1st Jan 2023

First question:-

Stock buy and sell

The cost of stock on each day is given in an array **A[]** of size **N**. Find all the days on which you buy and sell the stock so that in between those days your profit is maximum.[1]

Note: Output format is as follows - (buy_day sell_day) (buy_day sell_day)
For each input, the output should be in a single line, i.e. It's important to move to a new/next line for printing the output of other test cases.

Example 1:

```
Input:
N = 7
A[] = { 100, 180, 260, 310, 40, 535, 695 }

Output:
(0 3) (4 6)

Explanation 1:
We can buy stock on day 0,
and sell it on 3rd day,
which will give us maximum profit.
```

Example 2:

```
Input:
N = 10
A[] = {23, 13, 25, 29, 33, 19, 34, 45, 65, 67}

Output:
(1 4) (5 9)
```

Your Task:

Complete **stockBuySell()** function and print all the days with profit in a single line. And if there is no profit then print "**No Profit**". You do not require to return since the function is void.

Constraints:

1 <= T <= 100

2 <= N <= 10^4

0 <= A_i <= 10^4

CODE SECTION:-

```
// Program to find best buying and selling days
#include <bits/stdc++.h>

using namespace std;

// This function finds the buy sell schedule for maximum profit
void stockBuySell(int *, int);

// Driver program to test above functions
int main()
{
    int T;
    cin >> T;

    while (T--)
    {
        int n, i;
        cin >> n;
        int price[n];
        for (i = 0; i < n; i++)
            cin >> price[i];
        // function call
        stockBuySell(price, n);
    }
    return 0;
}

// } Driver Code Ends

// User function template for C++
// This function finds the buy sell schedule for maximum profit
void stockBuySell(int price[], int n)
{
    // code here

    // need revision

    int low = 0, high = 1, count = 0;
    while (high < n)
    {
        if (price[high] > price[high - 1])
            high++;
        else
        {
            if (low != high - 1)
            {
                cout << "(" << low << " " << high - 1 << ") ";
                low = high++;
                count++;
            }
            else
                low = high++;
        }
    }
    if (price[high - 1] > price[high - 2])
        cout << "(" << low << " " << high - 1 << ") ";
    else if (count == 0)
        cout << "No Profit";
    cout << endl;
}
```

Second question:-

Aggressive Cows (Medium) Accuracy: **59.57%**Submissions: **14K+**Points: **4**

You are given an **array** consisting of **n integers** which denote the position of a **stall**. You are also given an **integer k** which denotes the number of aggressive cows. You are given the task of **assigning stalls to k cows** such that the **minimum distance between any two of them is the maximum possible**.

The first line of input contains two space-separated integers **n** and **k**.

The second line contains **n** space-separated integers denoting the position of the stalls.

Example 1:

```
Input:
n=5
k=3
stalls = [1 2 4 8 9]
Output:
3
Explanation:
The first cow can be placed at stalls[0],
the second cow can be placed at stalls[2] and
the third cow can be placed at stalls[3].
The minimum distance between cows, in this case, is 3,
which also is the largest among all possible ways.
```

Example 2:

```
Input:
n=5
k=3
stalls = [10 1 2 7 5]
Output:
4
Explanation:
The first cow can be placed at stalls[0],
the second cow can be placed at stalls[1] and
the third cow can be placed at stalls[4].
The minimum distance between cows, in this case, is 4,
which also is the largest among all possible ways.
```

Your Task:

Complete the function int solve(), which takes integer n, k, and a vector stalls with n integers as input and returns the largest possible minimum distance between cows.

Expected Time Complexity: O(n*log(10^9)).

Expected Auxiliary Space: O(1).

Constraints:

2 <= n <= 10^5

2 <= k <= n

0 <= stalls[i] <= 10^9

Code section:-

```
//{ Driver Code Starts
//  Initial Template for C++
#include <bits/stdc++.h>
using namespace std;

// } Driver Code Ends
// User function Template for C++

class Solution
{
public:
    bool cancowsplace(vector<int> v, int n, int cows, int dis)
    {

        // this function will check whether the cows can stay at the given
stalls or not

        int co = v[0];
        int count = 1;

        for (int i = 1; i < n; i++)
        {

            if ((v[i] - co) >= dis)
            {
                count++;
                co = v[i];
            }

            if (count == cows)
            {
                return true;
            }
        }
```

```cpp
        return false;
    }



    int solve(int n, int k, vector<int> &stalls)
    {

        // Write your code here
        sort(stalls.begin(), stalls.end());

        int low, high, mid;
        int res = 0;

        low = 1;
        high = stalls[n - 1] - stalls[0];

        while (low <= high)
        {

            mid = (high + low) / 2;

            if (cancowsplace(stalls, n, k, mid))
            {
                res = mid;
                low = mid + 1;
            }

            else
            {

                high = mid - 1;
            }
        }

        return res;
    }
};
```

```cpp
//{ Driver Code Starts.

int main()
{
    int t = 1;
    cin >> t;

    // freopen ("output_gfg.txt", "w", stdout);

    while (t--)
    {
        // Input

        int n, k;
        cin >> n >> k;

        vector<int> stalls(n);
        for (int i = 0; i < n; i++)
        {
            cin >> stalls[i];
        }
        // char ch;
        // cin >> ch;

        Solution obj;
        cout << obj.solve(n, k, stalls) << endl;

        // cout << "~\n";
    }
    // fclose(stdout);

    return 0;
}
// } Driver Code Ends
```