Quicksort 1 - Partition

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 $equal = p$.

Example
$arr = [5, 7, 4, 3, 8]$

In this challenge, the pivot will always be at $arr[0]$, so the pivot is 5.

arr is divided into $left = \{4, 3\}$, $equal = \{5\}$, and $right = \{7, 8\}$.
Putting them all together, you get $\{4, 3, 5, 7, 8\}$. There is a flexible checker that allows the elements of $left$ and $right$ to be in any order. For example, $\{3, 4, 5, 8, 7\}$ is valid as well.

Given arr and $p = arr[0]$, partition arr into $left$, $right$, and $equal$ using the *Divide* instructions above. Return a 1-dimensional array containing each element in $left$ first, followed by each element in $equal$, followed by each element in $right$.

Function Description

Complete the *quickSort* function in the editor below.

quickSort has the following parameter(s):

- *int arr[n]:* $arr[0]$ is the pivot element

Returns

- *int[n]:* an array of integers as described above

Input Format

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

Constraints

- $1 \leq n \leq 1000$
- $-1000 \leq arr[i] \leq 1000$ where $0 \leq i < n$
- All elements are distinct.

Sample Input

```
STDIN       Function
-----       --------
5           arr[] size n =5
4 5 3 7 2   arr =[4, 5, 3, 7, 2]
```

Sample Output

```
3 2 4 5 7
```

Explanation

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

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

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

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

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

Return the array $\{32457\}$.

The order of the elements to the left and right of 4 does not need to match this answer. It is only required that 3 and 2 are to the left of 4, and 5 and 7 are to the right.

The contest has not yet started. It begins in an hour.

Submissions: 0
Max Score: 0
Difficulty: Easy

Rate This Challenge:
☆☆☆☆☆

More

C++

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

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);

/*
 * Complete the 'quickSort' function below.
 *
 * The function is expected to return an INTEGER_ARRAY.
 * The function accepts INTEGER_ARRAY arr as parameter.
 */

vector<int> quickSort(vector<int> arr) {

}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    string n_temp;
    getline(cin, n_temp);

    int n = stoi(ltrim(rtrim(n_temp)));

    string arr_temp_temp;
    getline(cin, arr_temp_temp);

    vector<string> arr_temp = split(rtrim(arr_temp_temp));

    vector<int> arr(n);

    for (int i = 0; i < n; i++) {
        int arr_item = stoi(arr_temp[i]);

        arr[i] = arr_item;
    }

    vector<int> result = quickSort(arr);

    for (size_t i = 0; i < result.size(); i++) {
        fout << result[i];

        if (i != result.size() - 1) {
            fout << " ";
        }
    }

    fout << "\n";

    fout.close();

    return 0;
}

string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
        s.end()
    );

    return s;
}

vector<string> split(const string &str) {
    vector<string> tokens;

    string::size_type start = 0;
    string::size_type end = 0;

    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push_back(str.substr(start, end - start));

        start = end + 1;
    }

    tokens.push_back(str.substr(start));

    return tokens;
}

```

Line: 1 Col: 1

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

Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy |