Quicksort 2 - Sorting

by HackerRank

Problem Submissions Leaderboard Discussions

In the previous challenge, you wrote a *partition* method to split an array into two sub-arrays, one containing smaller elements and one containing larger elements than a given number. This means you 'sorted' half the array with respect to the other half. Can you repeatedly use *partition* to sort an entire array?

Guideline

In Insertion Sort, you simply went through each element in order and inserted it into a sorted sub-array. In this challenge, you cannot focus on one element at a time, but instead must deal with whole sub-arrays, with a strategy known as "divide and conquer".

When *partition* is called on an array, two parts of the array get 'sorted' with respect to each other. If *partition* is then called on each sub-array, the array will now be split into four parts. This process can be repeated until the sub-arrays are small. Notice that when partition is called on just one of the numbers, they end up being sorted.

Can you repeatedly call partition so that the entire array ends up sorted?

Print Sub-Arrays

In this challenge, print your array every time your partitioning method finishes, i.e. whenever two subarrays, along with the pivot, are merged together.

- The first element in a sub-array should be used as a pivot.
- Partition the left side before partitioning the right side.
- The pivot should be placed between sub-arrays while merging them.
- Array of length 1 or less will be considered sorted, and there is no need to sort or to print them.

Note

Please maintain the original order of the elements in the left and right partitions while partitioning around a pivot element.

For example: Partition about the first element for the array A[]={5, 8, 1, 3, 7, 9, 2} will be {1, 3, 2, 5, 8, 7, 9}

Input Format

There will be two lines of input:

- n - the size of the array
- ar - the n numbers of the array

Output Format

Print every partitioned sub-array on a new line.

Constraints

$1 \leq n \leq 1000$
$-1000 \leq x \leq 1000, x \in ar$
Each number is unique.

Sample Input

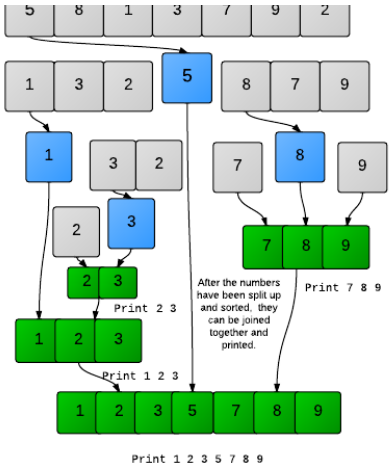
```
7
5 8 1 3 7 9 2
```

Sample Output

```
2 3
1 2 3
7 8 9
1 2 3 5 7 8 9
```

Explanation

This is a diagram of Quicksort operating on the sample array:



Task

The method **quickSort** takes in a parameter, ***ar***, an unsorted array. Use the Quicksort algorithm to sort the entire array.

Submissions: 19959
Max Score: 30
Difficulty: Easy

Rate This Challenge:

More

Current Buffer (saved locally, editable) C++

```cpp
#include <bits/stdc++.h>
using namespace std;

void quickSort(vector <int> &arr) {
    if (arr.size() <= 1) {
        return;
    }

    int pivot = arr[0];

    vector <int> left, right;

    for (int i = 0; i < arr.size(); i++) {
        if (arr[i] < pivot) {
            left.push_back(arr[i]);
        } else if (arr[i] > pivot) {
            right.push_back(arr[i]);
        }
    }

    quickSort(left);
    quickSort(right);

    int i;
    // merge arrays into arr (not actually merging but overwriting since we're dealing with &arr)
    for (i = 0; i < left.size(); i++) arr[i] = left[i];
    arr[i++] = pivot;
    for (int j = 0; j < right.size(); j++) arr[i++] = right[j];

    // print the now "merged" array
    for (auto i : arr) cout << i << " ";
    cout << "\n";
}

int main()
{
    int n;
    cin >> n;

    vector <int> arr(n);
    for(int i = 0; i < (int)n; ++i) {
        cin >> arr[i];
    }

    quickSort(arr);

    return 0;
}

```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code