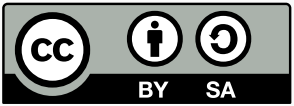
Introduction

Welcome

- Welcome to the FPC 2024 Introduction Session
- 2 parts
- First part gives introduction of programming contests
- We will discuss all last year's problems of the FPC in the second part
- Problem set is available on paper, try to read through it in the break

Who am I

- Alumnus, working in the Software Industry
- Involved in organizing programming contests since 2003 as volunteer
- “Coach” for TU Delft teams since NWERC 2003
- Twice coach on the World Finals



Introduction to Programming Contests

What is a programming contest?

- Team of 3 people
- Single computer
- Solve as many problems from the problem set (8 to 15 problems)
- In 5 hours
- In any order

What is a programming contest?

- Team of 3 people
- Single computer
- Solve as many problems from the problem set (8 to 15 problems)
- In 5 hours
- In any order
  - Solve it efficiently
  - do it as quickly as possible (under pressure)
  - and do it correctly (without bugs)

What is a programming contest?

- Team of 3 people
- Single computer
- Solve as many problems from the problem set (8 to 15 problems)
- In 5 hours
- In any order
  - Solve it efficiently
  - do it as quickly as possible (under pressure)
  - and do it correctly (without bugs)
- With limited documentation and no internet

How is score calculated?

- Sorted by number of problems solved

How is score calculated?

- Sorted by number of problems solved
- Sorted by the total time for solved problems

How is score calculated?

- Sorted by number of problems solved
- Sorted by the total time for solved problems
  - Time in minutes since the start of the contest
  - Penalty for each wrong attempt on a solved solution of 20 minutes

How is score calculated?

- Sorted by number of problems solved
- Sorted by the total time for solved problems
  - Time in minutes since the start of the contest
  - Penalty for each wrong attempt on a solved solution of 20 minutes
    - Penalty time is counts only if the problem is solved afterward.
    - Penalty time does not reduce your contest time.
    - Penalty time is not added for wrong attempts after the problem is solved.
    - No penalty for compile errors.

Example Scoreboard

DAPC 2022 — final standings

RANK	TEAM	SCORE	A	B	C	D	E	F	G	H	I	J	K	L
1	Delft University of Technology **Segfault go BRRRR** Delft University of Technology	12 1090	22 1 try	48 2 tries	41 1 try	94 1 try	66 1 try	7 1 try	190 3 tries	223 1 try	118 1 try	106 2 tries	85 1 try	10 1 try
2	**ADA Refactor** Delft University of Technology	11 1087	21 2 tries	44 2 tries	98 1 try	108 1 try	77 1 try	25 1 try	251 1 try	1 try	66 1 try	162 4 tries	129 1 try	6 1 try
3	**Chindia Targoviste** Delft University of Technology	11 1273	74 3 tries	31 1 try	188 1 try	112 1 try	61 1 try	14 1 try	226 1 try	1 try	142 2 tries	149 2 tries	175 1 try	21 1 try
4	**WA & Chill** Delft University of Technology	11 1424	9 1 try	67 1 try	111 1 try	181 1 try	131 4 tries	35 1 try	298 3 tries		154 1 try	102 3 tries	143 1 try	53 1 try
5	**Placeholder** Delft University of Technology	11 1589	50 3 tries	110 2 tries	76 1 try	181 1 try	224 1 try	19 1 try	289 4 tries		144 1 try	99 2 tries	160 1 try	17 5 tries
6	**Exponential Fenwick** Delft University of Technology	10 749	14 1 try	62 2 tries	139 4 tries	94 1 try	61 1 try	18 1 try	2 tries	3 tries	42 2 tries	99 1 try	116 1 try	4 1 try
7	**Dirty Bits Done Dirt Cheap** Delft University of Technology	10 1198	11 1 try	35 2 tries	64 1 try	129 1 try	191 1 try	22 1 try	2 tries		147 1 try	264 10 tries	105 1 try	30 1 try
8	**Sleetje3** Delft University of Technology	10 1311	30 1 try	94 2 tries	49 1 try	222 1 try	209 2 tries	19 1 try	7 tries	1 try	254 1 try	169 5 tries	79 1 try	66 1 try
9	**SMG** Delft University of Technology	10 1534	48 1 try	10 1 try	201 3 tries	169 1 try	219 3 tries	21 1 try	7 tries	1 try	267 8 tries	93 2 tries	192 4 tries	14 1 try
10	**Poland Mountain** Delft University of Technology	10 1626	65 3 tries	55 1 try	287 4 tries	217 1 try	119 1 try	20 1 try	1 try		253 1 try	172 2 tries	218 1 try	100 1 try
11	**NoDucksGiven** Delft University of Technology	10 1826	128 3 tries	147 1 try	73 1 try	257 1 try	234 6 tries	36 1 try	290 4 tries			175 2 tries	160 1 try	26 5 tries
12	**bits by dre** Delft University of Technology	9 1396	24 1 try	55 1 try	132 1 try	246 1 try	262 1 try	52 1 try				221 3 tries	234 3 tries	90 1 try

Freshman Programming Contest (FPC)

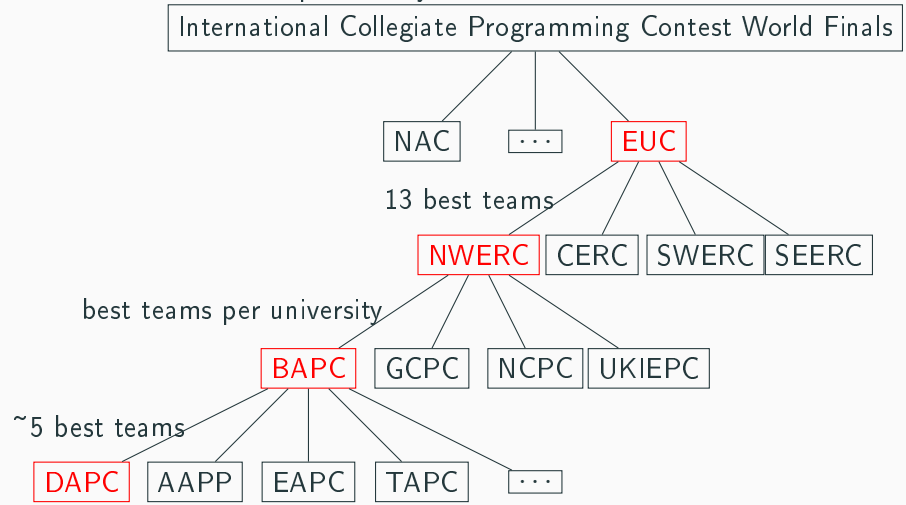
- Contest of 3 hours
- Simpler problems
- Preparation for the Delft Algorithm Programming Contest (DAPC)

Delft Algorithm Programming Contest (DAPC)

- Contest is 5 hours instead of 3 hours
- Official preliminary for International Collegiate Programming Contest (ICPC)
- Increased difficulty in every contest

Road to the world finals

The DAPC is an official preliminary of the ICPC.



Reading a problem

Problem structure

A typical problem has the following structure

- Problem description
- Input description
- Output description
- Example input/output
- A time limit in seconds

You are asked to write a program that solves the problem for all valid inputs within the time limit.

Example problem

Problem description
Write a program that multiplies pairs of integers.

Input description
The input consists of:

- One line with an integer t ($1 \leq t \leq 100$), the number of test cases.
- t lines, each with two integers a and b ($|a|, |b| \leq 10^6$), the numbers to multiply.

Output description
For each test case, output the value of $a \times b$.

Example problem

Sample input	Sample output
4 3 4 13 0 1 8 100 100	12 0 8 10000

Solution in C++

```cpp
#include <iostream>
using namespace std;

int main() {
    int t;
    cin >> t;
    for (int i = 0; i < t; i++) {
        int a, b;
        cin >> a >> b;
        cout << a * b << endl;
    }
    return 0;
}
```

Solution in Java

```java
import java.io.*;

class Problem {
  public static void main(String[] args) throws IOException {
    var input = new BufferedReader(new InputStreamReader(System.in));
    var cases = Integer.parseInt(input.readLine());
    for (int i = 0; i < cases; i++) {
      var line = input.readLine().split(" ");
      System.out.println(
        Integer.parseInt(line[0]) * Integer.parseInt(line[1])
      );
    }
  }
}
```

Solution in Kotlin and Python

```kotlin
fun main() {
  val t = readln().toInt();
  System.`in`.bufferedReader().lineSequence().take(t).forEach { line ->
    println(line.split(" ").map { it.toInt() }.let { (a, b) -> a * b })
  }
}
```

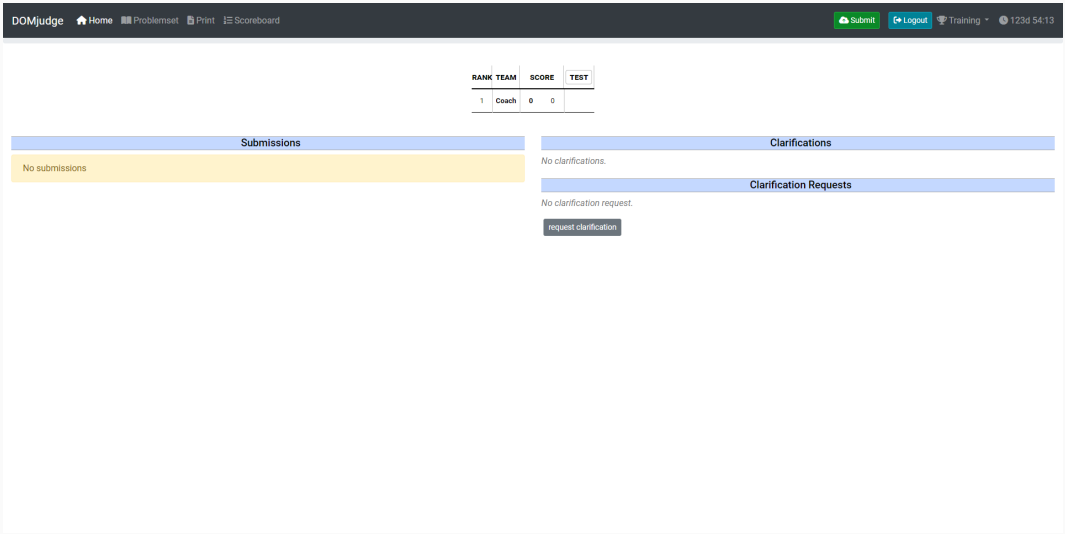
```python
t = int(input())
for t in range(t):
  numbers = list(map(int, input().split()))
  print(numbers[0] * numbers[1])
```

Introduction to DOMjudge

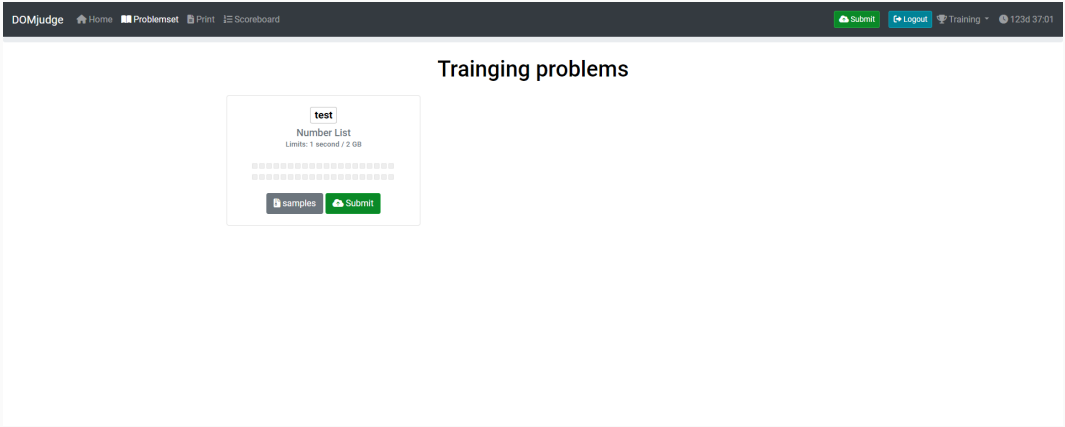
Submitting the Solution

- During the contest you submit to a contest control system
  - Usually DOMjudge, but sometimes Kattis or PC^2
- Submit solutions
- Ask questions about the problems or programming environment
- Read clarifications from the jury

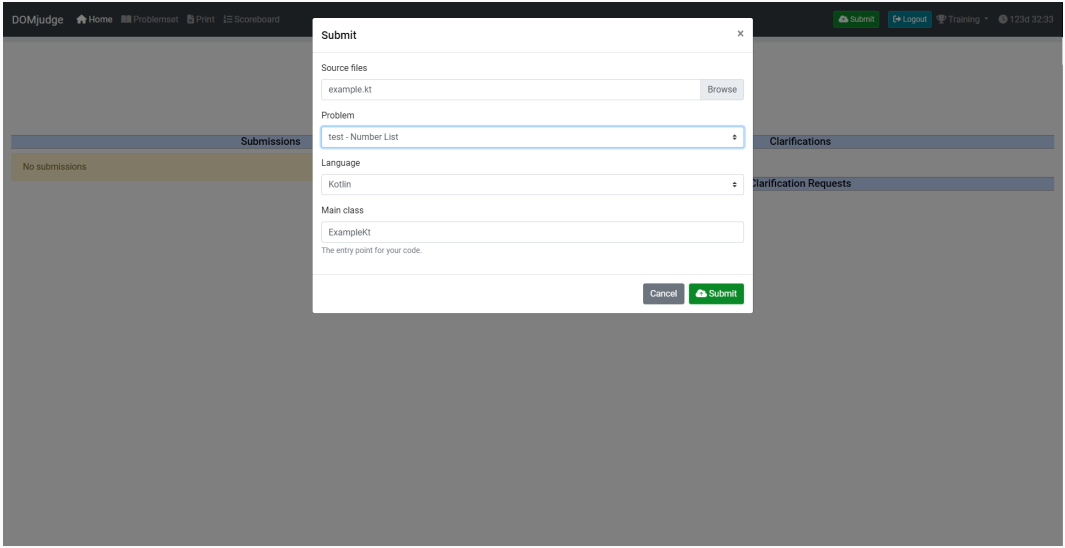
DOMjudge Interface - home



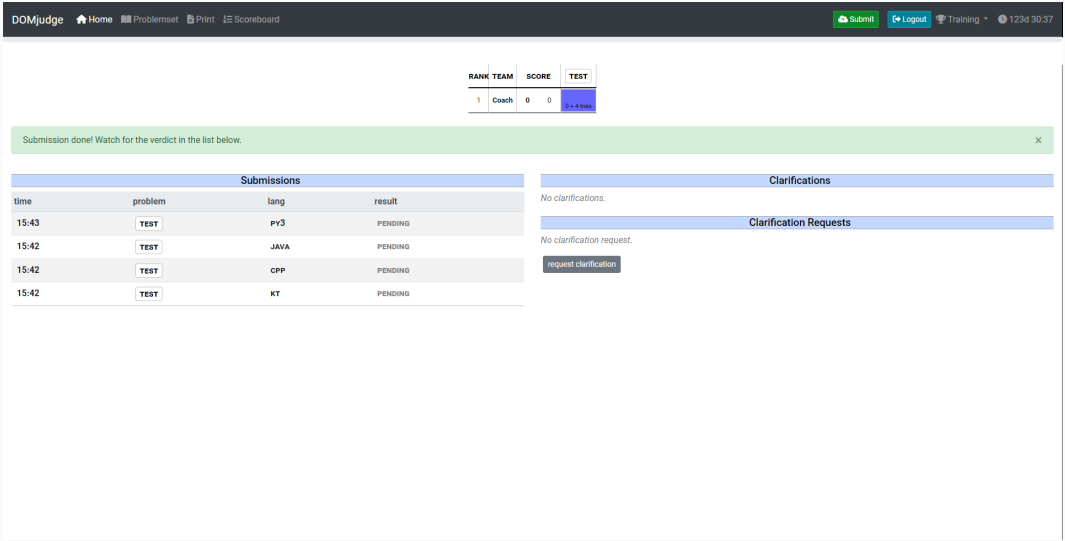
DOMjudge Interface - problems



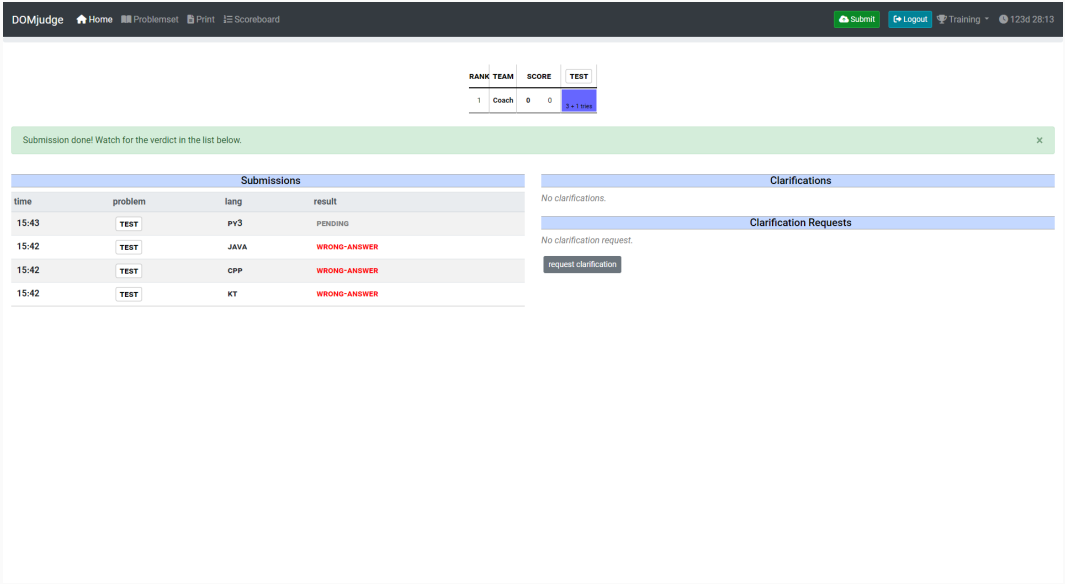
DOMjudge Interface - submit



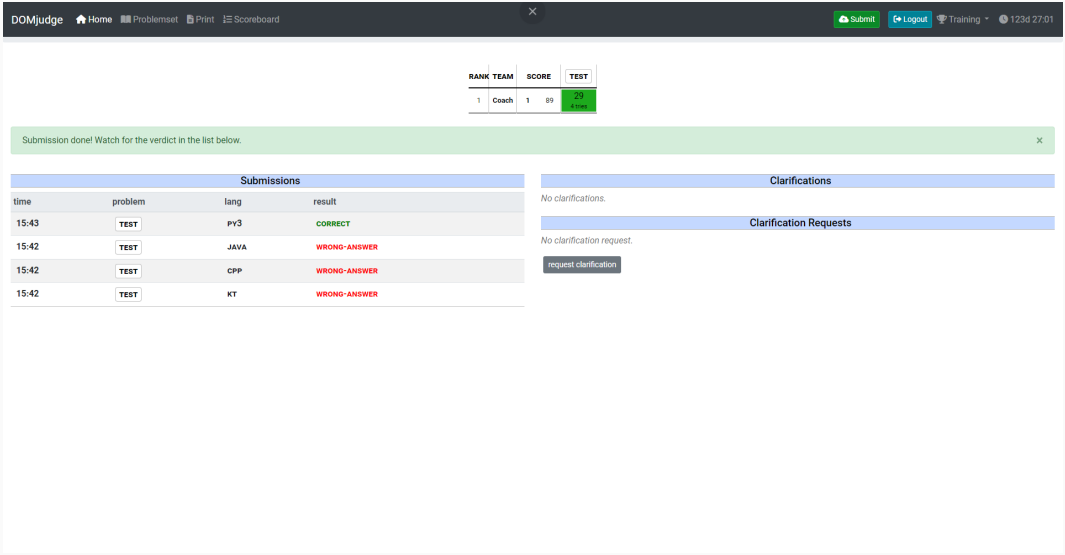
Are the solutions correct?



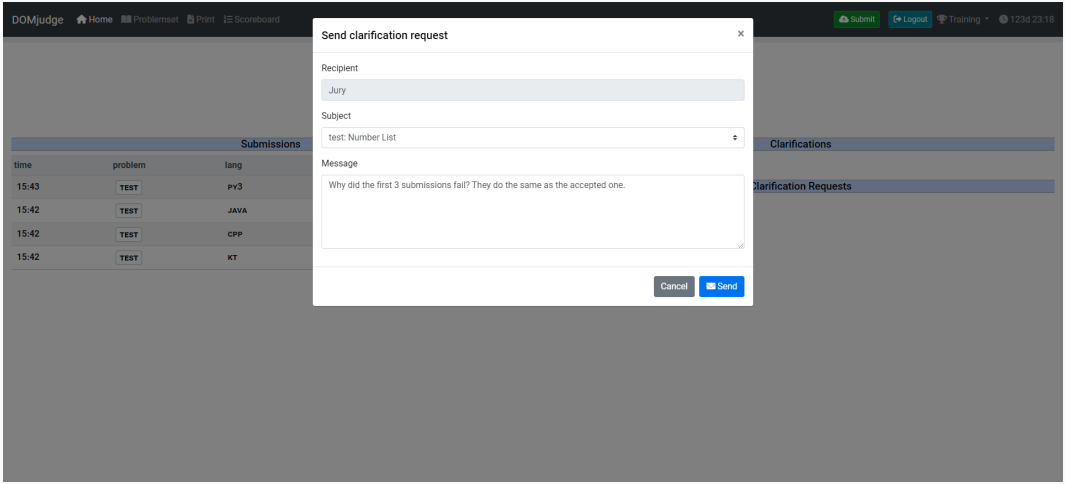
We made a whoopsie?



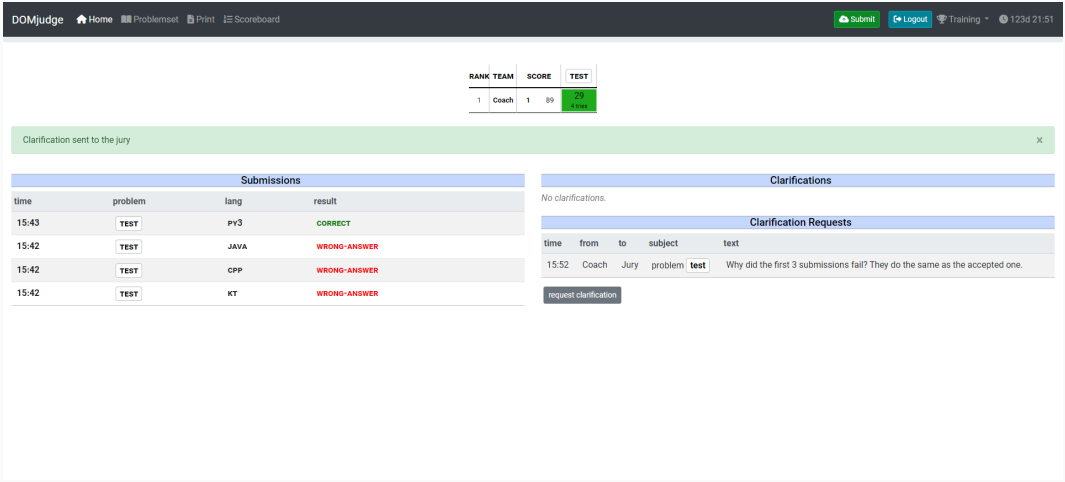
Or not


DOMjudge
Home
Problemset
Print
Scoreboard
Submit
Logout
Training
123d 27:01
RANK
TEAM
SCORE
TEST
1
Coach
1
89
-29
4 tries
Submission done! Watch for the verdict in the list below.
Submissions
time
problem
lang
result
15:43
TEST
PY3
CORRECT
15:42
TEST
JAVA
WRONG-ANSWER
15:42
TEST
CPP
WRONG-ANSWER
15:42
TEST
KT
WRONG-ANSWER
Clarifications
No clarifications.
Clarification Requests
No clarification request.
request clarification

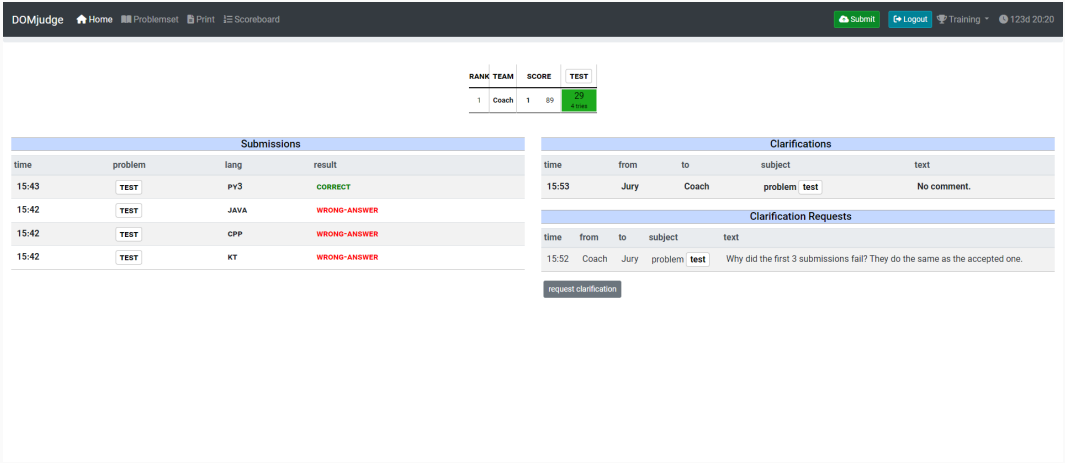
Let's ask the jury



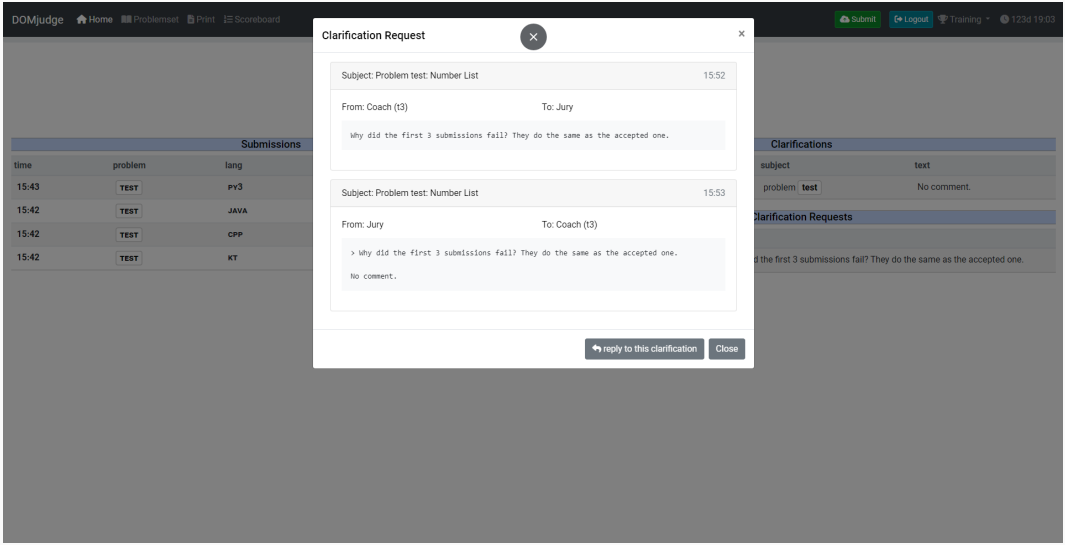
Let's hope they respond fast



We have a response



The jury is not helping us



Why did the 3 solutions fail?

- Let's check the input again: $|a|, |b| \leq 10^6$

Why did the 3 solutions fail?

- Let's check the input again: $|a|, |b| \leq 10^6$
- Worst case scenario: $a = 10^6$ and $b = 10^6$ giving $a \times b = 10^{12}$

Why did the 3 solutions fail?

- Let's check the input again: $|a|, |b| \leq 10^6$
- Worst case scenario: $a = 10^6$ and $b = 10^6$ giving $a \times b = 10^{12}$
- Does 10^{12} fit in a 32-bit `int`?

Why did the 3 solutions fail?

- Let's check the input again: $|a|, |b| \leq 10^6$
- Worst case scenario: $a = 10^6$ and $b = 10^6$ giving $a \times b = 10^{12}$
- Does 10^{12} fit in a 32-bit `int`?
- $\log_2 10^{12} \approx 40$, so **NO**, 40 bits don't fit in an `int`

Why did the 3 solutions fail?

- Let's check the input again: $|a|, |b| \leq 10^6$
- Worst case scenario: $a = 10^6$ and $b = 10^6$ giving $a \times b = 10^{12}$
- Does 10^{12} fit in a 32-bit `int`?
- $\log_2 10^{12} \approx 40$, so **NO**, 40 bits don't fit in an `int`
- Use `long` (`long`) when possible, except in Python

Solution in C++

```cpp
#include <iostream>
using namespace std;

int main() {
    int t;
    cin >> t;
    for (int i = 0; i < t; i++) {
        long long a, b;
        cin >> a >> b;
        cout << a * b << endl;
    }
    return 0;
}
```

Solution in Java

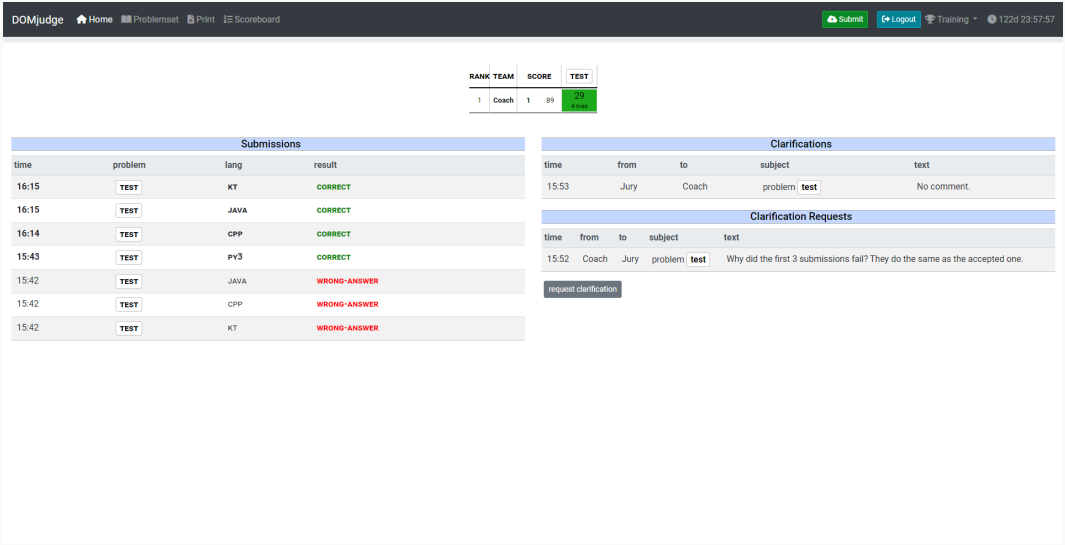
```java
import java.io.*;

class ProblemCorrect {
  public static void main(String[] args) throws IOException {
    var input = new BufferedReader(new InputStreamReader(System.in));
    var cases = Integer.parseInt(input.readLine());
    for (int i = 0; i < cases; i++) {
      var line = input.readLine().split(" ");
      System.out.println(
        Long.parseLong(line[0]) * Long.parseLong(line[1])
      );
    }
  }
}
```

Solution in Kotlin

```kotlin
fun main() {
  val t = readln().toInt();
  System.`in`.bufferedReader().lineSequence().take(t).forEach { line ->
    println(line.split(" ").map { it.toLong() }.let { (a, b) -> a * b })
  }
}
```

All solutions correct



Estimating problem complexity

About time limit

- The time limit specifies the time you program may run
- This includes JVM-startup and I/O
- High time limit signify
  - lots of I/O
  - Slower algorithms can be accepted
- Low limit signifies fast algorithms, usually the use of formulas
- You can use the time limit to check your code on your local machine

```
$ time myjava ProblemA < worst-case.in
```

About input size[1]

Based on the input size you can an idea of the time complexity.

$\mathcal{O}(n!)$	$n \leq 10$	$\mathcal{O}(n \log^2 n)$	$n \leq 10^5$
$\mathcal{O}(2^n)$	$n \leq 20$	$\mathcal{O}(n \log n)$	$n \leq 10^6$
$\mathcal{O}(n^3)$	$n \leq 500$	$\mathcal{O}(n)$	$n \leq 10^8$
$\mathcal{O}(n^2 \log n)$	$n \leq 1000$	$\mathcal{O}(\sqrt{n})$	$n \leq 10^{15}$
$\mathcal{O}(n^2)$	$n \leq 5000$	$\mathcal{O}(\log n)$	$n \leq 10^{18}$
$\mathcal{O}(n\sqrt{n})$	$n \leq 10^5$		

Warning: This is not guaranteed to be always the case!

[1] https://gcpc.nwerc.eu/primer.pdf

Tips, tricks and common mistakes

General tips

- Read the output specification carefully!
- Don't forget to remove debug prints!
- When integers get large, use 64-bit!
- Do not do string concatenation with + in a loop!
- Calling functions is more expensive than you might think!
- For Java, `BufferedReader` is faster than `Scanner`!
- Don't forget to eat and drink. Programming contest is a sport, and you need to be energized and focussed for the whole contest.

General Tactics

- Know each other's strengths and weaknesses like:
  - types of problems (math, geometry, search, strings, graphs, etc.)
  - debugging skills
  - coding speed and accuracy
- Parallelize
- Work on paper (e.g. pseudocode or flow diagrams)
- Debug on paper
- Use rubber duck debugging when stuck

Team Tactics

- Plot of the contest: 3 contestants, 1 computer

Team Tactics

- Plot of the contest: 3 contestants, 1 computer
- Several tactics how to divide the computer efficiently

Team Tactics

- Plot of the contest: 3 contestants, 1 computer
- Several tactics how to divide the computer efficiently
- Shuffle Tactic

- Designated Tactic

- No best solution: Pick and mix what works best for your team

Team Tactics

- Plot of the contest: 3 contestants, 1 computer
- Several tactics how to divide the computer efficiently
- Shuffle Tactic
  - Rotate around who sits behind the pc
  - After submitting a problem, switch around if someone has a solution
  - Useful when programming in different languages
- Designated Tactic

- No best solution: Pick and mix what works best for your team

Team Tactics

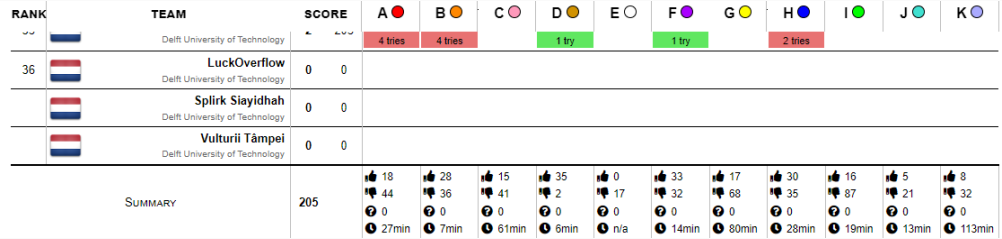
- Plot of the contest: 3 contestants, 1 computer
- Several tactics how to divide the computer efficiently
- Shuffle Tactic
  - Rotate around who sits behind the pc
  - After submitting a problem, switch around if someone has a solution
  - Useful when programming in different languages
- Designated Tactic
  - Dedicated person behind computer
  - Other team members work on paper or read along on screen
  - Useful for teams with different disciplines
- No best solution: Pick and mix what works best for your team

Selecting the first problem

- Decide on a reading tactic
  - Do we all start reading the first problem?
  - Or does one person start at the end?
- There are usually several “simple” problems in a set
- Be careful: the easiest problems usually contain some pitfall corner cases!

Finding the easiest problems by results

- After a few minutes of contest, the first balloons will be handed out
- Check the scoreboard or balloon colours to see which problem is solved most



RANK	TEAM	SCORE	A	B	C	D	E	F	G	H	I	J	K
	Delft University of Technology		4 tries	4 tries		1 try		1 try		2 tries			
36	LuckOverflow Delft University of Technology	0 0											
	Splirk Siayidhah Delft University of Technology	0 0											
	Vulturii Tâmpei Delft University of Technology	0 0											
	SUMMARY	205	18 44 0 27min	28 36 0 7min	15 41 0 61min	35 2 0 6min	0 17 0 n/a	33 32 0 14min	17 68 0 80min	30 35 0 28min	16 87 0 19min	5 21 0 13min	8 32 0 113min

- Or the problems page in DOMjudge (only newer versions)
- **Warning**: The first problem solved is not guaranteed the easiest!

My problem is wrong, what now

Print out the problem and let other people work on the computer, work out cases that might go wrong.

- When the result is Run Time Error (RTE):
  - Check for possible null pointers, array overflows, or integer overflow
  - Check the input specification, don't forget 0 can do unexpected things
- When the result is Time Limit Exceeded (TLE):
  - Check stop conditions, maybe an infinite loop?
  - Code is too slow, try optimizing or thinking of a faster solution
- When then result is Wrong Answer (WA):
  - Check for corner cases, don't forget zero
  - Check correctness of algorithm
- **Warning**: A problem can be RTE and TLE and WA at the same time, but only *one* is reported back!

Common Battle Plan

Start of contest Prepare computer, find and solve easiest problems, all problems should be read by at least a single team member.

First hours Prioritize solving the easiest problems, every team member works on their own problems

Mid contest Work on solving harder problems with 2 people, while the last person works alone on the last easy or specialized hard problems

End of the contest Work together with the whole team on a single problem, free submit mode

Common Errors

- Focusing on the first problem you think you can solve
- Not reading all problems in the set
- Debugging on the computer while another solution can be implemented
- Fighting who can solve which problem
- Not rewriting code when it gets to messy

Training your self

- If you want to try to make it to the World Finals, you can train for next year's DAPC
- Many online problem-solving websites:
  - December: Advent of Code (`https://adventofcode.com/`)
  - September–January: Universal Cup (`https://ucup.ac`)
  - Year round: Kattis Problem Archive (`https://open.kattis.com/`)
  - Year round: Codeforces (`https://codeforces.com/`)
- Several books available, listed on `https://chipcie.wisv.ch/resources`

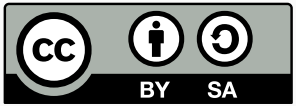
FPC 2023 problems

FPC 2023 problems

- We go through the problems in alphabetical order
- Implementations are left to the reader
- Reference solutions can be found in the CHipCie problem archive at `https://chipcie.wisv.ch/archive`

Admiring Droplets

- FPC 2023
- Time limit: 3s
- Difficulty: Very Easy
- Given n droplets on the same vertical line with size s (μL) on a window with distance y (mm) from the top. The velocity is is given by $v = \sqrt[6]{V}$ (v in m/s and V in m^3) and when two droplets meet they coalesce together ($V = s_{current} + s_{next}$). Calculate the time takes for the coalesced droplet to reach the bottom of the window.



Admiring Droplets

- **Observation:** $n \leq 10^5$, so we are looking for a $\mathcal{O}(n)$ solution

Admiring Droplets

- **Observation:** $n \leq 10^5$, so we are looking for a $\mathcal{O}(n)$ solution
- Simulate the droplets from highest to lowest

Admiring Droplets

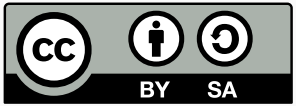
- **Observation:** $n \leq 10^5$, so we are looking for a $\mathcal{O}(n)$ solution
- Simulate the droplets from highest to lowest
- For the current droplet, calculate the time to reach the closest droplet

Admiring Droplets

- **Observation:** $n \leq 10^5$, so we are looking for a $\mathcal{O}(n)$ solution
- Simulate the droplets from highest to lowest
- For the current droplet, calculate the time to reach the closest droplet
- Merge the droplets together and calculate the new size
- Repeat until a single droplet is left and print the sum of the time
- **Pitfall:**
  - Be careful with unit conversion: $1\,m = 1000\,mm$, $1\,m^3 = 10^9\,mm^3$
  - Off by one errors

Beaking Spackwards

- FPC 2023
- Time limit: 1s
- Difficulty: Easy
- Given a number s, print a string containing exactly s palindromes



Beaking Spackwards

- **Observation:** $n \leq 10^9$, so we are looking for a solution faster $\mathcal{O}(n)$

Beaking Spackwards

- **Observation:** $n \leq 10^9$, so we are looking for a solution faster $\mathcal{O}(n)$
- **Observation:** a string "aaa...aaa" of size l has $\frac{l(l+1)}{2}$ palindromes

Beaking Spackwards

- **Observation:** $n \leq 10^9$, so we are looking for a solution faster $\mathcal{O}(n)$
- **Observation:** a string "aaa...aaa" of size l has $\frac{l(l+1)}{2}$ palindromes
- The length of the word should be $\lceil \sqrt{n} \rceil$

Beaking Spackwards

- **Observation:** $n \leq 10^9$, so we are looking for a solution faster $\mathcal{O}(n)$
- **Observation:** a string "aaa...aaa" of size l has $\frac{l(l+1)}{2}$ palindromes
- The length of the word should be $\lceil \sqrt{n} \rceil$
- Find the largest l where $l \leq n$ by search or using the formula $l = \lfloor \frac{\sqrt{8n+1}-1}{2} \rfloor$

$$\frac{l(l+1)}{2} \leq n \equiv l^2 + l \leq 2n \equiv l^2 + l - 2n \leq 0 \equiv l \leq \frac{-1 + \sqrt{1 + 4 \cdot 2n}}{2}$$

Beaking Spackwards

- **Observation:** $n \leq 10^9$, so we are looking for a solution faster $\mathcal{O}(n)$
- **Observation:** a string "aaa...aaa" of size l has $\frac{l(l+1)}{2}$ palindromes
- The length of the word should be $\lceil \sqrt{n} \rceil$
- Find the largest l where $l \leq n$ by search or using the formula $l = \lfloor \frac{\sqrt{8n+1}-1}{2} \rfloor$

$$\frac{l(l+1)}{2} \leq n \equiv l^2 + l \leq 2n \equiv l^2 + l - 2n \leq 0 \equiv l \leq \frac{-1 + \sqrt{1 + 4 \cdot 2n}}{2}$$

- Generate a string of length l with the same letter that is unused
- or append a non palindrome letter to increase the number of palindromes is the desired size
- Update n with the remaining length and repeat until $n = 0$ and print the string

Beaking Spackwards

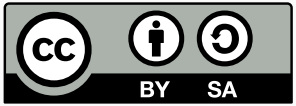
- **Observation:** $n \leq 10^9$, so we are looking for a solution faster $\mathcal{O}(n)$
- **Observation:** a string "aaa...aaa" of size l has $\frac{l(l+1)}{2}$ palindromes
- The length of the word should be $\lceil \sqrt{n} \rceil$
- Find the largest l where $l \leq n$ by search or using the formula $l = \lfloor \frac{\sqrt{8n+1}-1}{2} \rfloor$

$$\frac{l(l+1)}{2} \leq n \equiv l^2 + l \leq 2n \equiv l^2 + l - 2n \leq 0 \equiv l \leq \frac{-1 + \sqrt{1 + 4 \cdot 2n}}{2}$$

- Generate a string of length l with the same letter that is unused
- or append a non palindrome letter to increase the number of palindromes is the desired size
- Update n with the remaining length and repeat until $n = 0$ and print the string
- **Pitfall:** Be careful with slow string concatenations

Catchy Tunes

- FPC 2023
- Time limit: 3s
- Difficulty: Medium
- Given a list on n songs with their artist, generate an ordering where every song is followed with a song from a different artist.



Catchy Tunes

- **Observation:** $n \leq 10^5$, so we are looking for a $\mathcal{O}(n)$ solution
- At least half of the songs in the playlist are from a unique artist

Catchy Tunes

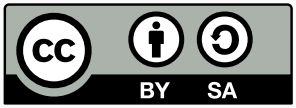
- **Observation:** $n \leq 10^5$, so we are looking for a $\mathcal{O}(n)$ solution
- At least half of the songs in the playlist are from a unique artist
- Group the songs in two lists during input if the artist is unique or not

Catchy Tunes

- **Observation:** $n \leq 10^5$, so we are looking for a $\mathcal{O}(n)$ solution
- At least half of the songs in the playlist are from a unique artist
- Group the songs in two lists during input if the artist is unique or not
- Alternate printing a song title from either list and remove it
- If one list is empty, keep printing from the other list

Dungeon of Darkness

- FPC 2023
- Time limit: 1s
- Difficulty: Hard
- **Interactive Problem**
- Using interactions, navigate a maze and find the exit.



What are Interactive Problems?

- Traditional problems give all the input at once, you solve and print all the output at once
- Interactive problems give input, you do work, print output, and you receive new input
- This process continues until you find the final answer
- The problem defines an interaction protocol
- The problem may have an interaction limit
- If an interactive problem may be in the set, an simple interactive problem will be included in the test session

Type of problems for Interactive Problems

- Search in a finite space
- Explore a maze
- Matching games
- Decision problems

Common pitfalls for Interactive problems

- Flush the output after every write
  - Only the output, not the input
  - Not flushing the output results in Time Limit Exceeded
- Verdict of a solution is not deterministic, but the following is guaranteed:
  - Wrong Answer means you printed something wrong
  - Runtime Error means you returned a non-zero exit code
  - If both occur, you will get either
- ICPC style contests don't have "Idleness Limit Exceeded", but a total runtime limit.

Flushing the output

C++: end your output with `std::endl` or `std::flush`

Python: use the flush parameter, like `print("abc", flush=True)`

Java/Kotlin: use a `java.io.BufferedWriter` and after each write use the `.flush()` method.

Interactive problems testing tool

- Most contests provide a testing tool to test the interaction with a testing tool
- This is usually called `testing_tool.py` in our region
- The header file tells you how to run run the testing tool, for example
  `$ python3 testing_tool.py -f 1.in python3 ./solution.py`
- Pitfall for Java/Kotlin: You should run the testing tool in the directory which contains the compiled class file
- **Wrong**:
  `~/$ python3 testing_tool.py -f 1.in java ./code/ProblemA`
- **Right**:
  `~/code/$ python3 testing_tool.py -f 1.in java ProblemA`

Dungeon of Darkness

- Get to the final room in a dungeon, where you only see the symbols of the doors leading from the current room

Dungeon of Darkness

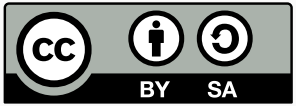
- Get to the final room in a dungeon, where you only see the symbols of the doors leading from the current room
- Use a Depth First Search (DFS) to delve deeper in the maze

Dungeon of Darkness

- Get to the final room in a dungeon, where you only see the symbols of the doors leading from the current room
- Use a Depth First Search (DFS) to delve deeper in the maze
- If the exit is in the room, go through to the exit
- If in a room, the exit is not there, move through a door you haven't visited yet
- If all doors are visited, move back to the door you entered the room through

Expected Eyes

- FPC 2023
- Time limit: 4s
- Difficulty: Very Easy
- Given n dices with x faces, calculate the expected value of throwing all dice at once.



Expected Eyes

- **Observation:** $n \cdot x \leq 64$, so we are looking for a solution faster than $\mathcal{O}(2^n)$
- **Observation:** High time limit signals brute force might be possible

Expected Eyes

- **Observation:** $n \cdot x \leq 64$, so we are looking for a solution faster than $\mathcal{O}(2^n)$
- **Observation:** High time limit signals brute force might be possible
- Calculate all possible throws for the combined dice and sum them up
- Divide them by the number of possible outcomes
- Complexity $\mathcal{O}(x_{max}^n)$ gets accepted due to high time limit

Expected Eyes

- **Observation:** $n \cdot x \leq 64$, so we are looking for a solution faster than $\mathcal{O}(2^n)$
- **Observation:** High time limit signals brute force might be possible
- Calculate all possible throws for the combined dice and sum them up
- Divide them by the number of possible outcomes
- Complexity $\mathcal{O}(x_{max}^n)$ gets accepted due to high time limit
- The expected value of two independent variables is $E[X + Y] = E[X] + E[Y]$

Expected Eyes

- **Observation:** $n \cdot x \leq 64$, so we are looking for a solution faster than $\mathcal{O}(2^n)$
- **Observation:** High time limit signals brute force might be possible
- Calculate all possible throws for the combined dice and sum them up
- Divide them by the number of possible outcomes
- Complexity $\mathcal{O}(x_{max}^n)$ gets accepted due to high time limit
- The expected value of two independent variables is $E[X + Y] = E[X] + E[Y]$
- The expected value of a dice d_k with k faces is

$$E[d_k] = \frac{1}{k} \cdot \sum_{k}^{x=1} x = \frac{1}{k} \cdot \frac{k(k+1)}{2} = \frac{k+1}{2}$$

Expected Eyes

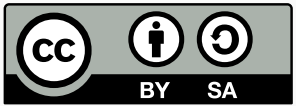
- **Observation:** $n \cdot x \leq 64$, so we are looking for a solution faster than $\mathcal{O}(2^n)$
- **Observation:** High time limit signals brute force might be possible
- Calculate all possible throws for the combined dice and sum them up
- Divide them by the number of possible outcomes
- Complexity $\mathcal{O}(x_{max}^n)$ gets accepted due to high time limit
- The expected value of two independent variables is $E[X + Y] = E[X] + E[Y]$
- The expected value of a dice d_k with k faces is

$$E[d_k] = \frac{1}{k} \cdot \sum_{k}^{x=1} x = \frac{1}{k} \cdot \frac{k(k+1)}{2} = \frac{k+1}{2}$$

- Sum the expected value of each dice gives a complexity of $\mathcal{O}(n)$

Feline Friendship

- FPC 2023
- Time limit: 2s
- Difficulty: Hard
- Given n cats and there preferred team partner, create teams of k and calculate the minimal number of cats you have to convince to not be in the team with their favourite player.



Feline Friendship

- **Observation:** $n \leq 2 \cdot 10^5$, so we are looking for a $\mathcal{O}(n \log n)$ solution

Feline Friendship

- **Observation:** $n \leq 2 \cdot 10^5$, so we are looking for a $\mathcal{O}(n \log n)$ solution
- Only a single team needs to be created (Example 1)
- Since cats have unique preferences, there already teams created following the cycle or preferences

Feline Friendship

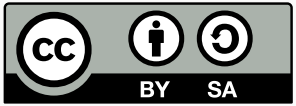
- **Observation:** $n \leq 2 \cdot 10^5$, so we are looking for a $\mathcal{O}(n \log n)$ solution
- Only a single team needs to be created (Example 1)
- Since cats have unique preferences, there already teams created following the cycle or preferences
- if the cycle for a team is l then:
  - $l = k$ No operations are needed for that team
  - $l > k$ A single operation suffice, convincing the k^{th} cat
  - $l < k$ For 2 teams of size a and b can be merged in size $\in [a + 1, a + b]$ in a single operation. start with the greatest cycle and merge the next longest cycle, repeat until the size is $\geq k$ and count the merges

Feline Friendship

- **Observation:** $n \leq 2 \cdot 10^5$, so we are looking for a $\mathcal{O}(n \log n)$ solution
- Only a single team needs to be created (Example 1)
- Since cats have unique preferences, there already teams created following the cycle or preferences
- if the cycle for a team is l then:
  - $l = k$ No operations are needed for that team
  - $l > k$ A single operation suffice, convincing the k^{th} cat
  - $l < k$ For 2 teams of size a and b can be merged in size $\in [a + 1, a + b]$ in a single operation. start with the greatest cycle and merge the next longest cycle, repeat until the size is $\geq k$ and count the merges
- Complexity is $\mathcal{O}(n)$ for finding disjoint cycles
- Complexity is $\mathcal{O}(n \log n)$ for sorting the cycles by length

Grid Lock

- FPC 2023
- Time limit: 6s
- Difficulty: Very Hard
- Given a grid with arrows, remove them one by one. When removing an arrow, it must not point to another arrow in the grid



Gridlock

- **Observation:** This problem has a large output (4000 lines), so not much can be said about the complexity

Gridlock

- **Observation:** This problem has a large output (4000 lines), so not much can be said about the complexity
- Every tile has a dependency on all its preceding tiles in the direction on the tile.
- Topological Sort the dependencies is too slow $\mathcal{O}(h \cdot w(h+w))$

Gridlock

- **Observation:** This problem has a large output (4000 lines), so not much can be said about the complexity
- Every tile has a dependency on all its preceding tiles in the direction on the tile.
- Topological Sort the dependencies is too slow $\mathcal{O}(h \cdot w(h + w))$
- For every tile, keep track of its neighbours

Gridlock

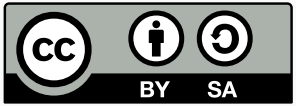
- **Observation:** This problem has a large output (4000 lines), so not much can be said about the complexity
- Every tile has a dependency on all its preceding tiles in the direction on the tile.
- Topological Sort the dependencies is too slow $\mathcal{O}(h \cdot w(h + w))$
- For every tile, keep track of its neighbours
  - Start by removing a tile with no dependencies
  - Update the dependencies of the neighbours by removing the tile and linking to new blocking tiles
  - For every neighbouring tile try to remove tile
  - Repeat until no more tiles can be removed or the board is empty

Gridlock

- **Observation:** This problem has a large output (4000 lines), so not much can be said about the complexity
- Every tile has a dependency on all its preceding tiles in the direction on the tile.
- Topological Sort the dependencies is too slow $\mathcal{O}(h \cdot w(h + w))$
- For every tile, keep track of its neighbours
  - Start by removing a tile with no dependencies
  - Update the dependencies of the neighbours by removing the tile and linking to new blocking tiles
  - For every neighbouring tile try to remove tile
  - Repeat until no more tiles can be removed or the board is empty
- This approach is $\mathcal{O}(h \cdot w)$

Hunting the Mavericks

- FPC 2023
- Time limit: 3s
- Difficulty: Medium
- Determine in which level to start your playthrough, so that you miss the least armour upgrades



Hunting the Mavericks

- **Observation:** $n + m = s \leq 2 \cdot 10^5$, so we are looking for a $\mathcal{O}(s \log s)$ solution

Hunting the Mavericks

- **Observation:** $n + m = s \leq 2 \cdot 10^5$, so we are looking for a $\mathcal{O}(s \log s)$ solution
- For each level i, precalculate
  - the number of armour upgrades it contains (c_i)
  - the number of armour that requires a weapon of a later level (r_i)

Hunting the Mavericks

- **Observation:** $n + m = s \leq 2 \cdot 10^5$, so we are looking for a $\mathcal{O}(s \log s)$ solution
- For each level i, precalculate
  - the number of armour upgrades it contains (c_i)
  - the number of armour that requires a weapon of a later level (r_i)
- Calculate initial number x of armour you miss if you start on level 1

Hunting the Mavericks

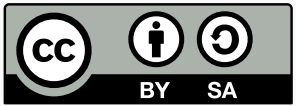
- **Observation:** $n + m = s \leq 2 \cdot 10^5$, so we are looking for a $\mathcal{O}(s \log s)$ solution
- For each level i, precalculate
  - the number of armour upgrades it contains (c_i)
  - the number of armour that requires a weapon of a later level (r_i)
- Calculate initial number x of armour you miss if you start on level 1
- Iterate over each level and calculate the number of missed weapons on level i, and update $x = x + r_i - c_i$
- Then output is the minimum value of x in $\mathcal{O}(n + m)$

Hunting the Mavericks

- **Observation:** $n + m = s \leq 2 \cdot 10^5$, so we are looking for a $\mathcal{O}(s \log s)$ solution
- For each level i, precalculate
  - the number of armour upgrades it contains (c_i)
  - the number of armour that requires a weapon of a later level (r_i)
- Calculate initial number x of armour you miss if you start on level 1
- Iterate over each level and calculate the number of missed weapons on level i, and update $x = x + r_i - c_i$
- Then output is the minimum value of x in $\mathcal{O}(n + m)$
- **Alternatively:** Use a segment tree to store the ranges in which you miss each armour, resulting in $\mathcal{O}(m \log n)$ and more code

Industry Improvements

- FPC 2023
- Time limit: 2s
- Difficulty: Medium
- Given a list of n boxes that need to be processed by a machine line in at most k runs, determine the minimum summed weight that the machine needs to handle in one run.



Industry Improvements

- **Observation:** $n \leq 10^5$, so we are looking for a $\mathcal{O}(n \log^2 n)$ solution

Industry Improvements

- **Observation:** $n \leq 10^5$, so we are looking for a $\mathcal{O}(n \log^2 n)$ solution
- Calculating the number of restarts for a machine for a given max weight is trivial, since you have to process the boxes in order

Industry Improvements

- **Observation:** $n \leq 10^5$, so we are looking for a $\mathcal{O}(n \log^2 n)$ solution
- Calculating the number of restarts for a machine for a given max weight is trivial, since you have to process the boxes in order
- If a max weight can process the boxes in less then k runs for capacity a, then it will also work for any higher max weight

Industry Improvements

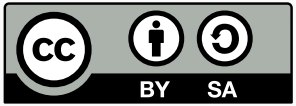
- **Observation:** $n \leq 10^5$, so we are looking for a $\mathcal{O}(n \log^2 n)$ solution
- Calculating the number of restarts for a machine for a given max weight is trivial, since you have to process the boxes in order
- If a max weight can process the boxes in less then k runs for capacity a, then it will also work for any higher max weight
- Binary search the solution over the range $[\min(x), \sum x]$
- Start in the middle, go to the right half is smaller then k, else go the left half.

Industry Improvements

- **Observation:** $n \leq 10^5$, so we are looking for a $\mathcal{O}(n \log^2 n)$ solution
- Calculating the number of restarts for a machine for a given max weight is trivial, since you have to process the boxes in order
- If a max weight can process the boxes in less then k runs for capacity a, then it will also work for any higher max weight
- Binary search the solution over the range $[\min(x), \sum x]$
- Start in the middle, go to the right half is smaller then k, else go the left half.
- This results in a complexity of $\mathcal{O}(n \log \sum x)$

Jurassic Park

- FPC 2023
- Time limit: 3s
- Difficulty: Very hard
- Given a set of uniform random points in a square, find the smallest perimeter among all triangles.



Jurassic Park

- **Observation:** $n \leq 10^3$, so we are looking for a $\mathcal{O}(n\sqrt{n})$ solution

Jurassic Park

- **Observation:** $n \leq 10^3$, so we are looking for a $\mathcal{O}(n\sqrt{n})$ solution
- The points are **uniform randomly** divided, so there are no nasty cases where all points are clustered

Jurassic Park

- **Observation:** $n \leq 10^3$, so we are looking for a $\mathcal{O}(n\sqrt{n})$ solution
- The points are **uniform randomly** divided, so there are no nasty cases where all points are clustered
- Divide the area in a grid of $\lfloor\sqrt{\frac{n}{3}}\rfloor \times \lfloor\sqrt{\frac{n}{3}}\rfloor$

Jurassic Park

- **Observation:** $n \leq 10^3$, so we are looking for a $\mathcal{O}(n\sqrt{n})$ solution
- The points are **uniform randomly** divided, so there are no nasty cases where all points are clustered
- Divide the area in a grid of $\lfloor\sqrt{\frac{n}{3}}\rfloor \times \lfloor\sqrt{\frac{n}{3}}\rfloor$
- Every square of the grid will have at least 3 points in it
- But the smallest triangle might be spanning the over the grid lines

Jurassic Park

- **Observation:** $n \leq 10^3$, so we are looking for a $\mathcal{O}(n\sqrt{n})$ solution
- The points are **uniform randomly** divided, so there are no nasty cases where all points are clustered
- Divide the area in a grid of $\lfloor\sqrt{\frac{n}{3}}\rfloor \times \lfloor\sqrt{\frac{n}{3}}\rfloor$
- Every square of the grid will have at least 3 points in it
- But the smallest triangle might be spanning the over the grid lines
- Calculate the all possible triangles in a by taking 3×3 grid tiles

Jurassic Park

- **Observation:** $n \leq 10^3$, so we are looking for a $\mathcal{O}(n\sqrt{n})$ solution
- The points are **uniform randomly** divided, so there are no nasty cases where all points are clustered
- Divide the area in a grid of $\lfloor\sqrt{\frac{n}{3}}\rfloor \times \lfloor\sqrt{\frac{n}{3}}\rfloor$
- Every square of the grid will have at least 3 points in it
- But the smallest triangle might be spanning the over the grid lines
- Calculate the all possible triangles in a by taking 3×3 grid tiles
- This will calculate the smallest possible triangle in $\mathcal{O}(n)$ time with high probability

Final remarks

- Resources for the contest are available on `https://chipcie.wisv.ch`
- Reference solutions, input and output for the problems can be found in the problem archive
- Good luck during the contest and have fun