Big Sorting

by _mfv_

Consider an array of numeric strings, $unsorted$, where each string is a positive number with anywhere from 1 to 10^6 digits. Sort the array's elements in *non-decreasing* (i.e., ascending) order of their real-world integer values and print each element of the sorted array on a new line.

Input Format

The first line contains an integer, n, denoting the number of strings in $unsorted$.
Each of the n subsequent lines contains a string of integers describing an element of the array.

Constraints

- $1 \leq n \leq 2 \times 10^5$
- Each string is guaranteed to represent a positive integer without leading zeros.
- The total number of digits across all strings in $unsorted$ is between 1 and 10^6 (inclusive).

Output Format

Print each element of the sorted array on a new line.

Sample Input 0

```
6
31415926535897932384626433832795
1
3
10
3
5
```

Sample Output 0

```
1
3
3
5
10
31415926535897932384626433832795
```

Explanation 0

The initial array of strings is $unsorted = [31415926535897932384626433832795, 1, 3, 10, 3, 5]$. When we order each string by the real-world integer value it represents, we get:

$$1 \leq 3 \leq 3 \leq 5 \leq 10 \leq 31415926535897932384626433832795$$

We then print each value on a new line, from smallest to largest.

Submissions: 463
Max Score: 20
Difficulty: Easy

Rate This Challenge:

More

Current Buffer (saved locally, editable) C#

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
class Solution {

     static int Comparar(string a, string b)
        {
            if (a.Length > b.Length) return 1;
            else if (b.Length > a.Length) return -1;

            for (int i = 0; i < a.Length; i++)
            {
                if (a[i] > b[i])
                {
                    return 1;
                }
                else if (b[i] > a[i])
                {
                    return -1;
                }
            }
            return 0;
        }


        static int partition(string[] arr, int low, int high)
        {
            string pivot = arr[high];
            string temp;
            int i = (low - 1); // index of smaller element
            for (int j = low; j <= high - 1; j++)
            {
                // If current element is smaller than or
                // equal to pivot
                if(Comparar(arr[j], pivot)<0) //if (arr[j] <= pivot)
                {
                    i++;

                    // swap arr[i] and arr[j]
                    temp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;
                }
            }

            // swap arr[i+1] and arr[high] (or pivot)
            temp = arr[i + 1];
            arr[i + 1] = arr[high];
            arr[high] = temp;

            return i + 1;
        }


        static void quickSort(string [] array, int start, int end)
        {
            if (start < end)
            {
                int pivotIndex = partition(array, start, end);

```

```
62                quickSort(array, start, pivotIndex - 1);
63                quickSort(array, pivotIndex + 1, end);
64            }
65        }
66
67        static void Main(string[] args)
68        {
69            //int n = int.Parse(Console.ReadLine());
70            //int[] arr = Array.ConvertAll(Console.ReadLine().Split(' '), e => int.Parse(e));
71            //quickSort(arr, 0, arr.Length - 1);
72
73            //string[] s = {
74            //"6",
75            //"31415926535897932384626433832795",
76            //"1",
77            //"3",
78            //"10",
79            //"3",
80            //"5"};
81
82            //quickSort(s, 0, s.Length - 1);
83
84            //foreach (string elem in s)
85            //{
86            //    Console.WriteLine(elem);
87            //}
88
89            int n = int.Parse(Console.ReadLine());
90            string[] s = new string[n];
91            for (int i = 0; i < n; i++)
92            {
93                s[i] = Console.ReadLine();
94            }
95            quickSort(s, 0, n - 1);
96
97            foreach (string elem in s)
98            {
99                Console.WriteLine(elem);
100           }
101
102
103           Console.ReadLine();
104       }
105
106
107   }
108
```

Line: 104 Col: 10

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

Congrats, you solved this challenge!

- ✔ Test Case #0
- ✔ Test Case #1
- ✔ Test Case #2
- ✔ Test Case #3
- ✔ Test Case #4
- ✔ Test Case #5
- ✔ Test Case #6
- ✔ Test Case #7

Next Challenge