Page No:
Date:

Algorithm	Program
- Design part of project	- Implementation part of project
- Should have domain Knowledge	- Programmers
- Any language	- Programming language
- H/w and OS independent	- H/w and OS dependent
- Analyse	- Testing

Priori Analysis	Posteriori Analysis
- Algorithm	- Program
- Independent of language	- Language dependent
- Hardware independent	- Hardware dependent
- Time & Space function	- Watch time & Bytes

Characteristics of algorithm
(i) i/p → 0 or more
(ii) o/p → atleast 1 o/p
(iii) Definiteness
(iv) Finiteness
(v) Effectiveness

How to analyse an algorithm?
(i) Time
(ii) Space
(iii) Network Consumption
(iv) Power
(v) CPU Registers

(Ex) Algorithm swap(a,b)
{
temp = a; ——— (1)
a = b; ——— (1)
b = temp; ——— (1)
}

$f(n) = 3$
$O(1)$

Space
a - 1
b - 1
temp - 1
$f(n) = 3$
$O(1)$

Page No:
Date:

Frequency Count Method

A	8	3	9	7	2
	0	1	2	3	4

space

A → n
n → 1
s → 1
i → 1

S(n) = n + 3
→ O(n)

Q (Sum of all elements in arr)

```
Algorithm Sum (A, n) {
    s = 0                          ——— 1
    for (i=0; i<n; i++) {          → 2n+2 = n+1
        s = s + A[i];              ——— n
    }
    return s;                      ——— 1
}
```

f(n) = 2n + 3
→ O(n)

Q. Sum of two matrices

```
Algorithm Add (A, B, n) {
    for (i=0; i<n; i++) {          → 2n+2 → n+1
        for (j=0; j<n; j++) {      → n(n+1)
            c[i,j] = A[i,j] + B[i,j];  → n(n)
        }
    }
```

f(n) = 2n² + 2n + 1
O(n²)

Space

A → n × n → n²
B → n × n → n²
c → n × n → n²
n → 1
i → 1
j → 1

S(n) = 3n² + 3
⇒ O(n²)

Q. Multiplication of two matrices

```
Algorithm multiply (A, B, n) {
    for (i=0; i<n; i++) {          → 2n+2 → n+1
        for (j=0; j<n; j++) {      → n(n+1) → n²+n
            c[i,j] = 0;            → n(n) → n²
            for (k=0; k<n; k++) {  → n × n × (n+1)
                c[i,j] = c[i,j] + A[i,k] * B[k,j];  → n × n × n
            }
        }
    }
```

Page No:
Date:

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

Time $\longrightarrow O(n^3)$

Space

A – n^2
B – n^2
C – n^2
n – 1
i – 1
j – 1
k – 1

$S(n) = 3n^2 + 4$
$\longrightarrow O(n^2)$

Q1
```
for (i=0; i<n; i++)        ⟶ 2n + 2 ⟶ n+1
{
    start;                 ⟶ n
}
```
$f(n) = 2n + 1$
$O(n)$.

Q2
```
for (i=n; i>0; i--)        ⟶ n-1
{
    start;                 ⟶ n
}
```
$f(n) = 2n - 1$
$O(n)$

Q3
```
for (i=1; i<n; i=i+2)
{
    start;                 ⟶ n/2
}
```
$O(n)$

Q4
```
for (i=0; i<n; i++)        ⟶ n+1
{
    for (j=0; j<n; j++)    ⟶ n(n+1)
    {
        start;             ⟶ n(n)
    }
}
```
$(n+1) + n(n+1) + n^2$
$n + 1 + n^2 + n + n^2$
$2n^2 + 2n + 1$
$O(n^2)$

$1 + 2 + 3 + 4 + \ldots + n$
$= n(n+1)/2 = n^2$
$O(n^2)$.

i	j	Stmt
0	0	0
1	0	1
	1	
2	0	2
	1	
	2	
3	0	3
	1	
	2	
	3	
		n

Page No:
Date:

Q5

```
P=0
for(i=1; P<=n; i++){
    P=P+i;
}
```

i	P
1	0+1 = 1
2	1+2 = 3
3	1+2+3 = 6
⋮	⋮
K	1+2+... K = K(K+1)/2

Assume $P > n$

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

$K(K+1)/2 > n$

$K^2 > n$

$K > \sqrt{n} \longrightarrow O(\sqrt{n})$

Q6

```
for(i=1; i<n; i=i*2)
{
    start;
}
```

i
1
$1 \times 2 = 2$
$2 \times 2 = 2^2$
⋮
2^K

Assume $i >= n$

$\because i = 2^K$

$\therefore 2^K >= n$

$2^K = n$

$K = \log_2 n \longrightarrow O(\log_2 n)$

```
for(i=0; i<=n; i++)
{
    start;
}
```

$i = 1+1+1+\cdots 1 = n$

$K = n$

Q7

```
for(i=1; i<n; i=i*2)
{
    stmt;
}
```

n=8 i	n=10 i
1	1
2	2
4	4
8 ✗	8
	16 ✗

(3) (4)

Page No:
Date:

Q8
```
for(i=n; i>=1; i=i/2)
{
  Start;
}
```

Assume i<1

$n/2^k < 1$

$n/2^k = 1$

$n = 2^k$

$k = \log_2 n$ → $O(\log_2 n)$

Q2 i/n

$n/2$

$n/2^2$

$n/2^3 \dots n/2^k$

Q9
```
for(i=0; i*i<n; i++)
{
  Stmt;
}
```

$i*i < n$

$i*i > -n$

$i^2 = n$

$i = \sqrt{n}$

$O(\sqrt{n})$

Q10
```
for(i=0; i<n; i++){
  Stmt;
} for(j=0; j<n; j++){
  Stmt;
}
```

n + n → $O(n)$

$f(n) = 2n$

Q11
```
P=0
for(i=1; i<n; i=i*2){
  P++;
}
for(j=1; j<p; j=j*2){
  Stmt;
}
```

$P = \log n$

$\log p$

$O(\log(\log n))$

Q12
```
for(i=0; i<n; i++){
  for(j=1; j<n; j=j*2){
    Stmt;
  }
}
```

n

$n * \log n$

$n * \log n$

$f(n) = 2n\log n + n$

$\longrightarrow O(n\log n)$

Page No:
Date:

Analysis of if and while

Q
```
i=0;          ——— (1)
while (i<n)   ——— (n+1)
    stmt;     ——— (n)
    i++;      ——— (n)
}
```
$3n+2 \longrightarrow O(n)$

Q
```
a=1;
while (a<b){
    stmt;
    a=a*2;
}
```

a	Terminate
1	$a \geq b$
$1 \times 2 = 2$	$a = 2^k$
$2 \times 2 = 2^2$	$2^k \geq b$
$2 \times 2 \times 2 = 2^3$	$2^k = b$
$\vdots$ 2^k	$k = \log_2 b$

$(\log n)$

Q
```
i=1;
k=1;
while (k<n){
    stmt;
    k=k+i;
    i++;
}
```

i	k
1	1
2	1+1=2
3	2+2=4
4	2+2+3
5	2+2+3+4
$\vdots$	$\vdots$
m	$2+2+3+4+\dots+m = m(m+1)/2$

$k \geq n$

$\frac{m(m+1)}{2} \geq n$

$m^2 \geq n \longrightarrow m = \sqrt{n}$ $\quad O(\sqrt{n})$

Q GCD of two numbers

```
while (m!=n)
    if (m>n)
        m=m-n;
    else
        n=n-m;
}
```

$m=16$; $n=2$

m	n	
16	2	
12	2	
10	2	
8	2	$k/2$
6	2	$n/2$
4	2	
2	2	$O(n)$

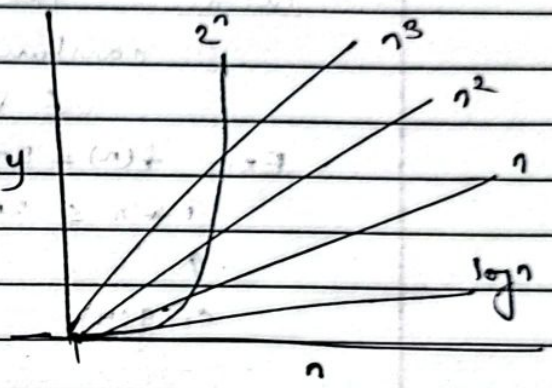
Page No:
Date:

Types of time functions:

$O(1)$ → Constant	$O(\log n)$ → Logarithmic
$O(n)$ → Linear	$O(n^2)$ → Quadratic
$O(n^3)$ → Cubic	$O(2^n)$ → Exponential

Compare class of function: $1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \ldots < 2^n < 3^n < \ldots < n^n$

$\log n$	n	n^2	2^n
0	1	1	2
1	2	4	4
2	4	16	16
3	8	64	256
3.1	9	81	512



Asymptotic Notation:

- O big-oh upperbound
- Ω big-o-mega lowerbound
- Θ theta average bound.

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \ldots < 2^n < 3^n < \ldots < n^n$

lower bound — n — average bound — upper bound

- **Big Oh** — The function $f(n) = O(g(n))$ iff $\exists$ +ve constants c and n_0 such that $f(n) \leq c * g(n)$ $\forall\ n \geq n_0$

Ex: $f(n) = 2n+3$ → let

$2n+3 \leq 2n+3n$

$2n+3 \leq 5n \longrightarrow n \geq 1$

$2n+3 \leq 2n^2+3n^2$

$2n+3 \leq 5n^2$

Nearest one most appropriate

$f(n) = O(n)$ ✓

$f(n) = O(n^2)$ ✓

$f(n) = O(2^n)$ ✓

$f(n) = O(\log n)$ ✗

• Big Omega – The function $f(n) = \Omega(g(n))$ iff $\exists$ +ve constants c and n_0 i.e $f(n) \geq c * g(n)$ $\forall\ n \geq n_0$

Ex. $f(n) = 2n+3$

$2n+3 \geq 1 * n \quad \forall\ n \geq 1$

$f(n)$ $\quad$ c $\quad$ $g(n)$

$f(n) = \Omega(n^2)$ ✓

$f(n) = \Omega(\log n)^2$ ✓

$f(n) = \Omega(n^2)$ ✗

• Theta – The function $f(n) = \Theta(g(n))$ iff $\exists$ +ve constants c_1, c_2 and n_0 so that $c_1 * g(n) \leq f(n) \leq c_2 * g(n)$

Ex. $f(n) = 2n+3$

$1 * n \leq 2n+3 \leq 5 * n$

$c_1\ g(n)$ $\quad$ $f(n)$ $\quad$ $c_2 g(n)$

$f(n) = \Theta(n)$ ✓

$g(n) = \Theta(n^2)$ ✗

Q $f(n) = 2n^2 + 3n + 4$

$2n^2 + 3n + 4 \leq 2n^2 + 3n^2 + 4n^2$ $\quad$ $f(n) = O(n^2)$

$2n^2 + 3n + 4 \leq 9n^2 \longrightarrow n \geq 1$

Ex: $f(n) = 2n^2 + 3n + 4$

$2n^2 + 3n + 4 \geq 1 * n^2$ $\quad$ $\Omega(n^2)$

$n^2 \leq 2n^2 + 3n + 4 \leq 9n^2$ $\quad$ $\Theta(n^2)$

Ex: $f(n) = n^2 \log n + n$

$1 * n^2 \log n \leq n^2 \log n + n \leq 10 n^2 \log n$

$O(n^2 \log n)$ $\quad$ $\Omega(n^2 \log n)$ $\quad$ $\Theta(n^2 \log n)$

Ex: $f(n) = n! = n \times (n-1) \times (n-2) \times \cdots \cdots 3 \times 2 \times 1$

$1 \times 1 \times 1 \cdots 1 \times 1 \leq 1 \times 2 \times 3 \times \cdots \times n \leq n \times n \times n \times \cdots n$

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

$O(n^n)$

$\Omega(1)$

Ex: $f(n) = \log n!$

$\log(1 \times 1 \times 1 \cdots 1) \leq \log(1 \times 2 \times 3 \times \cdots n) \leq \log(n \times n \times n \cdots n)$

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

$\Omega(1)$

$O(n \log n)$

Properties of asymptotic notations

① General properties

If $f(n)$ is $O(g(n))$ then $a * f(n)$ is $O(g(n))$

Ex: $f(n) = 2n^2 + 5$ is $O(n^2)$

then $7 \cdot f(n) = 7(2n^2 + 5)$

$= 14n^2 + 35 \longrightarrow O(n^2)$

② Reflexive

If $f(n)$ is given then $f(n)$ is $O(f(n))$

Ex: $f(n) = n^2 \longrightarrow O(n^2)$

③ Transitive

If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$, then $f(n) = O(h(n))$

Ex: $f(n) = n$, $g(n) = n^2$ $h(n) = n^3$

n is $O(n^2)$ and n^2 is $O(n^3)$

then n is $O(n^3)$

④ Symmetric

If $f(n)$ is $\Theta(g(n))$ then $g(n)$ is $\Theta(f(n))$

Ex: $f(n) = n^2$ $g(n) = n^2$

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

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

⑤ Transpose Symmetric

If $f(n) = O(g(n))$ then $g(n)$ is $\Omega(f(n))$

Ex: $f(n) = n$ $g(n) = n^2$

then n is $O(n^2)$ and n^2 is $\Omega(n)$

if $f(n) = O(g(n))$
and $f(n) = \Omega(g(n))$
$g(n) \leq f(n) \leq g(n)$
$\therefore f(n) = \Theta(g(n))$

if $f(n) = O(g(n))$
and $d(n) = O(e(n))$
then $f(n) + d(n)$

Ex

$f(n) = n = O(n)$
$d(n) = n^2 = O(n^2)$
$\therefore f(n) + d(n) = n + n^2$
$= O(n^2)$

Best, Worst and Average Case Analysis

* Linear Search

Best case time – searching key element present at first index
$B(n) = O(1)$

Worst case time – $O(n)$ ⟶ Searching a key at last index

$$\text{Avg case} = \frac{\text{All possible case time}}{\text{No. of cases}}$$

Avg. time $= (n+1)/2$

* Binary Search

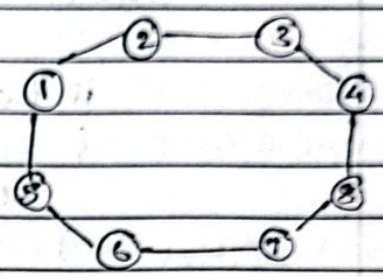
Best case time : $O(1)$
Worst case : $\log n$ $O(\log n)$

Page No:
Date:

Disjoint Sets

1. Disjoint sets and Operations
2. Detecting a cycle
3. Graphical representation
4. Array representation
5. Weighted Union & Collapsing ~~Find~~ Find.

Operations : 1. Find
2. Union



$S1 = \{1, 2, 3, 4\}$

$S2 = \{5, 6, 7, 8\}$

$S1 \cap S2 = \phi$

$S3 = \{1, 2, 3, 4, 5, 6, 7, 8\}$

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

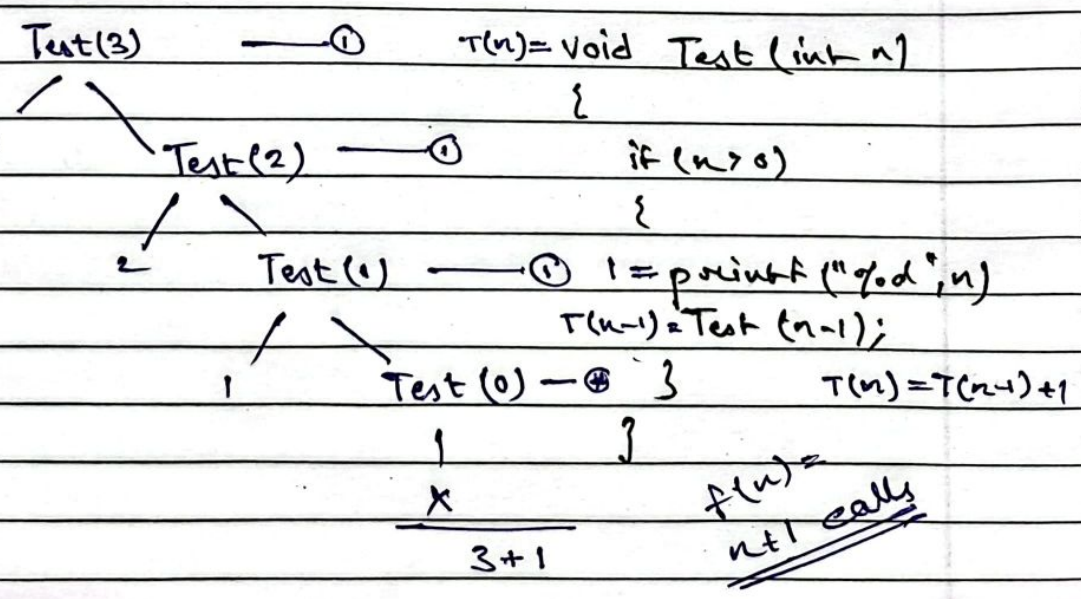
Page No:
Date:

Divide and Conquer

Algorithm that recursively breaks down a problem into two or more subproblems of the same or related type, until these become simple enough to be solved directly.

```
DAC(P)
{
  if (small(P))
  {
    S(P);
  }
  else
  {
    divide into P1, P2 .... Px
    Apply DAC(P1), DAC(P2) ....
    Combine (DAC(A'), DAC(P2) ....)
  }
}
```

(i) Binary Search
(ii) Finding maximum and minimum
(iii) MergeSort
(iv) Quicksort
(v) Strassen's Matrix Multiplication



Page No:
Date:

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

$$T(n) = \begin{cases} 1 & ; n=0 \\ T(n-1)+1 & ; n>0 \end{cases}$$

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

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

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

⋮ K times

$\boxed{T(n) = T(n-K) + K}$

Assuming $n-K=0$

$\therefore K=n$

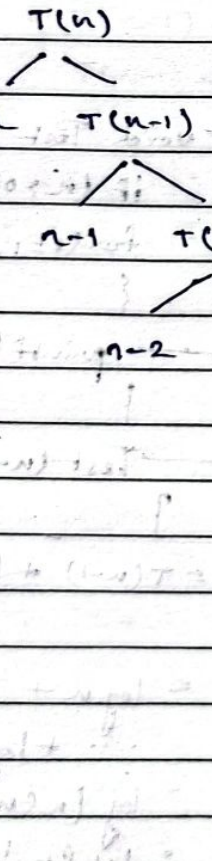
$T(n) = T(n-n) + n$

$T(n) = T(0) + n$

$T(n) = 1 + n \longrightarrow O(n)$

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

$$T(n) = \begin{cases} 1 & ; n=0 \\ T(n-1)+n & ; n>0 \end{cases}$$




```
T(n)   void Test(int n)
       {
1  ——    if(n>0)
       {
n+1 ——   for(i=0; i<n; i++)
       {
n  ——      pf("%d", n);
       }
T(n-1) ——  Test(n-1);
       }
       }
```

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

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

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

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

$O(n^2)$

Page No:
Date:

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1)+n & n>0 \end{cases}$$

$T(n) = T(n-1) + n$ — ①

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

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

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

$T(n) = T(n-2) + (n-1) + n$ — ②

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

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

⋮

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

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

Assume $n-k = 0 \Rightarrow n = k$ — ④

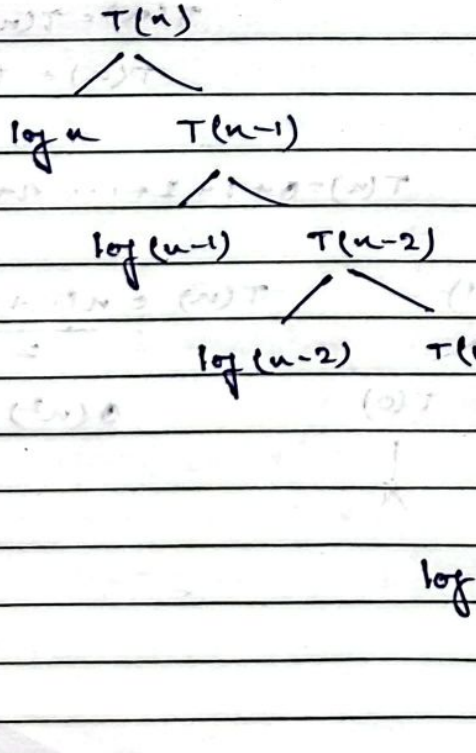
$T(n) = T(0) + 1 + 2 + 3 + \cdots + (n-1) + (n)$

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

$T(n) = \frac{n^2+n}{2} \longrightarrow O(n^2)$

Q

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1) + \log n & n>0 \end{cases}$$




```
T(n) —— void Test (int n) {
          if (n>0) {
          for(i=1; i<n; i=i*2)
          {
log(n) ——   printf(" %d", i);
          }
T(n-1) ——  Test(n-1);
          }
```

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

$= \log n + \log(n-1) + \log(n-2) + \cdots + \log 2 + \log 1$

$= \log [n(n-1) \times \cdots \times 2 \times 1]$

$= \log(n!)$

$\Rightarrow O(n \log n)$

Page No:
Date:

$$T(n) = \begin{cases} 1 & ; n = 0 \\ T(n-1) + \log n & ; n > 0 \end{cases}$$

$T(n) = T(n-1) + \log n$ ——— ①

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

$T(n) = T(n-2) + \log(n-1) + \log n$ ——— ②

$T(n) = [T(n-3) + \log(n-2)] + \log(n-1) + \log n$

$T(n) = T(n-3) + \log(n-2) + \log(n-1) + \log n$ ——— ③

⋮

$T(n) = T(n-k) + \log(n-(n-1)) + \log(n-(n-2)) + \cdots \log(n-1) + \log n$

$T(n-1) = T(n-k) + [\log 1 + \log 2 + \log 3 + \cdots (\log(n-1)) + \log(n)]$

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

$T(n) = T(0)$

$T(n-1) = T(0) + \log n!$

$T(n-1) = 1 + \log n!$ ⟶ $\theta(n \log n)$

$T(n) = T(n-1) + 1$ ——— $O(n)$

$T(n) = T(n-1) + n$ ——— $O(n^2)$

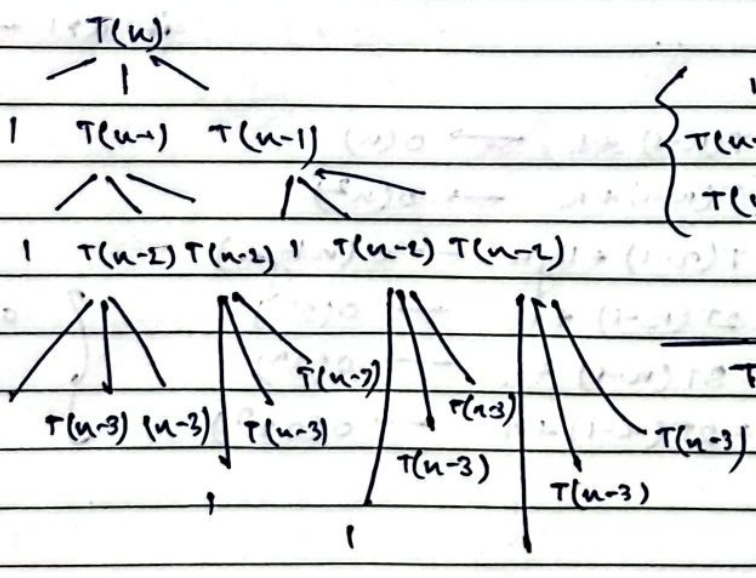
$T(n) = T(n-1) + \log n$ ——— $O(n \log n)$

$T(n) = T(n-1) + n^2$ ——— $O(n^3)$

$T(n) = T(n-2) + 1$ ——— $n/2$ ——— $O(n)$

$T(n) = T(n-100) + n$ ——— $O(n^2)$

$$T(n) = \begin{cases} 1 & , n = 0 \\ 2T(n-1) + 1 & , n > 0 \end{cases}$$




```
T(n) — Algorithm Test(int n)
{
  if (n > 0)
  {
    1      — printf("%d", n);
    T(n-1) — Test(n-1);
    T(n-1) — Test(n-1);
  }
}
```

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

$1+2^1+2^2+2^3+\cdots+2^k$

$= 2^{k+1} - 1$

Assume $n-k=0$

$n=k$

$2^{n+1}-1 \longrightarrow O(2^n)$

$$T(n) = \begin{cases} 1 & ; n=0 \\ 2T(n-1)+1 & ; n>0 \end{cases}$$

$T(n) = 2T(n-1)+1$ —— ①

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

$= 2^2 T(n-2) + 2 + 1$ —— ②

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

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

⋮

$T(n) = 2^k T(n-k) + 2^{k-1} + 2^{k-2} + \cdots 2^2 + 2 + 1$ —— ④

Assume $n-k=0$

$n=k$

$2^2 T(0) + 1 + 2 + 2^2 + \cdots 2^{k-1}$

$= 2^2 \times 1 + 2 - 1$

$= 2^2 + 2^n - 1$

$= 2^{2+1} - 1 \longrightarrow O(2^n)$

Master theorem for decreasing function

$T(n) = aT(n-b) + f(n)$ → General form for RR

$a>0,\ b>0$ and $f(n) = O(n^k)$ where $k \geq 0$

if $a=1 \longrightarrow O(n^{k+1})$

$O(n * f(n))$

if $a>1 \longrightarrow O(n^k a^n)$

$T(n) = T(n-1) + 1 \longrightarrow O(n)$

$T(n) = T(n-1) + n \longrightarrow O(n^2)$

$T(n) = T(n-1) + \log n \longrightarrow O(n \log n)$

$T(n) = 2T(n-1) + 1 \longrightarrow O(2^n)$

$T(n) = 3T(n-1) + 1 \longrightarrow O(3^n)$

$T(n) = 2T(n-1) + n \longrightarrow O(n 2^n)$

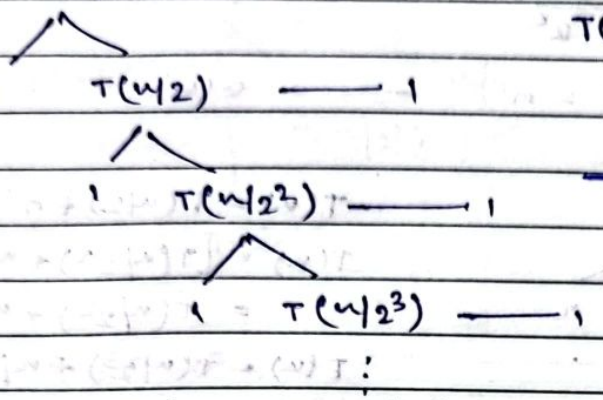
} $O(n^k a^{n/b})$, $O(f(n) a^{n/b})$

Page No:
Date:

Recurrance Relation Dividing Function:

Tree

$$T(n) = \begin{cases} 1 & n=1 \\ T(n/2)+1 & n>1 \end{cases}$$



$n/2^k = 1$

$T(n/2^k)$.

$n = 2^k$

$k = \log_2 n$

$O(\log n)$

```
T(n) ---- Algorithm Test (int n) {
                if (n>1) {
1 ---------- printf("%d", n);
T(n/2) ------ Test (n/2);
                }
T(n) = T(n/2) + 1
```

Substitution

$$T(n) = \begin{cases} 1 & n=1 \\ T(n/2)+1 & n>1 \end{cases}$$

$T(n) = [T(n/2^2)+1]+1$ ——— (1)

$= T(n/2^2)+2$ ——— (2)

$T(n) = T(n/2^3)+3$ ——— (3)

⋮

$T(n) = T(n/2^k)+k$ ——— (4)

Assume, $n/2^k = 1 \Rightarrow n = 2^k \Rightarrow k = \log n$

$T(n) = T(1) + \log n$

$T(n) = 1 + \log n$

$O(\log n)$

Ans.

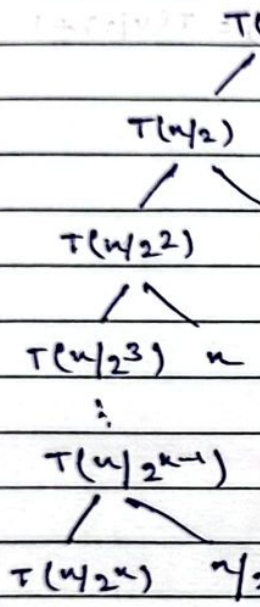
Page No:
Date:

Q $T(n) = n + \frac{n}{2} + \frac{n}{2^2} + \frac{n}{2^3} + \cdots + \frac{n}{2^k}$

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

$= n \sum_{i=0}^{k} \frac{1}{2^k} \to 1$

$= n \times 1 = n \longrightarrow O(n)$



$T(n) = T(n/2) + n$ —— ①

$T(n) = [T(n/2^2) + n/2] + n$

$= T(n/2^2) + n/2 + n$ —— ②

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

⋮

$T(n) = T(n/2^k) + n/2^{k-1} + n/2^{k-2}$

$+ \cdots + n/2 + n$ —— ④

$n/2^k = 1 \Rightarrow n = 2^k \Rightarrow k = \log_2 n$

$T(n) = T(1) + n\left[\frac{1}{2^{k-1}} + \frac{1}{2^{k-2}} + \cdots \frac{1}{2} + 1\right]$

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

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

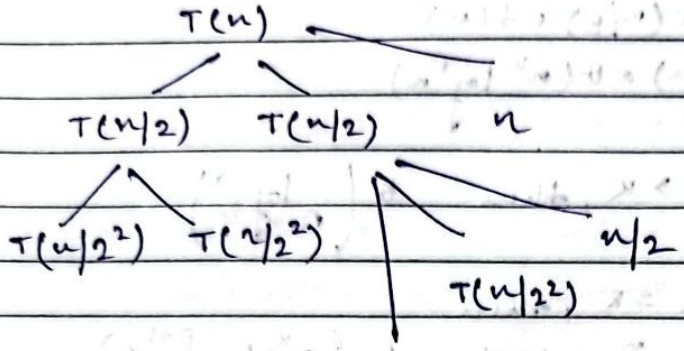
$T(n) \longrightarrow O(n)$

Q
```
Void test(int n) {          ⟶ T(n)
  if(n>1) {
  for(i=0; i<n; i++){       ⎫
    stmt;                   ⎭ n
  }
  test(n/2);                —— T(n/2)
  test(n/2);                —— T(n/2)
}
}
```

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

Page No:
Date:

$$T(n) = \begin{cases} 1 & ; n = 1 \\ 2T(n/2) + n & ; n > 1 \end{cases}$$



Assume $n/2^k = 1$

$n = 2^k$

$k = \log n$

$\longrightarrow O(n \log n)$

Ans.

$$T(n) = 2T(n/2) + n \quad \text{——} ①$$

$$= 2[2T(n/2^2) + n/2] + n$$

$$= 2^2 T(n/2^2) + n + n \quad \text{——} ②$$

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

$$= 2^3 T(n/2^3) + 3n \quad \text{——} ③$$

$$\vdots$$

$$2^k T(n/2^k) + kn$$

$$T(n/2^k) = T(1)$$

$$\therefore n/2^k = 1 \Rightarrow n = 2^k \Rightarrow k = \log n$$

$$T(n) = 2^k T(1) + kn$$

$$= n \times 1 + n \log n$$

$$O(n \log n)$$

Ans.

Master Theorem for Dividing functions :-

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

$a \geq 1, b > 1 \qquad f(n) = \theta(n^k \log^p n)$

Case 1: if $\log_b a > k$, then $\theta(n^{\log_b a})$

Case 2: if $\log_b a = k$, then
- if $p > -1$, $\theta(n^k \log^{p+1} n)$
- if $p = -1$, $\theta(n^k \log\log n)$
- if $p < -1$, $\theta(n^k)$

Case 3: if $\log_b a < k$, if $p \geq 0$, $\theta(n^k \log^p n)$
if $p < 0$ $\theta(n^k)$

Q $T(n) = 2T(n/2) + 1$

$a = 2, b = 2 \qquad \log_2^2 = 1 > k = 0$

$f(n) = \theta(1)$

$= \theta(n^0 \log^0 n)$

$k = 0, p = 0$

Q $T(n) = 4T(n/2) + n \qquad \log_2^4 = 2 > k = 1$

$a = 4, b = 2$

$\rightarrow \theta(n^2)$

Q $T(n) = 8T(n/2) + n \qquad \log_2^8 = 3 > k = 1$

$a = 8, b = 2$

$\rightarrow \theta(n^3)$

Q $T(n) = 9T(n/3) + 1 \qquad \log_3^9 = 2 > k = 0$

$a = 9, b = 3$

$\rightarrow \theta(n^2)$

Page No:
Date:

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

$\log_3^9 = 2 \Rightarrow k = 2 \longrightarrow \theta(n^2)$

Q. $T(n) = 2T(n/2) + n^1$

$\log_2^2 = 1 \Rightarrow k = 1 \quad p = 0 \longrightarrow \theta(n \log n)$

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

$\log_2^4 = 2 \quad k = 2 \longrightarrow \theta(n^2 \log n)$

Q. $T(n) = 4T(n/2) + n^2 \log n$

$\log_2^4 = 2 \quad k = 2 \longrightarrow \theta(n^2 \log^2 n)$

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

$\log_2^2 = 1 \quad k = 1 \quad p = -1 \longrightarrow \theta(n \log\log n)$

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

$\log_2^2 = 1 \quad k = 1 \quad p = -2 \longrightarrow \theta(n)$

Q. $T(n) = T(n/2) + n^2$

$\log_2^1 = 0 < k = 2 \longrightarrow \theta(n^2)$

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

$\log_2^2 = 1 < k = 2 \longrightarrow \theta(n^2 \log^2 n^0)$

Q. $T(n) = 4T(n/2) + n^3$

$\log_2^4 = 2 < k = 3 \longrightarrow \theta(n^3)$

Page No:
Date:

Root function

$$T(n) = \begin{cases} 1 & , n = 2 \\ T(\sqrt{n}) + 1 & , n > 2 \end{cases}$$

$T(n) = T(\sqrt{n}) + 1$

$T(n) = T(n^{1/2}) + 1$ —— (1)

$T(n) = T(n^{1/2^2}) + 2$ —— (2)

$T(n) = T(n^{1/2^3}) + 3$ —— (3)

⋮

$T(n) = T(n^{1/2^n}) + n$ —— (4)

Assume, $n = 2^m$ ⟶ $m = \log_2 n$

$T(2^m) = T(2^{m/2^k}) + k$

Assume, $T(m/2^k) = T(2^1)$

$\therefore m/2^k = 1$

$\Rightarrow m = 2^k$

$k = \log m$

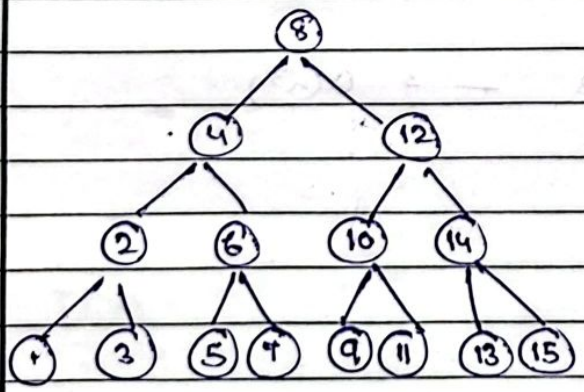
$k = \log\log_2 n$

$\theta(\log\log_2 n)$

```
T(n) —— void Test (int n){
            if (n>2) {
1 —— stmt;
T(√n) —— Test(√n);
            }
        }
T(n) = T(√n) + 1
```

Binary Search ⟶ must be in sorted order

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	3	6	8	12	14	17	25	29	31	36	42	47	53	55	62



min ⟶ $O(1)$

max ⟶ $O(\log n)$

```
int BinSearch (A, n, key){
    l=1, h=n;
    while (l ≤ h) {
        mid = (l+h)/2;
        if (key == A[mid])
            return mid;
        if (key < A[mid])
            h = mid-1;
        else
            l = mid+1;
    } return 0;
}
```

Page No:
Date:

Recursive algorithm for binary search

$$T(n) = \begin{cases} 1 & , n=1 \\ T(n/2)+1 & , n>1 \end{cases}$$

```
T(n) — Algorithm RBinSearch (l,h,key)
{
  if (l==h){
    if (A[l] == key) {
      return l;
    else
      return 0;
  } else {
1 —  mid = (l+h)/2;
1 —  if (key == A[mid])
       return mid;
1 —  if (key < A[mid])
       return RBinSearch
       (l, mid-1, key);
T(n/2) else
       return RBinSearch
       (mid+1, h, key);
  }
}
```

Merge Sort

* 2 way merge sort (iterative)

	1	2	3	4	5	6	7	8
A	9	3	7	5	6	4	8	2

(1st) 3,9 5,7 4,6 2,8 → (n)

(2nd) 3579 2468 → (n)

(3rd) 23456789 → (n)

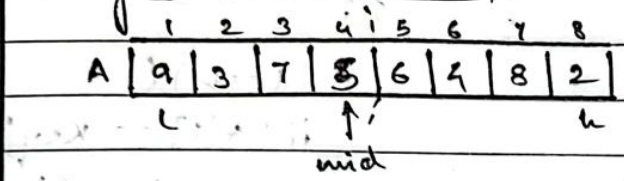
$8 = 2^3 \rightarrow \log_2 8 = 3$

No. of passes = $\log n$

$\theta(n \log n)$

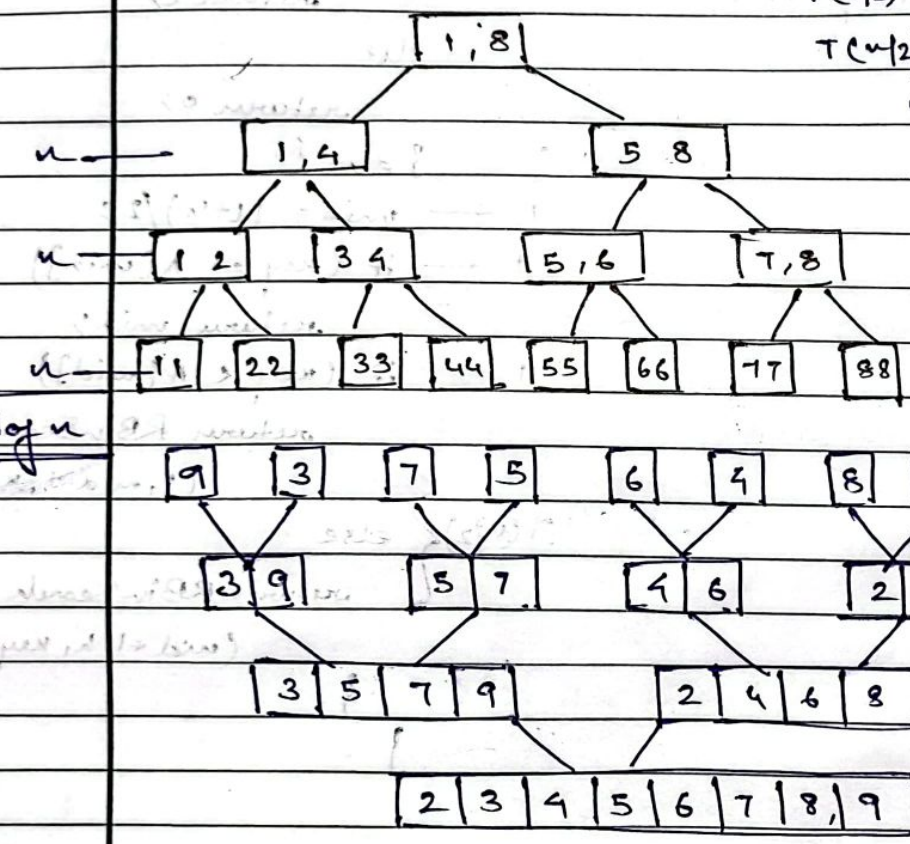
```
Algorithm Merge (A,B,m,n)
{
  i=1; j=1; k=1;
  while (i<=m && j<=n){
    if (A[i] < B[i])
      c[k++] = A[i++];
    else
      c[k++] = B[j++];
  }
  for ( ; i<=m; i++)
    c[k++] = A[i];
  for ( ; i<=n; j++)
    c[k++] = B[j]
}
```

Merge Sort (Recursive)




```
T(n) Algorithm MergeSort(l,h){
     if(l<h){
①——   mid = (l+h)/2;
T(n/2)—— MergeSort(l,mid);
T(n/2)—— MergeSort(mid+1,h);
n——    Merge(l,mid,h);
     }
}
```

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



$$T(n) = \begin{cases} 1 & , n=1 \\ 2T(n/2)+n & , n>1 \end{cases}$$

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

$$T(n) = \begin{cases} 1 & , n=1 \\ 2T(n/2)+n & , n>1 \end{cases}$$

$a=2, \; b=2, \; f(n)=n$

$\log_b^a = \log_2^2 = 1 \qquad n^k = n^1$

$\log_2^2 = k$

$\theta(n \log n)$ Ans.

- **Pros:** ① Large size list ② Linked list ③ External Sorting Support ④ Stable
- **Cons** ① Extra Space (not inplace Sort) ② No small problem ③ Recursive

Page No:
Date:

Quick Sort

A | 10 | 80 | 30 | 90 | 40 |

↑ Pivot

- Compare 10 with the pivot, and since its less than pivot arrange it accordingly.

| 10 | 80 | 30 | 90 | 40 |

- Compare 80 with the pivot, it is greater than the pivot

| 10 | 80 | 30 | 90 | 40 |

80 > Pivot → Hence No Swap, Move to 30

- Compare 30 with pivot, it is less than pivot so arrange it accordingly.

| 10 | 80 | 30 | 90 | 40 |

30 < pivot → Swap 80 with 30

| 10 | 80 | 30 | 90 | 40 |

- Compare 90 with the pivot, it is greater than pivot

| 10 | 30 | 80 | 90 | 40 |

90 > Pivot → Hence No Swap, End traversal

- Arrange the pivot in its correct position

| 10 | 30 | 80 | 90 | 40 |

Swap 80 & 40 →

PIVOT

| 10 | 30 | 40 | 90 | 80 |

↓

| 10 | 30 | 40 | 80 | 90 |

Page No:
Date:

~~Partition (l,h){~~
~~pivot = a[l];~~
~~i = l-1;~~

```
Partition (l, h) {
    pivot = A[l];
    i=l; j=h;
    while (i<j){
    do {
        i++;
    } while (A[i] ≤ pivot) {
      do {
          j--;
    } while (A[i] > pivot) {
      if (i<j){
          swap(A[i], A[j]);
        }
    }

QuickSort(l,h) {
  if (l < h) {
    j = Partition (l,h);
    QuickSort (l,j);
    QuickSort (j+1, h);
  }
}
```

Best case → O(n log n)
Worst case → O(n²)

GREEDY METHOD

- Mainly used for optimization problems
  - Greedy method
  - Dynamic Programming
  - Branch and Bound

Algorithm Greedy (a, n)
{
for i=1 to n do
{
x = select (a);
if feasible (x) Then
{
solution = solution + x;
}
}
}

n=5

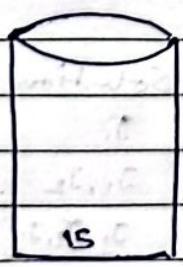
a	a_1	a_2	a_3	a_4	a_5
	1	2	3	4	5

Knapsack Problem

n=7
m=15

Objects:	1	2	3	4	5	6	7
Profit:	10	5	15	7	6	18	3
Weight:	2	3	5	7	1	4	1
P/W:	5	1.3	3	1	6	4.5	3
	(1	2/3	1	0	1	1	1)
	x_1	x_2	x_3	x_4	x_5	x_6	x_7

Constraint
$\sum x_i w_i \leq m$
Objective
max $\sum x_i P_i$



15-1=14
14-2=12
12-4=8
8-5=3
3-1=2
2-2=0

$= 1\times2 + 2/3\times3 + 1\times5 + 0\times7 + 1\times1 + 1\times4 + 1\times1$
$= 2+2+5+0+1+4+1$
$= 15$

$\sum x_i w_i = 15$

Profit = $1\times10 + 2/3\times5 + 1\times15 + 1\times6 + 1\times18 + 1\times3$
$= 54.6$

$\sum x_i P_i = 54.6$

Goal optimize
Profit ↑
Cost minimize
Time ↓

Page No:
Date:

Job Sequencing with Deadlines

n=5

Jobs	J_1	J_2	J_3	J_4	J_5
Profits	20	15	10	5	1
Deadlines	2	2	1	3	3

$20 + 15 + 5 = 40$

J_1 J_2 J_4

$0 \xrightarrow{J_2} 1 \xrightarrow{J_1} 2 \xrightarrow{J_4} 3$

$\{J_2 \rightarrow J_1 \rightarrow J_4\}$

$\{J_1 \rightarrow J_2 \rightarrow J_4\}$

0 — 1 — 2 — 3

Jobs Consider	Slot Assign	Solution	Profit
–	–	ϕ	0
J_1	[1,2]	J_1	20
J_2	[0,1] [1,2]	J_1, J_2	20+15
J_3	[0,1] [1,2]	J_1, J_2	20+15
J_4	[0,1] [1,2] [2,3]	J_1, J_2, J_4	20+15+5
J_5	[0,1] [1,2] [2,3]	J_1, J_2, J_4	40

n=7

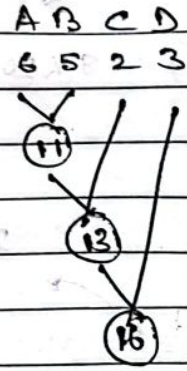
Jobs	J_1	J_2	J_3	J_4	J_5	J_6	J_7
Profits	35	30	25	20	15	12	5
Deadlines	3	4	4	2	3	1	2

$0 \xrightarrow[20]{J_4} 1 \xrightarrow[25]{J_3} 2 \xrightarrow[35]{J_1} 3 \xrightarrow[30]{J_2} 4 \rightarrow 110$

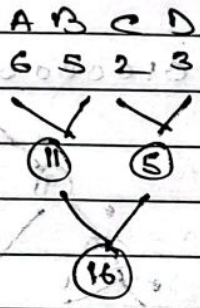
Jobs Consider	Slot Assign	Solution	Profit
J_1	[2,3]	J_1	35
J_2	[2,3] [3,4]	J_1, J_2	35+30
J_3	[2,3] [3,4] [1,2]	J_1, J_2, J_3	35+30+25
J_4	[2,3] [3,4] [1,2] [0,1]	$J_1 J_2 J_3 J_4$	35+30+25+20
J_5	"	"	110
J_6	"	"	110
J_7	"	"	110

Optimal Merge Pattern

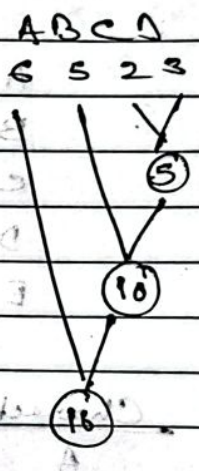
List → A B C D
Size → 6 5 2 3



11+13+16
= 40

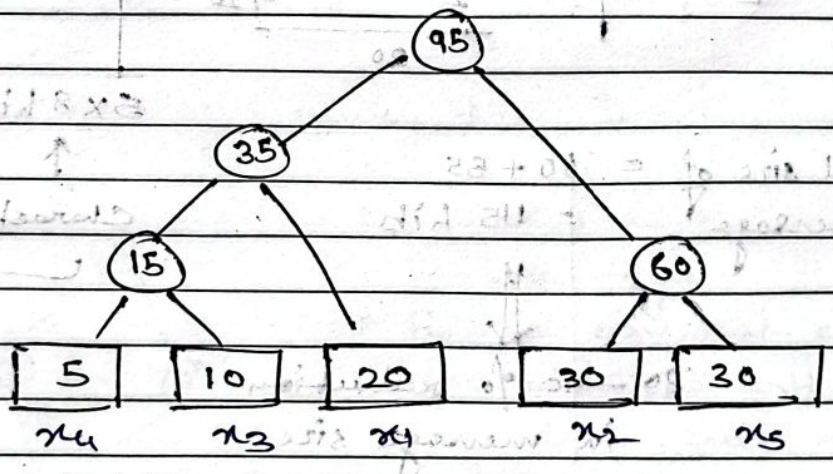


11+5+16
= 32



5+10+16
= 31

Lists →	x_1	x_2	x_3	x_4	x_5
sizes →	20	30	10	5	30



15+60+35+95 = 205

Total cost of merging = $\sum d_i * x_i$

Page No:
Date:

Huffmann Coding

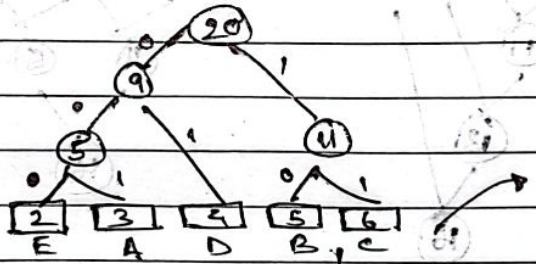
Message → BCCABBDDAECCBBAEDDCC

length → 20

ASCII → 8 bits

A	65	01000001
B	66	
C	67	
D	68	
E	69	

8 × 20 = 160 bits



20 × 3 = 60 bits

Character	Count/Frequency		Code
A	3	3/20	000
B	5	5/20	001
C	6	6/20	010
D	4	4/20	011
E	2	2/20	100
	20		

5 × 8 bits ↑ characters

5 × 3 ↑ codes

40 + 15 = 55 bits

Total size of message = 60 + 55 = 115 bits

30 – 40 % reduction in message size.

Char	Count	Code	
A	3	001	3×3 = 9
B	5	10	5×2 = 10
C	6	11	6×2 = 12
D	4	01	4×2 = 8
E	2	000	2×3 = 6
8×5 = 40 bits	20	12 bits	45 bits

msg → 45 bits

tree table → 52 bits

97 bits

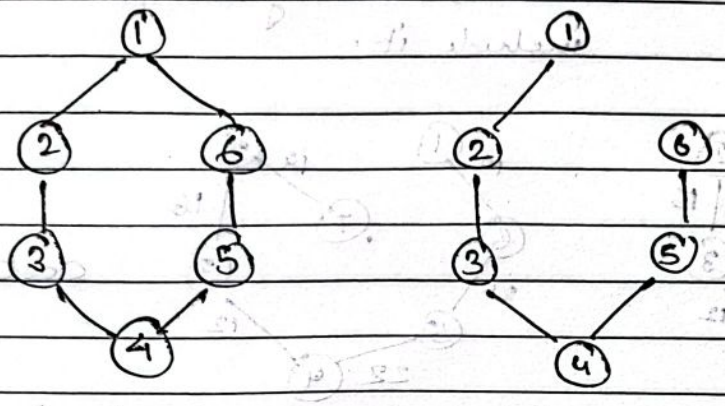
$\sum di * fi$

= 3×2 + 3×3 + 2×4 + 2×5 + 2×6

= 45 bits

Page No:
Date:

Minimum Cost Spanning Tree



$S \subseteq G$

$S = (V' E')$

$V' = V$

$|E'| = |V| - E$

$G = (V, E)$

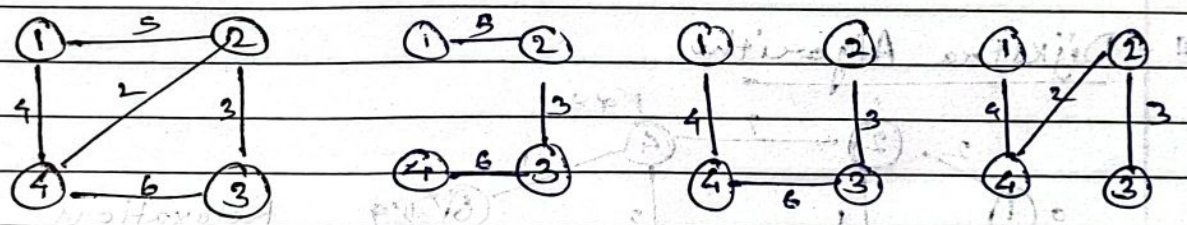
$V = \{1, 2, 3, 4, 5, 6\}$

$E = \{(1,2)(3,4) \dots\}$

$|V| = n = 6$

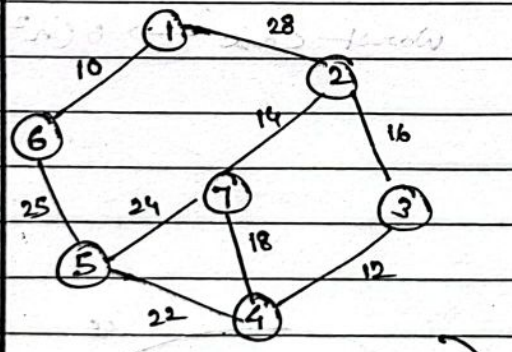
$|V| - 1 = 5$

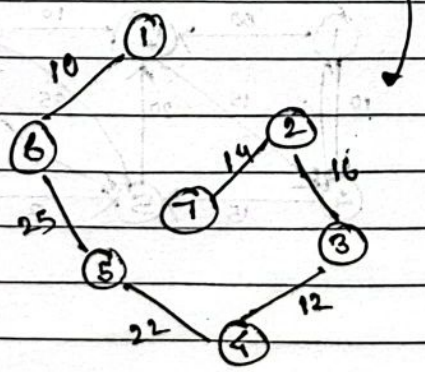
Q



cost = 14 cost = 13 cost = 9

Minimum Cost of Spanning tree through Prims Algorithm

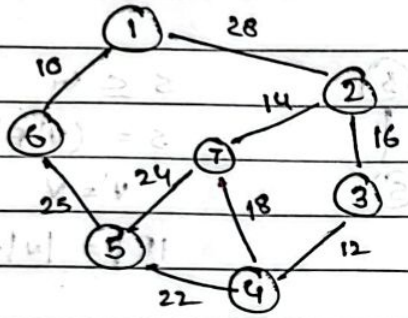


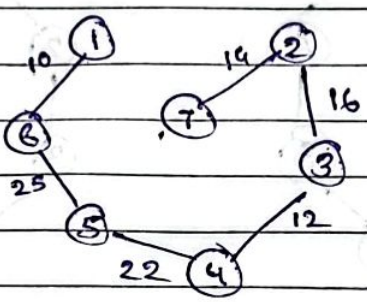


* Initialize an MST with randomly chosen vertex
* Find all edges that connects the tree with new vertices. From the edges found, select min. edge and add to the tree

Cost = 99

Kruskal's Method — Always select the minimum cost but if it is forming a cycle don't include it.



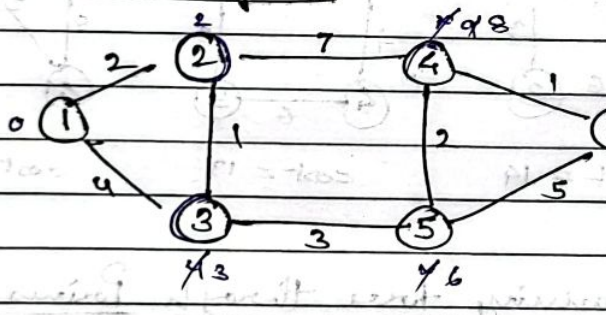


Cost = 99

$\theta(|V||E|)$

$\theta(n \cdot e) = \theta(n^2)$

Dijkstra Algorithm



Relaxation

if $(d[u] + c(u,v) < d[v])$

$d[v] = d[u] + c(u,v)$

v	d[v]
2	2
3	3
4	8
5	6
6	8

Worst Case $\longrightarrow \theta(n^2)$

Starting vertex = 1

Selected vertex	2	3	4	5	6
1	50	45	(10)	∞	∞
5	50	45	10	(25)	∞
2	(45)	45	10	25	∞
3	45	(45)	10	25	∞
6	45	45	10	25	(∞)