Today's content

- Tower of Hanoi
- Sorting Basics
- Selection Sort
- Bubble Sort
- Insertion Sort {idea}

Tower of Hanoi

→ Given 3 towers A, B and C

→ There are n-disks placed on tower A.

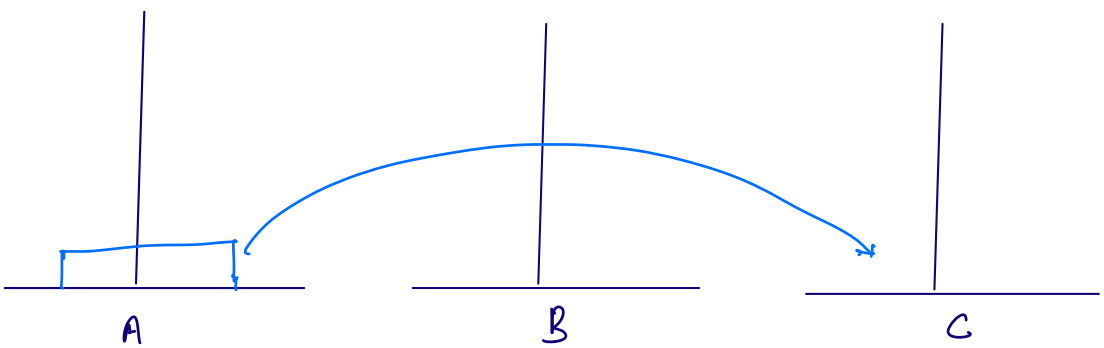
→ Move All the disks from A to C. {using B}

Note: ① Only one disk can be moved at a time.

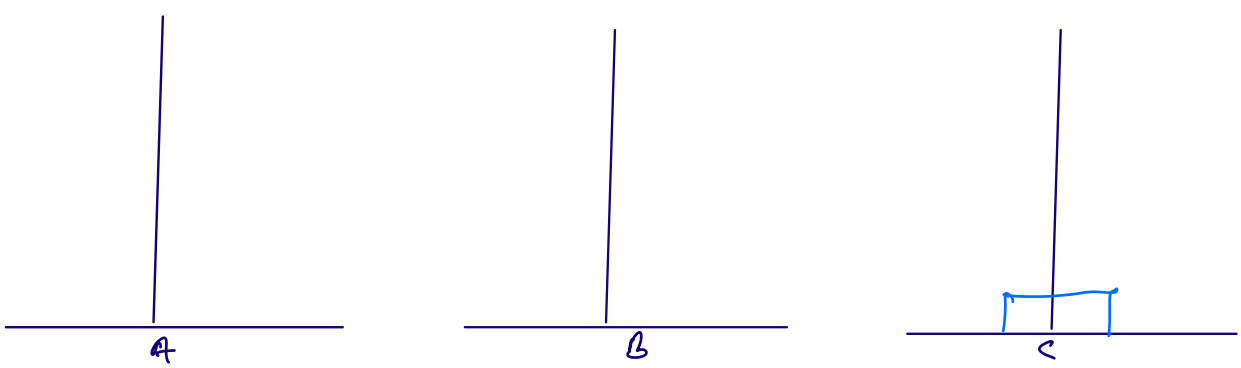
② larger disk can't be placed on a smaller disk.

Q → Print movement of the disks.

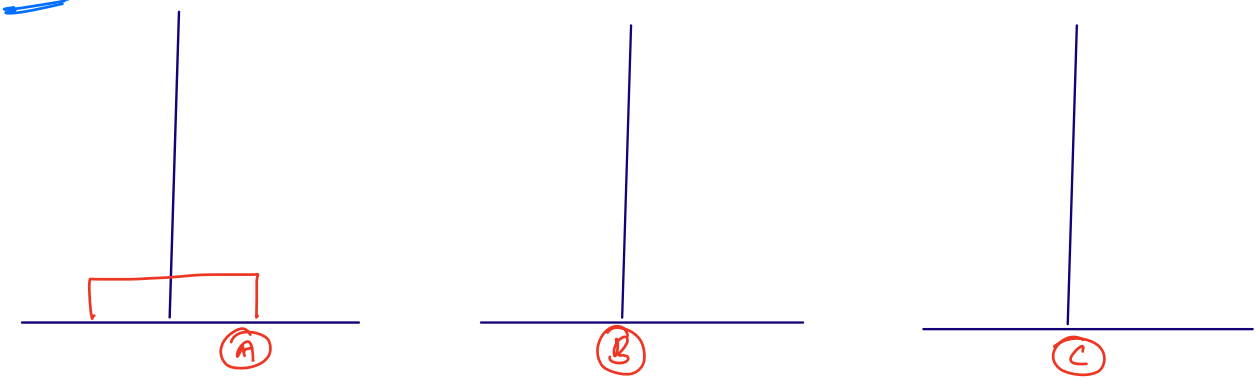
Eg (N=1)

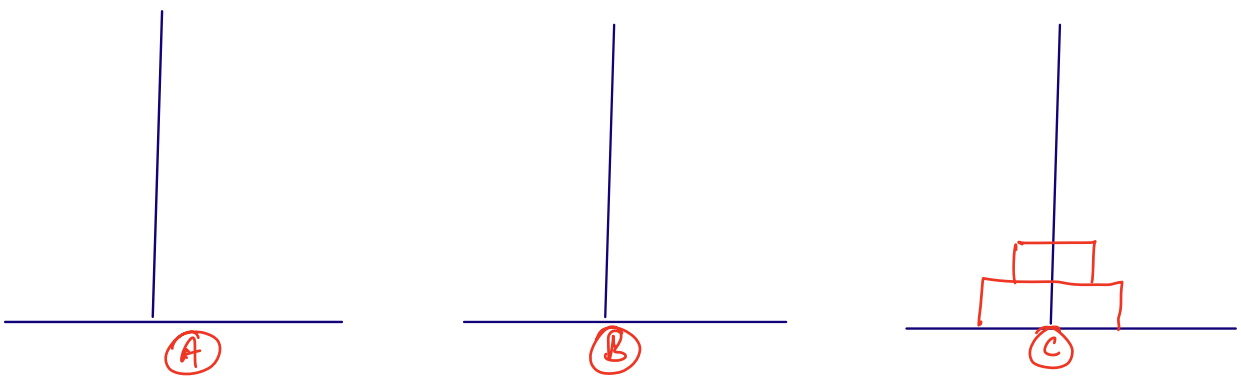


{move disk 1 from A to C}



N=2.



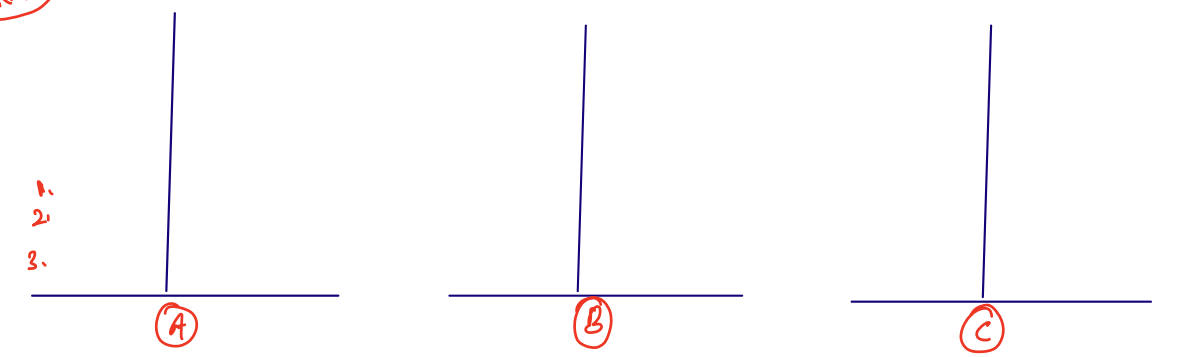


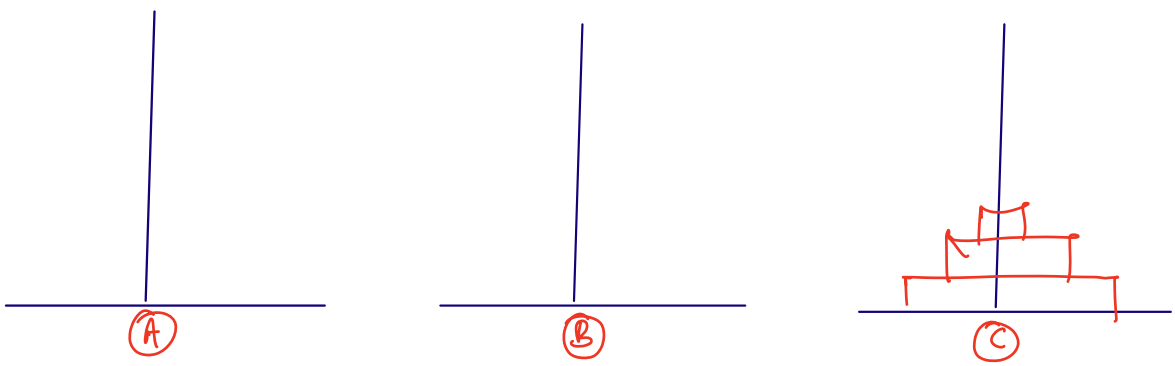
① Move disk 1 from A to B.

② Move disk 2 from A to C.

③ Move disk 1 from B to C.

N=3

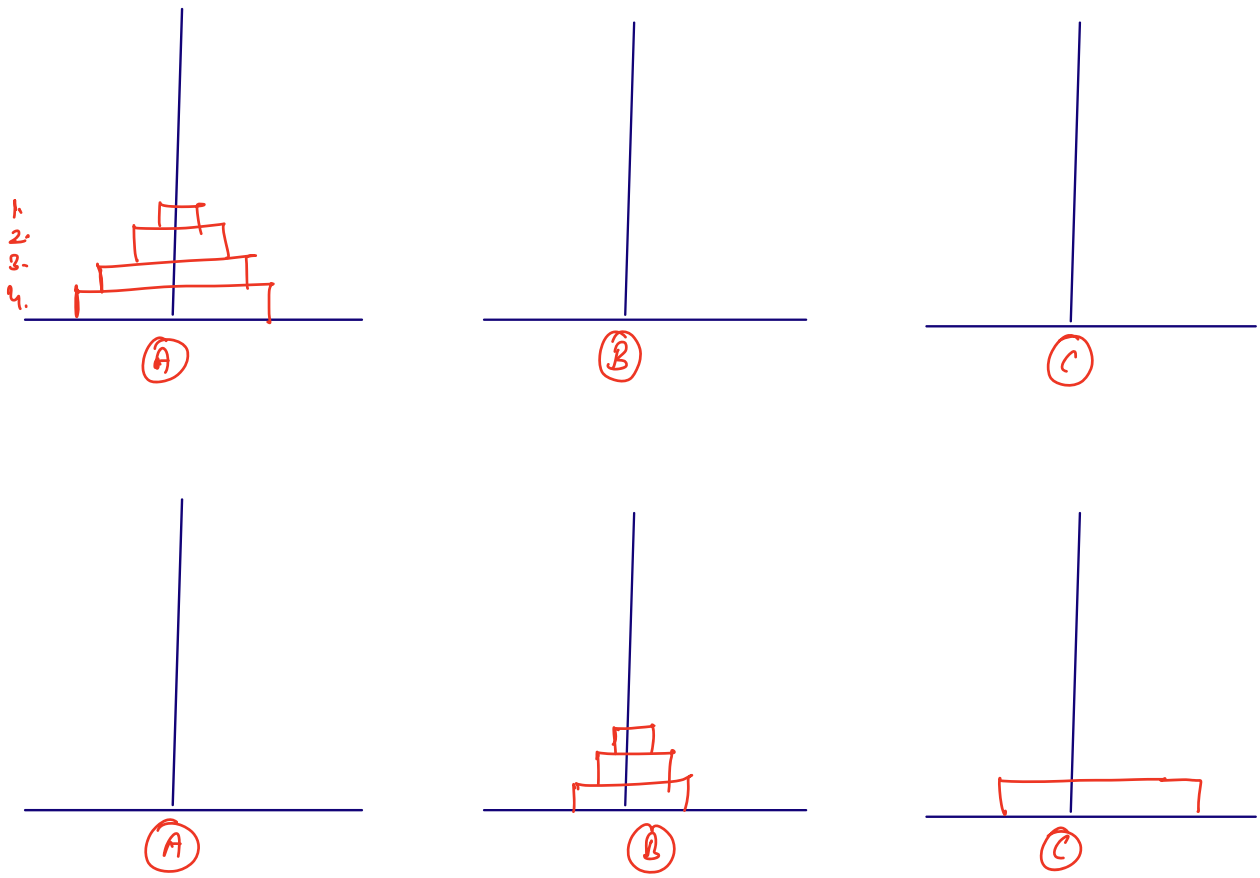




① Move disk 1 from A to C.
② Move disk 2 from A to B.
③ Move disk 1 from C to B.
④ Move disk 3 from A to C.
⑤ Move disk 1 from B to A
⑥ Move disk 2 from B to C
⑦ Move disk 1 from A to C.

o/p.

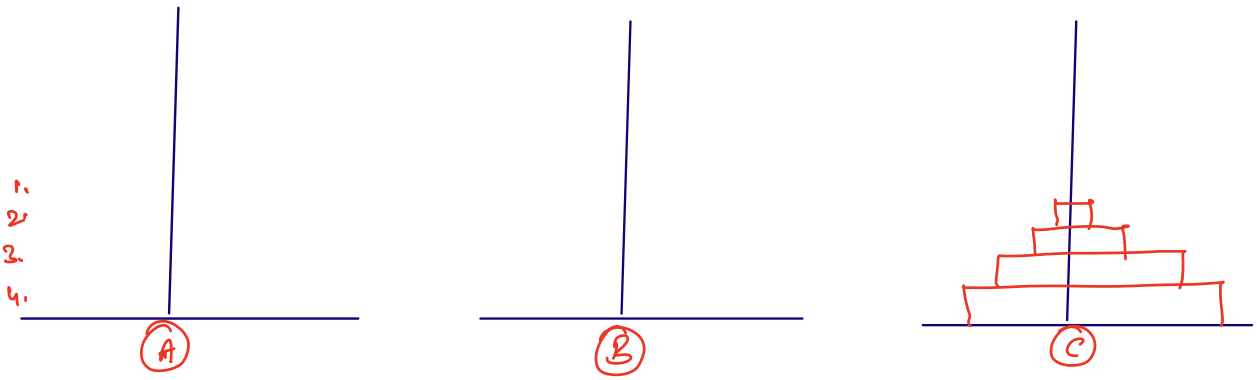
N=4



① Move 2 disks from A to B {using C} → sub-problem

② Move disk 4 from A to C.

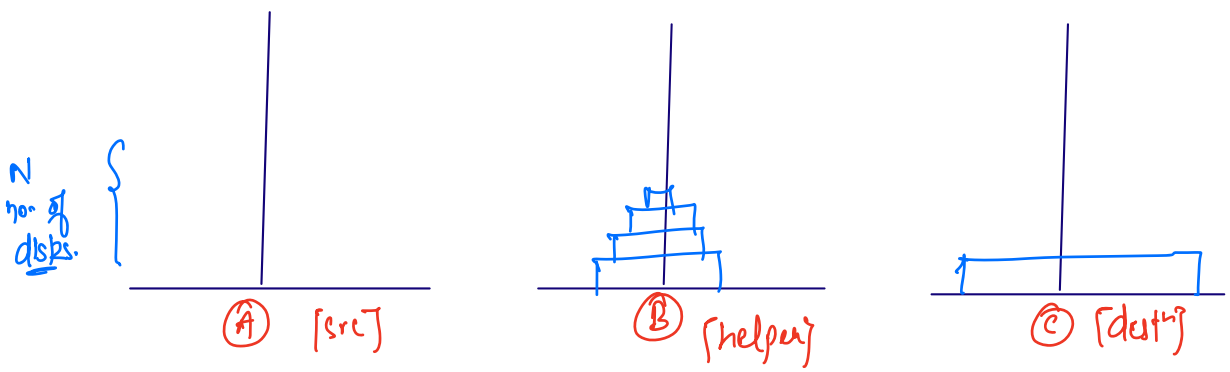
③ Move 2 disks from B to C {using A} → sub-problem.



① disk 1 from A to B
② disk 2 from A to C
③ disk 1 from B to C
④ disk 3 from A to B
⑤ disk 1 from C to A
⑥ disk 2 from C to B
⑦ disk 1 from A to B
⑧ disk 4 from A to C

⑨ disk 1 from B to C
⑩ disk 2 from B to A
⑪ disk 1 from C to A
⑫ disk 3 from B to C
⑬ disk 1 from A to B
⑭ disk 2 from A to C
⑮ disk 1 from B to C

total no. of movements : $2^N - 1$.



//Assm → Given N, print steps of moving N disks from S → D using helper & following all rules.

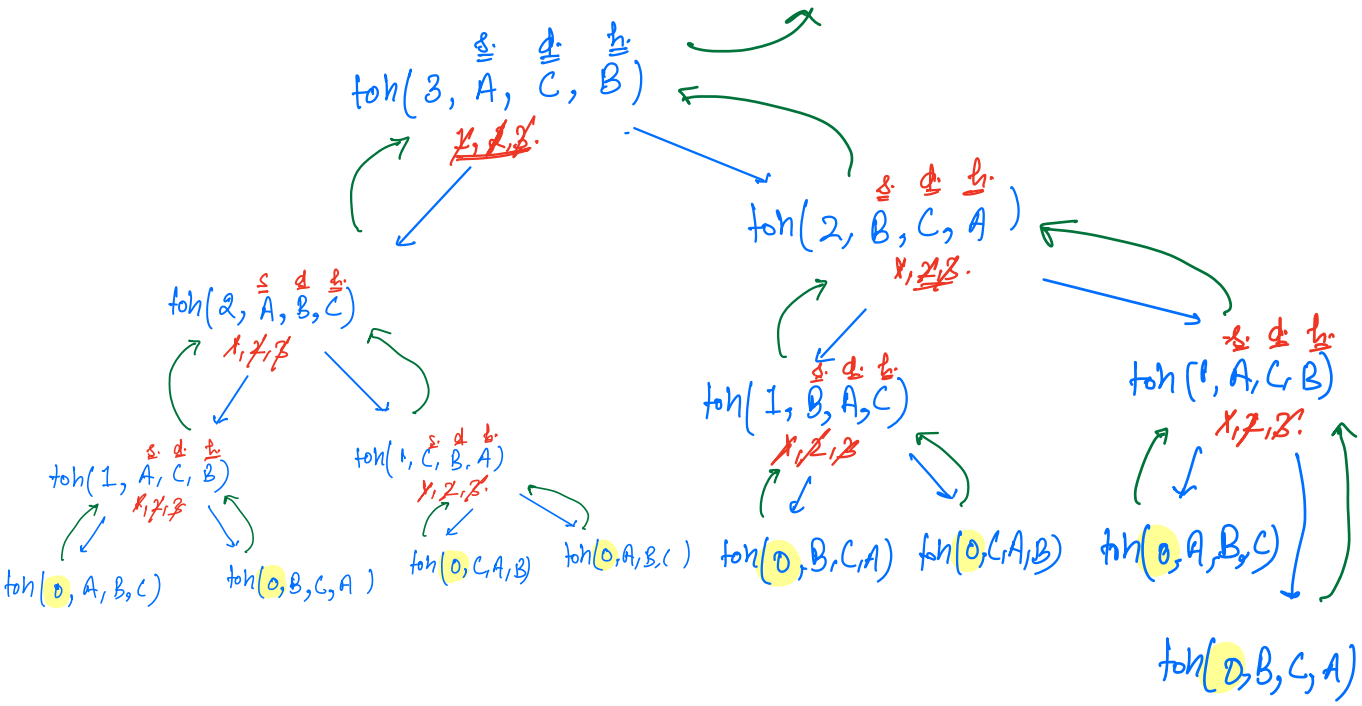
```
void toh( int N, char s(A), char d(C), char h(B)){
    B.Cond  if (N==0) {return}
    (1)     toh(N-1, s, h, d);
    (2)     print(Nth disk from s to d);
    (3)     toh(N-1, h, d, s);
}
```

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

$$T(N) = 2T(N-1) + 1.$$

$$\{T(0) = 1\}$$


```
void toh( int N, char s(A), char d(C), char h(B)){
    B.Cond  if (N==0) {return}
    (1)     toh(N-1, s, h, d);
    (2)     print(Nth disk from s to d);
    (3)     toh(N-1, h, d, s);
}
```



Move disk 1 from A to C.
Move disk 2 from A to B.
Move disk 1 from C to B.
Move disk 3 from A to C.
Move disk 1 from B to A.
Move disk 2 from B to C.
Move disk 1 from A to C.

Time Complexity.

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

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

$$T(N) = 2\left[2T(N-2) + 1\right] + 1$$

$$T(N) = 2^2 T(N-2) + (2^2 - 1)$$

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

$$T(N) = 4\left[2T(N-3) + 1\right] + 3$$

$$T(N) = 2^3 T(N-3) + (2^3 - 1)$$

Generalisation

$$T(N) = 2^K T(N-K) + (2^K - 1)$$

$$N - K = 0 \Rightarrow K = N$$

$$T(N) = 2^N T(0) + (2^N - 1)$$

$$= 2^N + 2^N - 1 = 2 \cdot 2^N - 1$$

T.C → $O(2^N)$

S.C → $O(N)$

Sorting. → arranging the data in inc/dec order based on parameter.

Eg → {4 8 10 12 15 20} → inc based on value

Eg2. → {100, 90, 85, 27, 3, -10} → dec. based on value.

Eg3 → {1, 3, 5, 7, 4, 9, 6, 10, 12} → inc. order based on no. of factors

factors : 1 2 2 2 3 3 4 4 6

Sorting Algorithms

↓

Stable Sorting.

In-place sorting.

→ without using any extra space

→ SC : O(1)

Actors.	Remuneration
SRK	40 CRS
Akshay Kumar	30 CRS
Rajnikanth	40 CRS
P.C	25 CRS
Alia Bhatt	30 CRS
Salman Khan	50 CRS
Yash	40 CRS.

Arrange on basis of Remuneration

P.C → 25 Cr

Akshay Kumar → 30 Cr

Alia Bhatt → 30 Cr

SRK → 40 Cr

Rajnikanth → 40 Cr

Yash → 40 Cr

Salman Khan → 50 Cr

Stable Sorting :

2 data points having same parameter value, their relative order before sorting & after sorting should be same.

Eg. → arr → [10 2 5 7 2 9]

After sorting → [2 2 5 7 9 10]

Q) Given N array elements. Find k^{th} smallest element.

arr[8]: { 2, 8, -1, 7, 4, 5, 11, 3 }

k=1	k=2	k=3
-1	2	3

idea → Sort it in increasing order → ans = arr[k-1]

{ T.C → O(NlogN)
S.C → O(1) }

k=5.

arr[8]: { 2, 8, -1, 7, 4, 5, 11, 3 }
idx: 0 1 2 3 4 5 6 7

	idx of minimum element	
i=0 :	[0,7]	2
i=1 :	[1,7]	2
i=2 :	[2,7]	7
i=3 :	[3,7]	4
i=4 :	[4,7]	5

swap (arr[0], arr[2])
swap (arr[1], arr[2])
swap (arr[2], arr[7])
swap (arr[3], arr[4])
swap (arr[4], arr[5])

i=5 : → [0,5] is sorted.
i=6 : → [0,6] is sorted.
i=7 : → [0,7] is sorted. → whole array is sorted.

Selection Sort.

```
void selectionSort ( arr, N){
    for ( i=0 ; i< N ; i++) {
        // find minimum elemt & its index in [i, N-1].
        minVal = arr[i], minIdx = i
        for( j = i+1 ; j < N ; j++) {
            if (arr[j] < minVal) {
                minVal = arr[j]
                minIdx = j
            }
        }
        swap arr[i] with arr[minIdx]
    }
}
```

T.C → $O(N^2)$
S.C → $O(1)$

Data: 2 2 1 4 —(if stable)→ 1 2 2 4

Algo. 2_0 2_1 1_2 4_3 → { Selection sort is not a stable sorting. }

Bubble Sort.

arr[6]: { 9 7 2 5 3 9 }

: { -1 7 2 5 3 9 }

: { -1 2 3 5 7 9 }

: { -1 2 3 5 7 9 }

max element will be present at its correct posn.

last 2 max elements at correct posn.

last 3 max elements at correct posn.

Pseudo-code:

```
void bubbleSort ( arr , N ) {
    for ( i = 1 ; i < N ; i++) {
        for( j = 0 ; j < N-i ; j++) {
            if (arr[j] > arr[j+1]){
                swap(arr , j , j+1);
            }
        }
    }
}
```

T.C → $O(N^2)$
S.C → $O(1)$

arr → { 2 2 1 4 }
1 2

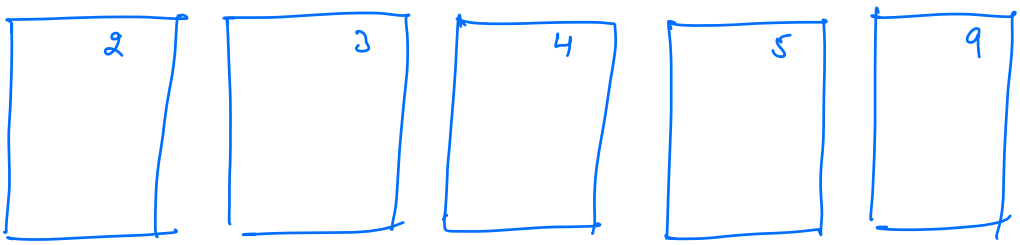
arr → { 2 1 2 4 }
1 2

arr → { 1 2 2 4 }

{ → stable sorting ✓
→ in-place sorting ✓ }

Insertion Sort:

↳ arrangement of cards.



arr →

0	1	2	3	4	5
9	3	5	2	4	7

arr →

0	1	2	3	4	5
3	9	5	2	4	7

5 9

arr →

0	1	2	3	4	5
3	5	9	2	4	7

2 3 5 9

arr →

0	1	2	3	4	5
2	3	5	9	4	7

4 5 9

arr →

0	1	2	3	4	5
2	3	4	5	9	7

j-1 j

7 9

arr[j-1] > arr[j]

Pseudo-code:

```
void insertionSort (arr, N) {
    for (i = 1 ; i < N ; i++) {
        for (j = i ; j > 0 ; j--) {
            if (arr[j-1] > arr[j]) {
                swap(arr, j, j-1)
            }
            else {
                break;
            }
        }
    }
}
```

{ T.C → $O(N^2)$
S.C → $O(1)$ }

arr → | 2 | 3 | 5 | 9 | 12 | 20 |
0 1 2 3 4 5

→ stable ? (#todo)

⇒ { H.W till Combinations. } → P.S.S. on friday.

N=30

→ perfect squares will remain open ?

⇓

no. of factors are odd.