

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 $O(n^2)$ comparisons, though this behaviour is rare.

```

1. #include <bits/stdc++.h>
2. using namespace std;
3. void swap(int* a, int* b)
4. {
5.     int t = *a;
6.     *a = *b;
7.     *b = t;
8. }
9. int partition (int arr[], int low, int high)
10. {
11.     int pivot = arr[high];
12.     int i = (low - 1);
13.     for (int j = low; j <= high - 1; j++)
14.     {
15.         if (arr[j] < pivot)
16.         {
17.             i++;
18.             swap(&arr[i], &arr[j]);
19.         }
20.     }
21.     swap(&arr[i + 1], &arr[high]);
22.     return (i + 1);
23. }
24. void quickSort(int arr[], int low, int high)
25. {
26.     if (low < high)
27.     {
28.         int pi = partition(arr, low, high);
29.         quickSort(arr, low, pi - 1);
30.         quickSort(arr, pi + 1, high);
31.     }
32. }
33. void printArray(int arr[], int size)
34. {
35.     int i;
36.     for (i = 0; i < size; i++)
37.         cout << arr[i] << " ";
38.     cout << endl;
39. }
40. int main()
41. {
42.     int arr[] = {15,27,1,3,49,10};
43.     int n = sizeof(arr) / sizeof(arr[0]);
44.     quickSort(arr, 0, n - 1);
45.     cout << "Sorted array: \n";
46.     printArray(arr, n);
47.     return 0;
48. }

```

Success #stdin #stdout 0.01s 5504KB

 comments (0)

 stdin

 copy

Standard input is empty

 stdout

 copy

Sorted array:

1 3 10 15 27 49