue data structure.	(2) In this searching we use stack datastructure.
(3). It give 100% result.	(3) It does not give 100% result.
(4) It is more suitable for searching vertices are closer to given source.	(4). It is more suitable when there as away solution away from source.
(5). There is no concept of backtracking.	(5). It is a recursive algorithm that use backtracking.

Applications:→

(a) BFS:→ Bipartite graph and minimum no of Nodes Path, networking, and GPS.

(b) DFS:→ acyclic graph, topological Sort Scheduling Problems.

Ques→02 Which data structure are used to implement BFS and DFS and Why?

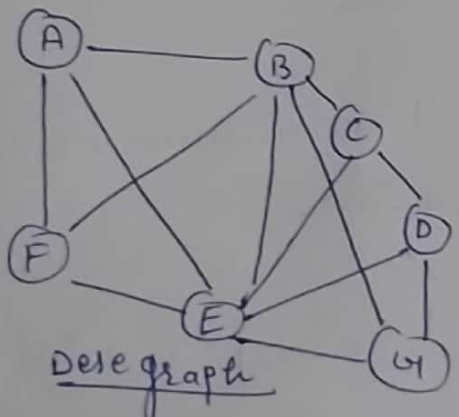
↳ For implementing BFS We use a Queue data-structure for finding minimum no. of nodes path between source node to destination node. We use Queue because things don't have to be processed immediately. but have to be processed in FIFO order. Like BFS :. Searching for nodes level traverse. Ex it search nodes w.r.t their distance from source. For this Queue is better in ~~any~~ ito use BFS.

For implemeting DFS We use →stack data-structure it traverse a graph in depthward motion and stack-10 remember to get the next nodes to start a search when dead end occurse in any iteration.

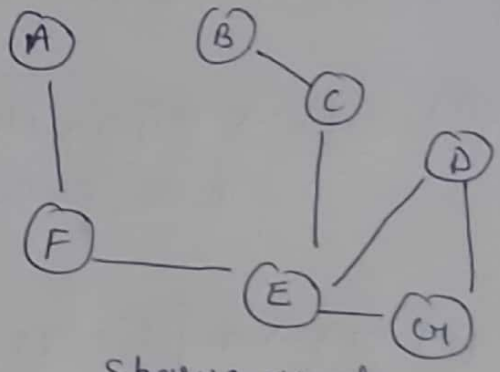
Ques→03 What do you mean by Spars and dense graphs? Which representation of graph is better for Sparse and dense graph?

↳ Dense graph is a graph in which no. of nodes is close to maximal no of edges.

↳ sparse graph is a graph in which no of edges in very less.



Dese graph



Sparse graph.

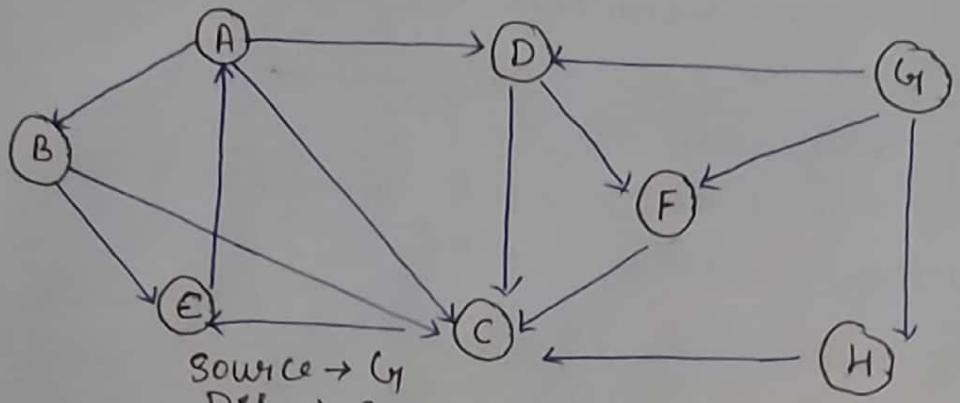
·) For Sparse graph it is prefered to use Adjency list.

·) For dense graph it is prefered to Adjency Matrix.

(2) Union :→ It takes two elements as input and find representation of this sets using the find operation. and finally Puts either one of the tree under Set node of other tree effectively Merging the trees and set.

```
void union(int a, int b)
{
    arr[a] = b
}
```

Ques → 6 Run BFS and DFS on graph.



BFS :→

Child	G	H	D	F	C	E	A	B
Parent	-	G	G	G	H	C	E	A

Path → G → H → C → E → A → B

DFS :→

PUSH	POP
~~G~~	G
D	F
H	C
~~F~~	E
~~C~~	A
~~E~~	B
~~A~~	
~~B~~	

Path → G → F → C → E → A → B.

Ques→04 How can you detect a cycle in graph using BFS and DFS.

Ans→ For detecting cycle in a graph using BFS we need to use Kahn's algorithm for topological sorting.

The steps required are:-

(1). Compute in-degree (no of incoming edges) for each of vertex present in graph and initialize count of visited nodes as 0.

(2) Pick all vertices with in degree as 0 and add them in queue.

(3) Remove a vetex from queue and then.
- increment count of visited nodes by 1.
- Decrease in-degree by 1 for all neighbouring nodes.
- If in-degree of neighbouring nodes reduced to zero then add in queue.

(4). Repeat 3. until queue is Empty.

(5). If count of visited nodes is not equal to no of nodes in graph. has cycle, otherwise not.

Ques→05 What do you mean by disjoint set data structure? Explain and ~~representations~~ operations along with example which can be performed on disjoint set.

Ans→ A disjoint set is a data structure that keeps track of set of element partitioned into several disjoint subsets. In other words. a disjoint set in a group of sets where no item can be in more than one set.

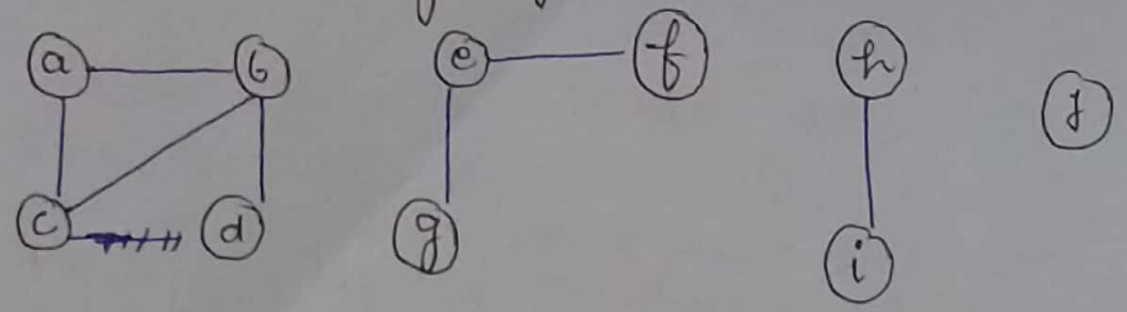
3 operations→

(i) Find → can be implemented by recursively traversing the parent array until we hit a node who is parent to itself.

```
int find (int V) {
  if (V == arr[V])
    return V;
  else return find(arr[V]);
}
```

Ques → 7 find out no of connected components and vertices in each component using disjoint set data structure.

Ans →



V = {a} {b} {c} {d} {e} {f} {g} {h} {i} {j}

= {a,b} {a,c}, {b,c}, {b,d}, {e,f}, {e,g} {h,i}, {j}

(a,b) {a,b} {c} {e} {f} {g} {h} {i} {j}.

(a,c) {a,b,c} {d} {e} {f} {g} {h} {i} {j}.

(b,c) {a,b,c} {d}, {e}, {f}, {g}, {h}, {i}, {j}.

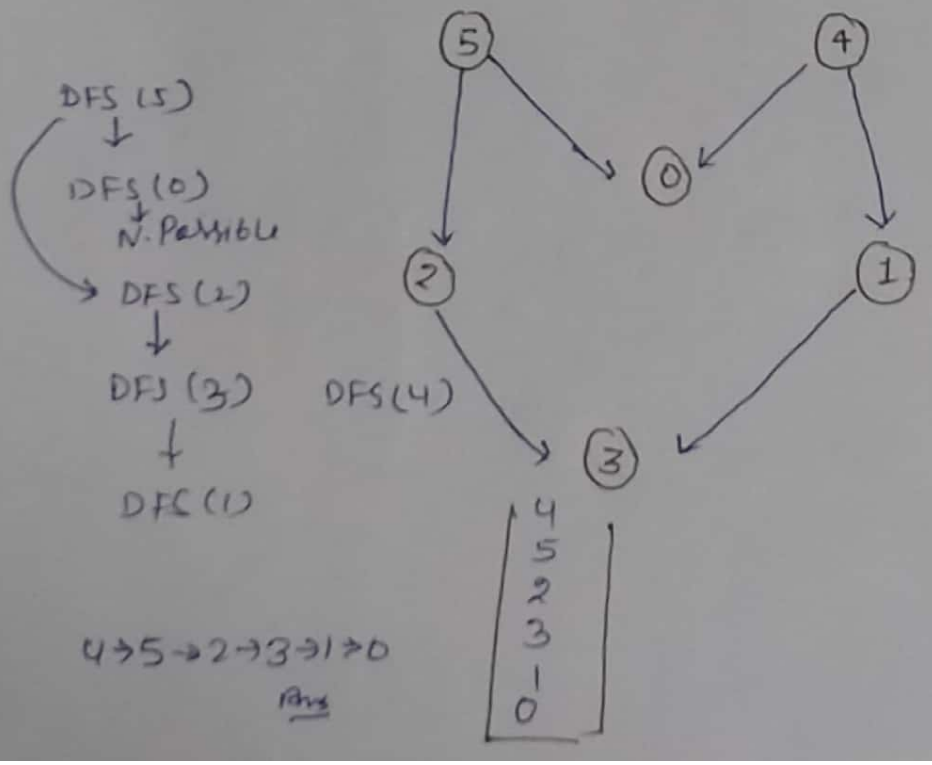
(b,d) {a,b,c,d} {e} {f} {g} {h} {i} {j}.

(e,g) {a,b,c,d} {e,f} {g} {h} {i} {j}

(h,i) {a,b,c,d} {e,f,g} {h,i} {j}

no. of connected components → 03.

Ques → 8 Apply topological sort and DFS on graph having vertices from 0 to 5.



Ques→9. Heap data structure can be used to implement priority queue Name few graph algorithm where you need to use Priority queue and why?

Ans Yes, Heap data structure can be used to implemented priority queue, It will take O(logn) time to insert and delete each element in priority queue. Based on heap structure priority queue has two types max priority queue based on max heap and min heap priority queue based on min heap Heaps provide better performance compare to array etc.

Tutorial 6

DATE _____
PAGE _____

1.

Minimal spanning tree

A minimal spanning tree or minimum weight weight spanning tree is a subset of the edges of the connected, edge-weighted and undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight.

Application :-

1. Design local area network.
2. In constructig highways or railways spanning
3. Laying pipelines.
4. telecommunication network.

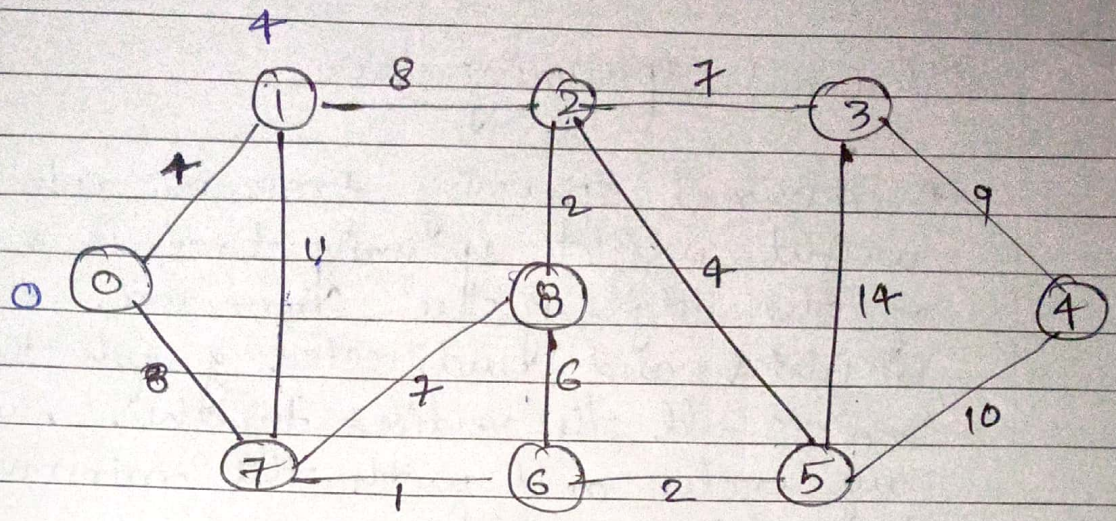
2.

Algo's.	Time Complexity	Space Complexity
Prims	$O(V^2)$	$O(V+E)$
Krushkal	$O(E \log V)$	$O(\log E)$
Dijkstra	$O(V+E)$	$O(V+E)$
Bellman ford	$O(VE)$	$O(V)$

DATE _____

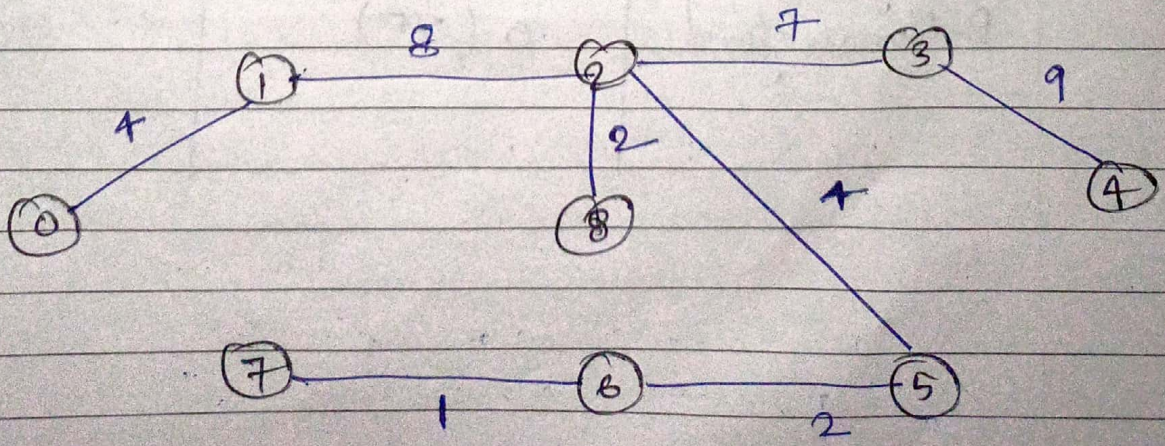
PAGE _____

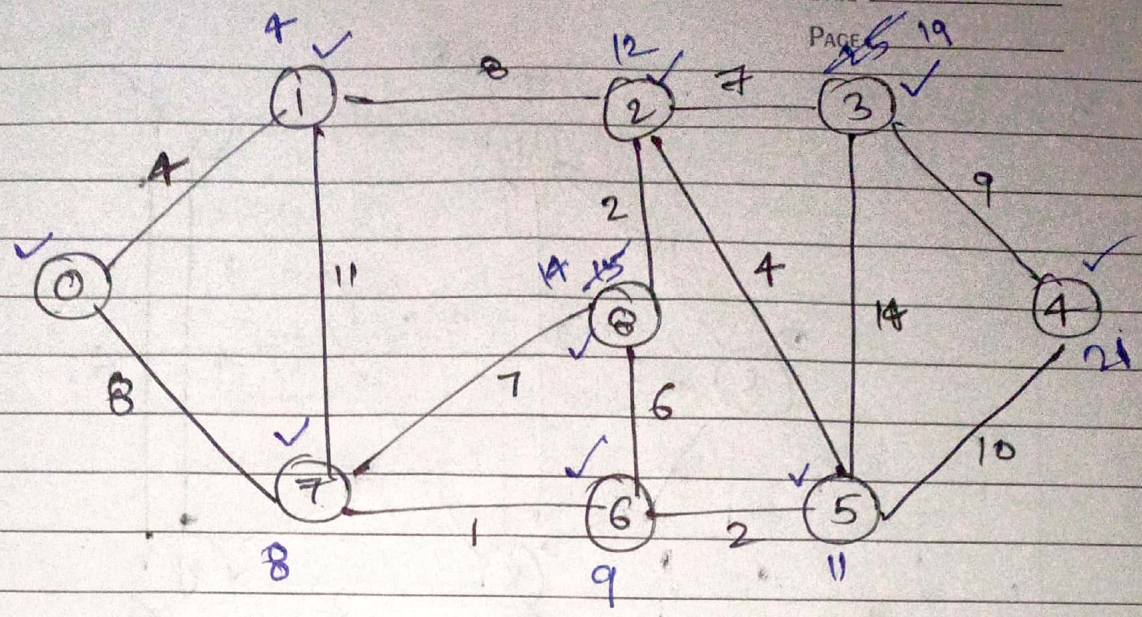
3

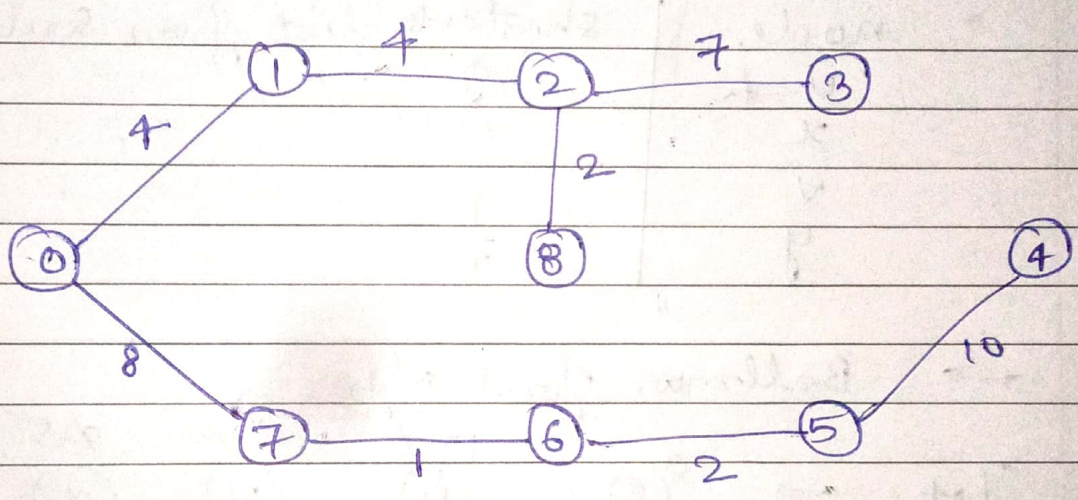


Path	Weight
7 → 6	1
6 → 5	2
2 → 8	2
0 → 1	4
2 → 5	4
8 → 6	6
2 → 3	7
7 → 8	7
1 → 2	8
3 → 4	9.
5 → 4	10
1 → 7	11
3 → 5	14

0 → 7 8







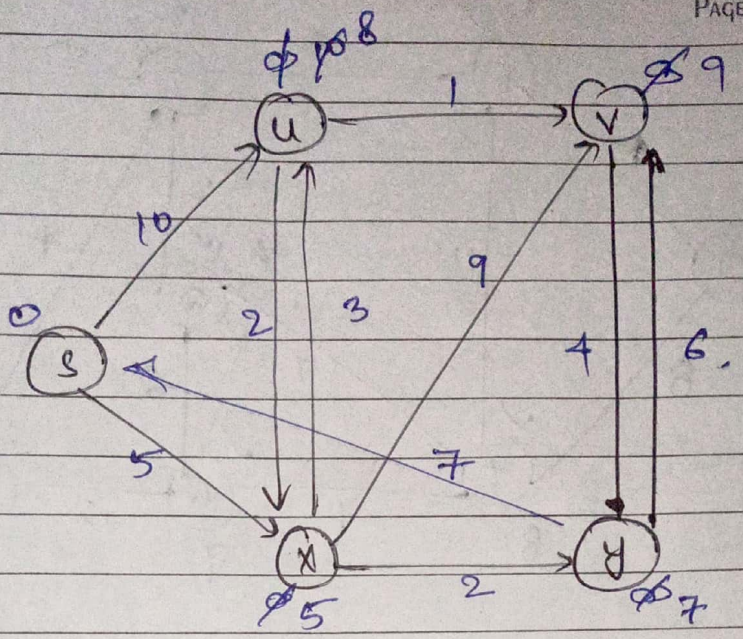
4 (i) The shortest path may change. The reason is there may be different no. of edges in different paths from s to t. for shortest path of weight 15 and 5 edge and another 2 edge and 25 weight. weight of shortest path inc. by 5^10 and become 15+50 and similiarly other path become 25+20. so shortest path weight as 45.

(ii) If we multiply all edges weight by 10, the shortest path does not change. The no. of edge on path does not matter.

DATE _____

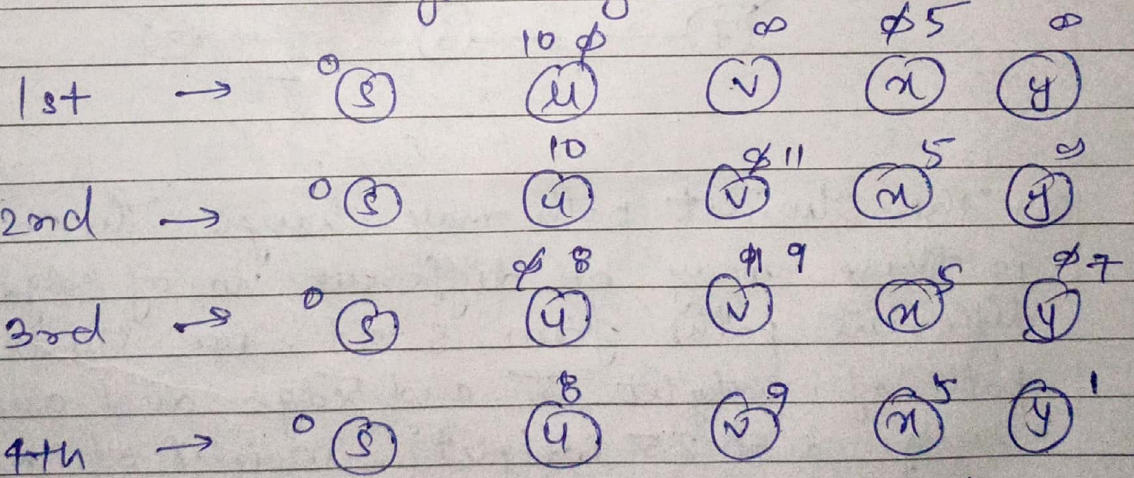
PAGE _____

5//



node	shortest dist from source node
u	8
x	5
v	9
y	7

→ Bellman ford algo.



final graph:-