



Bubble sort.

```

for (int i=0; i < n-1; i++)
{
    int flag=0;
    for (int j=0; j < n-i-1; j++)
    {
        if (arr[j] > arr[j+1])
        {
            int temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
            flag=1;
        }
        if (flag==0) break;
    }
}

```

size of the array

means that the list is sorted and no need to traverse it again and again.

no of rounds = total no of elements - 1

with every round we need not compare the last element because with every round largest no gets placed at the end

36 19 29 12 5

⇒ Round 1 ⇒ i=0

j=0 36 19 29 12 5
 j=1 19 36 29 12 5
 j=2 19 29 36 12 5
 j=3 19 29 12 36 5

19 29 12 5 36

⇒ Round 3 ⇒ i=2

j=0 19 12 5 29 36
 j=1 12 19 5 29 36

12 5 19 29 36

19 29 12 5 36

⇒ Round 2 ⇒ i=1

j=0 19 29 12 5 36
 j=1 19 29 12 5 36
 j=2 19 12 29 5 36

19 12 5 29 36

⇒ Round 4 ⇒ i=3

j=0 12 5 19 29 36
 j=1 5 12 19 29 36

Selection Sort

Combination of searching and sorting.

finds the smallest element in the array and puts it in the beginning

In every iteration of selection sort, the minimum element is picked from the unsorted array and moved to the sorted subarray.

38	52	9	18	6	62	13
----	----	---	----	---	----	----

→ idle sabse chota element kounsa?
(always 1 the smallest element)

6 | 52 9 18 38 62 13

6 9 | 52 18 38 62 13

6 9 13 | 18 38 62 52

6 9 13 18 | 38 62 52 → unsorted list
sorted list

6 9 13 18 38 | 62 52

6 9 13 18 38 52 62

```
int min;  
for (int i = 0; i < n; i++)  
{  
    min = i;  
    for (int j = i + 1; j < n; j++)  
    {  
        if (a[j] < a[min])  
        {  
            min = j;  
        }  
    }  
    int temp = arr[i];  
    arr[i] = arr[min];  
    arr[min] = temp;  
}
```

searches for smaller element than at ith index.

Insertion Sort

It works the way we sort playing cards in our hand. We choose one card and insert it in its position (ascending or descending)

5	1	6	2	4	3
---	---	---	---	---	---

5	1	6	2	4	3
---	---	---	---	---	---

temp

1	5	6	2	4	3
---	---	---	---	---	---

temp

1	5	6	2	4	3
---	---	---	---	---	---

temp

1	2	5	6	4	3
---	---	---	---	---	---

temp

1	2	4	5	6	3
---	---	---	---	---	---

temp

1	2	3	4	5	6
---	---	---	---	---	---

0	1	2	3	4	5
5	1	6	2	4	3

temp

Working Of Insertion Sort

1. We start by making the second element of the given array, i.e. element at index 1, the key. The key element here is the new card that we need to add to our existing sorted set of cards (as explained cards example before).
2. We compare the key element with the element(s) before it, in this case, element at index 0:
 - 2.1 If the key element is less than the first element, we insert the key element before the first element.
 - 2.2 If the key element is greater than the first element, then we insert it after the first element.
3. Then, we make the third element of the array as key and will compare it with elements to its left and insert it at the right position.
4. And we go on repeating this, until the array is sorted.

```
ints temp, j;
for (ints i = 1; i < n; i++)
{
    temp = arr[i]; // detect the temp variable
    j = i;
    while (j > 0 & arr[j-1] > temp)
    {
        // performs swapping till
        // temp ac chota element
        // namil jaye.
        arr[j] = arr[j-1];
        j = j-1; // j--;
    }
    arr[j] = temp;
}
```