Todays Content

1. Sorting Intro
2. Bubble Sort
3. Selection Sort
4. Insertion Sort

Sorting: Arranging data in inc/dec order according to a parameter

	0	1	2	3	4	5	
Ex1: A[] = {	2	3	9	12	17	19	} : Inc based on value
Ex2: A[] = {	19	6	5	2	-1	-19	} : Dec based on value
Ex3: A[] = {	2	7	4	9	6	10	} : Inc based on factors
#factors =	2	2	3	3	4	4	

Sorting library? TC: $O(N \log_2 N)$ To sort N elements.

1. To sort int arr[]:

```
sort(arr, arr+n);  # It will sort arr[] from arr[0] to arr[N-1] in Inc
sort(arr, arr+n, greater<Datatype>());  # It will sort arr[] in Dec
# include <function>  # for greater
```

2. To sort vector<int> v;

```
sort(v.begin(), v.end());  #Sort in increasing order
sort(v.begin(), v.end(), greater<Datatype>());  # Sort in Decreasing Order
# include <function>  # for greater
```

Stable Sorting: When 2 data points have same value, their relative order should be same, before & after sorting, that type of sorting algo is called stable sorting Algo.

Ex: ar[5] = {	1	5	2	6	2	} #Sort inc based on value;
#Sort1 = {	1	2	2	5	6	} # Sort1 is stable sort
#Sort2 = {	1	2	2	5	6	} # Sort2 is not stable.

Inplace Sorting:

Sorting Algo with SC: O(1) it is considered as Inplace Sorting.

All Sorting Algorithms

#Basic:	Bubble Sort	Selection Sort	Insertion Sort
TC :	$O(N^2)$	$O(N^2)$	
Stable Sort :	✓	*	
Inplace Sort :	✓	✓	

#Optimized:	Merge Sort	Quick Sort	Heap Sort
TC :			
Stable Sort :			
Inplace Sort :			

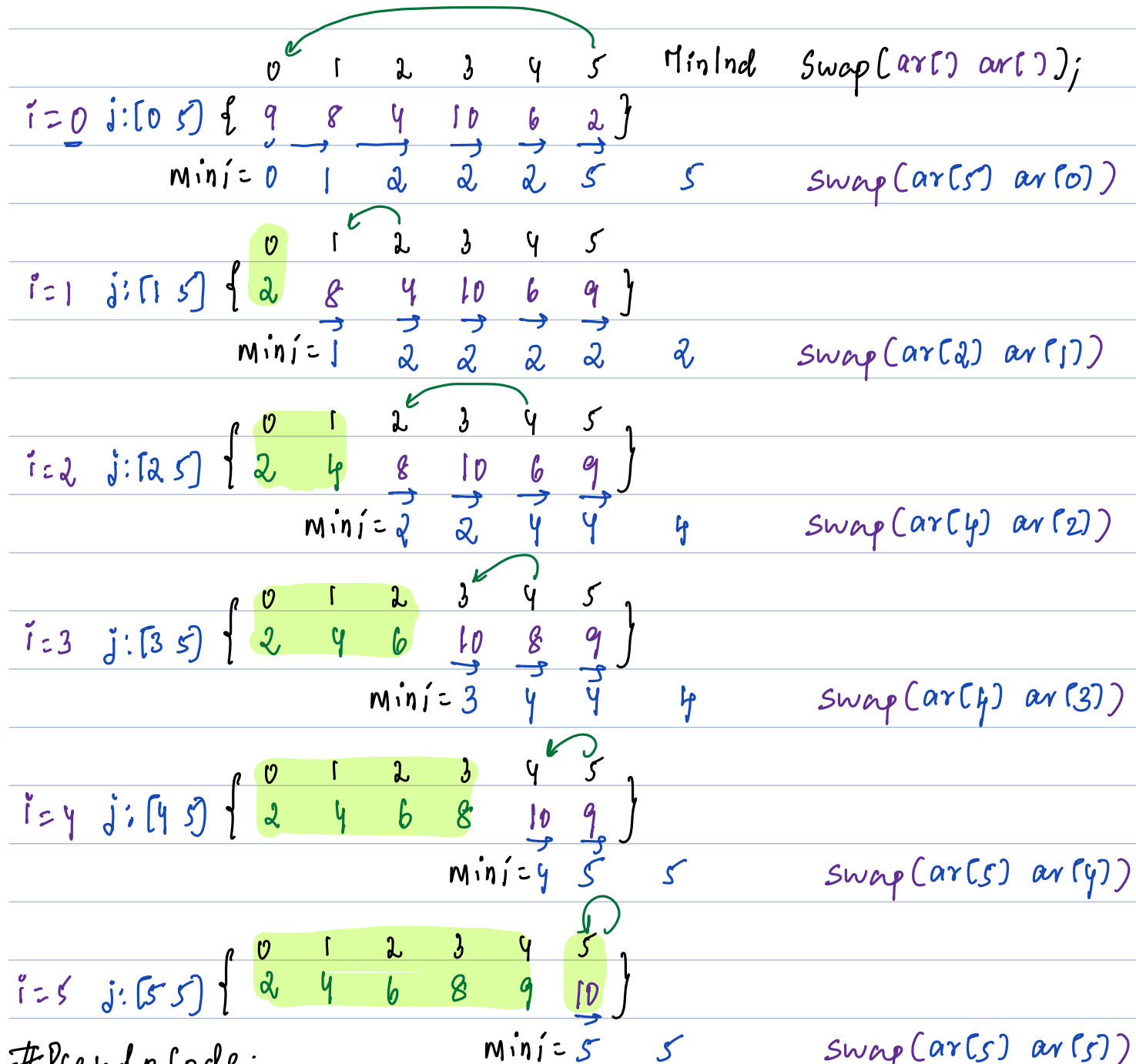
#Linear:	Count Sort	Radix Sort
TC :		
Stable Sort :		
Inplace Sort :		

Selection Sort:

At each iteration:

Iterate & select min[index with min v] & keep in it's sorted position.

	0	1	2	3	4	5
ar[] = {	9	8	4	10	6	2 }

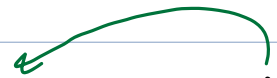


#Pseudo code:

```
i : 0 to N-1;
    mini = i;
    # j = [i.. N-1] # Iterate & calculate index with min value;
    swap ar[mini] & ar[i]
```

```
void selectim(int[] ar, int N){ TC: O(N^2)  SC: O(1)
    for(int i=0; i<N; i++){
        #Calculate mini;
        int mini = i;  # iterate from [i..N-1]
        for( int j=i; j<N; j++){
            if( ar[j] < ar[mini]){
                mini = j;
            }
        }
        swap ar[mini] & ar[i]
    }
}
```

By Run:

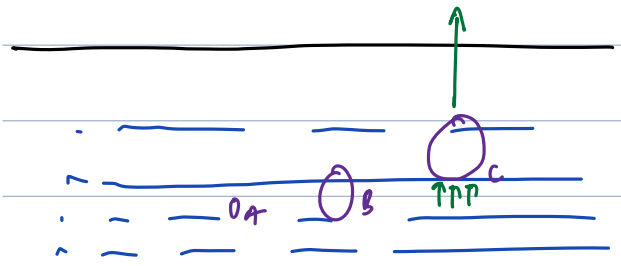


	0	1	2	3	4	5	MinInd	Swap(ar[] ar[]);
i=0 j:[0 5] {	6	5	6	3	8	7 }		
mini =	0	1	1	3	3	3	3	swap ar[3] ar[0]

	0	1	2	3	4	5
i=0 j:[0 5] {	3	5	6	6	8	7 }

Obs: In abve case, 6 6 are changing relative order,
Stability is not maintained in selectim srt.

#Idea Bubble Sort: Bigger bubble will come out on top.



Bubble Sort At each iteration: We compare adj elements, if they are not in correct order (Swap) we swap elements

#Generalization ar[N]

	0	1	2	3	4	5	
ar[] :	9	8	4	10	6	4	i j < N-i-1
i=0:	8	4	9	6	4	10	i=0 j<5; # j=5 stop
i=1:	4	8	6	4	9	10	i=1 j<4; # j=4 stop
i=2:	4	6	4	8	9	10	i=2 j<3; # j=3 stop
i=3:	4	4	6	8	9	10	i=3 j<2; # j=2 stop
i=4:	4	4	6	8	9	10	i=4 j<1; # j=1 stop
i=5:	4	4	6	8	9	10	i=5 j<0; # j=0 stop

```
void bubbleSort(int[] ar, int N){ TC: O(N^2)  SC: O(1)
    for(int i=0; i<N; i++){
        int swap=0;
        for(int j=0; j<N-i-1; j++){
            #Comp ar[j] & ar[j+1]
            if( ar[j] > ar[j+1] ){
                swap ar[j] & ar[j+1]
                swap++;
            }
        }
        if(swap==0){ break; }
    }
}
```

#Optimization: 2

ar[] :	8	3	5	10	9
i=0:	3	5	8	9	10
i=1:	3	5	8	9	10

#Obs:

for each iteration:

Calculate no: of swaps

if swaps == 0:

ar[] sorted break it