Selection sort breaks the input list in two parts: the sorted part which initially is empty, and the unsorted part which initially contains the list of all elements. The algorithm then selects the minimum value of the whole unsorted file, swaps it with the first unsorted value, and increases the sorted part by one.

The time complexity of selection sort is O(n^2) in all cases.

The selection sort is best used when you want to:

- Sort a small list of items in ascending order;
- The cost of swapping values is insignificant;
- To make sure that all the values in the list have been checked.

Selection sort performs very well on small lists(already sorted or nearly sorted items) and does not require a lot of space for its operations.

However, the number of iterations made during the sorting is n-squared, where n is the total number of elements in the list, so it will perform poorly when working on bigger lists.

In [25]:

```
def selection_sort(array):
    count = 0
    for i in range(len(array)-1): # A)
        print(array)
        minimum = array[i] # B)
        minimum_index = i
        for j in range(i+1,len(array)): # C)
            count += 1
            if array[j] < minimum:
# D)
                minimum = array[j]
                minimum_index = j
        if minimum_index != i:
# E)
            array[minimum_index], array[i] = array[i], array[minimum_index]
    return (f'\nSorted array:{array} \nNumber of comparisons = {count}\n')
```

In [26]:

```
data = [-2, 45, 0, 11, -9]
print(selection_sort(data))
```

```
[-2, 45, 0, 11, -9]
[-9, 45, 0, 11, -2]
[-9, -2, 0, 11, 45]
[-9, -2, 0, 11, 45]

Sorted array:[-9, -2, 0, 11, 45]
Number of comparisons = 10
```

Here is a brief explanation of the implementation above:

A) It is -1, because when only 1 item remains, the array will be sorted already;

B) On the first loop, we set the minimum element to be the ith element, and the minimum index to be the ith index;

C) Then, we check the array from the i+1th element to the end of its length;

D) If it finds an item that is smaller than the minimum element, it reassign the minimum element and the minimum index;

E) If minimum index changes, then it swaps that item with the ith element.