Theme: Sorting

Date: / /

Sat Sun Mon Tue wed Thu Fri

Part 1 ← Insertion Sort
" Bubble "
" Selection "

Selection Sort

→ we select minimums

13 16 24 52 20 9 → 9 13 20 24 ....

Step 1: Get the minimum of the entire array

→ Select minimum & swap ←

swap at index 0 & minimum index {0-n-1}
swap at " 1 & " " {1-n-1}
swap at " 2 & " " {2-n-1}
⋮
n-2

```
for (i=0 ; i<n-2 ; i++) {        mini = ~~arr~~ i;
    for (j=1; j<n-1 ; j++) {
        if (arr[j] < arr[mini]){
            mini = j ; }
    swap(arr[mini], arr[i]);
}
```

Theme:

Date: / /

Sat Sun Mon Tue Wed Thu Fri

How to swap ?

arr[i] ⟷ arr [mini]

temp = arr [mini]

arr [mini] = arr [i]

arr [i] = temp

```
selection_sort(int arr[], int n)
{  for(i=0, i<n-2; i++){
     int min = i;
     for(j=i; j<n-1; j++)
     {  if(arr[j] < arr[min]) min = j;}
   //SWAP
     int temp = arr[min];
     arr[min] = arr[i];
     arr[i] = temp;
}
```

Time complexity → n + n-1 + n-2 + n-3 + ... 2+1

↳ $O(n^2)$ → $\frac{n \times (n+1)}{2}$

Best
Worst
Avg.

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

Theme:

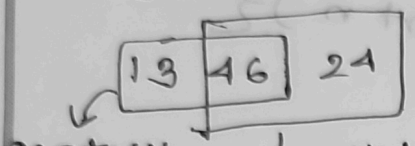
Date: / /

Sat Sun Mon Tue wed Thu Fri

Bubble Sort

→ pushes maximum to the last

→ By adjacent swaps



are they in sorted order? Y → no change

↳ Not sorted = Swap

i
0 → n − 1
0 → n − 2
0 → n − 3
0 → n − 4
⋮
0 → 1

```
for (i=n-1; i >= 1 ; i--) {
    for (j=0; j<= i-1 ; j++) {
        if ( a[j] > a[j+1]) { swap }
    }
}
```

i−1 → otherwise we get runtime error because its not present in the array

```
void bubble_sort(int arr[], int n) {
    for (i=n-1; i>=0 ; i--) {
        didswap = 0
        for(j= 0; j<= i-1 ; j++) {
            if (arr[j] > arr[j+1]) {
                int temp = arr[j+1];
                arr[j+1] = arr[j];
                arr[j] = temp ;
                didswap = 1;
            }
        }
        if (didswap == 0) break; //never ran
    }
}
```

what if its already sorted?

correct order = no swap

TC → O(N)

Best case with optimized

TC → $O(n^2)$ → worst / Avg.

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Insertion Sort

-> Takes an element & place it in its correct order.

```
void insertionSort(int arr[], int n){
    for(i=0, i<=n-1; i++){
        int j = i;
        while(j>0 && arr[j-1] > arr[j])
            int temp = arr[j-1]
            arr[j-1] = arr[j]
            arr[j] = temp;

            j--;
        }
    }
```

TC → 0 + 1 + 2 + 3 + 4

$n \times \frac{n+1}{2} = O(n^2)$ → Avg / worst

Best case → O(N) No swaps happened. (Sorted)