Your Quicksort 1 - Partition submission got 10.00 points. Share Tweet

Try the Next Challenge | Try a Random Challenge

Quicksort 1 - Partition

by HackerRank

Problem Submissions Leaderboard Discussions

The previous challenges covered Insertion Sort, which is a simple and intuitive sorting algorithm with a running time of $O(n^2)$. In these next few challenges, we're covering a *divide-and-conquer* algorithm called Quicksort (also known as *Partition Sort*). This challenge is a modified version of the algorithm that only addresses partitioning. It is implemented as follows:

Step 1: Divide
Choose some pivot element, p, and partition your unsorted array, arr, into three smaller arrays: $left$, $right$, and $equal$, where each element in $left < p$, each element in $right > p$, and each element in $equal = p$.

For example: Assume $arr = [5, 7, 4, 3, 8]$
The pivot is at $arr_0 = 5$
arr is divided into $left = \{4, 3\}$, $equal = \{5\}$, and $right = \{7, 8\}$.
Putting them all together, you get $\{4, 3, 5, 7, 8\}$. Another valid solution is $\{3, 4, 5, 8, 7\}$.

Given arr and $p = ar[0]$, partition arr into $left$, $right$, and $equal$ using the *Divide* instructions above. Then print each element in $left$ followed by each element in $equal$, followed by each element in $right$ on a single line. Your output should be space-separated and does not have to maintain ordering of the elements within the three categories.

Input Format

The first line contains n, the size of the array arr.
The second line contains n space-separated integers describing arr (the unsorted array). The first integer (corresponding to $ar[0]$) is your pivot element, p.

Constraints

- $1 \leq n \leq 1000$
- $-1000 \leq a_i \leq 1000, a_i \in ar, 0 \leq i < n$
- All elements will be unique.

Output Format

On a single line, print the partitioned numbers (i.e.: the elements in $left$, then the elements in $equal$, and then the elements in $right$). Each integer should be separated by a single space.

Sample Input

```
5
4 5 3 7 2
```

Sample Output

```
3 2 4 5 7
```

Explanation

$ar = [4, 5, 3, 7, 2]$
Pivot: $p = ar[0] = 4$.
$left = \{\}$; $equal = \{4\}$; $right = \{\}$

$ar[1] = 5 > p$, so it's added to $right$.
$left = \{\}$; $equal = \{4\}$; $right = \{5\}$

$ar[2] = 3 < p$, so it's added to $left$.
$left = \{3\}$; $equal = \{4\}$; $right = \{5\}$

$ar[3] = 7 > p$, so it's added to $right$.
$left = \{3\}$; $equal = \{4\}$; $right = \{5, 7\}$

$ar[4] = 2 < p$, so it's added to $left$.
$left = \{3, 2\}$; $equal = \{4\}$; $right = \{5, 7\}$

We then print the elements of $left$, followed by $equal$, followed by $right$, we get: `3 2 4 5 7`.

You don't need to maintain ordering, so another valid solution would be `2 3 4 5 7`.

Easy Submitted 54180 times
Max Score 10

Need Help?

View Discussions

View Top Submissions

RATE THIS CHALLENGE

Download problem statement

Download sample test cases

Suggest Edits

Current Buffer (saved locally, editable) C++14

```
#include <bits/stdc++.h>

using namespace std;


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

    vector<int> equals;
    vector<int> left;
    vector<int> right;

    equals.push_back(arr[0]);

    for (i = 1; i < n; i++) {
        if (arr[i] > arr[0]) right.push_back(arr[i]);
        else if (arr[i] == arr[0]) equals.push_back(arr[i]);
        else left.push_back(arr[i]);
```

```
    }
    for(i=0;i<left.size();i++) cout << left[i] << " ";
    for(i=0;i<equals.size();i++) cout << equals[i] << " ";
    for(i=0;i<right.size();i++) cout << right[i] << " ";
    cout << endl;
    return 0;
}
```

Line: 1 Col: 1

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