using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

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();

}

//static void Main()

//{

// int[] costos = { 5, 9, 1, 3, 4 };

// Ice[] ice = new Ice[costos.Length];

// for (int i = 0; i < costos.Length; i++)

// {

// ice[i] = new Ice(costos[i], i + 1);

// }

// Quicksort(ice, 0, ice.Length - 1);

// foreach (Ice elem in ice)

// {

// Console.WriteLine(elem.costo + " " + elem.indice);

// }

// Console.ReadLine();

//}

}

}