Practice Compete Jobs Rank Leaderboard bor0s

Dashboard > Algorithms > Sorting > Quicksort 1 - Partition

Badge Progress ★ ★

Points: 651.00 Rank: 47018

Quicksort 1 - Partition

by HackerRank

Problem | Submissions | Leaderboard | Discussions

The previous challenges covered Insertion Sort, which is a simple and intuitive sorting algorithm with an average case performance of $O(n^2)$. In these next few challenges, we're covering a *divide-and-conquer* algorithm called Quicksort (also known as *Partition Sort*).

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

Challenge
Given ar and $p = ar[0]$, partition ar 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.

Note: There is no need to sort the elements in-place; you can create two lists and stitch them together at the end.

Input Format

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

Constraints

- $1 \leq n \leq 1000$
- $-1000 \leq x \leq 1000, x \in ar$
- All elements will be unique.
- Multiple answer can exists for the given test case. Print any one of them.

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: $p = ar[0] = 4$.
$left = \{\}$; $equal = \{4\}$; $right = \{\}$

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

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

$ar[3] = 7 \geq p$, 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`.

This example is only one correct answer based on the implementation shown, but it is not the only correct answer (e.g.: another valid solution would be `2 3 4 5 7`).

Solved score: 10.00pts

Submissions: 38412
Max Score: 10
Difficulty: Easy

Rate This Challenge:
☆☆☆☆☆

More

Current Buffer (saved locally, editable)

C++

```
#include <map>
#include <set>
#include <list>
#include <cmath>
#include <ctime>
#include <deque>
#include <queue>
#include <stack>
#include <bitset>
#include <cstdio>
#include <vector>
#include <cstdlib>
#include <numeric>
#include <sstream>
#include <iostream>
#include <algorithm>
using namespace std;
void partition(vector <int>  ar) {
    int pivot = ar[0];
    vector <int> left, right;

    for (int i = 0; i < ar.size(); i++) {
        if (ar[i] < pivot) {
            left.push_back(ar[i]);
        } else if (ar[i] > pivot) {
            right.push_back(ar[i]);
        }
    }
    for (auto i : left) cout << i << " ";
    cout << pivot;
    for (auto i : right) cout << " " << i;
   // Enter code for partitioning and printing here.
}
int main(void) {
   vector <int>  _ar;
   int _ar_size;
   cin >> _ar_size;

     for(int _ar_i=0; _ar_i<_ar_size; _ar_i++) {
        int _ar_tmp;
        cin >> _ar_tmp;
        _ar.push_back(_ar_tmp);
    }

   partition(_ar);

   return 0;
}

```

Line: 1 Col: 1

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

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature