(1) Bubble sort

A → | 8 | 5 | 7 | 3 | 2 |

Ist Pass ⇒

8	5	5	5	5
5	8	7	7	7
7	7	8	3	3
3	3	3	8	2
2	2	2	2	(8)

IInd pass ⇒

5	5	5	5
7	7	3	3
3	3	7	2
2	2	2	(7)
8	8	8	(8)

~~One pass — Highest.~~

One pass — Greatest no. sorted
2 passes — 2 greatest no.'s sorted
⋮
n passes — n numbers sorted.

n elements — (n−1) passes

n elements

(n-1) passes

$\frac{n(n-1)}{2}$ comparisions

$\rightarrow O(n^2)$

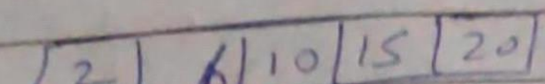
```
Void Bubble Sort (int A[], int n)
{
  for (i=0; i<n-1; i++)
  {
    for (j=0; j<n-1; j++)
    {
      if (A[j] > A[j+1])
      {
        Swap (A[j], A[j+1]);
      }
    }
  }
}
```

Bubble Sort time

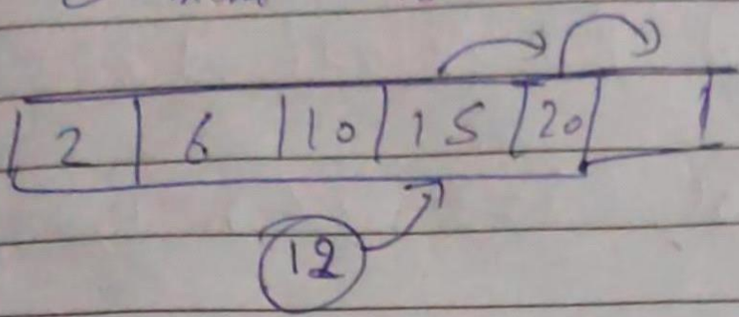
Min - $O(n)$

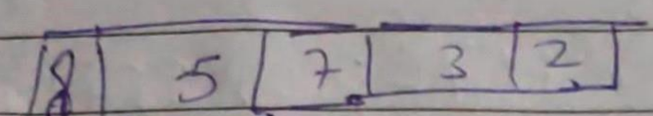
Max. - $O(n^2)$

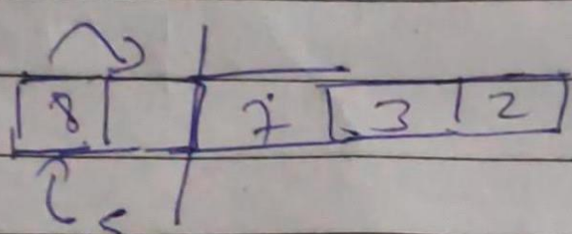
Insertion sort

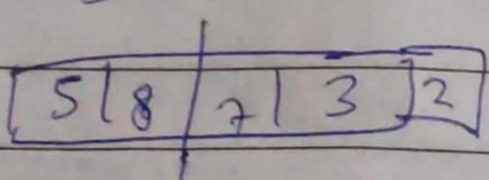
A ⇒ 

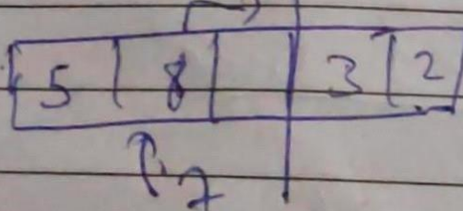
element = 12

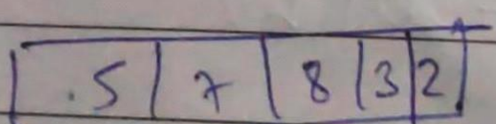


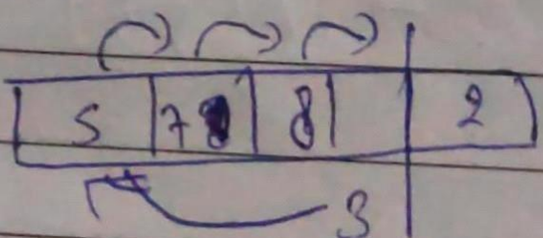
A ⇒ 

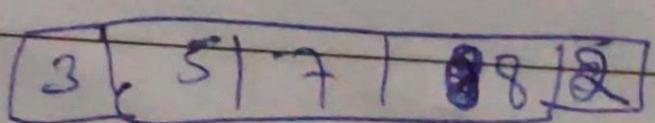
1st pass ⇒ 



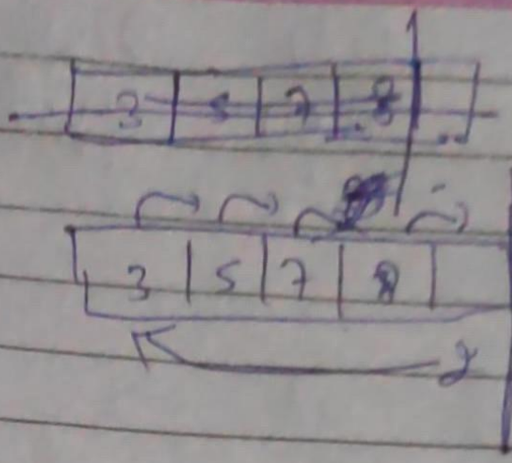
IInd pass ⇒ 



IIIrd pass ⇒ 



4th pass



No. of passes ⇒ n - 1

∴ Comparisions = $\frac{n(n-1)}{2} \in O(n^2)$

```
Void insertion Sort(int A[], int n)
{
    for (i=1; i<n; i++)
    {
        j = i-1
        x = A[i];
        while (A[j] > x && j > -1)
        {
            A[j+1] = A[j]
            j--;
        }
        A[j+1] = x;
    }
}
```

	Min	Max
Time	$O(n)$	$O(n^2)$
Space	$O(1)$	$O(n^2)$

Quick Sort

Pivot

(50) 70 60 90 40 80 10 20 30
i ——— j

(50) 30 60 90 40 80 10 20 70
i ——— j

(50) 30 20 90 40 80 10 60 70
i ——— j

(50) 30 20 10 40 80 90 60 70
j — i

(40 30 20 10) (50) (80 90 60 70)
↓
Partition

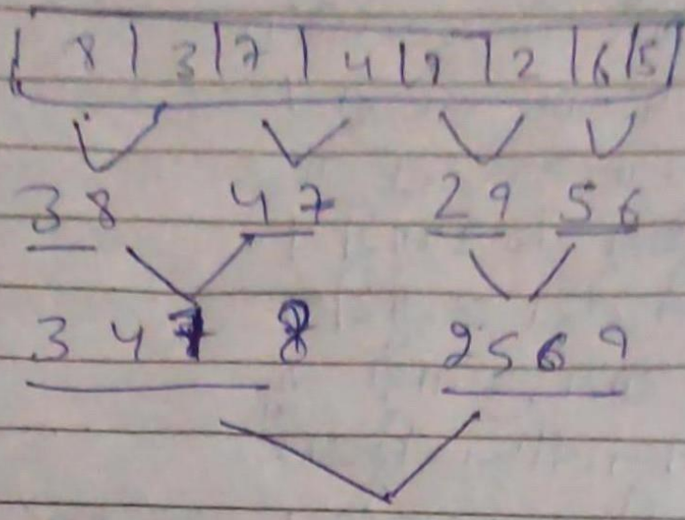
```
int Partition (int A[], int l, int r)
{
    int pivot = A[l];
    int i = l, j = r;
    do
    {
        do { i++; } while (A[i] <= pivot);
        do { j--; } while (A[j] > pivot);
        if (i < j)
            swap (A[i], A[j]);
    }
    while (i < j);
    swap (A[l], A[j]);
    return j;
}
```

Best case → Time $O(n \log n)$

Worst case → time $O(n^2)$

Merge Sort

A [8|3|7|4|9|2|6|5]
0 1 2 3 4 5 6 7




```
Void Merge Sort (int A[], int n)
{
    int p, i, l, mid, h;
    for (p=2; p<n; p=p*2)
    {
        for (i=0; i+p-1<n); i = i+p)
        {
            l = i;
            h = i+p-1;
            mid = (l+h)/2;
            Merge (A, l, mid, h);
        }
        if((p/2)<n)
            merge (A, 0, p/2, n-1)
    }
```

```
if (A[i] < b[j])
    c[k++] = A[i++];

Void merge (int A[], int l, int mid, int h)
{   int i = l, j = mid+1, k = l;
    int b[100];
    while (i <= mid && j <= h)
    {   if (A[i] < A[j])
            b[k++] = A[i++];
        else
            B[k++] = A[j++];
    }
    for (; i <= mid; i++)
        B[k++] = A[i];
    for (; j <= h; j++)
        B[k++] = A[j];
    for (i = l; i <= h; i++)
        A[i] = B[i];
}
```