Binary Search: Ice Cream Parlor

by dheeraj

Check out the resources on the page's right side to learn more about binary search. The video tutorial is by Gayle Laakmann McDowell, author of the best-selling interview book Cracking the Coding Interview.

Each time Sunny and Johnny take a trip to the Ice Cream Parlor, they pool together m dollars for ice cream. On any given day, the parlor offers a line of n flavors. Each flavor, i, is numbered sequentially with a unique ID number from 1 to n and has a cost, c_i, associated with it.

Given the value of m and the cost of each flavor for t trips to the Ice Cream Parlor, help Sunny and Johnny choose two *distinct* flavors such that they spend their entire pool of money (m) during each visit. For each trip to the parlor, print the ID numbers for the two types of ice cream that Sunny and Johnny purchase as two space-separated integers on a new line. You must print the smaller ID first and the larger ID second.

Note: Two ice creams having unique IDs i and j *may* have the same cost (i.e., $c_i \equiv c_j$).

Input Format

The first line contains an integer, t, denoting the number of trips to the ice cream parlor. The $3t$ subsequent lines describe all of Sunny and Johnny's trips to the parlor; each trip is described as follows:

1. The first line contains m.
2. The second line contains n.
3. The third line contains n space-separated integers denoting the cost of each respective flavor. The i^{th} integer corresponds to the cost, c_i, for the ice cream with ID number i (where $1 \leq i \leq n$).

Constraints

- $1 \leq t \leq 50$
- $2 \leq m \leq 10^4$
- $2 \leq n \leq 10^4$
- $1 \leq c_i \leq 10^4$, where $i \in [1, n]$
- It is guaranteed that there will always be a unique solution.

Output Format

Print two space-separated integers denoting the respective ID numbers for the two distinct flavors they choose to purchase, where the smaller ID is printed first and the larger ID is printed second. Recall that each ice cream flavor has a unique ID number in the inclusive range from 1 to n.

Sample Input

```
2
4
5
1 4 5 3 2
4
4
2 2 4 3
```

Sample Output

```
1 4
1 2
```

Explanation

Sunny and Johnny make the following two trips to the parlor:

1. The first time, they pool together $m = 4$ dollars. There are five flavors available that day and flavors 1 and 4 have a total cost of $1 + 3 = 4$. Thus, we print `1 4` on a new line.

2. The second time, they pool together $m = 4$ dollars. There are four flavors available that day and flavors 1 and 2 have a total cost of $2 + 2 = 4$. Thus, we print `1 2` on a new line.

Submissions: 3292
Max Score: 35
Difficulty: Medium

Rate This Challenge:
★★★★★

Need Help?

6:22

Binary Search

More

Current Buffer (saved locally, editable)

C#

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


 public class Ice
        {
            public int costo;
            public int indice;

            public Ice() { }

            public Ice(int costo, int indice)
            {
                this.costo = costo;
                this.indice = indice;
            }
        }

        private static void Quicksort(Ice[] vector, int primero, int ultimo)
        {
            int i, j, central;
            Ice pivote;
            central = (primero + ultimo) / 2;
            pivote = vector[central];

            i = primero;
            j = ultimo;
            do
            {
                while (vector[i].costo < pivote.costo) i++;
                while (vector[j].costo > pivote.costo) j--;
                if (i <= j)
                {
                    Ice temp;
                    temp = vector[i];
                    vector[i] = vector[j];
                    vector[j] = temp;
                    i++;
                    j--;
                }
            } while (i <= j);

            if (primero < j)
            {
                Quicksort(vector, primero, j);
            }
            if (i < ultimo)
            {
                Quicksort(vector, i, ultimo);
            }
        }

        // A iterative binary search function. It returns location of x in
        // given array arr[l..r] if present, otherwise -1
        static int BinarySearch(Ice[] arr, int l, int r, int x)
        {
            //int l = 0, r = arr.Length - 1;

            while (l <= r)
            {
                int m = l + (r - l) / 2;

                // Check if x is present at mid
                if (arr[m].costo == x)
                {
                    return m;
                }

                // If x greater, ignore left half
                if (arr[m].costo < x)
                {
                    l = m + 1;
                }

                // If x is smaller, ignore right half
                else
                {
                    r = m - 1;
                }
            }

            // if we reach here, then element was not present
            return -1;
        }


        static void Main(string[] args)
        {
            int t = Convert.ToInt32(Console.ReadLine());
            for (int a0 = 0; a0 < t; a0++)
            {
                int m = Convert.ToInt32(Console.ReadLine());
                int n = Convert.ToInt32(Console.ReadLine());
                string[] a_temp = Console.ReadLine().Split(' ');
                int[] a = Array.ConvertAll(a_temp, e => int.Parse(e));

                Ice[] iceCream = new Ice[n];
                for (int i = 0; i < n; i++)
                {
                    iceCream[i] = new Ice(a[i], i + 1);
                }

                Quicksort(iceCream, 0, n - 1);

                for (int i = 0; i < n; i++)
                {
                    int complemento = m - iceCream[i].costo;

                    int indiceComplemento = BinarySearch(iceCream, i + 1, iceCream.Length - 1, complemento);

                    if (indiceComplemento != -1)
                    {
                        Console.WriteLine(Math.Min(iceCream[i].indice, iceCream[indiceComplemento].indice) + " "
                            + Math.Max(iceCream[i].indice, iceCream[indiceComplemento].indice));
                        break;
                    }


                }

            }
           // Console.ReadLine();
        }



}

```

Line: 121 Col: 18

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

Next Challenge

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