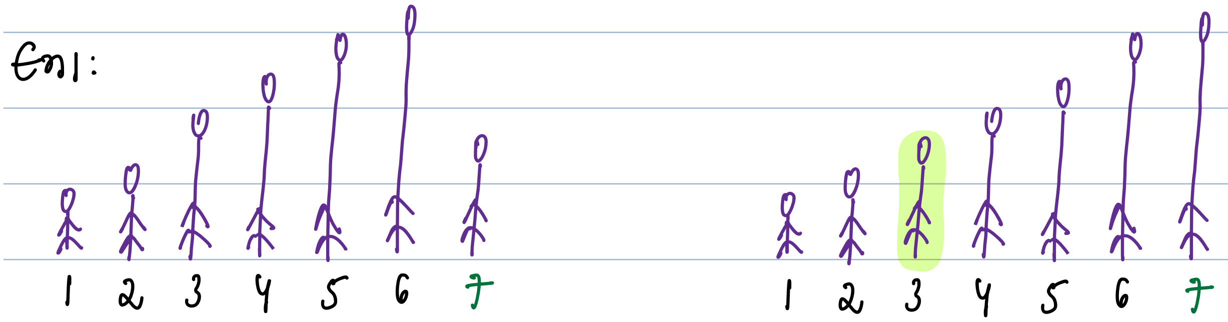
Todays Content

1. Insertim sort
2. Merge 2 sorted arrays
3. Merge 2 sorted subarrays.

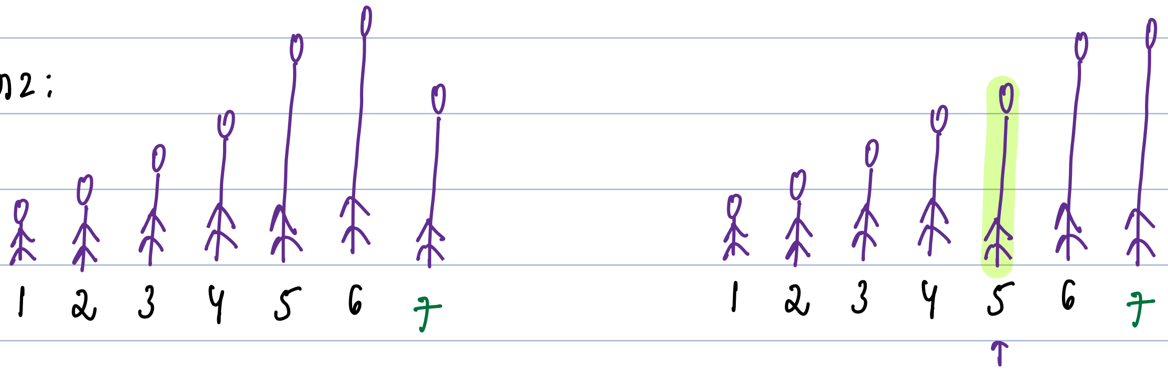
Insertion Sort :

We insert 1 element in existing sorted data to make entire data sorted

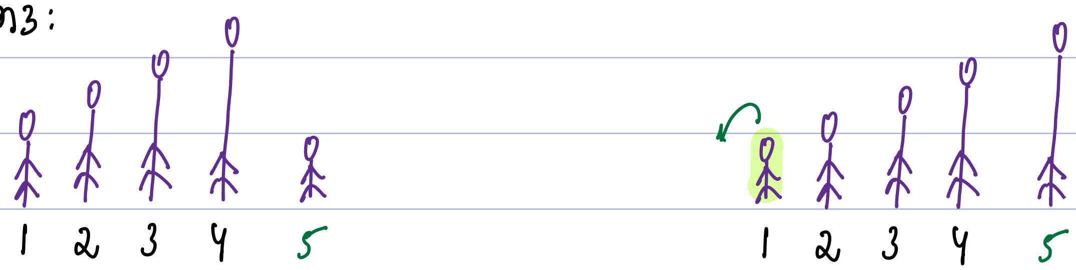
Ex1:



Ex2:



Ex3:



```
while(nper!=start && pper > nper){
    swap pper & nper
}
```

Insertion Sort:

0 ‖ 1 2 3 4 5

ar[ ] : { 10 ‖ 9 4 8 6 2 }

Dry Run: Sort Insert

0 1 ‖ 2 3 4 5

i=0 [0 0] ar[1] ar[ ] : { 9 10 ‖ 4 8 6 2 }

j ← j

0 1 2 ‖ 3 4 5

i=1 [0 1] ar[2] ar[ ] : { 4 9 10 ‖ 8 6 2 }

j ← j ← j

0 1 2 3 ‖ 4 5

i=2 [0 2] ar[3] ar[ ] : { 4 8 9 10 ‖ 6 2 }

j ← j ← j

0 1 2 3 4 ‖ 5

i=3 [0 3] ar[4] ar[ ] : { 4 6 8 9 10 ‖ 2 }

j ← j ← j ← j

0 1 2 3 4 5 ‖

i=4 [0 4] ar[5] ar[ ] = { 2 4 6 8 9 10 ‖ }

j ← j ← j ← j ← j ← j

i=5 [0 5] # Entire arr[ ] is sorted stop process.

Note: when i is at last_index stop process

```
void Insertion( int [] ar, int N){   TC: O(N^2)  SC: O(1)
    for(int i=0; i< n-1; i++){
        # [0..i] is sorted  Insert arr[i+1];
        int j= i+1;
        while( j>0 && ar[j-1] > ar[j] ){
            swap ar[j-1] & ar[j];
            j--;
        }
    }
}
```

TODO: Take an almost sorted array & apply BS, SS, IS & compare iterations.

Q. Given 2 sorted arrays A[N] B[M] create C[N+M] which contains overall sorted data

	0	1	2	3
A[4] :	{ 7	10	11	14 }

	0	1	2
A[4] :	{ 3	6	10 }

	0	1	2
B[3] :	{ 3	8	9 }

	0	1	2	3
B[3] :	{ 5	14	20	25 }

	0	1	2	3	4	5	6
C[7] :	{ 3	7	8	9	10	11	14 }

	0	1	2	3	4	5	6
C[7] :	{ 3	5	6	10	14	20	25 }

Idea1:

1. Create C[N+M]
2. Copy A[N] → C[] & B[M] → C[]
3. Sort C[N+M]

TC1: $O(1+N+M+\underbrace{(N+M)^2}_{BS/IS/SC}) = O(N+M)^2$

TC2: $O(1+N+M+\underbrace{(N+M)\log(N+M)}_{\text{Inbuilt Sort}}) = O(N+M)\log(N+M)$

#Idea2: At each step

Compare min of A[] & B[] & keep min among them in C[]

Note: 2 keep track of min in A[] & B[] we use 2 variables

Note: 3rd variable to keep a track of index in C[]

#DryRun1:

	0	1	2	3	4
A[5] : {	~~7~~	~~9~~	~~11~~	14	18 }

N

P1

Note: P2 == M : stop

copy remaining ele to C[]

	0	1	2	3
B[4] : {	~~3~~	~~8~~	~~10~~	~~12~~ }

M

P2

	0	1	2	3	4	5	6	7	8
C[9] : {	3	7	8	9	10	11	12	14	18 }

P3

#DryRun2:

Note: P1 == N : stop

copy remaining ele to C[]

	0	1	2	3
A[4] : {	~~7~~	~~9~~	~~11~~	~~14~~ }

P1

	0	1	2	3	4	5	6	7
B[8] : {	~~3~~	~~8~~	~~10~~	~~12~~	15	16	18	20 }

P2

	0	1	2	3	4	5	6	7	8	9	10	11
C[12] : {	3	7	8	9	10	11	12	14	15	16	18	20 }

P3

```
int[]  merge( vector<int> &A, vector<int> &B){ TC: O(N+M)
                                                SC: O(1)
    int N= A.size(), M= B.size();
    vector<int> c(N+M, 0);

    int P1=0, P2=0, P3=0;
    while( P1<N && P2<M ){# To compare both P1 & P2 should be in array
        if( A[P1] < B[P2]){
            C[P3] = A[P1]; P3++; P1++;
        }
        else{
            C[P3] = B[P2]; P3++; P2++;
        }
    }

    while( P1 < N){
        C[P3] = A[P1]; P3++; P1++;
    }

    while( P2<M){
        C[P3] = B[P2]; P3++; P2++;
    }
}
```

38: Merge 2 consective sorted subarrays #Include

Given ar[N] elements & 3 indices s, m, e

#Subarray [s.. m] is sorted

#Subarray [m+1.. e] is sorted # s.. m m+1.. e

#Sort entire subarray from [s.. e] in arr[]

		0	1	2	3	4	5	6	7	8	9	10	11	
ar[12] =	{	4	8	~~1~~	~~2~~	8	9	11	~~3~~	~~4~~	~~7~~	13	0	}
						P_1						P_2		

# s	m	e
2	6	9

tmp[e-s+1] =

		0	1	2	3	4	5	6	7	
tmp[8] =	{	-1	2	3	4	7	8	9	11	}
							P_3			

#Copy tmp[0-7] = ar[2-9]

				s	s+1	s+2					e			
		0	1	2	3	4	5	6	7	8	9	10	11	
ar[12] =	{	4	8	-1	2	3	4	7	8	9	11	13	0	}

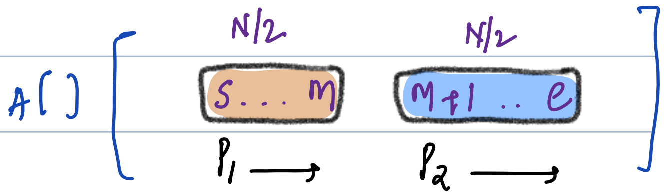
#ar : {s..m} is sorted {M+1..e} is sorted

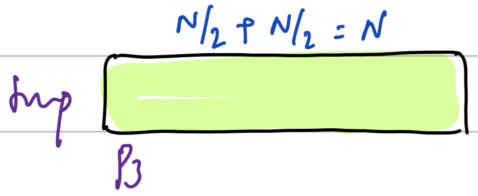
Sort entire subarray {s..e}

TC: O(N+N) = O(N)

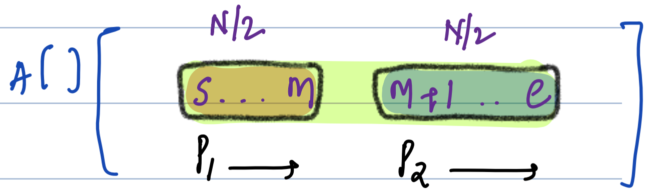
SC: O(N)

```
void merge( int ar[], int s, int m, int e){
    int tmp[e-s+1]; # 0... e-s
    int P1=s, P2 = M+1, P3 =0;

    while( P1<=m && P2<=e ){
        if( A[P1] < A[P2] ){
            tmp[P3]= A[P1]; P3++; P1++;
        }
        else{
            tmp[P3]= A[P2]; P3++; P2++;
        }
    }
    while( P1<=m){
        tmp[P3]= A[P1]; P3++; P1++;
    }

    while( P2<=e){
        tmp[P3]= A[P2]; P3++; P2++;
    }

    # Copy tmp[ 0 ... e-s] → ar[s.. e]

    for( int i=s; i<=e; i++){
        ar[i]= tmp[i-s]
    }
}
```





Copy tmp[] → A[] : N



#DryRun:

i	ar[i]= tmp[i-s]
s	ar[s]= tmp[0]
s+1	ar[s+1]= tmp[1]
s+2	ar[s+2]= tmp[2]