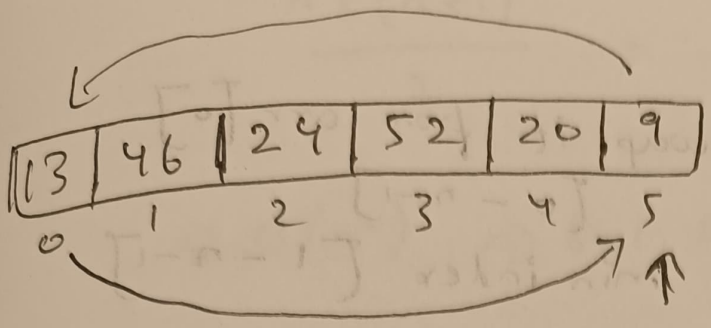
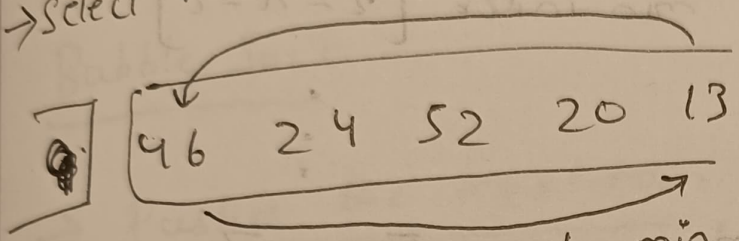
Selection Sort

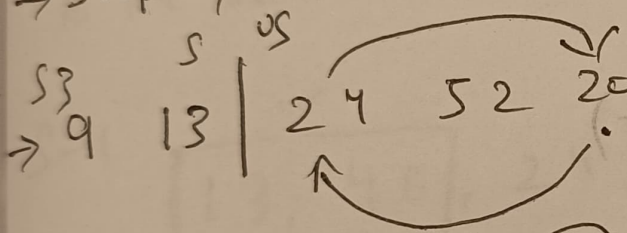


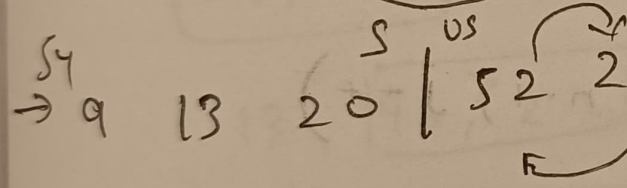
ascending order → 9, 13, 20, 24, 46, 52

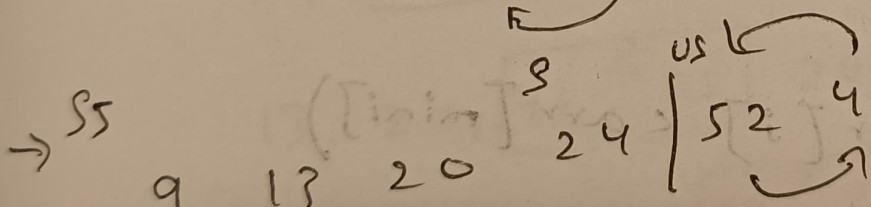
→ select minimums and swap (step 1)



→ Step 2, select next min and swap







→ 9 13 20 24 46 52 ⇒ Sorted

Get the min and swap it

Pseudo Code

observation:

→ Figure out mini, swap it from arr.[0] [0 - n-1]

→ Swap at index 1, min index [1 - n-1]

→ swap at " 2, min index [2 - n-1]

⋮ ⋮

n-2, [n-1]

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

// to swap

```
temp = arr[mini]
arr[mini] = arr[i]      arr[i] = temp;
```

$n + n-1 + n-2 \ldots\ldots 2+1$

T.C ⇒ $\frac{n \times (n+1)}{2} = \frac{n^2}{2} + \frac{n}{2}$

⇒ $O(n^2)$ ⇒ Best, Average, Worst.

Bubble Sort

→ Pushes the maximum to the left and opposite to the selection sort ↳ to the last by adjacent steps.

S-1 [13, 46], 24, 52, 20, 9

13, [46, 24], 52, 20, 9

13, 24, [46, 52], 20, 9

13, 24, 46, [52, 20], 9

13, 24, 46, 20, [52, 9]

13, 24, 46, 20, 9, [52]

So on...

now if you observe the maximum is at the right.

S-2 13, 24, 20, 46, 9, 52 ⇒ 13, 24, 20, 9, [46, 52]

S-3 j, j+1

13, [24, 20], 9, 46, 52

13, 20, [24, 9], 46, 52

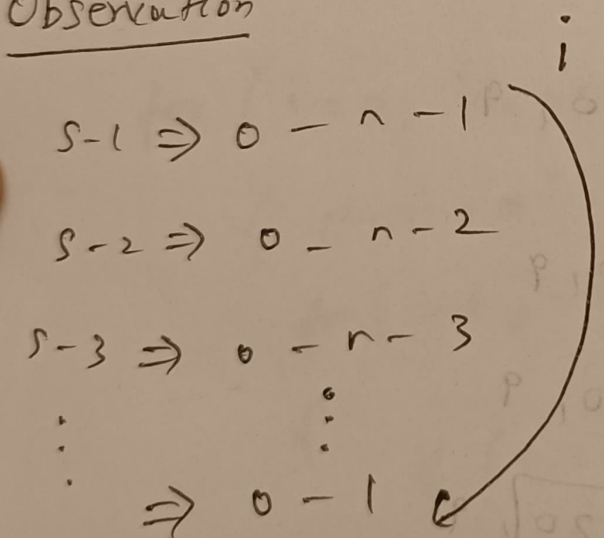
13, [20, 9], [24, 46, 52]

S-4 [13, 9], 20, 24, 46, 52

S-5 9, [13, 20, 24, 46, 52]

S

Observation




```
for (i = n-1; i>=1; i--)        → Not to sort last element.
                  i-1
{  for (j = 0; j <= i-1; j++)
   {  if (a[j] > a[j+1])
         swap
   }
}
```

T.C ⇒ n n-1 n-2 n-3 ... 2 1

$O(n^2)$ ⇒ Worst Case

If the arr is 2 3 5 15 20 ⇒ Best Case

T.C ⇒ O(N) ⇒ Best Case

Insertion Sort

Take an element and place it in the correct position.

14 9 15 12 6 8 13

↳ 9 14 15 12 6 8 13

↳ 9 12 14 15 6 8 13

↳ 6 9 12 14 15 8 13

... So on.

6 8 9 12 13 14 15

* Everyone right shifts by one.
* Swap happen.

Worst ⇒ $O(n^2)$

Best Case ⇒ $O(n)$

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