ADA LAB **Kunal Sachdeva**

06-04-2021 **19115045**

4th semester CSE

QUICK SORT

```
#include <bits/stdc++.h>
using namespace std;
void swap(int* a, int* b)
{
	int t = *a;
	*a = *b;
	*b = t;
}
int partition (int arr[], int low, int high)
{
	int pivot = arr[high];
	int i = (low - 1);
	for (int j = low; j <= high - 1; j++)
	{
		if (arr[j] < pivot)
		{
			i++;
			swap(&arr[i], &arr[j]);
		}
	}
	swap(&arr[i + 1], &arr[high]);
	return (i + 1);
}
void quickSort(int arr[], int low, int high)
{
	if (low < high)
	{
```

```
                int pi = partition(arr, low, high);
                quickSort(arr, low, pi - 1);
                quickSort(arr, pi + 1, high);
        }
}
void printArray(int arr[], int size)
{
        int i;
        for (i = 0; i < size; i++)
                cout << arr[i] << " ";
        cout << endl;
}
int main()
{
        int arr[] = {15,27,1,3,49,10};
        int n = sizeof(arr) / sizeof(arr[0]);
        quickSort(arr, 0, n - 1);
        cout << "Sorted array: \n";
        printArray(arr, n);
        return 0;
}
```

Time Complexity:

Mathematical analysis of quicksort shows that, on average, the algorithm takes O(n log n) comparisons to sort n items. In the worst case, it makes O(n2) comparisons, though this behaviour is rare.

```cpp
#include <bits/stdc++.h>
using namespace std;
void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}
int partition (int arr[], int low, int high)
{
    int pivot = arr[high];
    int i = (low - 1);
    for (int j = low; j <= high - 1; j++)
    {
        if (arr[j] < pivot)
        {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}
void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}
int main()
{
    int arr[] = {15,27,1,3,49,10};
    int n = sizeof(arr) / sizeof(arr[0]);
    quickSort(arr, 0, n - 1);
    cout << "Sorted array: \n";
    printArray(arr, n);
    return 0;
}
```

Success #stdin #stdout 0.01s 5504KB

comments (0)

stdin

copy

```
Standard input is empty
```

stdout

copy

```
Sorted array:
1 3 10 15 27 49
```