Minimum Loss

by nabila_ahmed

Problem Submissions Leaderboard Discussions Editorial

Lauren has a chart of distinct projected prices for a house over the next n years, where the price of the house in the i^{th} year is p_i. She wants to purchase and resell the house at a minimal *loss* according to the following rules:

- The house cannot be sold at a price greater than or equal to the price it was purchased at (i.e., it must be resold at a loss).
- The house cannot be resold within the same year it was purchased.

Find and print the *minimum* amount of money Lauren must lose if she buys the house and resells it within the next n years.

Note: It's guaranteed that a valid answer exists.

Input Format

The first line contains an integer, n, denoting the number of years of house data.
The second line contains n space-separated long integers describing the respective values of $p_1, p_2, \ldots, p_n$.

Constraints

- $2 \leq n \leq 2 \times 10^5$
- $1 \leq p_i \leq 10^{16}$
- All the prices are distinct.
- It's guaranteed that a valid answer exists.

Subtasks

- $2 \leq n \leq 1000$ for 50% of the maximum score.

Output Format

Print a single integer denoting the minimum amount of money Lauren must lose if she buys and resells the house within the next n years.

Sample Input 0

```
3
5 10 3
```

Sample Output 0

```
2
```

Explanation 0

Lauren buys the house in year 1 at price $p_1 = 5$ and sells it in year 3 at $p_3 = 3$ for a minimal loss of $5 - 3 = 2$.

Sample Input 1

```
5
20 7 8 2 5
```

Sample Output 1

```
2
```

Explanation 1

Lauren buys the house in year 2 at price $p_2 = 7$ and sells it in year 5 at $p_5 = 5$ for a minimal loss of $7 - 5 = 2$.

Current Buffer (saved locally, editable) C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
class Solution
{
    static void quicksort(long[] vector, int[] paralelo, int primero, int ultimo)
        {
            int i, j, central;
            long pivote;
            central = (primero + ultimo) / 2;
            pivote = vector[central];
            i = primero;
            j = ultimo;
            do
            {
                while (vector[i] < pivote) i++;
                while (vector[j] > pivote) j--;
                if (i <= j)
                {
                    long temp;
                    temp = vector[i];
                    vector[i] = vector[j];
                    vector[j] = temp;

                    int t2 = paralelo[i];
                    paralelo[i] = paralelo[j];
                    paralelo[j] = t2;

                    i++;
                    j--;
                }
            } while (i <= j);

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


```

```
46          static void Main(string[] args)
47          {
48              int n =  int.Parse(Console.ReadLine());
49              long[] p =  Array.ConvertAll(Console.ReadLine().Split(' '), e => long.Parse(e));
50
51
52
53              /* la idea es ordenar el array, y asociarle un array paralelo
54               * para almacenar los indices, hago un array paralelo porque,
55               * si lo ordeno y luego lo recorro de punta a punta
56               * para saber cual es la minima diferencia entre p[i] - p[i-1]
57               * entonces el indice p[i-1] puede estar despues o antes en el
58               array no ordenado, y yo necesito que p[i] sea menor que p[i-1]
59               y ADEMAS que el indice i-1 sea mayor que i en el array original,
60               por eso hago un array paralelo almacenando los indices y lo ordeno
61               como se ordenan los arrays con los paralelos usando el quicksort
62               porque sino da que excede el tiempo limite.
63               * Si hago un bucle i con un bucle j interno tambien excede el tiempo limite
64               en ese caso tomaria O(n^2),
65               pero haciendolo así toma O(nLog n) para el quicksort
66               y O(n) para verificar de punta a punta la minima diferencia
67               * entre p[i-1] y p[i]*/
68
69
70              int[] indices = new int[n];
71              for (int i = 0; i < n; i++)
72              {
73                  indices[i] = i;
74              }
75
76              quicksort(p, indices, 0, n - 1);
77
78
79              long min_dif = int.MaxValue;
80              for (int i = 1; i < n; i++)
81              {
82                  long dif = p[i] - p[i - 1];
83
84                  if (indices[i - 1] > indices[i])
85                  {
86                      min_dif = Math.Min(min_dif, dif);
87                  }
88              }
89
90              Console.WriteLine(min_dif);
91
92              //Console.ReadLine();
93
94          }
95
96      }
```

Line: 95 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
✔ Test Case #12	✔ Test Case #13	✔ Test Case #14

✔ Test Case #15

Next Challenge