Insertion Sort

Insertion Sort builds the sorted array **one element at a time**, just like sorting playing cards in your hand. You take one card (element) from the unsorted pile and **insert** it into its correct position among the already sorted ones.

Basically, we “pick” and element and assume that now its place is empty. We then check the array for a place where the element fits, and insert it there, shifting the elements, hence the name.

🔹 Algorithmic Steps (in pseudocode / algorithm format)

Algorithm: InsertionSort(A, n)

Input: Array A of n elements

Output: Sorted array A in ascending order

```
1. for i ← 1 to n-1 do
2.     key ← A[i]
3.     j ← i - 1
4.     while j ≥ 0 and A[j] > key do
5.         A[j + 1] ← A[j]
6.         j ← j - 1
7.     end while
8.     A[j + 1] ← key
9. end for
10. return A
```

Implementation in C++

```cpp
#include <iostream>
using namespace std;

void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];     // current element to insert
        int j = i - 1;

        // Move elements of arr[0..i-1] that are greater than key
        // to one position ahead of their current position
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

int main() {
    int arr[] = {12, 11, 13, 5, 6};
    int n = sizeof(arr) / sizeof(arr[0]);

    insertionSort(arr, n);
```

```
    cout << "Sorted array: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;

    return 0;
}
```

Time Complexity

Case	Comparisons	Shifts	Time
Best	n-1	0	$O(n)$
Average	$\sim n^2/4$	$\sim n^2/4$	$O(n^2)$
Worst	$n^2/2$	n-1	$O(n^2)$

Space complexity: O(1); in-place sorting

Stable? ✅ Yes, equal elements stay in order

Adaptive? ✅ Yes, fewer shifts if array is nearly sorted