Princeton Competitive Programming

Fall 2023
Welcome to Fall 23
— Intro to Competitive Programming
— Problem Solving session
A few words about competitive programming

What is competitive programming?

- Solving complex problems with algorithms and math under tight time limits
- Participants use programming languages to create fast and correct solutions

Why should I do competitive programming?

- Learn problem-solving and coding skills, readying you for industry challenges
- Improve efficiency and creativity, tackling complex problems under a time limit
- Boost your resume, unlocks job and internship opportunities, and practice for interviews

What if I don’t want to compete?

- Our events prioritize problem-solving and learning, with optional competition
- Look out for upcoming interview prep sessions in the fall semester, designed to specifically address interview aspects (details coming soon)
We have a website: https://competitive-programming.cs.princeton.edu/

We have a discord: _________________________

“Division II”:
- For people with little or no competitive programming experience
- Problems will often have associated readings for you to learn new concepts

“Division I”:
- For people with at least some competitive programming experience
- Problems of varying difficulties to target different levels of experience.

Divisions are informal and you can switch whenever you want

Problem solving sessions are supposed to be interactive and collaborative

Last 30 minutes of each session will be used for solutions discussion

Most weeks will start with a short (~30 minute) talk on some topic (some advanced some beginner)
Logistics

This week’s problems

Past problems (you can try to solve any problem at any time)

Click here to be able to submit

Go to our group on codeforces and click on “Join”

https://codeforces.com
ICPC and Competing

- **Local Qualifier:**
  Local selection contest at Princeton in the **first week of October**
  Participate as a team or individually in this contest
  5 teams will be selected to represent Princeton at the next stage.

- **Greater New York Regional Contest:**
  Will be held on **Sunday, October 29** at Columbia University
  Top ~5 teams qualify for the next stage, but only one per university

- **North American Championship:**
  Will happen at some point at the **end of the Spring semester** in the USA
  The first ~18 teams qualify for the next stage

- **ICPC World Finals:**
  The last stage, date and location is yet to be announced

Note: You don’t have to compete in order to benefit from these sessions!
Welcome to Fall 23

Intro to Competitive Programming

Problem Solving session
Intro to Competitive Programming

**Problem Statement**: Detailed problem statement describing the task

**Input Format**: Input will be provided in specific format. You need to read from standard input

**Output Format**: Expected format for the output also specified. Write to standard output

**Testing**: Output must exactly match the expected output. Even a single character difference or extra whitespace can result in a “Wrong Answer” verdict

**Multiple Test Cases**: Each problem has multiple test cases to evaluate correctness and efficiency

**Samples**: You’ll get sample input and output to help you understand the problem’s requirements

**Hidden Test Cases**: Hidden test cases evaluate your solution. You do not have access to these test cases, so your code should handle all possible inputs

**Languages and Libraries**: You can use any programming language and any default library

**126/226 Libraries**: If you are used to the special 126/226 libraries, these are NOT available on codeforces
A. To My Critics

- Time limit per test: 1 second
- Memory limit per test: 256 megabytes
- Standard input
- Standard output

Sunnet has three digits \(a\), \(b\), and \(c\).

Since math isn't his strongest point, he asks you to determine if you can choose any two digits to make a sum greater or equal to 10.

Output "YES" if there is such a pair, and "NO" otherwise.

**Input**
The first line contains a single integer \(t\) \((1 \leq t \leq 1000)\) — the number of test cases.
The only line of each test case contains three digits \(a\), \(b\), \(c\) \((0 \leq a, b, c \leq 9)\).

**Output**
For each test case, output "YES" if such a pair exists, and "NO" otherwise.

You can output the answer in any case (for example, the strings "yEs", "yes", "Yes" and "YES" will be recognized as a positive answer).

**Example**

<table>
<thead>
<tr>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>YES</td>
</tr>
<tr>
<td>8 1 2</td>
<td>NO</td>
</tr>
<tr>
<td>4 4 5</td>
<td>YES</td>
</tr>
<tr>
<td>9 9 9</td>
<td>NO</td>
</tr>
<tr>
<td>0 0 0</td>
<td>YES</td>
</tr>
<tr>
<td>8 5 3</td>
<td>YES</td>
</tr>
</tbody>
</table>

**Note**
For the first test case, by choosing the digits 8 and 2 we can obtain a sum of 8 + 2 = 10 which satisfies the condition, thus the output should be "YES".

For the second test case, any combination of chosen digits won't be at least 10, thus the output should be "NO" (note that we can not choose the digit on the same position twice).

For the third test case, any combination of chosen digits will have a sum equal to 18, thus the output should be "YES".
B. Balanced Round

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are the author of a Codeforces round and have prepared $n$ problems you are going to test. A problem having difficulty $a_i$. You will do the following process:

- remove some (possibly zero) problems from the list;
- rearrange the remaining problems in any order you wish.

A round is considered balanced if and only if the absolute difference between the difficulty of any two consecutive problems is at most $k$ (ties or equal ties are allowed).

What is the minimum number of problems you have to remove so that an arrangement of problems is balanced?

Input
The first line contains a single integer $t$ ($1 \leq t \leq 1000$) — the number of test cases.

The first line of each test case contains two positive integers $w$ ($0 \leq w \leq 10^9$) and $k$ ($1 \leq k \leq 10^9$) — the number of problems, and the maximum allowed absolute difference between consecutive problems.

The second line of each test case contains $w$ space-separated integers $a_i$ ($1 \leq a_i \leq 10^9$) — the difficulty of each problem.

Note that the sum of $w$ over all test cases doesn't exceed $10^9$.

Output
For each test case, output is a single integer — the minimum number of problems you have to remove so that an arrangement of problems is balanced.

Example

Input
7
7
5 3 4 5 6
3 2
2 0 1 2 3 4
3 2 9 12 5 17 32
4 4 6 8
3 2 3 18 19 8
3 2 18 5 2
8 1 2 4 6 7 3

Output
0
0
1
0
1
1
1

Note
For the first test case, we can remove the first 2 problems and construct a set using problems with the difficulties $\{4, 5, 6\}$, with difficulties between adjacent problems equal to $|5 - 4| = 1 \leq 1$ and $|6 - 5| = 1 \leq 1$.

For the second test case, we can take the single problem and compose a round using the problems with difficulty $1$. 
Welcome to Fall 23
Intro to Competitive Programming
Problem Solving session