4 Sep '19

Analysis of Algorithms

Lecture #2

We analyze the algorithms on the basis of some factors and papameters.

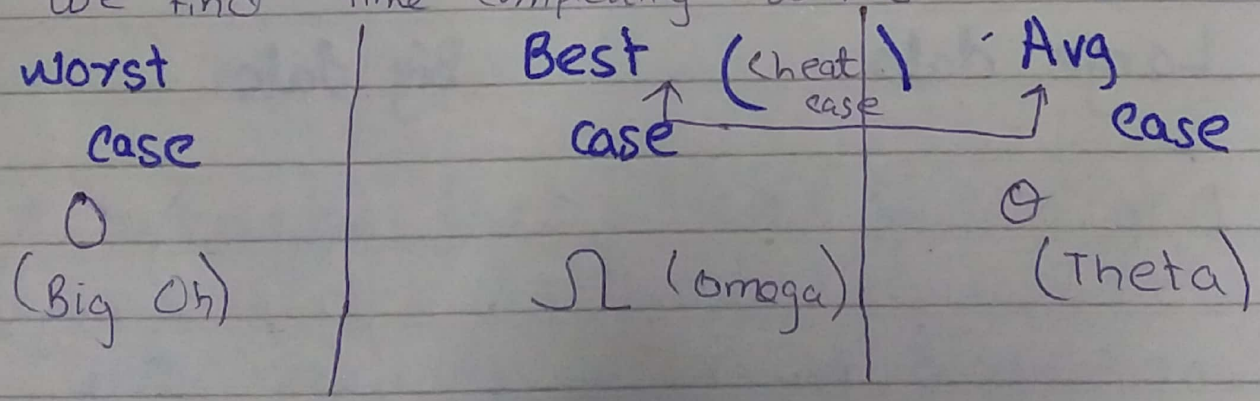
1) Time Complexity.
2) Space Complexity.

• Mathematical Analysis:-

In mathematical analysis we have some notation symbols called Asympatic Notation.

Mathematical analysis
↳ order of Growth (Performance)

We find time Complexity w.r.t worst case.



1) Iterative Method:- (family Algo)

For Brute Force we use Iterative method for calculating time Complexity

```
{
    flag = 0;                                   (C1)
    find = val;                                 (C2)
(looping instruction 1 more time) for (i=1; i<=N; i++)   (C3)
    {
        if (a[i] == find)                       (C4)
(looping body)  {
            flag = 1;
            break;
        }
    }
```

Time	Cost
1	C_1
1	C_2
$(N+1)$	C_3
N	C_4

$$T(n) = C_1 \times 1 + C_2 \times 1 + C_3 \times (N+1) + C_4 (N)$$

$$= C_1 + C_2 + C_3 N + C_3 + C_4 \cdot N$$

$$= (C_1 + C_2 + C_3) + (C_3 + C_4) N$$

$$= (C_3 + C_4) N \quad \text{(ignore constant)}$$

$$= N$$

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

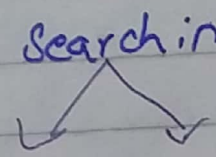
Iterative Method:-

↳ using Step Method.

⇒ Brute Force Algo are used for

↳
↳ Searching (Linear search)
↳ Sorting

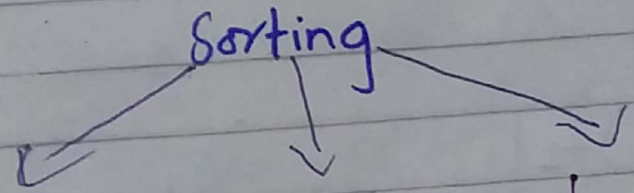
⇒



Linear Search

Binary Search (Divide & Conqure)

⇒



Iterative	Divide & Conqure	Non. Comparison
↳ Bubble sort	↳ Merge sort	↳ Radix sort
↳ Insertion sort	↳ Quick sort.	↳ BUCKET sort
↳ Selection sort		↳ COUNTING sort.

Example..

Design an Algo/ Pseudo Code for bubble cort & find its worst time Complexity & show each step.

Bubble Sort Pseudo Code:-

(C1) for i ← 1 to length [A] - 1
(C2) { for j ← 1 to length [A] - i
(C3) if (arr[j] > arr[j+1])
Swap arr [i] ←→ arr [j]
} [End if]
[End for]
[End for]

Time	Cost
$(N-1)+1 = N$	C_1
$(N-1) * [(N-i)+1]$	C_2
$(N-1) \times [(N-i+1)-1] + (C_3)$ loop 1	C_3
$[N-1] \times [N-i] * (1)$ loop 1 loop 2 $(N-1)(N-i)$	
(loop1) (loop2) x (instruction struct) $(N-1) \times (N-i) \times 1$ $(N-1)(N-i)$	C_4

insertion
selection

$T(n) : C_1 \times N + C_2 (N-1)(N-i+1)$
$C_3 (N-1)(N-i) + C_4 (N-1)(N-i)$

$= C_1 N + C_2 (N^2 - Ni + N - N + i - 1) + C_3$
$(N - Ni - N + i) + C_4 (N^2 - Ni - N + i)$

$= C_1 N + C_2 N^2 - C_2 Ni + C_2 i - C_2 + C_3 N - C_3 Ni$
$- C_3 N + C_3 i + C_4 N^2 - C_4 Ni - C_4 N + C_4 i$

$= (C_1 - C_2) + N (C_3 - C_4) + 1 (C_2 + C_3 + C_4) +$
$Ni (-C_2 - C_3 - C_4) + N^2 (C_2 + C_4)$

$= N^2 (C_2 + C_4)$

$T(n) = O(N^2)$

Analysis Of Algorithms

Brute force algo:-

Question:-

Design Algo for selection sort & find the worst time complexity using step count method.

Algo for selection sort:-

1) n ← length[A]
2) for j ← 1 to n-1
3) min ← j
4) for i ← (j+1) to n
5) if (A[i] < A[min])
6) min ← i
[End for]
7) exchange A[j] ↔ A[min]
[End for]

Time	Cost
1	C_1
$(n-1)+1 = n$	C_2
$(n-1)$	C_3
$(n-1)(n+1-j)$	C_4

$(n+1)-j$

$(n-1)(n-j-1+1)$

$(n-1)(n-j)$	C_5
$(n-1)(n-j)$	C_6
$(n-1)$	C_7

$$T(n) = C_1(1) + C_2(N) + C_3(n-1) + C_4((n-1)(n+1-j) + C_5(n-1)(n-j) + C_6(n-1)(n-j) + C_7(n-1)$$

$$= C_1 + C_2N + C_3N - C_3 + C_4[n^2 + n - nj - n - 1 + j] + C_5[n^2 - nj - n + j] + C_6[n^2 - nj - n + j] + C_7N - C_7$$

$$= N^2$$

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

Insertion Sort:-

$n+1$ ⇒ we add 1 to make loop finite & sub 1 bcz 2 is not included

Algo :-

	Time	Cost
1) for $j \leftarrow 2$ to n	$(n+1-1) = N$	C_1
2) key $\leftarrow A[j]$	$(n-1)$	C_2

→ sentinal

while (i<=10) { print i; i++ } — control in which starting & termination condition is given

i depends on j

3) $i \leftarrow j-1$	$(n-1)$	C_3
4) while (i>0) and A[i]>key (sentinal loop)	$\sum_{j=2}^{n} t_j$	C_4
5) $A[j+1] \leftarrow A[i]$	$\sum_{j=2}^{n} (t_j - 1)$	C_5
6) $i \leftarrow i-1$ [wend]	$\sum_{j=2}^{n} (t_j - 1)$	C_6
7) $A[i+1] \leftarrow$ key [ENDFOR]	$(n-1)$	C_7

$$T(n) = C_1(N) + C_2(n-1) + C_3(n-1) + C_4\left(\sum_{j=2}^{n} t_j\right)$$

$$+ C_5\left(\sum_{j=2}^{n} (t_j - 1)\right) + C_6\left(\sum_{j=2}^{n} (t_j - 1)\right) + C_7\left(\sum_{j=2}^{n} (t_j - 1)\right)$$

$$\boxed{\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1} \qquad \boxed{\sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}}$$

$$\sum_{j=2}^{n} (j) - \sum_{j=2}^{n} (1) =$$

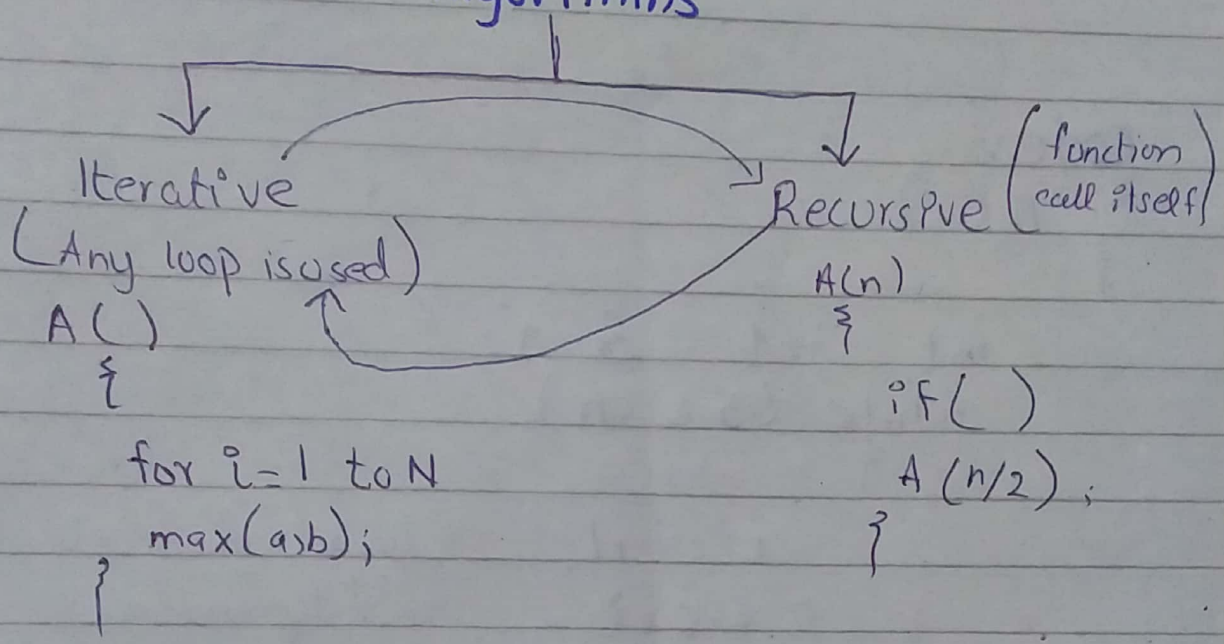
$$= C_1N + C_2N - C_2 + C_3N - C_3 + C_4\left(\frac{n(n+1)}{2} - 1\right) + C_5\left(\frac{n(n-1)}{2}\right)$$

$$C_6\left(n\,\frac{(n-1)}{2}\right) + C_7(n-1)$$

$$= O(n^2)$$

Analysis Of Algorithm

Lecture



⇒ Recursive program can be converted into iterative & vice versa.

⇒ We use recursive Equations to analyze mathematically & in iterative we use step-count method for mathematical analysis.

① for i=1 to n → n+1
cout "Pakistan" → n

n+1+n = 2n+1 = 2n = n

(2) for i = 1 to n
for j = 1 to n
cout << " Pakistan";

$O(n^2)$

Code :-

```
A( )
{
    int i=1 , S =1;
    while (S <=n)
    {
        i = i+1;   // linear Increment
        S = S+i    // dependency
        cout "Pakistan";
    }
}
```

when dependency is present then we have to unroll.

i =	1,	2,	3 ,	4 ,	5	.... K
S =	1,	3,	6 ,	10,	15,	.. S>n

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

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

$\frac{K^2+K}{2} > n \quad = \frac{K^2}{2} + \frac{K}{2} > n$

$K^2 > n$

$O(\sqrt{n})$

Example:-

```
A()
{
① for(int i=1; i*i<=n; i++)
②     cout<<"Pak";
}
```

① $\sqrt{n}+1$

② $\sqrt{n}$

---

④

```
A()
{
for(int i=1; i<=n; i++)      // Independent
  for(int j=1; j<=i; j++)    // dependent
    for(int k=1; k<=100; k++) // Independent
      cout<<"Pak";
}
```

unroll

i = 1	i = 2	i = 3	i = 4	i = 5
j = 1	j = 2 time	j = 3 times	j = 4 times	j = 5
k = 1 × 100 times	k = 2 × 100 = 200	k = 3 × 100 = 300	k = 400	k = 500

---

i = n

j = n times

k = n × 100 times

$= 1\times100 + 2\times100 + 3\times100 + 4\times100 + 5\times100 + \cdots + n\times100$

$= 100(1+2+3+4+\cdots+n)$

$= 100\left(\frac{n(n+1)}{2}\right)$

$= 100\left(\frac{n^2+n}{2}\right)$

$= \frac{100n^2}{2} + \frac{100n}{2}$

$= 50n^2 + 50n$

$= 50n^2$

$= n^2$

---

(5)

```
A()
{
for (int i=1; i<n; i++)
for (int j=1; j<=i^2; j++)
for (int k=1; k<=n/2; k++)
    cout << "Pakistan";
}
```

unroll

$i=1$	$i=2$	$i=3$	$i=4$
$j=1$	$j=4$ times	$j=9$	$j=16$
$k=1\times\frac{n}{2}$	$k=4\times\frac{n}{2}$	$k=9\times\frac{n}{2}$	$k=16\times\frac{n}{2}$

$i=5$	$i=n$
$j=25$	$j=n^2$
$k=25 \times \frac{n}{2}$	$k=n^2 \times \frac{n}{2}$

$$= 1 \times \frac{n}{2} + 4 * \frac{n}{2} + 9 * \frac{n}{2} + 16 * \frac{n}{2} + 25 * \frac{n}{2}$$

$$+ \text{ ---- } n^2 * \frac{n}{2}$$

$$= \frac{n}{2}\left(1+4+9+16+25+\text{ ---- } n^2 \right)$$

$$= \frac{n}{2}\left(1+(2)^2+(3)^2+(4)^2+(5)^2 \text{ ---- } (n)^2 \right)$$

$$= \frac{n}{2}\left(\frac{n(n+1)(2n+1)}{6} \right)$$

$$= O(n^4)$$

$i=1$	$i=2$	$i=2$	—
$j=1$	$j=2$	$j=3$	—
$K=1$	$k=4$	$k=9$	---

$1 \times 100 + 4 \times 100 + 9 \times 100 + 16 \times 100$ ... n times

$100\left(1+4+9+\text{ ---- } n^2 \right)$

$100\left(1+(2)^2+(3)^2 \text{ ---- } (n)^2 \right)$

n

(6)

```
A ← for ( i = 1 ; i <= n , i++ )        ⇒ Indepen
D ← for ( j = 1 ; j <= i ; j++ )
D ← for ( k = 1 ; k = j^2 ; k++
          for (q = 1 ; q <= 100 ; q++)
```

(7)

```
A()
{
    for (int i=1;  i<=n;  i=i*2)
        cout << "Pak";
}
```

(k iteration required to reach n)

i = 1, 2, 4, 8, 16, 32, 64 .... k)

$2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6 \ldots 2^n$

$k = 2^n$

$$\boxed{k = \log_2 n}$$

(8)

```
A()
{
    for (int i=n/2 ; i<=n ; i++)
    for (int j=1 ; j<=n/2 ; j++)
    for (int k=1; k<=n; k=k*2)
        cout << "Pak";
}
```

$n/2$

$n/2$

$\log_2 n$

(All are Independent so cross multiply)

$$= \left(\frac{n}{2}\right) * \left(\frac{n}{2}\right) * \left(\log_2 n\right)$$

$$= \left(\frac{n^2}{2} \log_2 n\right)$$

$$= \frac{1}{2}\left(n^2 \cdot \log_2 n\right)$$

$$= O\left(n^2 \log_2 n\right)$$

```
for ( i=1 ; i<=n ; i++ )        → Independent
for ( j=1 ; j<=n; j = j*2 )     → "  "
for ( k=1 ; k<=n ; k = k*2 )    → "  "
        cout << "Pak";
```

$$= (n)\left(\log_2 n\right)\left(\log_2 n\right)$$

$$= n\left(\log_2 n\right)^2$$

$$= O\left(n(\log n)^2\right)$$

Analysis of Algo

Types of Algo:-

Brute force
- ↳ Bubble sort
- ↳ Insertion
- ↳ Selection
- ↳ Radix
- ↳ Bucket
- ↳ Counting

- Design Algo
- Dry Run

- Mathematically Analyze
- Correctness of Algo

Radix Sort → Least Significant bit

1	2	3
	4	7
9	1	1
	1	2

L.S.B

priority

9	1	1
	1	2
1	2	3
	4	7

bcz 1 is smallest

Bucket Sort

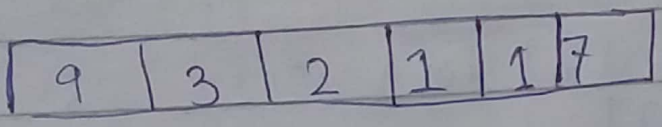
arrange using link list

9	1	1
0	1	2
1	2	3
0	4	7

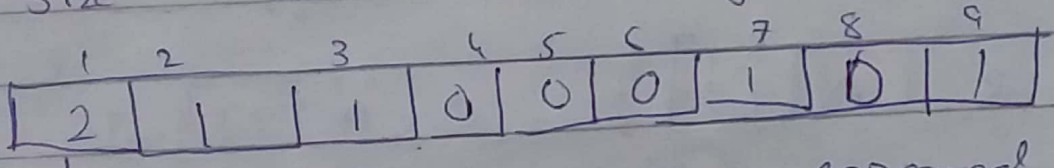
0	1	2
0	4	7
1	2	3
9	1	1

Counting Sort:- (Numerical & smaller data)

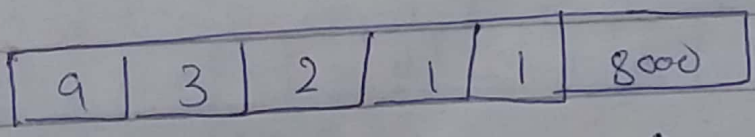
↳ Count array → size
↳ largest value in array

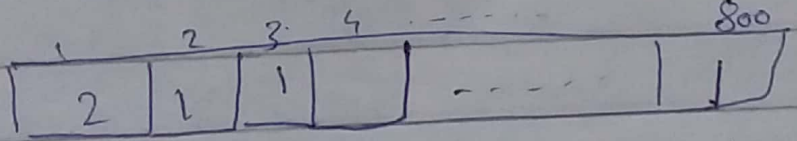


size of this array is "9"



↳ no. of occurence appeared in unsorted array.

if 



↳ wastage of memory
↳ Computational Cost

Algo:

for i ← 1 to (k) → size of array

2) C[i] ← 0] Independent

n times
2) for j ← 1 to length[A] (no. of occurence)
C[A[j]] ← C[A[j] +1]

(k-1)
3) for i ← 2 to k → index of sorted array
2 se is lye shuru kya h k agr neechy i ki jga 1 hith dy to 0 ijega jo k array mein nhi hy
C[i] ← C[i] + C[i-1]

n
4) j ← length[A] to 1
B[C[A[j]]] ← A[j]

n+n+k-1+n

2n+2K-1
C[A[j]] ← C[A[j]]-1
2(n+k)
(n+k)

1	2	3	4	5	6	7	8	9	10	11
7	1	3	1	2	4	5	7	2	4	3

Solution:-

Size = 7

prevent from Grabage value

1	2	3	4	5	6	7
0	0	0	0	0	0	0

k=7
C[i] ← 0

(j=1)

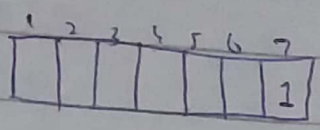
C[A[j]] ← C [A[j]] +1

C [A [1]] ← C [A [j]] + 1

C [7] ← C [7] + 1

C [7] ← 0 + 1

C [7] ← 1

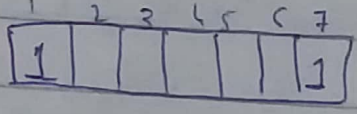


C [A [2]] ← C [A [2]] + 1

C [1] ← C [1] + 1

C [1] ← 0 + 1

C [1] ← 1

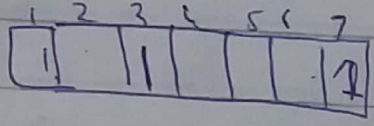


A [A [3]] ← C [A [3]] + 1

C [3] ← C [3] + 1

C [3] ← 0 + 1

C [3] ← 1



A [A [6]] ← C [A [6]] + 1

C [4] ← C [4] + 1

C [4] ← 1 + 1

C [4] ← 2

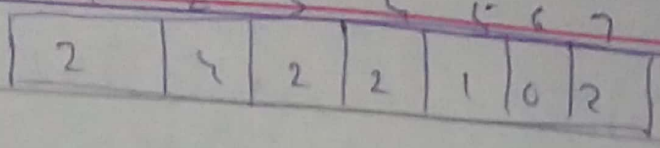
C [i] ← C [i] + C [i-1]

C [2] ← C [2] + C [2-1]

C [2] ← C [2] + C [1]

C [2] ← C [2] + C [2]

4

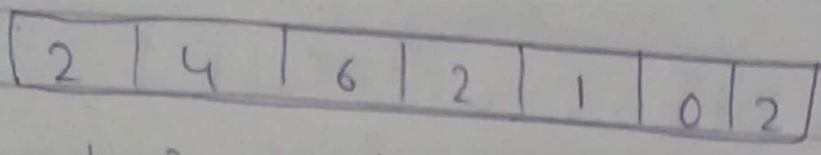


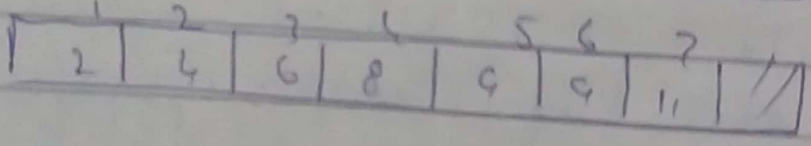
C[3] ← C[3] + C[3-1]

C[3] ← C[3] + C[2]

C[3] ← 2 + 4

C[3] ← 6



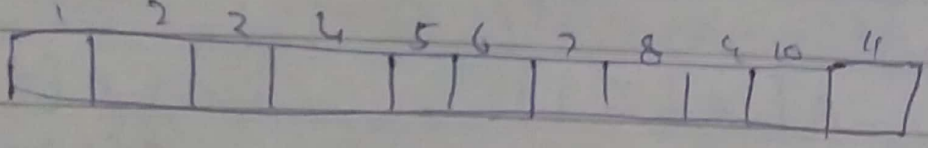


B[C[A[j]]] ← A[j]

B[C[A[11]]] ← A[11]

B[C[3]] ← 3

B[6] ← 3



⇒ C[A[j]] ← C[A[j]] - 1

C[A[11]] ← C[A[11]] - 1

C[3] ← C[3] - 1

C[3] ← 6 - 1

C[3] ← 5

Analysis of Algo

Types of Algo:-

1) Brute Force
2) Divide & Conqure (handle heap amount of data)
   - ↳ Searching ⇒ Binary Sort
   - ↳ Sorting ⇒ Quick Sort, Merge Sort

i) Algo Design
ii) Dry-Run
iii) Mathematical Analysis:
- ↳ Master Theorem
- ↳ Back Substitution
- ↳ Recursion Tree

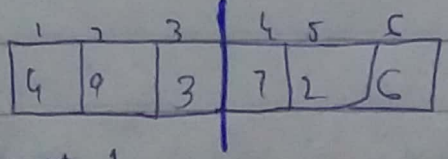
Draw backs of Brute Force:-

1) Only handle small amount of data.

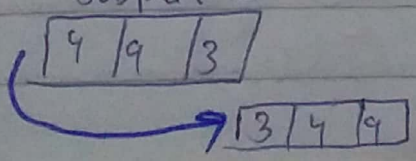
Divide & Conqure:-

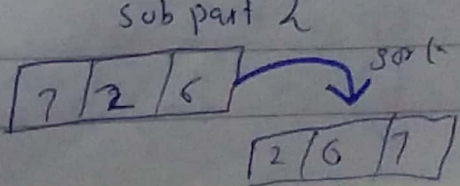
1) Divide the problem into sub parts (2^n)
2) Solve each sub-problem independently.
3) Combine each solved part to get solution.

1)

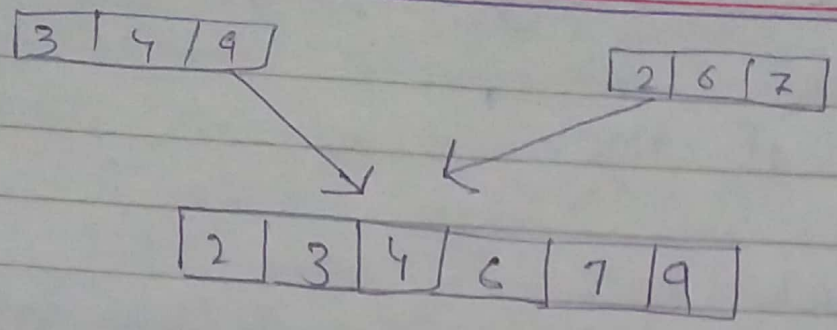


2) Sort





3)



⇒ Quick Sort ⇒ Partioning Algo

⇒ Merge Sort ⇒ To merge two sorted Data structure.

MASTER THEOREM:-

$$T(n) = a\,T(n/b) + \Theta(n^k \log^p n)$$

where

$a \geq 1$, $b > 1$, $k \geq 0$, p = real number

1) if $a > b^k$ $\quad T(n) = \Theta(n^{\log_b^a} \log^{p+1} n)$

2) if $a = b^k$

a) if $p > -1$, $T(n) = \Theta(n^{\log_b^a} \cdot \log^{p+1} n)$

b) if $p = -1$, $T(n) = \Theta(n^{\log_b^a} \cdot \log \cdot \log n)$

c) if $p < -1$, $T(n) = \Theta(n^{\log_b^a})$

3) if $a < b^k$

a) if $p \geq 0$, $T(n) = \Theta(n^k \log^p n)$

b) if $p < 0$, $T(n) = \Theta(n^k)$

1) Solve following recursive equations using master theorem.

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

$a = 3, \quad b = 2, \quad k = 2, \quad P = 0$

a	b^k
3	2^2
$3 < 4$	$a < b$

Condition (3), subcase (3.a) is valid

$T(n) = \Theta(n^k \log^p n)$

$= \Theta(n^2 \log^0 n)$

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

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

$a = 4, \quad b = 2, \quad k = 2, \quad P = 0$

a	b^k
4	2^2
4	4

Condition 2, subcase (2.1) is valid

$T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$

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

$n \log_b a = n^2$

③ $T(n) = T(n/2) + n^2$

$a = 1 \quad , b = 2 \quad , k = 2 \; , P = 0$

a	b^k
1	2^2
1	4

$1 < 4$

Condition (3) Subcase 3.c is valid

$T(n) = \Theta(n^2 \log^0 n)$

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

④ $T(n) = 2^n T(n/2) + n^n$

$a = 2^n$

Unsolvable by master Theorem because a = real number

⑤ $T(n) = 2T(n/2) + n \log n$

$a = 2 \quad , b = 2, \quad k = 1 \quad , P = 1$

a	b^k
2	2^1

$2 = 2$

Condition 2 Subcase 2.a is valid

$T(n) = \Theta(n^{\log_2^2} \log^{1+1} n)$

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

⑥ $T(n) = 16T(n/4) + n$

$a = 16, \quad b = 4, \quad k = 1, \quad p = 0$

a	b^k
16	4^1
16	> 4

Condition (1) is valid.

$$T(n) = \Theta\left(n^{\log_4^{16}}\right)$$

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

Equations:-

① $T(n) = 2T(n/2) + n/\log n$

② $T(n) = 2T(n/4) + n^{0.51}$

③ $T(n) = 0.5T(n/2) + 1/n$ (unsolveable bcz a is less than 1)

④ $T(n) = 6T(n/3) + n^2 \log n$

⑤ $T(n) = 64T(n/8) - n^2 \log n$ (unsolvable bcz of -ve sign)

⑥ $T(n) = 7T(n/3) + n^2$

⑦ $T(n) = 4T(n/2) + \log n$

⑧ $T(n) = \sqrt{2}\,T(n/2) + \log n$

① $2T(n/2) + n/\log n$

$a = 2 \rightarrow b = 2, \; k = 1 \rightarrow p = 1$

a	b^k
2	2^1
2	= 2

$n = \frac{\log n}{\log n}^{-1}$

Condition (2), subcase (2.a) is valid

$$T(n) = \left(n^{\log_b^a} \cdot \log^{p+1} n\right)$$

$$= \left(n^{\log_2^2} \cdot \log^{1+1} n\right)$$

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

Analysis of Algo

↳ Divide & Conqure
- ↳ Master Theorem
- ↳ Back substitution
- ↳ Recursive Tree

$T(n) = \underbrace{T(n-1)}_{\text{recursive part}} + \underbrace{1}_{\text{non-recursive part}}$ (Recursive Equation)

Now we eleminate T. from Equation.

↳ Back Substitution
- ↳ forward substitution
- ↳ Backward substitution
- ↳ Mapl substitution
- ↳ Characteristics Substitution.

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

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

Now replace "n" by $(n-2)$

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

Now replace "n" by $(n-3)$

$$T(n-3) = T(n-3-1) + 1$$
$$T(n-3) = T(n-4) + 1 \quad \text{---(4)}$$

Put Eq (2) in Eq (1)

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

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

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

Put Eq (3) in Eq A

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

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

$$\boxed{T(n) = T(n-3) + 3}$$

Put Eq (4) in eq.

$$T(n) = [T(n-4) + 1] + 3$$

$$= T(n-4) + 1 + 3$$

$$\boxed{T(n) = T(n-4) + 4}$$

⋮ k iteration

$$\boxed{T(n) = T(n-k) + k} \quad \text{---} \quad (F)$$

$\therefore n - k = 1$ Replace $(n-1)$ by k.

$$\boxed{k = n-1}$$

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

Put $k = n-1$ in (F)

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

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

$= T(1) + n-1$

$= 1 + n - 1$

$= \theta n$

Base Condition:-

$T(n) = T(n-1) + n$ if $x > 1$

$T(1) = 1$ $n = 1$

Solution:-

Using back substitution method

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

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

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

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

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

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

$$T(n-3) = T(n-4) + n-3 \quad \text{---(4)}$$

Put Eq 2 in Eq 1

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

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

Put Eq (3) in eq

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

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

Put Eq 4 in

$T(4) = [T(n-4) + n-3] + n-2 + n-1 + n$

$T(4) = T(n-4) + (n-3) + n-2 + n-1 + n$

$T(n) = T(n-(k+1)) + n-3 + n-2 + n-1 + 4$

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

$(k+1) \ldots 4\ 3\ 2\ 1\ 0$

$= T(n-(k+1)) + (n-k) + \ldots + n-3 + n-2 + n-1 + n$

$n-(k+1) = 1$

$n-k-1 = 1$

$n-1 = 1-k$

$n-1-1 = +k$

$n-2 = k$

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

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

Put Eq (3) in eq

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

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

Put Eq 4 in

$T(4) = [T(n-4) + n-3] + n-2 + n-1 + n$

$T(4) = T(n-4) + (n-3) + n-2 + n-1 + n$

$\vdots$

$T(n) = T(n-(k+1)) + n-3 + n-2 + n-1 + 4$

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

$(k+1) \ldots 4\ 3\ 2\ 1\ 0$

$= T(n-(k+1)) + (n-k) + \ldots + n-3 + n-2 + n-1 + n$

$n-(k+1) = 1$

$n-k-1 = 1$

$n-1 = 1-k$

$n-1-1 = +k$

$n-2 = k$

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

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

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

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

$= \frac{n(n+1)}{2}$ (sum of 10 digits)

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

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

Example:-

$T(n) = 2T(n/2) + 2 \quad : T(1) = 1$

$\boxed{T(n) = 2T(n/2) + 2}$ ⟶ (1)

$\boxed{T(n/2) = 2T(n/4) + 2}$ ⟶ (2)

$\boxed{T(n/4) = 2T(n/8) + 2}$ ⟶ (3)

$\boxed{T(n/8) = 2T(n/16) + 2}$ ⟶ (4)

Now

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

Put Eq 2 in

$T(n) = 2[2T(n/4) + 2] + 2$

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

Put eq(3) in eq

$T(n) = 4[2T(n/8) + 2] + 4 + 2$

$T(n) = 8T(n/8) + 8 + 4 + 2$

$T(n) = 8(T(n/8)) + 8 + 4 + 2$

Put Eq(4) in Eq

$T(n) = 8[2T(n/16) + 2] + 8 + 4 + 2$

$T(n) = 16T(n/16) + 16 + 8 + 4 + 2$

$k=1, \quad T(n) = 2^1 T(n/2^1) + 2^1$

$k=2, \quad T(n) = 2^2 T(n/2^2) + 2^2 + 2^1$

$k=3, \quad T(n) = 2^3 T(n/2^3) + (2^3 + 2^2 + 2^1)$

$k=4, \quad T(n) = 2^4 T(n/2^4) + (2^4 + 2^3 + 2^2 + 2^1)$

$= 2^k T(n/2^k) + 2^4 + 2^3 + 2^2 + 2^1$

$2^k = n$

$= nT(n/n) + 2^4 + 2^3 + 2^2 + 2^1$

$= nT(1) + 2^4 + 2^3 + 2^2 + 2^1$

$= 2n$

$T(n) = O(n)$

Analysis of Algorithms

Divide & Conqure
- ↳ Master Theorem
- ↳ Back Substitution
- ↳ Recursion Tree

Divide & Conqure:

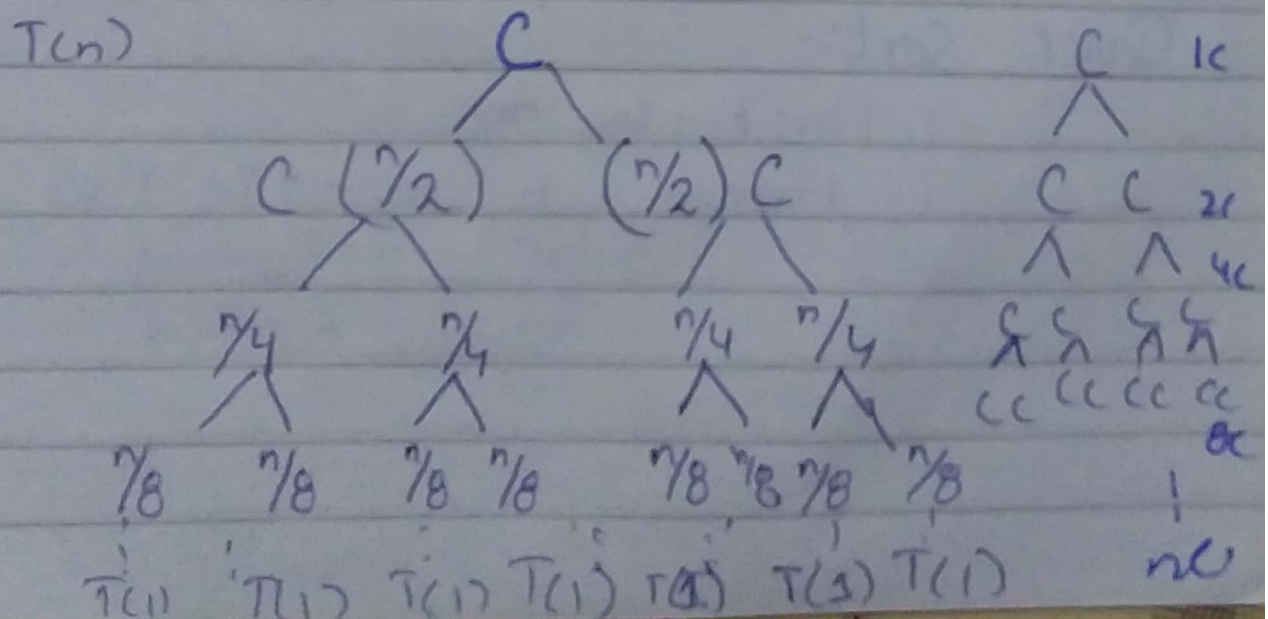
Handle Heap amount of data

- Searching
  - ↳ Binary Search
- Sorting
  - ↳ MERGE sort
  - ↳ Quick sort

RECURSION TREE

- ↳ Construct Tree
- ↳ Develop Series (calculate nth) N time)

$$T(n) = 2T(n/2) + C \quad \text{if } n > 1$$
$$T(1) = 1 \text{ (base condition)} \quad \text{if } n = 1$$



Harmonic
Arithmetic.

$= 1C + 2C + 4C + 8C + \cdots\cdots + n(C)$

$= C(1 + 2 + 4 + 8 + 16 + \cdots\cdots + n)$.

Geometric Progression

$$G.P = \frac{a(r^n - 1)}{r - 1}$$

$a = 1$, $r = 2$, $n = (k+1)$ terms

$$C = \left(\frac{1(2^{k+1} - 1)}{2 - 1}\right)$$

$$= C(2^{k+1} - 1)$$

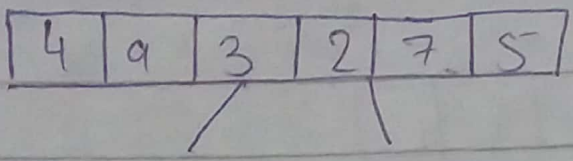
$$= C(2^n - 1)$$

$$= 2^n - 1$$

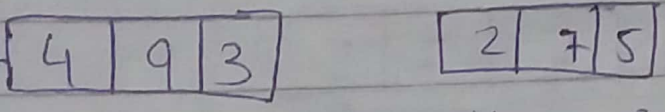
$$T(n) = \Theta(n)$$

Quick Sort:-

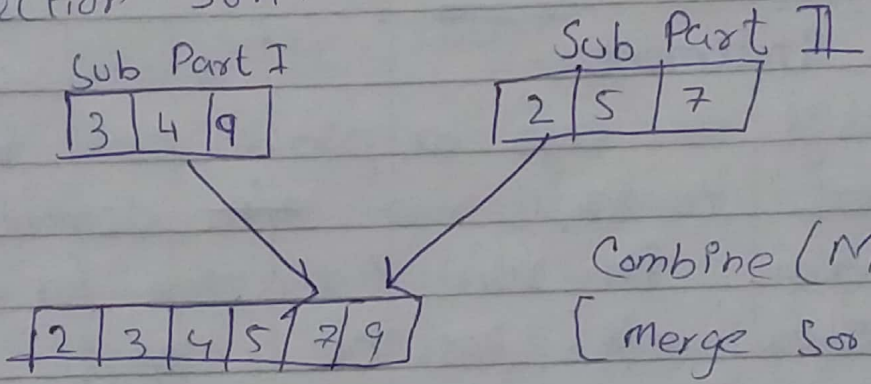
- Partitioning Algo
- Partition with respect to Pivot Element.



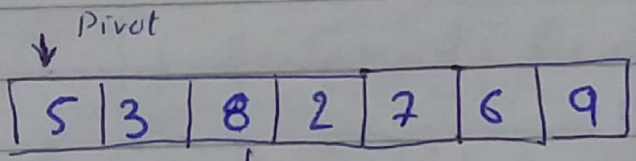
For Partitioning we use Quick Sot.



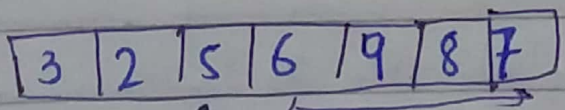
For Solution we use either insertion or selection sort.

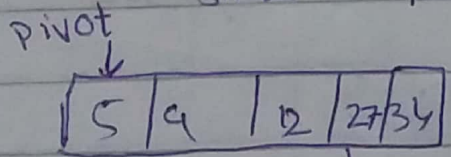


Example

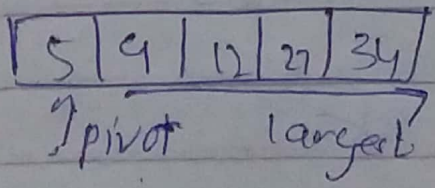


Quick sort

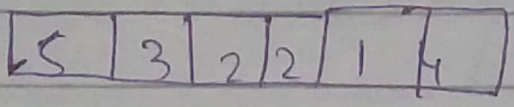


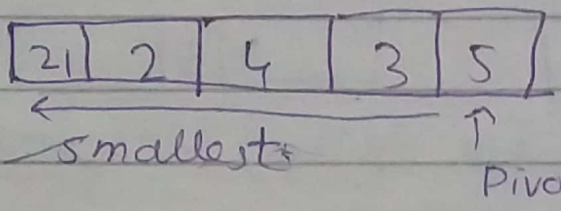


Quick sort.



⇒





Problem Statement:

↳ Design an algo or pseudo code for Quick sort to Partitionion ~~algo~~ elements

↳ Perform Dry run Execution for each & show all possible steps

5, 3, 8, 6, 4, 7, 3,)

Pseudo Code:-

1) x ← A(p]
2) q ← p
3) for s ← p+1 to r
4) if (A[s] < x)
5) q ← q+1
6) Swap A[q] with A[s]
[Swap if]
[End if]

7) Swap A[p] with A[q].

S = 2, 3, 4, 5, 6, 7, 8

if(A[s]

x	p	r	q	s	if (A[s] ≤ x)
5	1	8	1	1	if (A[1] < x)
5	1	8	1	2	if (A[2] < x) (T) if (3 < 5)

			2	3	if (A[3] < x)
					if (8 < 5)
					(false)

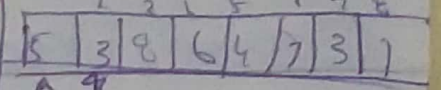
			2	4	if (A[4] < x)
					if (6 < 5)
					false

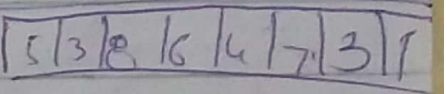
			3	5	if (A[5] < x)
					if (4 < 5)
					True

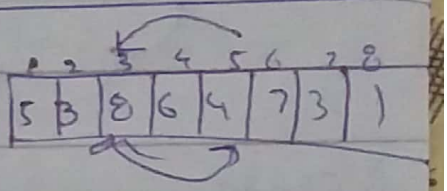
			3	6	if (A[6] < x)
					if (7 < 5)
					False

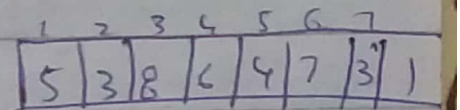
			4	7	if (A[7] < x)
					if (3 < 5)
					(True)

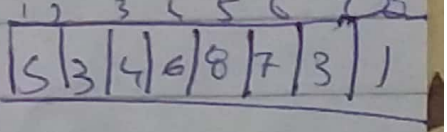
			5	8	if (A[8] < x)
					if (1 < 5)
					True

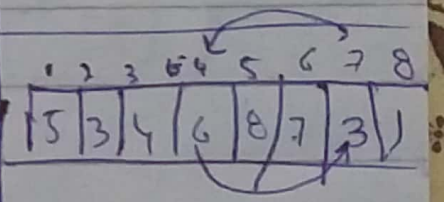


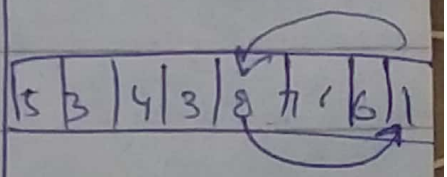


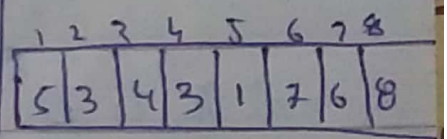






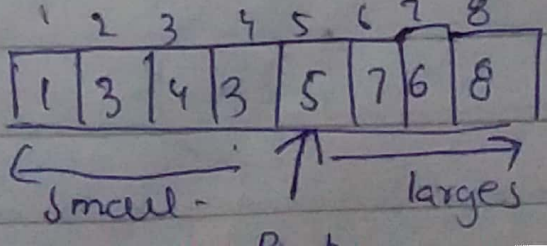






Swap A[1] with A[5]

BST Heap

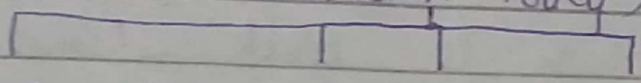


Analysis of Algorithms.

RED BLACK TREE (RBT)

↳ Binary Tree

↳ Self Balancing Tree (Rotation Policy)



Right Left Left - Left Left right Right - Right

PROPERTIES of RBT:

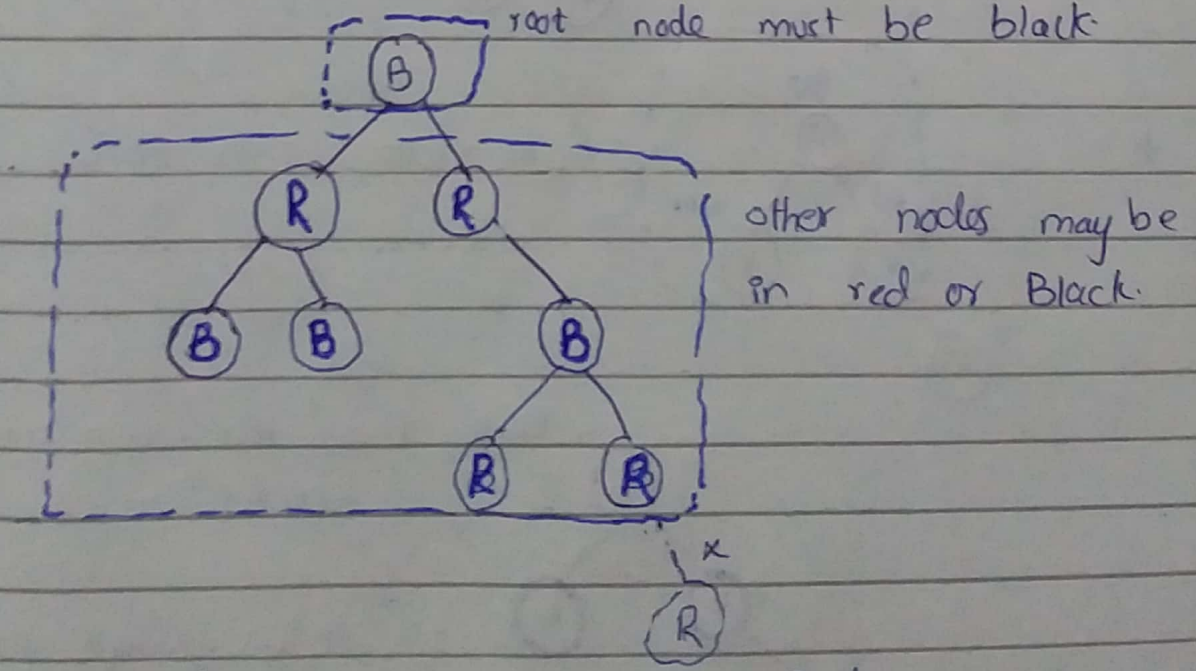
i) Root node is always in Black colar

ii) Each node (other than root) may be red or Black

iii) No of Black node in each path must be equal.

iv) No (R-R / Red - Red) Relationship.
(Parent → Red & Child → Red ⇒ Not allowed).

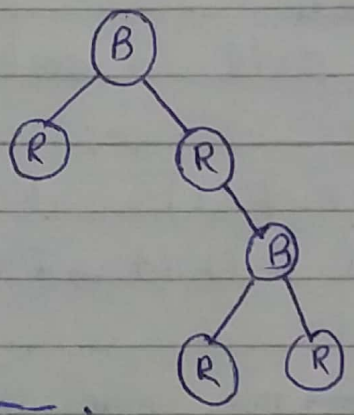
RBT

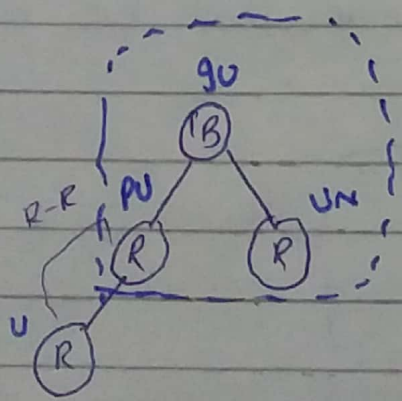


v) Inserted new node is in Red color by default.

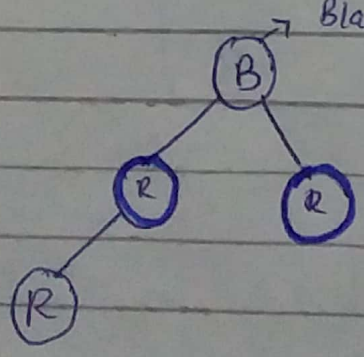
INSERTION Policies in RBT:-

↳ Create the Black node as the root node if tree is empty
↳ Insert New node as Red
↳ If parent is Black then insert
↳ If Parent is Red then,
  ↳ If sibling is Red, then re-color
  ↳ If sibling is Black, then rotate anti clockwise & change color
    - Sibling is empty / Black
    - Rotate & change the color.





* Re-color this Red

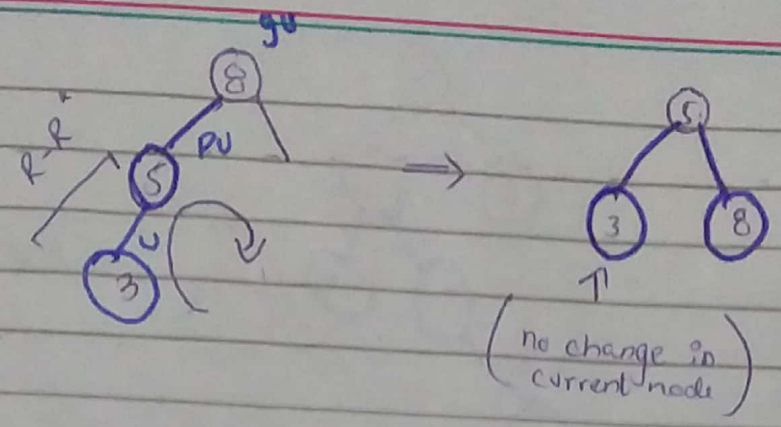


→ Black bcz root node is always black.

→ only change color not the values

○ → Black

● → Red



- Sibling is empty
- Rotation
- Left Rotate

(no change in current node)

Example:

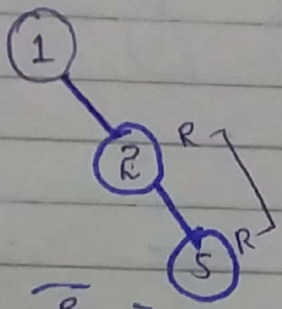
Statement:

Construct "RBT" of given data.

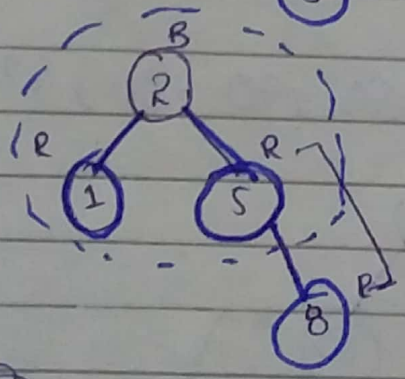
1, 2, 5, 8, 7, 4

○ → Black

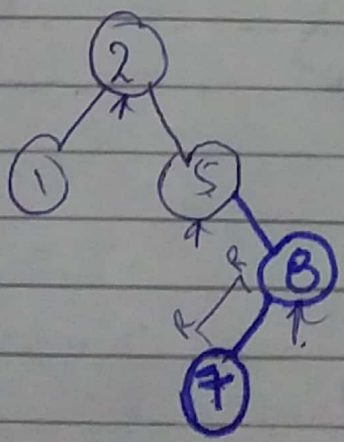
● → Red



Rotation:

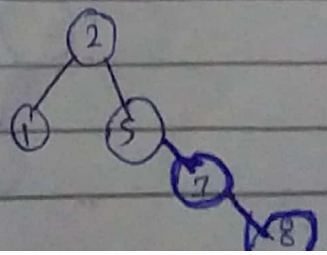


* Color change

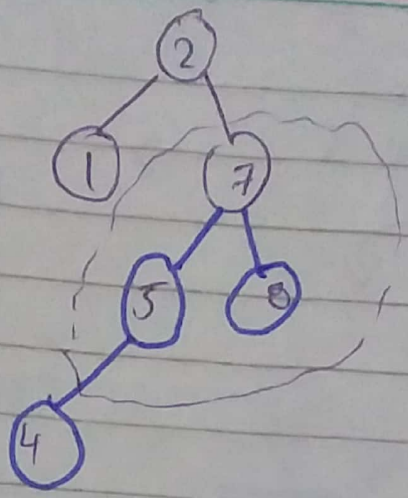


→ Double Rotation

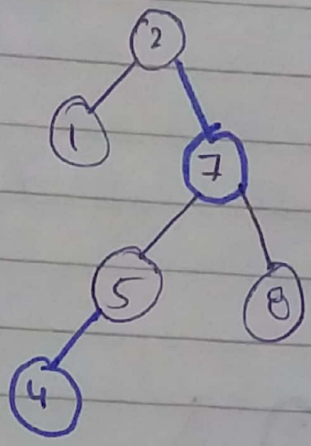
(1)



2)

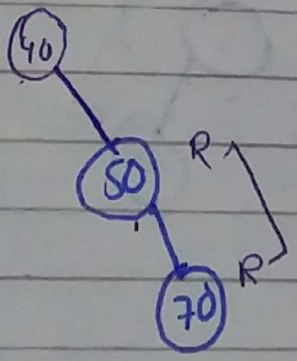


Change color

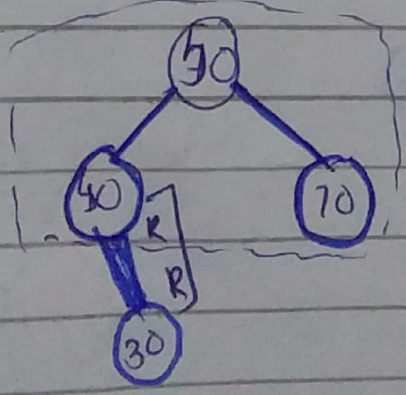


Example:

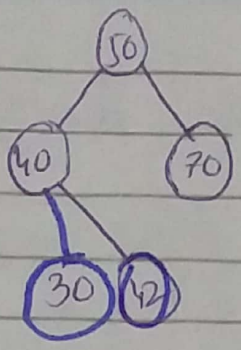
40, 50, 70, 30, 42, 15

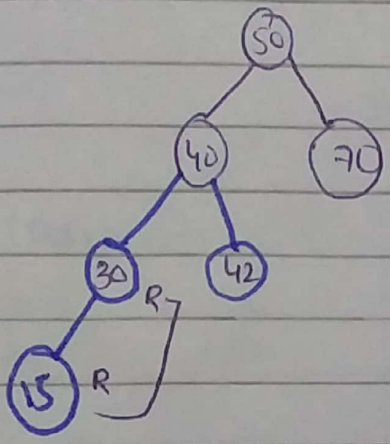


Rotate

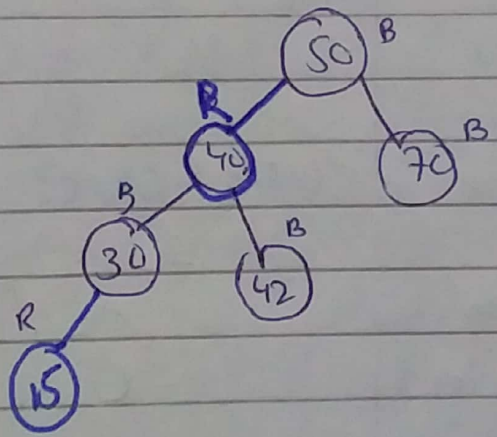


Change color

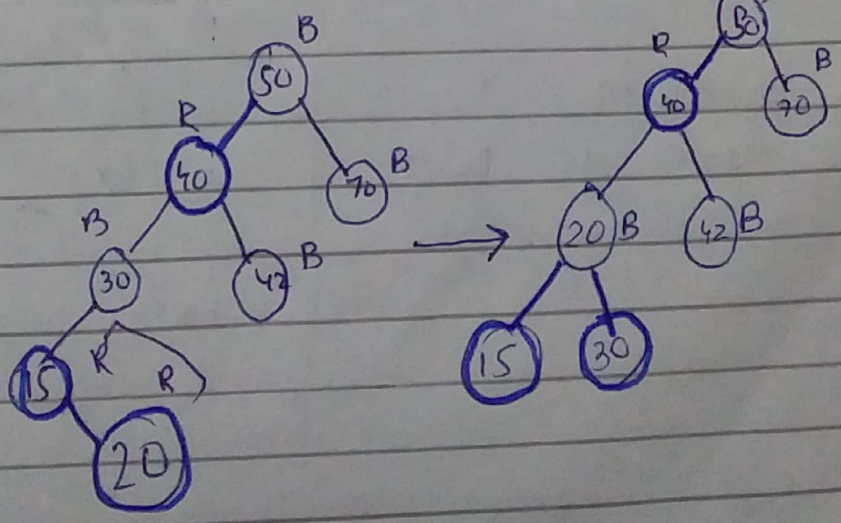




Change Colour



Insert 20



AOA

TREE STRUCTURE:

- Binary Search Tree
- Red Black Tree
- AVL Tree
- Heap

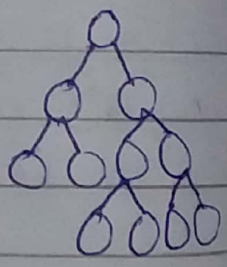
i) insertion
ii) Deletion
iii) Traversing
iv) Searching
v) Sorting (Hea

Linear / Non-Linear:

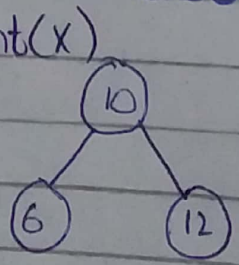
- Tree → Tree
- Graph

General Tree

Binary Tree (Each node has maximum 2 childs)



Binary Search Tree (BST):



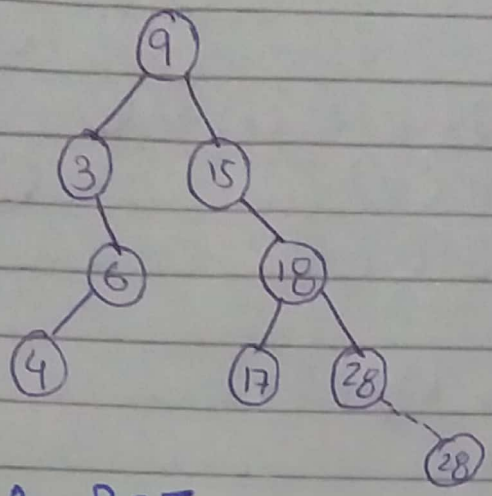
- In-order
- Pre-order
- Post-order

Left ≤ parent right ≥ parent

Full Fill both the conditions.

Construction of BST:

9, 5, 3, 18, 6, 4, 17, 28 (add 28)

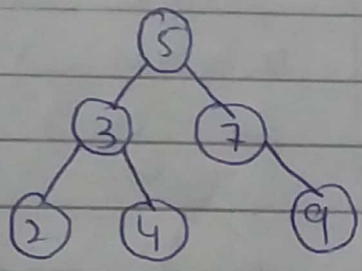


Traversal in BST:-

In-order : Left, root, Right

Pre-order : Root, Left, Right

Post order : left, Right, Root



- In-order :- (Always in ascending order)

  2, 3, 4, 5, 7, 9

- Pre. order :

  5, 3, 2, 4, 7, 9

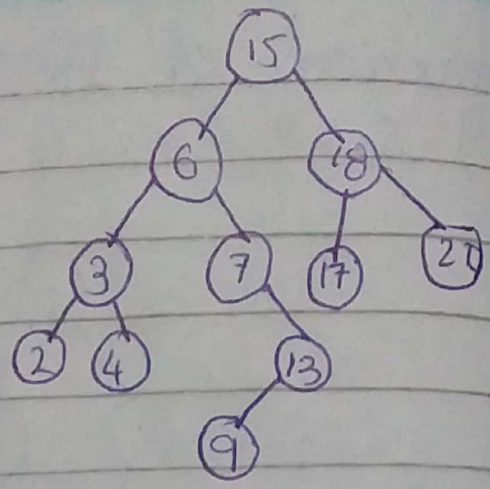
- Post order :

  2, 4, 3, 9, 7, 5

SEARCHING IN BST :-

Find 13 in BST.

Psuedo Code:-



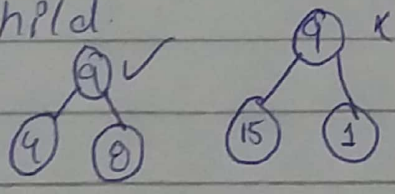

```
1) while x ≠ Nil && k ≠ key(x)
2)     if k < key(x)
3)         x ← left(x)
4)     else
           x ← Right(x)
5)     return x.
```

HEAP:

↳ Left Justified Binary Tree

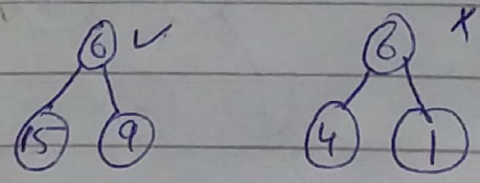
↳ **Max Heap** (By Default)

parent has max value then its child.



Min Heap

parent node has minimum value than its child.



Construct Max Heap:

9, 13, 18, 27, 16, 4, 56, 11