

Algorithm

- Design part of project
- Should have domain knowledge

- Any language
- H/w and OS independent
- Analyse

Program

- Implementation part of project
- Programmers

- Programming language
- H/w and OS dependent
- Testing

Priiori Analysis

- Algorithm
- Independent of language
- Hardware independent
- Time & Space function

Posteriori Analysis

- Program
- Language dependent
- Hardware dependent
- Watch time & Bytes

Characteristics of algorithm

- I O/p \rightarrow 0 or more
- II O/p \rightarrow at least 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; -----②
b = temp; -----③
}

```

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

Space

a - 1

b - 1

temp - 1

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

Frequency Count Method

A	8	3	9	7	2
	0	1	2	3	1

Space

$$A \rightarrow n$$

$$n \rightarrow 1$$

$$s \rightarrow 1$$

$$i \rightarrow 1$$

$$s(n) = n + 3$$

$$\rightarrow O(n)$$

Q (Sum of all elements in arr)

Algorithm Sum (A, n) {

$$s = 0 \quad \text{①}$$

for ($i=0$; $i < n$; $i++$) { $\rightarrow \frac{2n+2}{2} = n+1$

$$s = s + A[i]; \quad \text{②}$$

} \rightarrow return s; $\rightarrow \text{③}$

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

Q. Sum of two matrices

Algorithm Add (A, B, n) {

for ($i=0$; $i < n$; $i++$) { $\rightarrow 2n+2 \rightarrow n+1$

for ($j=0$; $j < n$; $j++$) { $\rightarrow n(n+1)$

$$c[i, j] = A[i, j] + B[i, j]; \rightarrow n^2$$

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

$$O(n^2)$$

Space

$$A \rightarrow n \times n \rightarrow n^2$$

$$B \rightarrow n \times n \rightarrow n^2$$

$$c \rightarrow n \times n \rightarrow n^2$$

$$n \rightarrow 1$$

$$i \rightarrow 1$$

$$j \rightarrow 1$$

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

$$\Rightarrow O(n^2)$$

Q Multiplication of two matrices

Algorithm multiply (A, B, n) {

for ($i=0$; $i < n$; $i++$) { $\rightarrow 2n+2 \rightarrow n+1$

for ($j=0$; $j < n$; $j++$) { $\rightarrow n(n+1) \rightarrow n^2+n$

$$c[i, j] = 0; \rightarrow n(n) \rightarrow n^2$$

for ($k=0$; $k < n$; $k++$) { $\rightarrow n \times n \times (n+1)$

$$c[i, j] = c[i, j] + A[i, k] * B[k, j]; \rightarrow n \times n \times n$$

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

Time $\rightarrow O(n^3)$

Space

$$A - n^2$$

$$B - n^2$$

$$C - n^2$$

$$n - 1$$

$$i - 1$$

$$j - 1$$

$$k - 1$$

$$S(n) = 3n^2 + 4$$

$$\rightarrow O(n^2)$$

Q1 $\text{for}(i=0; i < n; i++) \rightarrow n+1$

{

$$\text{start;} \rightarrow n$$

{

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

~~O(n)~~ $\cdot O(n)$

Q2 $\text{for}(i=n; i > 0; i--) \rightarrow n-1$

{

$$\text{start;} \rightarrow n$$

{

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

$O(n)$

Q3 $\text{for}(i=1; i < n; i=i+2)$

{

$$\text{start;} \rightarrow \frac{n}{2}$$

{

$$O(n)$$

Q4 $\text{for}(i=0; i < n; i++) \rightarrow n+1$

{

$\text{for}(j=0; j < n; j++) \rightarrow n(n+1)$

{

$$\text{start;} \rightarrow n(n)$$

{

$$1+2+3+4+\dots+n$$

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

$$O(n^2)$$

i	j	stmt
0	0	0
1	0	1
2	0	2
3	0	3
2	1	
1	1	
0	1	
2	2	
1	2	
0	2	
3	2	
2	3	
1	3	
0	3	

Q5 $P = 0$ $\text{for } (i=1; P < n; i++) \{$ $P = P + i;$

{

i	P
1	$0+1 = 1$
2	$1+2 = 3$
3	$1+2+3 = 6$
⋮	⋮

Assume $P > n$

$K \cdot 1+2+\dots+n = K(K+1)/2$

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

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

$K^2 > n$

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

Q6 $\text{for } (i=1; i < n; i=i*2)$

{

start;

}

$1 \times 2 = 2$

$2 \times 2 = 2^2$

Assume $i >= n$

$\because i = 2^K$

$\therefore 2^K >= n$

$2^K = n$

$K = \log_2 n$

i

$O(1)$

$O(2)$

$O(4)$

$O(8)$

$O(16)$

$O(32)$

$O(64)$

$O(128)$

$O(256)$

$O(512)$

$O(1024)$

$O(2048)$

$O(4096)$

$O(8192)$

$O(16384)$

$O(32768)$

$O(65536)$

$O(131072)$

$O(262144)$

$O(524288)$

$O(1048576)$

$O(2097152)$

$O(4194304)$

$O(8388608)$

$O(16777216)$

$O(33554432)$

$O(67108864)$

$O(134217728)$

$O(268435456)$

$O(536870912)$

$O(1073741824)$

$O(2147483648)$

$O(4294967296)$

$O(8589934592)$

$O(17179869184)$

$O(34359738368)$

$O(68719476736)$

$O(137438953472)$

$O(274877906944)$

$O(549755813888)$

$O(1099511627776)$

$O(219902325552)$

$O(439804651104)$

$O(879609302208)$

$O(1759218604416)$

$O(3518437208832)$

$O(7036874417664)$

$O(14073748835328)$

$O(28147497670656)$

$O(56294995341312)$

$O(112589990682624)$

$O(225179981365248)$

$O(450359962730496)$

$O(900719925460992)$

$O(1801439850921984)$

$O(3602879701843968)$

$O(7205759403687936)$

$O(14411518807375872)$

$O(28823037614751744)$

$O(57646075229503488)$

$O(115292150459006976)$

$O(230584300918013952)$

$O(461168601836027904)$

$O(922337203672055808)$

$O(1844674407344111616)$

$O(3689348814688223232)$

$O(7378697629376446464)$

$O(14757395258752892928)$

$O(29514790517505785856)$

$O(59029581035011571712)$

$O(11805916207002314344)$

$O(23611832414004628688)$

$O(47223664828009257376)$

$O(94447329656018514752)$

$O(188894659312037029504)$

$O(377789318624074059008)$

$O(755578637248148118016)$

$O(1511157274496296236032)$

$O(3022314548992592472064)$

$O(6044629097985184944128)$

$O(12089258195970369888256)$

$O(24178516391940739776512)$

$O(48357032783881479553024)$

$O(96714065567762959106048)$

$O(193428131135525918212096)$

$O(386856262270651836424192)$

$O(773712524541303672848384)$

$O(1547425049082607345696768)$

$O(3094850098165214691393536)$

$O(6189700196330429382787072)$

$O(12379400392660858765574144)$

$O(24758800785321717531148288)$

$O(49517601570643435062296576)$

$O(99035203141286870124593152)$

$O(19807040628257374024986304)$

$O(39614081256514748049972608)$

$O(79228162513029496099945216)$

$O(15845632522605899219989032)$

$O(31691265045211798439978064)$

$O(63382530090423596879956128)$

$O(126765060180847193759912256)$

$O(253530120361694387519824512)$

$O(507060240723388775039649024)$

$O(101412048144677755007929048)$

$O(202824096289355510015858096)$

$O(405648192578711020031716192)$

$O(811296385157422040063432384)$

$O(162259277031484408012664776)$

$O(324518554062968816025329552)$

$O(649037108125937632050659104)$

$O(1298074216259688664101318208)$

$O(2596148432519377328202636416)$

$O(5192296865038754656405272832)$

$O(10384593730077089312810545664)$

$O(20769187460154178625621091328)$

$O(41538374920308357251242182656)$

$O(83076749840616714502484365112)$

$O(166153499681233429004968730224)$

$O(332306999362466858009937460448)$

$O(664613998724933716019874920896)$

$O(1329227997448667432039749811792)$

$O(2658455994897334864079498123584)$

$O(5316911989794669728158996247168)$

$O(10633823979589339456317992494336)$

$O(21267647959178678912635984988672)$

$O(42535295918357357825271969977344)$

$O(85070591836714715650543939954688)$

$O(170141183674285431301087879909376)$

$O(340282367348570862602175759818752)$

$O(680564734697141725204351519637504)$

$O(136112946939428345040870303927508)$

$O(272225893878856690081740607855016)$

$O(544451787757713380163481215710032)$

$O(108890357551542676032664423542064)$

$O(217780715103085352065328846784128)$

$O(435561430206170704130657693568256)$

$O(871122860412341408261315387136512)$

$O(1742245720824682816522630774273024)$

$O(3484491441649365633045261548546048)$

$O(6968982883298731266090523097092096)$

$O(13937965766597462532181046194184192)$

$O(27875931533194925064362092388368384)$

$O(55751863066389850128724184776736768)$

$O(111503726132779700257448369553535536)$

$O(22300745226555940051489673910707112)$

$O(44601490453111880102979347821414224)$

$O(89202980906223760205958695642828448)$

$O(178405961812447520411917391285656896)$

$O(356811923624895040823834782571313792)$

$O(713623847249790081647669565142627584)$

$O(1427247694495980163295339130285255168)$

$O(2854495388991960326590678260570510336)$

$O(5708990777983920653181356521141020672)$

$O(11417981555967841306362732542282041344)$

$O(22835963111935682612725465084564082688)$

$O(45671926223871365225450930168128165376)$

$O(91343852447742730450901860336256321552)$

$O(182687704895485460901803720672512643104)$

$O(36537540979097092180360744134502528608)$

$O(73075081958194184360721488268755057216)$

$O(14615016391638836872144297653751014432)$

$O(29230032783277673744288595307502028864)$

$O(58460065566555347488577190615004057728)$

$O(11692013113311069497715438123000811556)$

$O(23384026226622138995430876246001623112)$

$O(46768052453244277990861752492003246224)$

$O(93536104906488555981723504984006492448)$

$O(1870722098129771119634470099$

Q8 $\text{for}(i=n; i>=1; i=i/2)$

i/n

$i/2$

Start:

}

Assume $i \in 1$

$$n/2^k < 1$$

$$n/2^k = 1$$

$$n = 2^k$$

$$k = \log_2 n$$

$\rightarrow O(\log_2 n)$

Q9 $\text{for}(i=0; i< n; i++)$

}

Start:

}

$i < i < n$

$i < i > -n$

$O(\sqrt{n})$

$$i^2 = n$$

$$i = \sqrt{n}$$

Q10 $\text{for}(i=0; i<n; i++) \{$

Start:

}

$\text{for}(j=0; j<n; j++) \{$

Start:

}

$\} \rightarrow n$

$\} \rightarrow n$

$\} \rightarrow n$

$\} \rightarrow n$

Q11 $P = 0$

$\text{for}(i=1; i<n; i=i*2) \{$

$P++;$

$\rightarrow P = \log n$

}

$\text{for}(j=1; j < p; j=j*2) \{$

Start:

$\rightarrow \log P$

}

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

Q12 $\text{for}(i=0; i<n; i++) \{ \rightarrow n$

$\text{for}(j=1; j < n; j=j*2) \{ \rightarrow n * \log n$

Start: $\rightarrow n * \log n$

}

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

$\rightarrow O(n \log n)$

Analysis of if and while

Q i=0; — ①
 while (i < n) — (n+1)
 stmt; — ②
 i++; — ③

$3n + 2 \rightarrow O(n)$

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

Terminate
 $a > b$
 $2^n > b$
 $K = \log_2 b$ } (gⁿ)

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

i	1	2	3	4	5	...
k	1	1+1=2	1+2=3	1+3=4	1+4=5	...

$m \times (2+2+3+4+\dots+n) = m(m+1)/2$

(Complexity)

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

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

Q GCD of two numbers

while (m != n)

if (m > n)

m=m-n;

else

n=n-m;

}

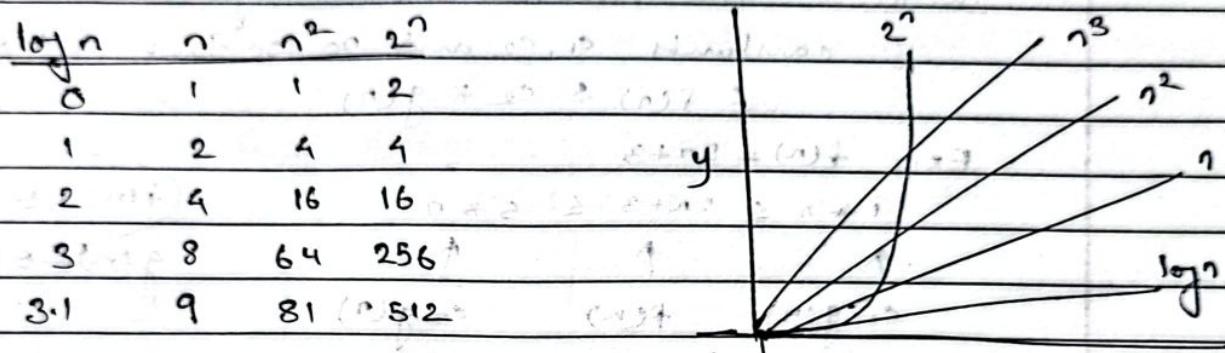
12	2	K/2
10	2	
8	2	
6	2	
1	2	

$n/2$
 $O(n)$

① Types of time functions:

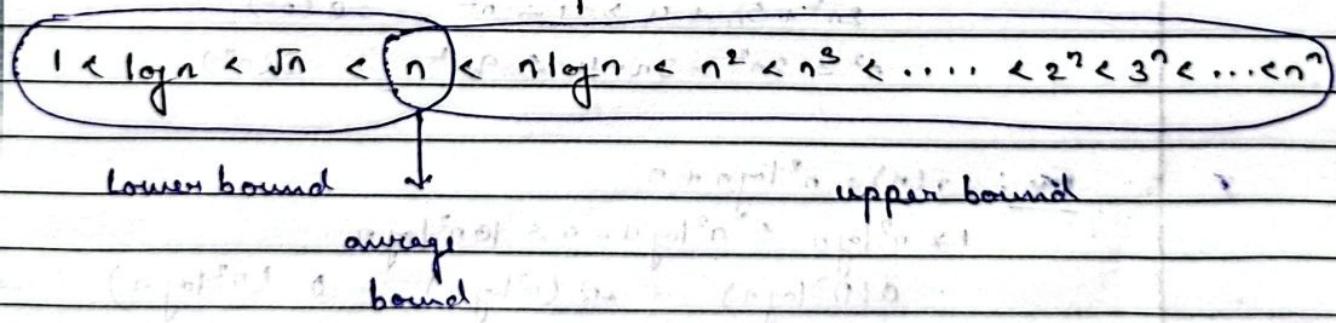
$O(1)$ → Constant	$O(\log n)$ → Logarithmic
$O(n)$ → Linear	$O(n^2)$ → Quadratic
$O(n^3)$ → Cubic	$O(2^n)$ → Exponential

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



Asymptotic Notation:

- big-oh upperbound
- big-o-mega lowerbound
- Theta average bound.



- Big Oh — The function $f(n) = O(g(n))$ iff ∃ the constants c and n_0 such that $f(n) \leq c \cdot g(n)$

↑ n_1, n_0

Nearest one most appropriate

Ex: $f(n) = 2n + 3$ ↗ let

$f(n) = O(n)$ ↗

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

$f(n) = O(n^2)$ ↗

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

$f(n) = O(2^n)$ ↗

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

$f(n) = O(\log n)$ ↗

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

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

$$\text{Ex. } f(n) = 2n+3$$

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

$$c * g(n)$$

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

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

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

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

$$\text{Ex. } f(n) = 2n+3$$

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

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

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

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

Q

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

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

$$2n^2 + 3n + 4 \leq 9n^2$$

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

$$\text{Ex: } f(n) = 2n^2 + 3n + 4$$

$$2n^2 + 3n + 4 \geq 1 * n^2$$

$$n^2 \leq 2n^2 + 3n + 4 \leq 9n^2$$

$$\Theta(n^2)$$

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

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

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

$$\text{Ex: } f(n) (= n!) = n \times (n-1) \times (n-2) \times \dots \times 3 \times 2 \times 1$$

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

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

$$\therefore (n^2 \log n)^2 + \Theta(n^2)$$

$$\Theta(1)$$

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

$$\log(1 \times 2 \times 3 \times \dots \times n) \leq \log(n \times n \times n \dots \times n) \\ 1 \leq \log n! \leq \log n^{\frac{n}{2}} \\ \Theta(1) \\ O(n \log n)$$

Properties of asymptotic notations

(I) General properties

If $f(n)$ is $\alpha(g(n))$ then $a(f(n))$ is $\alpha(g(n))$

$$\text{Ex: } f(n) = 2n^2 + 5 \text{ is } O(n^2) \\ \text{then } 7 \cdot f(n) = 7(2n^2 + 5) \\ = 14n^2 + 35 \rightarrow O(n^2)$$

(II) Reflexive

If $f(n)$ is given then $f(n)$ is $\alpha(f(n))$

$$\text{Ex: } f(n) = n^2 \rightarrow O(n^2)$$

(III) Transitive

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

$$\text{Ex: } f(n) = n, g(n) = n^2, h(n) = n^3$$

n is $\alpha(n^2)$ and n^2 is $\alpha(n^3)$

then n is $\alpha(n^3)$

(IV) Symmetric

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

$$\text{Ex: } f(n) = n^2, g(n) = n^2$$

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

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

(V) Transpose Symmetric

If $f(n) = \alpha(g(n))$ then $g(n) = \omega(f(n))$

$$\text{Ex: } f(n) = n, g(n) = n$$

then n is $\alpha(n^2)$ 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 $d(n) = \Theta(g(n))$
 then $f(n) + d(n)$

(Ex)

$$\begin{aligned}f(n) &= n = O(n) \\d(n) &= n^2 = \Theta(n^2) \\ \therefore f(n) + d(n) &= n + n^2 \\&= O(n^2)\end{aligned}$$

* 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

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

$$\text{Avg. time} = (n+1)/2$$

* Binary Search

Best case time: $O(1)$

Worst case: $\log n$ $O(\log n)$

■ Disjoint Sets

1. Disjoint Sets and Operations
2. Detecting a cycle
3. Graphical representation
4. Array representation
5. Weighted Union & Collapsing first find.

Operations :

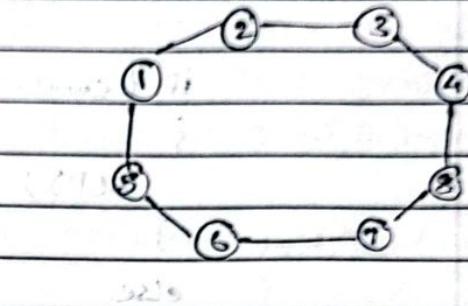
1. Find
2. Union

$$S_1 = \{1, 2, 3, 4\}$$

$$S_2 = \{5, 6, 7, 8\}$$

$$S_1 \cap S_2 = \emptyset$$

$$\left(\begin{array}{l} S_3 = \{1, 2, 3, 4, 5, 6, 7, 8\} \\ S_1 \cup S_2 = \{1, 2, 3, 4, 5, 6, 7, 8\} \end{array} \right)$$



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 P_1, P_2, \dots, P_x

 Apply DAC(P_1), DAC(P_2) ...

 Combine (DAC(A'_1), DAC(P_2))

 }

}

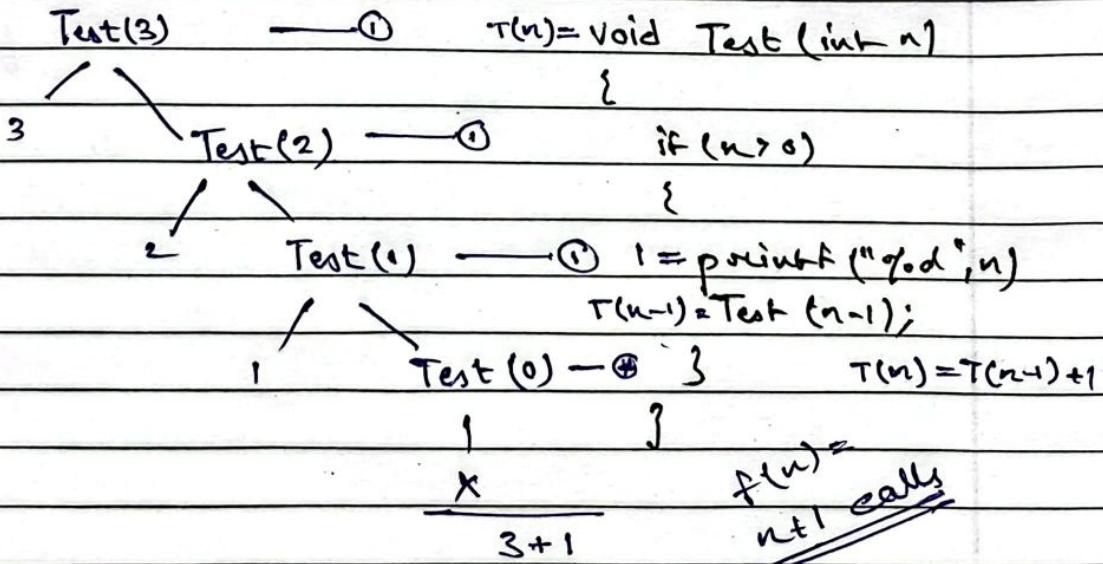
① Binary Search

② Finding maximum and minimum

③ Mergesort

④ Quicksort

⑤ Strassen's Matrix Multiplication



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

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

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

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

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

\vdots K times $T(n) = T(n-k) + k$

Assuming $n-k=0$

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

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

RR: $T(n) = T(n-1) + 1 + (n-1) + \dots + 1 + T(0) \rightarrow O(n^2) - void Test(int n)$

$$RR: T(n) = T(n-1) + 1 + \dots + 1 + T(0) \rightarrow O(n)$$

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

$$T(n)$$

$n+1 \leftarrow$ for ($i=0$; $i < n$; $i++$)

$n \leftarrow pt("of d", n);$

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

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

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

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

$$T(2)$$

$$T(n) = 0 + 1 + 2 + \dots + n + 1 + n = n(n+1)$$

$$2$$

$$T(1)$$

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

$$1 \leftarrow T(0)$$

$$O(n^2)$$

$$(0)^*$$

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

$$T(n) = T(n-1) + n \quad \text{--- (1)}$$

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

$$\therefore T(n) = [T(n-2) + (n-1)] + n = T(n-2) + (n-1) + n \quad \text{--- (2)}$$

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

$$T(n) = [T(n-3) + (n-2)] + (n-1) + n \quad \text{--- (3)}$$

$$T(n) = T(n-3) + (n-2) + (n-1) + n \quad \text{--- (3)}$$

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

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

$$\text{Assume } n-k=0 \Rightarrow n=k \quad \text{--- (4)}$$

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

$$T(n) = \frac{n(n+1)}{2} + 1 \quad \text{--- (4)}$$

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

Q $T(n) \rightarrow \text{void Test (int } n)$ {

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

if ($n>0$) {
for ($i=1; i<n; i=i+2$) {
}}

$T(n)$

$\log(n) \rightarrow \text{printf ("%.d", i);}$

$\log n \quad T(n-1)$

$T(n-1) \rightarrow \text{Test}(n-1);$

$\log(n-1) \quad T(n-2)$

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

$\log(n-2) \quad T(n-3)$

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

$\vdots \quad T(2)$

$\dots + \log 2 + \log 1$

$\log 2 \quad T(1)$

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

$\log_1 \quad T(0)$

$\Rightarrow O(n \log n)$

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

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

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

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

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

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

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

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

$$\text{Now } n-k=0 \Rightarrow n=k$$

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

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

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

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

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

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

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

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

$$T(n) = T(n-100) + n \rightarrow O(n^2)$$

$T(n)$ — Algorithm Test (introduction)

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

if ($n>0$)

$T(n)$

1
|
 $T(n-1)$ $T(n-1)$

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

{ 1 — printf("%d", n);

$T(n-1)$ — Test($n-1$);

$T(n-2)$ — Test($n-1$);

}

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

$$1 + 2^1 + 2^2 + 2^3 + \dots + 2^n = 2^{n+1} - 1$$

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

Assume $n-k=0$

$$n=k$$

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

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

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

$$= 2^2 T(n-2) + 2 + 1 \quad \text{--- (2)}$$

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

$$T(n) = 2^3 T(n-3) + 2^2 + 2 + 1 \quad \text{--- (3)}$$

$$T(n) = 2^k T(n-k) + 2^{k-1} + 2^{k-2} + \dots + 2^2 + 2 + 1 \quad \text{--- (4)}$$

Assume $n-k=0$

$$n=k$$

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

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

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

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

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

Master theorem for decreasing functions

$T(n) = aT(n-b) + f(n) \rightarrow$ general form for RR

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

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

$O(n+kf(n))$

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

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

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

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

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

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

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

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

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

Recurrence Relation Dividing Functions:

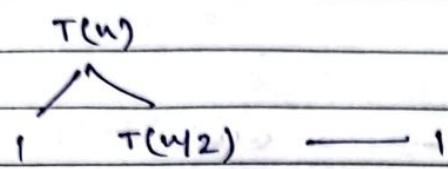
Tree

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

$T(n)$ — Algorithm Test (int n) {
if ($n \geq 1$) {

 1 — printf("%d", n);

$T(n/2)$ — Test($n/2$);



Q

$$T(n) = n + \frac{n}{2} + \frac{n}{2^2} + \frac{n}{2^3} + \dots + \frac{n}{2^K}$$

$$\begin{aligned} T(n) &= n \left[1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^K} \right] \\ &= n \sum_{i=0}^{K-1} \frac{1}{2^i} \\ &= n \times 1 = n \quad \rightarrow O(n) \end{aligned}$$

T(n)

$$\begin{array}{c} / \quad \backslash \\ T(n/2) \quad n \end{array}$$

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

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

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

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

$$\begin{array}{c} / \quad \backslash \\ T(n/2^2) \quad n \end{array}$$

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

$$+ \dots + n/2 + n \quad \text{--- ④}$$

$$\therefore n/2^K = 1 \Rightarrow n = 2^K \Rightarrow K = \log_2 n$$

$$T(n/2^{K-1})$$

$$\begin{array}{c} / \quad \backslash \\ T(n/2^K) \quad n/2^{K-1} \end{array} \quad T(n) = T(1) + n \left[\frac{1}{2^{K-1}} + \frac{1}{2^{K-2}} + \dots + \frac{1}{2} \right]$$

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

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

$$T(n) \rightarrow O(n)$$

Q

Void test(int n) { $\rightarrow T(n)$

if (n > 1) {

for (i=0; i<n; i++) {

 stmt;

}

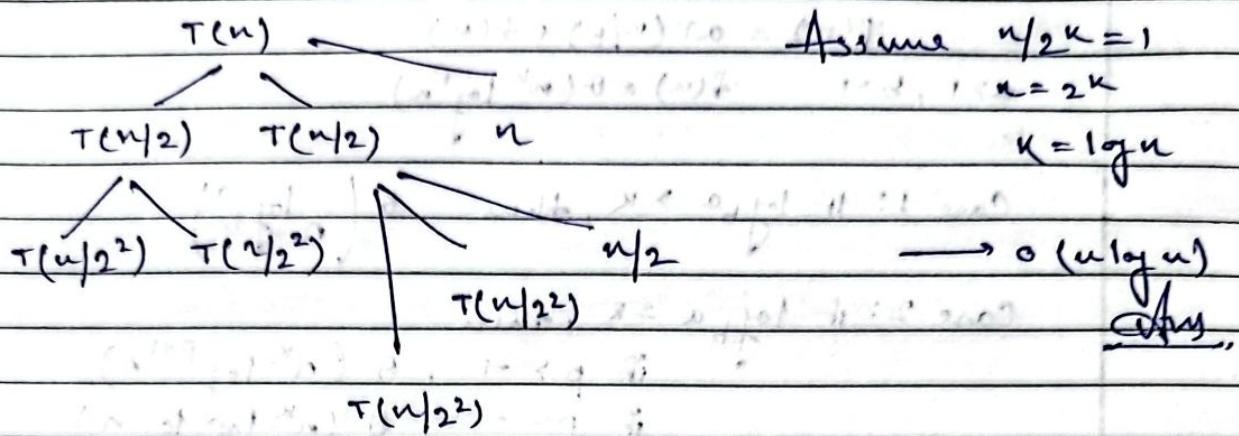
 Test(n/2); $\rightarrow T(n/2)$

 Test(n/2); $\rightarrow T(n/2)$

}

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

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



$$T(n) = 2T(n/2) + n \quad \dots \textcircled{1}$$

$$= 2[2T(n/2^2) + n/2] + n \quad \dots \textcircled{2}$$

$$= 2^2 [2T(n/2^3) + (n/2^2)] + 2n \quad \dots \textcircled{3}$$

$$= 2^3 T(n/2^3) + 3n \quad \dots \textcircled{3}$$

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

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

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

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

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

$\Theta(n \log n)$

Ans.

Master Theorem for Dividing functions :-

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

$$a \geq 1, b \geq 1 \quad f(n) = \Theta(n^k \log^p n)$$

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

Case 2: If $\log_b a = k$, then

$$\text{if } p > -1, \Theta(n^k \log^{p+1} n)$$

$$\text{if } p = -1, \Theta(n^k \log \log n)$$

$$\text{if } p < -1, \Theta(n^k)$$

Case 3: If $\log_b a < k$, if $p > 0$, $\Theta(n^k \log^p n)$
 if $p < 0$, $\Theta(n^k)$

Q $T(n) = 2T(n/2) + 1$
 $a=2, b=2$
 $f(n) = \Theta(1)$
 $= \Theta(n^0 \log^0 n)$
 $k=0, p=0$

Q $T(n) = 4T(n/2) + n$
 $a=4, b=2$
 $\rightarrow \Theta(n^2)$

Q $T(n) = 8T(n/2) + n$
 $a=8, b=2$
 $\rightarrow \Theta(n^3)$

Q $T(n) = 9T(n/3) + 1$
 $a=9, b=3$
 $\rightarrow \Theta(n^2)$

$$\underline{Q} \quad T(n) = 9T(n/3) + n^2$$

$$\log_3 9 = 2 \Rightarrow k=2 \rightarrow \Theta(n^2)$$

$$\underline{Q} \quad T(n) = 2T(n/2) + n^1$$

$$\log_2 2 = 1 \Rightarrow k=1 \quad p=0 \rightarrow \Theta(n \log n)$$

$$\underline{Q} \quad T(n) = 4T(n/2) + n^2$$

$$\log_2 4 = 2 \quad k=2 \rightarrow \Theta(n^2 \log n)$$

$$\underline{Q} \quad T(n) = 4T(n/2) + n^2 \log n$$

$$\log_2 4 = 2 \quad k=2 \rightarrow \Theta(n^2 \log^2 n)$$

$$\underline{Q} \quad T(n) = 2T(n/2) + n/\log n$$

$$\log_2 2 = 1 \quad k=1 \quad p=-1 \rightarrow \Theta(n \log \log n)$$

$$\underline{Q} \quad T(n) = 2T(n/2) + n/\log^2 n$$

$$\log_2 2 = 1 \quad k=1 \quad p=-2 \rightarrow \Theta(n)$$

$$\underline{Q} \quad T(n) = T(n/2) + n^2$$

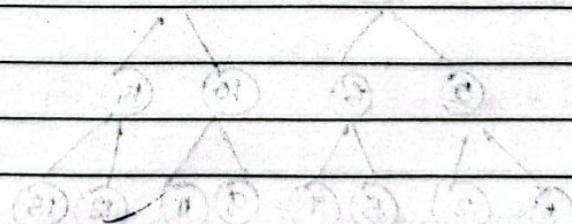
$$\log_2 1 = 0 \quad k=2 \rightarrow \Theta(n^2)$$

$$\underline{Q} \quad T(n) = 2T(n/2) + n^2 \log^2 n$$

$$\log_2 2 = 1 < k=2 \rightarrow \Theta(n^2 \log^2 n)$$

$$\underline{Q} \quad T(n) = 4T(n/2) + n^3$$

$$\log_2 4 = 2 < k=3 \rightarrow \Theta(n^3)$$



1st level - 3

2nd level - 6

(1,2,3) 3 + 3 = 6

(4,5,6,7) 4 + 4 = 8

Root function

$T(n) \rightarrow \text{void Test (int } n\})$

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

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

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

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

$$T(n) = T(n^{1/2^3}) + 3 \quad \text{--- (3)}$$

:

:

$$T(n) = T(n^{1/2^m}) + m \quad \text{--- (4)}$$

$$\text{Assume, } n = 2^m \rightarrow m = \log_2 n$$

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

Assume,

$$+ (m/2^k) = T(2^1)$$

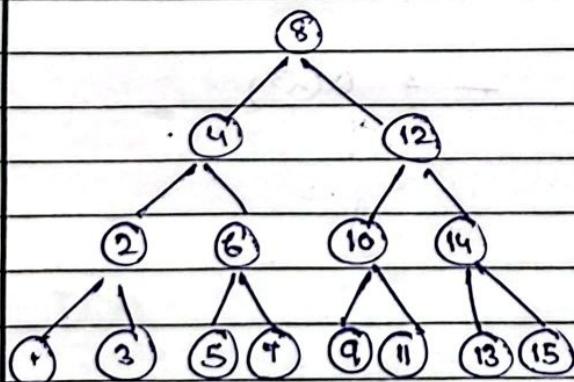
$$\therefore m/2^k = 1 \Rightarrow O(\log \log_2 n)$$

$$\Rightarrow m = 2^k \Rightarrow K = \log \log_2 n$$

$$K = \log m$$

Binary Search \rightarrow must be in sorted order

A	3	6	8	12	14	17	25	29	31	36	42	47	53	55	62
0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	



min $\rightarrow O(1)$

max $\rightarrow O(\log n)$

$\text{int BinSearch (A, n, key)}$

$L=1, h=n;$

while ($L \leq h$) {

mid = $(L+h)/2;$

if (key == A[mid])

return mid;

if (key < A[mid])

$h=mid-1;$

else

$L=mid+1;$

}

}

- 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 ($l \geq h$) {

 if ($A[l] == key$) {
 return l ;

 else

 return 0;

 } else {

 mid = $(l+h)/2$;

 if ($key == A[mid]$)

 return mid;

 if ($key < A[mid]$)

 return RBinSearch

 } else {
 T($n/2$) } else

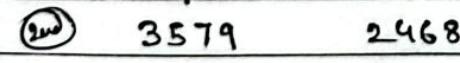
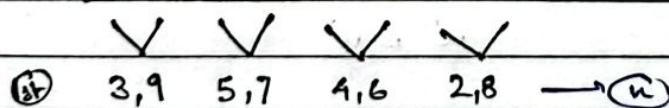
 return RBinSearch

 (mid+1, h, key);

Merge Sort

* 2 way merge sort (iterative)

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



No. of passes = $\log_2 n$

$\Theta(n \log n)$

Algorithm Merge (A, B, m, n)

i=1; j=1; k=1;

while ($i \leq m \& j \leq n$) {

 if ($A[i] < B[j]$)

 c[k++] = A[i++];

 else

 c[k++] = B[j++];

}

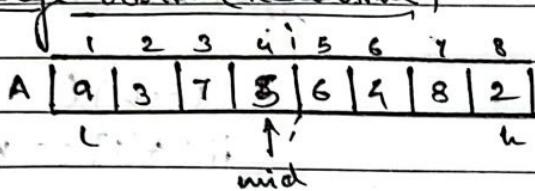
for (; i <= m; i++) {

 c[k++] = A[i];

for (; j <= n; j++) {

 c[k++] = B[j];

Merge Sort (Recursive)



$T(n) \rightarrow$ Algorithm: MergeSort(l, h) {

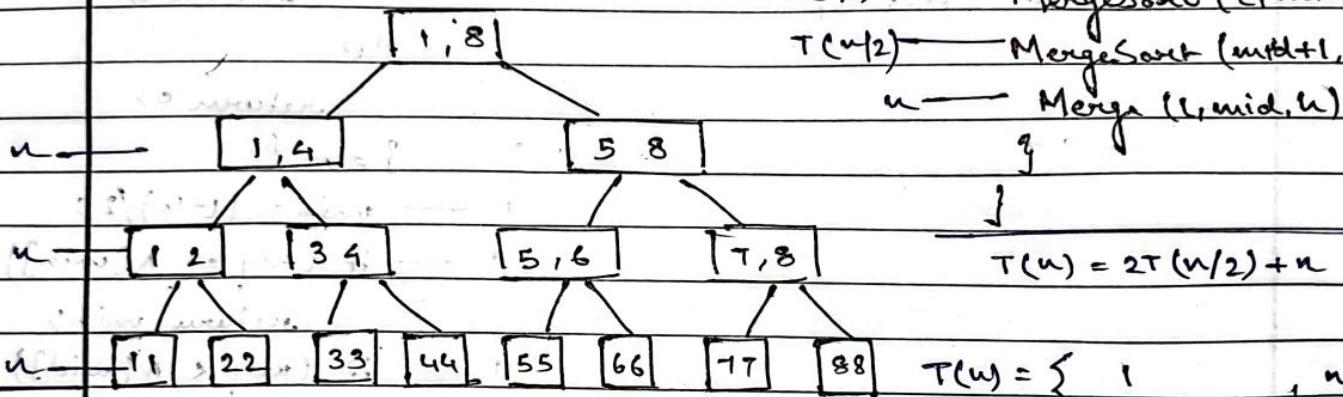
if ($l < h$) {

(i) $mid = (l+h)/2;$

(ii) $T(n/2) \rightarrow$ MergeSort(l, mid);

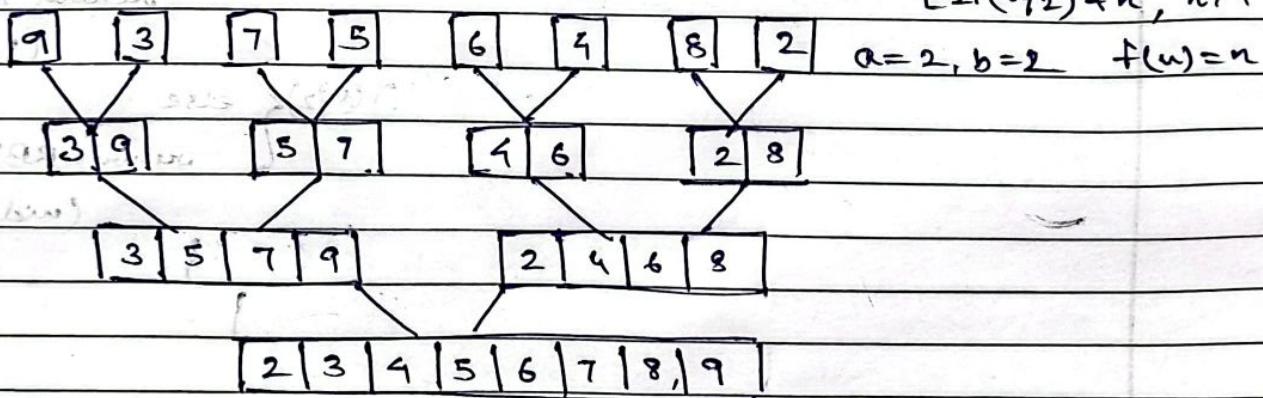
(iii) $T(n/2) \rightarrow$ MergeSort($mid+1, h$);

(iv) $n \rightarrow$ Merge(l, mid, h);



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

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



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

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

$$\log_2^a = \log_2^2 = 1$$

$$\log_2^2 = k$$

$$k = \log_2 n$$

$$n^k = n! \quad \checkmark \quad \checkmark$$

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

- Pros:
 - (i) Large size list
 - (ii) linked list

- Cons:
 - (i) Extra Space (not in-place Sort)
 - (ii) No small problem
 - (iii) Recursive

QuickSort

A

10	80	30	90	40
----	----	----	----	----

↑ Pivot

- Compare 10 with the pivot, and since it's 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
----	----	----	----	----



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 → $O(n^2)$



BREEDY METHOD

→ Mainly used for optimization problems

- Breedy method
- Dynamic Programming
- Branch and Bound

$n=5$

Algorithm Breedy (a, n)

a	a_1	a_2	a_3	a_4	a_5
i	1	2	3	4	5

for $i=1$ to n do

{

$x = \text{select}(a);$

 if feasible(x) Then

{

 solution = solution + $x;$

}

}

Knapsack Problem

Object: 1 2 3 4 5 6 7

$n=7$ Profit: 10 5 15 7 6 18 3

$m=15$ Weight: 2 3 5 7 11 9 1

↓ ↓ ↓ ↓ ↓ ↓ ↓

P/W: 5 1.3 3 1 6 4.5 3

(1 2/3 1 0 1 1 1)

Constraint

$$\sum x_i w_i \leq m$$

Objective

$$\max \sum x_i p_i$$

$$\begin{aligned} 15 - 1 &= 14 \\ 14 - 2 &= 12 \\ 12 - 4 &= 8 \\ 8 - 5 &= 3 \\ 3 - 1 &= 2 \\ 2 - 2 &= 0 \end{aligned}$$

$$\sum x_i w_i = 15$$

$$\text{Profit} = 1 \times 10 + 2/3 \times 5 + 1 \times 15 + 1 \times 6$$

$$+ 1 \times 18 + 1 \times 3$$

$$= 54.6$$

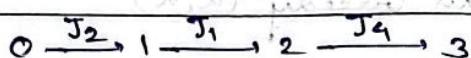
$$\sum x_i p_i = 54.6$$

~~to optimize~~
~~don't maximize~~
~~total cost~~
~~total~~

■ 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
	$\underbrace{20+15+5 = 40}$				
	J_1	J_2	J_3		



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

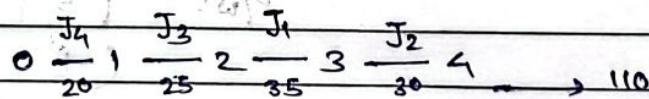
$$\{ J_1 \rightarrow J_2 \rightarrow J_4 \}$$

$0 \longrightarrow 1 \longrightarrow 2 \longrightarrow 3$

Jobs Consider	Slot Assignment	Solution	Profit
-	-	\emptyset	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



Jobs Consider	Slot Assignment	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

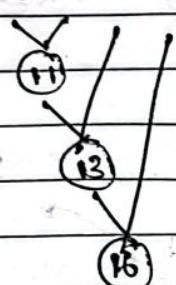
Optimal Merge Pattern

List $\rightarrow A B C D$, sizes $\rightarrow 6 \ 5 \ 2 \ 3$ \rightarrow optimal

Size $\rightarrow 6 \ 5 \ 2 \ 3$ \rightarrow step 1

A B C D

6 5 2 3

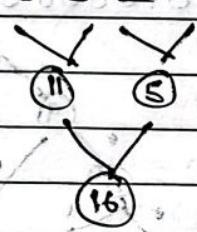


$$11 + 13 + 16$$

$$= 40$$

A B C D

6 5 2 3



$$11 + 5 + 16$$

$$= 32$$

A B C D

6 5 2 3



$$5 + 10 + 16$$

$$= 31$$

Lists $\rightarrow x_1$

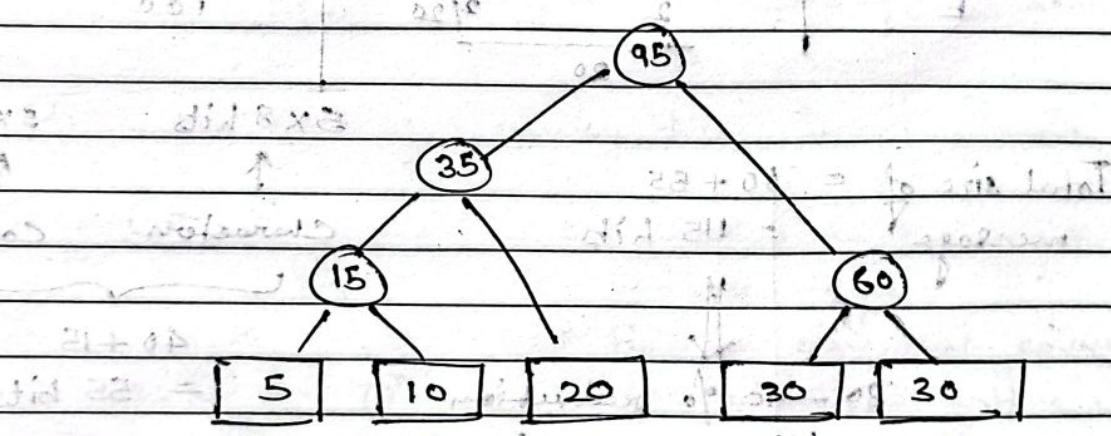
x_2

x_3

x_4

x_5

sizes $\rightarrow 20 \ 30 \ 10 \ 5 \ 30$



$$15 + 60 + 35 + 95 = 205$$

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

cost of

merging

	1	2	3	4	5
d_i	100	81	21	3	1
x_i	100	81	21	3	1
$d_i * x_i$	10000	6589	441	9	1
Total cost	10849	7000	450	10	100

Huffman Coding

Message → BCCABBDAAECCBRAEEDC

length → 20

ASCII → 8 bits

A	65	01000001
B	66	
C	67	(9)
D	68	(5)
E	69	(1)

$$8 \times 20 = 160 \text{ bits}$$

$$20 \times 3 = 60$$

<u>Character</u>	<u>Count/Frequency</u>	<u>Code</u>
A	$3 \text{ out of } 20 = \frac{3}{20}$	000
B	5 $\frac{5}{20}$	001
C	6 $\frac{6}{20}$	010
D	4 $\frac{4}{20}$	011
E	2 $\frac{2}{20}$	100
	20	

$$\text{Total size of message} = 60 + 55 = 115 \text{ bits}$$

5x8 bits 5x3

Character

Gadis

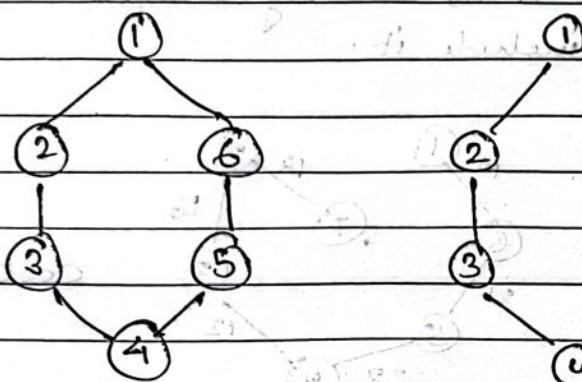
$$40 + 15$$

= 55 bits

in merge size.

<u>Char</u>	<u>Count</u>	<u>Code</u>		<u>msg → 45 bits</u>
A	3	001	$3 \times 2 = 6$	+ tree table → <u>52 bits</u>
B	5	10	$5 \times 2 = 10$	<u>97 bits</u>
C	6	11	$6 \times 2 = 12$	$\Sigma d_i \times f_i$
D	4	01	$4 \times 2 = 8$	$= 3 \times 2 + 3 \times 3 + 2 \times 4$
E	2	000	$2 \times 3 = 6$	$+ 2 \times 5 + 2 \times 6$
$8 \times 5 = 40$	20	12 bits	45 bits	$= 45 \text{ bits}$
				<u>bits</u>

Minimum Cost Spanning Tree



$$S \subseteq W$$

$$S = (V' E')$$

$$V' = V$$

$$|E'| = |N| - E$$

$$G = (V, E)$$

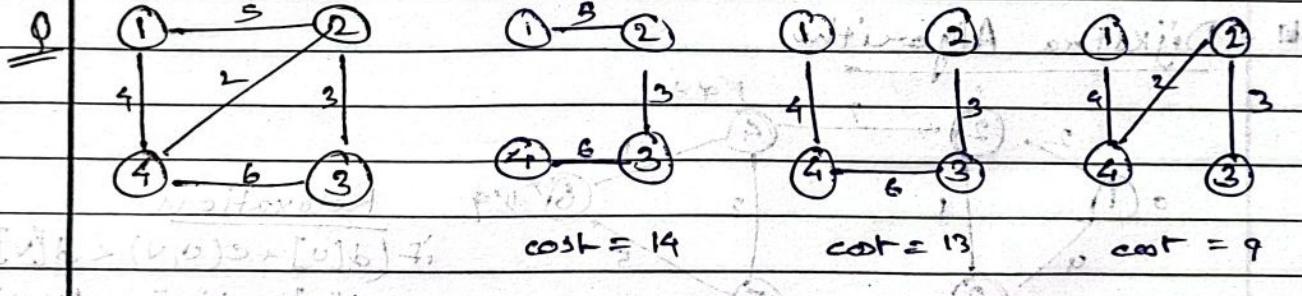
$$|V| = n = 6$$

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

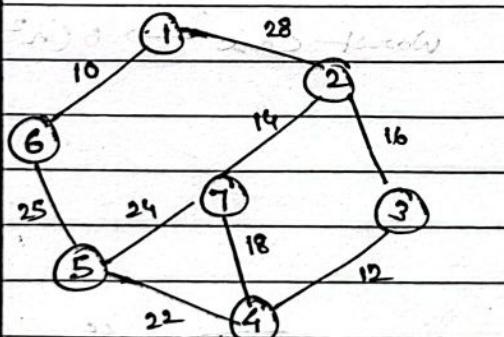
$$|V| - 1 = 5$$

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

$$(1, 2) \rightarrow (2, 1) \quad (3, 4) \rightarrow (4, 3)$$

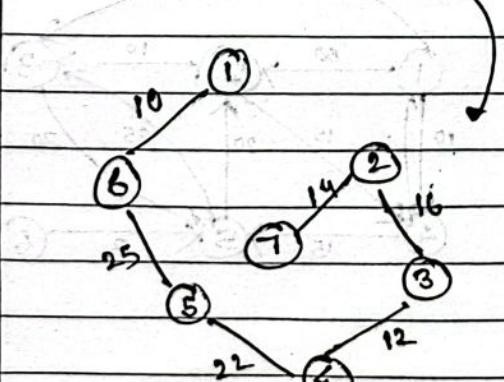


Minimum Cost of Spanning tree through Prim's Algorithm



* 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



$$\text{Cost} = 99$$

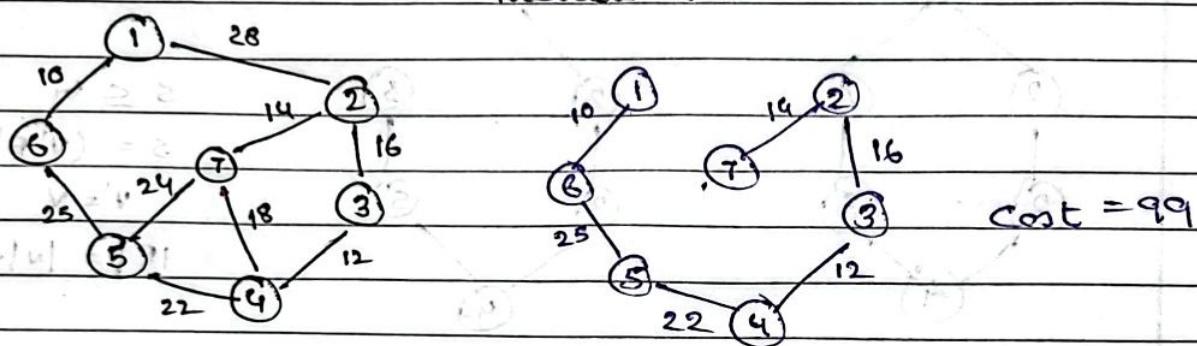
01 20 01 02 03 04 05

06 07 01 02 03 04 05

08 09 01 02 03 04 05

06 07 01 02 03 04 05

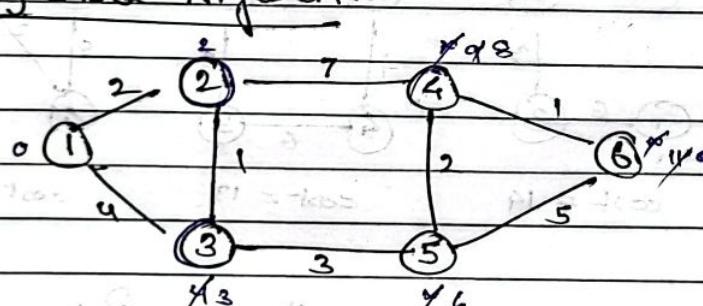
Kruskal's Method — Always select the minimum cost but if it's forming a cycle don't include it.



$$\Theta(n \cdot |E|)$$

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

Dijkstra Algorithm



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

Time complexity $\Theta(n^2)$ and $\Theta(n \cdot e)$

Worst Case $\rightarrow \Theta(n^2)$

With distance start cap 2, 3, 4, 5, 6, 7, 8

With weight + no. how and 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30

Distance after value update, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30

Weight after value update, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30

Starting vertex = 1

Selected vertex	2	3	4	5	6
1	50	45	10	00	00
5	50	45	10	25	00
2	45	45	10	25	00
3	45	45	10	25	00
6	45	45	10	25	20

