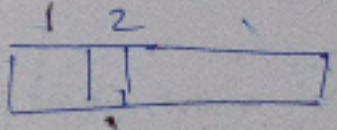
□ Insertion Sort ⇒ (Function)

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



pseudocode ⇒

```
Insertion-SORT(A)
  for j = 2 to A.length
      key = A[j]
  //Insert A[j] into the sorted sequence
                          A[1.....j-1]
      i = j-1
  {while(i>0 and A[i] > key)
          A[i+1] = A[i];
          i = i-1
  A[i+1] = key
```

Best-case Complexity — $O(n)$

It occurs when the array is already sorted.

Average-case Complexity — $O(n^2)$

It occurs when the array elements are in jumbled order that is not properly ascending and properly descending.

Worst-case Complexity — $O(n^2)$

It occurs when array elements are required to be sorted in reverse order.
That means suppose you have to sort the array elements in ascending order, but its elements are in descending order.

www.javatpoint.com/insertion-sort.

n = 6

12	31	25	8	32	17
0	1	2	3	4	5

i = 1

temp = a[1] = 31

j = 1 - 1 = 0

while (0 >= 0 && 31 <= a[j] a[j])

{

False

X

}

a[0+1] = temp = 31

12	31	25	8	32	17

i = 2

temp = a[2] = 25

j = 2 - 1 = 1

while (1 >= 0 && 25 <= 31)

{

a[2] = a[1]

j = 1 - 1 = 0

}

12	31	31	8	32	17
0	1	2	3	4	5

0 >= 0 & 25 <= 12

X

False

a[0+1] = 25

12	25	31	8	32	17
0	1	2	3	4	5

i = 3

temp = a[3] = 8

j = 3 - 1 = 2

while (2 >= 0 && 8 <= a[2] = 31)

{

a[2+1] = a[2]

12	25	31	31	32	17
0	1	2	3	4	5

j = j - 1 = 2 - 1 = 1

}

while (1 >= 0 & 8 <= a[1] = 25

a[2] = a[1]

12	25	25	31	32	17

j = 1 - 1 = 0

0 >= 0 & 8 <= 12

a[1] = a[0]

12	12	25	31	32	17

j = 0 - 1 = -1

a[0] = 8

i = 4

temp = a[4] = 32

j = 4 - 1 = 3

while (3 >= 0 && 32 <= a[3] = 31)

{

false

}

X

a[4] = 32

8	12	25	31	32	17
0	1	2	3	4	5

i = 5

temp = a[5] = 17

j = 5 - 1 = 4

while (4 >= 0 && 17 <= 32)

{

a[4+1] = a[4]

8	12	25	31	32	32
0	1	2	3	4	5

j = 4 = 4 - 1 = 3

}

while (3 >= 0 && 17 <= 31)

{

a[3+1] = a[3]

8	12	25	31	31	32
0	1	2	3	4	5

j = 3 - 1 = 2

}

while (2 >= 0 && 17 <= 25)

{

a[2+1] = a[2]

8	12	25	25	31	32
0	1	2	3	4	5

j = 2 - 1 = 1

}

while (1 >= 0 && 17 <= 12)

{

false

}

a[1+1] = 17

8	12	17	25	31	32
0	1	2	3	4	5

→

8	12	25	31	32	17