Maximum Perimeter Triangle

by zemen

Problem | Submissions | Leaderboard | Discussions | Editorial

Given n sticks of lengths $l_0, l_1, \ldots l_{n-1}$, use 3 of the sticks to construct a non-degenerate triangle with the maximum possible perimeter. Then print the lengths of its sides as 3 space-separated integers in non-decreasing order.

If there are several valid triangles having the maximum perimeter:

1. Choose the one with the *longest maximum side* (i.e., the largest value for the longest side of any valid triangle having the maximum perimeter).
2. If more than one such triangle meets the first criterion, choose the one with the *longest minimum side* (i.e., the largest value for the shortest side of any valid triangle having the maximum perimeter).
3. If more than one such triangle meets the second criterion, print any one of the qualifying triangles.

If no non-degenerate triangle exists, print `-1`.

Input Format

The first line contains single integer, n, denoting the number of sticks.
The second line contains n space-separated integers, $l_0, l_1, \ldots, l_{n-1}$, describing the respective stick lengths.

Constraints

- $3 \leq n \leq 50$
- $1 \leq l_i \leq 10^9$

Output Format

Print 3 *non-decreasing* space-separated integers, a, b, and c (where $a \leq b \leq c$) describing the respective lengths of a triangle meeting the criteria in the above *Problem Statement*.

If no non-degenerate triangle can be constructed, print `-1`.

Sample Input 0

```
5
1 1 1 3 3
```

Sample Output 0

```
1 3 3
```

Sample Input 1

```
3
1 2 3
```

Sample Output 1

```
-1
```

Explanation

Sample Case 0:
There are 2 possible unique triangles:

1. $(1, 1, 1)$
2. $(1, 3, 3)$

The second triangle has the largest perimeter, so we print its side lengths on a new line in non-decreasing order.

Sample Case 1:
The triangle $(1, 2, 3)$ is degenerate and thus can't be constructed, so we print `-1` on a new line.

Current Buffer (saved locally, editable) C#

```
using System;
using System.Collections.Generic;
using System.IO;
class Solution {
    static void Main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution */

          int n = int.Parse(Console.ReadLine());

            int[] lados = Array.ConvertAll(Console.ReadLine().Split(' '), e => int.Parse(e));

            //int[] lados = { 1, 1, 1, 3, 3 };
            //int[] lados = { 6,5,4,3,2,1 };

            Array.Sort(lados);
            Array.Reverse(lados);


            string ans = "-1";

            for (int i = 0; i < lados.Length - 2; i++)
            {
                //Console.WriteLine(lados[i] + " " + lados[i + 1] + " " + lados[i + 2]);
                if (lados[i + 1] + lados[i + 2] > lados[i])
                {
                    ans = "";
                    ans += lados[i+2] + " " + lados[i + 1] + " " + lados[i];
                    break;
                }
            }

            Console.WriteLine(ans);

    }
}
```

Line: 17 Col: 1

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
- ✔ Test Case #8
- ✔ Test Case #9
- ✔ Test Case #10
- ✔ Test Case #11

Next Challenge