NWERC 2024 Test Session

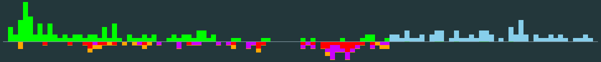
Solutions presentation

The NWERC 2024 jury

November 23, 2024

C: Consolidating Windows

Problem author: The NWERC 2024 jury

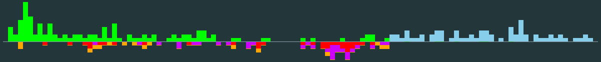


Problem

Given the side lengths of two smaller squares *a* and *b*, calculate the side length of a square with the same area as the two smaller squares combined.

C: Consolidating Windows

Problem author: The NWERC 2024 jury



Problem

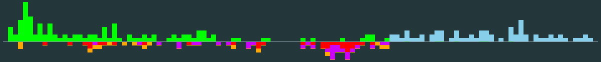
Given the side lengths of two smaller squares a and b, calculate the side length of a square with the same area as the two smaller squares combined.

Solution

Calculate $\sqrt{a^2 + b^2}$.

C: Consolidating Windows

Problem author: The NWERC 2024 jury



Problem

Given the side lengths of two smaller squares a and b, calculate the side length of a square with the same area as the two smaller squares combined.

Solution

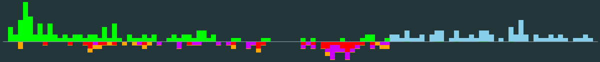
Calculate $\sqrt{a^2 + b^2}$.

Pitfalls

32-bit `floats` do not cover a precision of 10^{-8}, so you need to use at least 64-bit `doubles`.

C: Consolidating Windows

Problem author: The NWERC 2024 jury



Problem

Given the side lengths of two smaller squares a and b, calculate the side length of a square with the same area as the two smaller squares combined.

Solution

Calculate $\sqrt{a^2 + b^2}$.

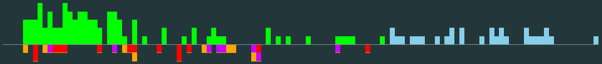
Pitfalls

32-bit `floats` do not cover a precision of 10^{-8}, so you need to use at least 64-bit `doubles`.

Statistics: 267 submissions, 79 accepted, 67 unknown

A: Alternative Encryption

Problem author: Thomas Beuman



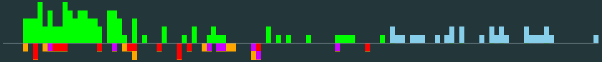
Problem

This is a multi-pass problem, where in each pass, you should:

1. Encrypt text, such that the length stays the same and every character differs.
2. Decrypt the text that you encrypted, such that you retrieve the original input.

A: Alternative Encryption

Problem author: Thomas Beuman



Problem

This is a multi-pass problem, where in each pass, you should:

1. Encrypt text, such that the length stays the same and every character differs.
2. Decrypt the text that you encrypted, such that you retrieve the original input.

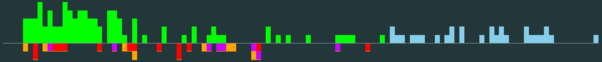
Solution

Some of the many possible solutions (there were some resubmissions):

- (70×) Use a Caesar cipher with offset $1 \leq x < 26$ for encrypting, and offset $26 - x$ for decrypting.
- (14×) Use a Caesar cipher with offset 13 for both encrypting and decrypting.
- (2×) Assuming 0-based `char` values, XOR the last bit of each value (‘`a`’ $\leftrightarrow$ ‘`b`’, ‘`c`’ $\leftrightarrow$ ‘`d`’, . . .).
- (1×) Atbash: Mirror the characters (‘`a`’ $\leftrightarrow$ ‘`z`’, ‘`b`’ $\leftrightarrow$ ‘`y`’, . . .) for both encrypting and decrypting.
- (1×) Generate a (seeded) random permutation to encrypt, and use its inverse to decrypt.

A: Alternative Encryption

Problem author: Thomas Beuman



Problem

This is a multi-pass problem, where in each pass, you should:

1. Encrypt text, such that the length stays the same and every character differs.
2. Decrypt the text that you encrypted, such that you retrieve the original input.

Solution

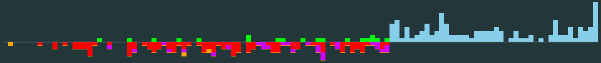
Some of the many possible solutions (there were some resubmissions):

- (70$\times$) Use a Caesar cipher with offset $1 \leq x < 26$ for encrypting, and offset $26 - x$ for decrypting.
- (14$\times$) Use a Caesar cipher with offset 13 for both encrypting and decrypting.
- (2$\times$) Assuming 0-based `char` values, XOR the last bit of each value (`'a'` $\leftrightarrow$ `'b'`, `'c'` $\leftrightarrow$ `'d'`, ...).
- (1$\times$) Atbash: Mirror the characters (`'a'` $\leftrightarrow$ `'z'`, `'b'` $\leftrightarrow$ `'y'`, ...) for both encrypting and decrypting.
- (1$\times$) Generate a (seeded) random permutation to encrypt, and use its inverse to decrypt.

Statistics: 150 submissions, 78 accepted, 29 unknown

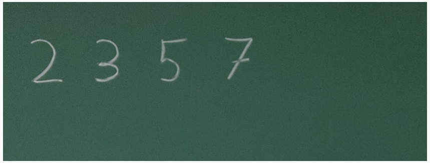
B: Blackboard

Problem author: Takuki Kurokawa



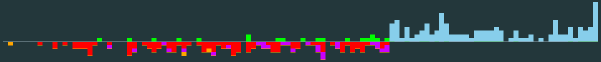
Problem

You are given a list of numbers on a blackboard. Repeatedly split one of the numbers into two parts until the largest number is at most $p\%$ larger than the smallest one.



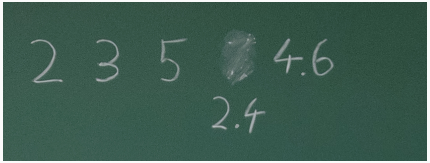
B: Blackboard

Problem author: Takuki Kurokawa



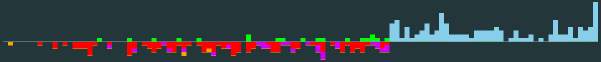
Problem

You are given a list of numbers on a blackboard. Repeatedly split one of the numbers into two parts until the largest number is at most $p\%$ larger than the smallest one.



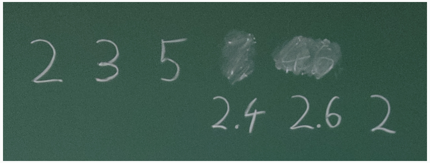
B: Blackboard

Problem author: Takuki Kurokawa



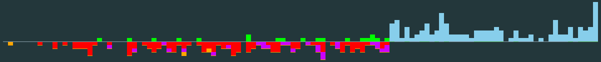
Problem

You are given a list of numbers on a blackboard. Repeatedly split one of the numbers into two parts until the largest number is at most $p\%$ larger than the smallest one.



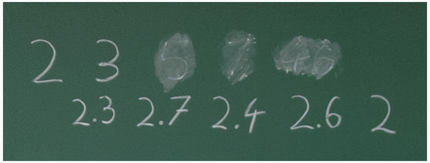
B: Blackboard

Problem author: Takuki Kurokawa



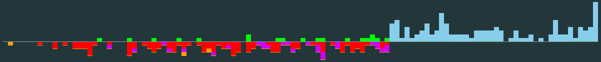
Problem

You are given a list of numbers on a blackboard. Repeatedly split one of the numbers into two parts until the largest number is at most $p\%$ larger than the smallest one.



B: Blackboard

Problem author: Takuki Kurokawa

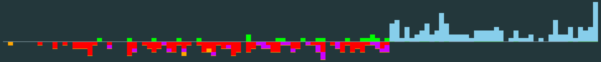


Insights

- It's always optimal to split each number into equal parts.
- $p = 0$ (all numbers must be equal) is a corner case:
  - $\rightsquigarrow$ Make all numbers equal to the greatest common divisor.

B: Blackboard

Problem author: Takuki Kurokawa



Insights

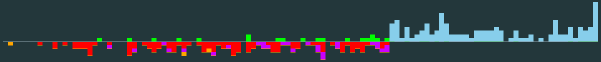
- It's always optimal to split each number into equal parts.
- $p = 0$ (all numbers must be equal) is a corner case:
  $\rightsquigarrow$ Make all numbers equal to the greatest common divisor.

Solution

- Maintain a priority queue containing fractions:
  - Numerators are the original numbers.
  - Denominators say how many parts they are split into.
- Repeatedly take the largest fraction and increase its denominator.

B: Blackboard

Problem author: Takuki Kurokawa



Insights

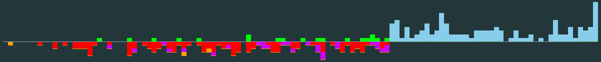
- It's always optimal to split each number into equal parts.
- $p = 0$ (all numbers must be equal) is a corner case:
  - $\rightsquigarrow$ Make all numbers equal to the greatest common divisor.

Solution

- Maintain a priority queue containing fractions:
  - Numerators are the original numbers.
  - Denominators say how many parts they are split into.
- Repeatedly take the largest fraction and increase its denominator.
- This is too slow if you just increment by one at a time.
- To make it fast enough, always compute the smallest denominator needed to make it at most $p\%$ larger than the current smallest one.

B: Blackboard

Problem author: Takuki Kurokawa



Insights

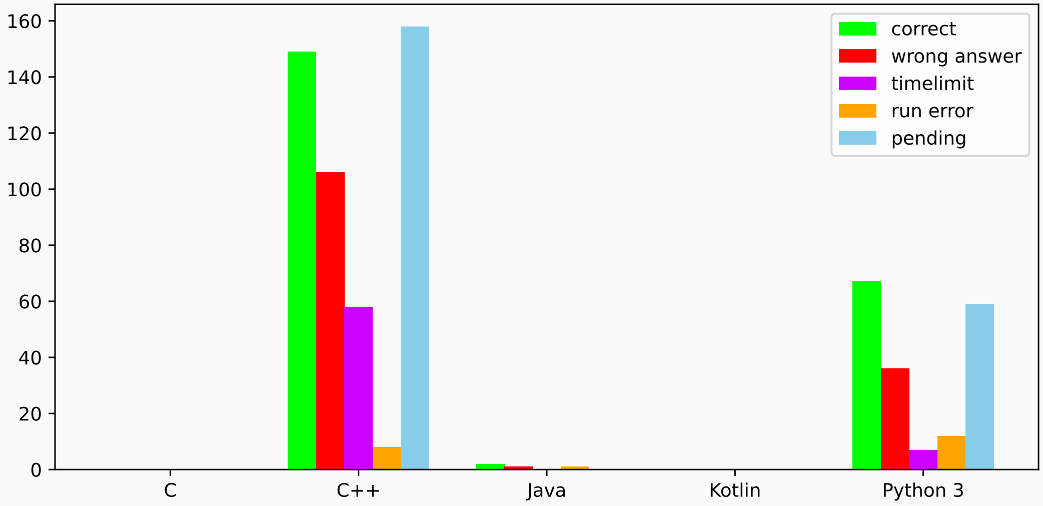
- It's always optimal to split each number into equal parts.
- $p = 0$ (all numbers must be equal) is a corner case:
  - $\rightsquigarrow$ Make all numbers equal to the greatest common divisor.

Solution

- Maintain a priority queue containing fractions:
  - Numerators are the original numbers.
  - Denominators say how many parts they are split into.
- Repeatedly take the largest fraction and increase its denominator.
- This is too slow if you just increment by one at a time.
- To make it fast enough, always compute the smallest denominator needed to make it at most p% larger than the current smallest one.

Statistics: 268 submissions, 14 accepted, 121 unknown

Language stats



Systems update

Systems update

- Everything appears to be working as expected!

Systems update

Systems update

- Everything appears to be working as expected!
- You can remap keys as much as you like, but we will reset your laptop before tomorrow.
  - We will not provide help using `xmodmap`, please look up the correct commands before tomorrow and use them at your own risk.

Systems update

Systems update

- Everything appears to be working as expected!
- You can remap keys as much as you like, but we will reset your laptop before tomorrow.
    - We will not provide help using `xmodmap`, please look up the correct commands before tomorrow and use them at your own risk.
- Reminders about printing:
    - Printing from Code::Blocks does not work.
    - The print command is `printfile <file>`

General remarks

General remarks

- The memory limit of your submission is 2 GiB (also see 2024.nwerc.eu/systems).
    - Note that exceeding the memory limit gives a `RUN-ERROR`.

General remarks

General remarks

- The memory limit of your submission is 2 GiB (also see 2024.nwerc.eu/systems).
  - Note that exceeding the memory limit gives a `RUN-ERROR`.
- The judging is typically case- and whitespace-insensitive (no guarantees though).

General remarks

General remarks

- The memory limit of your submission is 2 GiB (also see 2024.nwerc.eu/systems).
  - Note that exceeding the memory limit gives a `RUN-ERROR`.
- The judging is typically case- and whitespace-insensitive (no guarantees though).
- How to flush standard output: please check the documentation of your programming language before tomorrow.

General remarks

General remarks

- The memory limit of your submission is 2 GiB (also see 2024.nwerc.eu/systems).
  - Note that exceeding the memory limit gives a `RUN-ERROR`.
- The judging is typically case- and whitespace-insensitive (no guarantees though).
- How to flush standard output: please check the documentation of your programming language before tomorrow.
- You *will not* get a time penalty if your submission has a compilation error.

General remarks

General remarks

- The memory limit of your submission is 2 GiB (also see 2024.nwerc.eu/systems).
  - Note that exceeding the memory limit gives a `RUN-ERROR`.
- The judging is typically case- and whitespace-insensitive (no guarantees though).
- How to flush standard output: please check the documentation of your programming language before tomorrow.
- You *will not* get a time penalty if your submission has a compilation error.
- You *will* get a time penalty if your submission fails on a sample case.

General remarks

General remarks

- The memory limit of your submission is 2 GiB (also see 2024.nwerc.eu/systems).
    - Note that exceeding the memory limit gives a `RUN-ERROR`.
- The judging is typically case- and whitespace-insensitive (no guarantees though).
- How to flush standard output: please check the documentation of your programming language before tomorrow.
- You *will not* get a time penalty if your submission has a compilation error.
- You *will* get a time penalty if your submission fails on a sample case.
- You can find the samples and contest PDF in your home folder.

General remarks

General remarks

- The memory limit of your submission is 2 GiB (also see 2024.nwerc.eu/systems).
  - Note that exceeding the memory limit gives a `RUN-ERROR`.
- The judging is typically case- and whitespace-insensitive (no guarantees though).
- How to flush standard output: please check the documentation of your programming language before tomorrow.
- You *will not* get a time penalty if your submission has a compilation error.
- You *will* get a time penalty if your submission fails on a sample case.
- You can find the samples and contest PDF in your home folder.
- Other jury advice: 2024.nwerc.eu/jury-advice

For tomorrow

Tomorrow:

- you will get three copies of the problem set, scrap paper, and pens;

For tomorrow

Tomorrow:

- you will get three copies of the problem set, scrap paper, and pens;
- you are *NOT* allowed to have your bags or any electronic equipment on you (except for medical reasons);

For tomorrow

Tomorrow:

- you will get three copies of the problem set, scrap paper, and pens;
- you are *NOT* allowed to have your bags or any electronic equipment on you (except for medical reasons);
- you *MUST* wear your shirt and badge visibly;

For tomorrow

Tomorrow:

- you will get three copies of the problem set, scrap paper, and pens;
- you are *NOT* allowed to have your bags or any electronic equipment on you (except for medical reasons);
- you *MUST* wear your shirt and badge visibly;
- after the contest, you must take everything with you.