ADA LAB **Kunal Sachdeva**

07-04-2021 **19115045**

4th Sem CSE

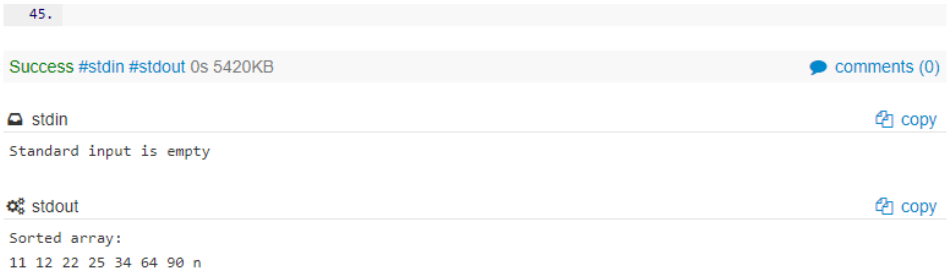
BUBBLE SORT

```
#include <bits/stdc++.h>
using namespace std;
void swap(int *xp, int *yp)
{
	int temp = *xp;
	*xp = *yp;
	*yp = temp;
}
void bubbleSort(int arr[], int n)
{
	int i, j;
	bool swapped;
	for (i = 0; i < n-1; i++)
	{
		swapped = false;
		for (j = 0; j < n-i-1; j++)
		{
			if (arr[j] > arr[j+1])
			{
			swap(&arr[j], &arr[j+1]);
			swapped = true;
			}
		}
		// IF no two elements were swapped by inner loop, then break
		if (swapped == false)
			break;
	}
}
```

```
void printArray(int arr[], int size)
{
	int i;
	for (i=0; i < size; i++)
		printf("%d ", arr[i]);
	printf("n");
}
int main()
{
	int arr[] = {64, 34, 25, 12, 22, 11, 90};
	int n = sizeof(arr)/sizeof(arr[0]);
	bubbleSort(arr, n);
	printf("Sorted array: \n");
	printArray(arr, n);
	return 0;
}
```

edit fork download copy

```
#include <bits/stdc++.h>
using namespace std;
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}
void bubbleSort(int arr[], int n)
{
    int i, j;
    bool swapped;
    for (i = 0; i < n-1; i++)
    {
        swapped = false;
        for (j = 0; j < n-i-1; j++)
        {
            if (arr[j] > arr[j+1])
            {
            swap(&arr[j], &arr[j+1]);
            swapped = true;
            }
        }
        // IF no two elements were swapped by inner loop, then break
        if (swapped == false)
            break;
    }
}
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("n");
}
int main()
{
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    bubbleSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```



Time Complexity:

Worst and Average Case Time Complexity: O(n*n). Worst case occurs when array is reverse sorted.

Best Case Time Complexity: O(n). Best case occurs when array is already sorted.

INSERTION SORT

```
#include <bits/stdc++.h>
using namespace std;
void insertionSort(int arr[], int n)
{
	int i, key, j;
	for (i = 1; i < n; i++)
	{
		key = arr[i];
		j = i - 1;
		while (j >= 0 && arr[j] > key)
		{
			arr[j + 1] = arr[j];
			j = j - 1;
		}
		arr[j + 1] = key;
	}
}
void printArray(int arr[], int n)
{
	int i;
	for (i = 0; i < n; i++)
		cout << arr[i] << " ";
	cout << endl;
}
int main()
```

{

int arr[] = { 12, 11, 13, 5, 6 };

int n = sizeof(arr) / sizeof(arr[0]);

insertionSort(arr, n);

printArray(arr, n);

return 0;

}

Time Complexity: O(n*2)

edit fork download copy

```cpp
#include <bits/stdc++.h>
using namespace std;
void insertionSort(int arr[], int n)
{
    int i, key, j;
    for (i = 1; i < n; i++)
    {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key)
        {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
void printArray(int arr[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}
int main()
{
    int arr[] = { 12, 11, 13, 5, 6 };
    int n = sizeof(arr) / sizeof(arr[0]);
    insertionSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

Success #stdin #stdout 0s 5460KB comments (0)

stdin copy

```
Standard input is empty
```

stdout copy

```
5 6 11 12 13
```