GCPC 2019
Presentation of solutions

Jury and Testers

Thanks to the jury:

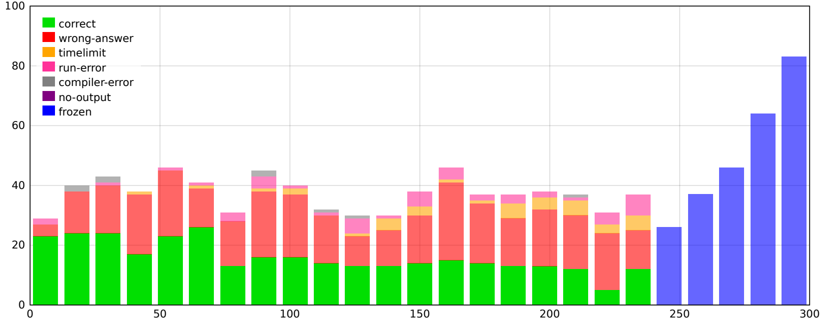
- Paul Wild (FAU)
- Michael Baer (FAU)
- Alexander Dietsch (FAU)
- Philipp Reger (FAU)
- Gregor Schwarz (TUM)
- Tobias Meggendorfer (TUM)
- Christian Müller (TUM)
- Gregor Behnke (Ulm)
- Julian Baldus (UdS)

Thanks to our test readers:

- Bakhodir Ashirmatov (GAU)
- Stefan Kraus (FAU)
- Simon Rainer (FAU)
- Alexander Raß (FAU)
- Stefan Toman (Google)

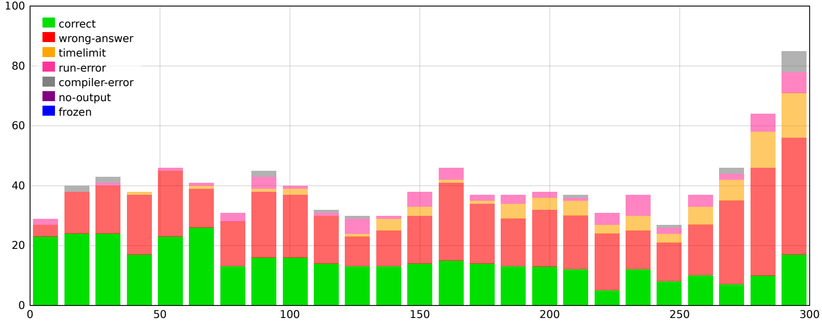
And many thanks to all the volunteers here
and at all other contest sites!

Statistics



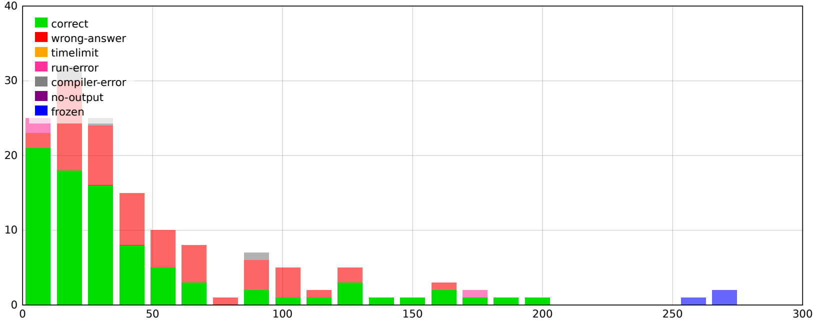
correct
wrong-answer
timelimit
run-error
compiler-error
no-output
frozen
0
20
40
60
80
100
0
50
100
150
200
250
300

Statistics



correct
wrong-answer
timelimit
run-error
compiler-error
no-output
frozen
100
80
60
40
20
0
0
50
100
150
200
250
300

J – Jazz Enthusiast



J – Jazz Enthusiast

Problem

Given the n song lengths of a playlist (in `m:ss` format) and a crossfade of c seconds, how long does it take to listen to the entire playlist (in `hh:mm:ss` format)?

J – Jazz Enthusiast

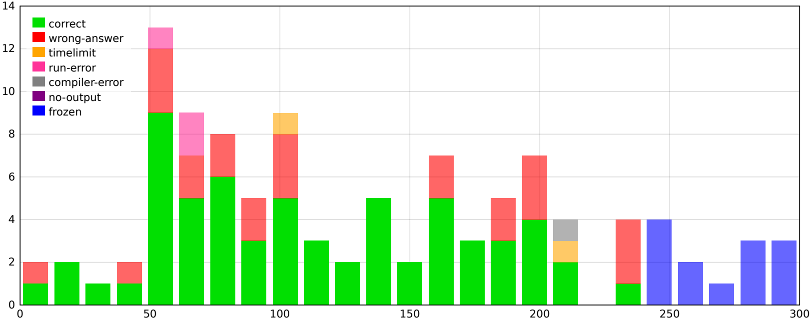
Problem

Given the n song lengths of a playlist (in `m:ss` format) and a crossfade of c seconds, how long does it take to listen to the entire playlist (in `hh:mm:ss` format)?

Solution

- Convert `m:ss` input lengths into (only) seconds.
- Subtract $(n-1) \cdot c$ seconds.
- Convert to `hh:mm:ss` format.
- Careful: `1:00` minus 10 seconds should not become `1:-10`!

A – Assessing Genomes



A – Assessing Genomes

Problem

Given two sets of n DNA strands, determine their repetition score (length of the smallest repeating substring). Then, find a perfect matching, minimising the squared difference of the matched pairs' repetition scores.

A – Assessing Genomes

Problem

Given two sets of n DNA strands, determine their repetition score (length of the smallest repeating substring). Then, find a perfect matching, minimising the squared difference of the matched pairs' repetition scores.

- Two sub-problems: (i) Repetition score, (ii) Matching

A – Assessing Genomes

Solution (i) Repetition score

- Simple brute-force approach sufficient.
- Better: simplification, based on the following observations:
  1. A string consists of the same pattern repeated multiple times if and only if the string is a non-trivial rotation of itself.
  2. If x and y are strings of the same length, then x is a rotation of y if and only if x is a substring of yy.
- Pseudocode: $\text{score} = (s + s).\text{find}(s, 1, 2 \cdot \text{len}(s) - 1)$

A – Assessing Genomes

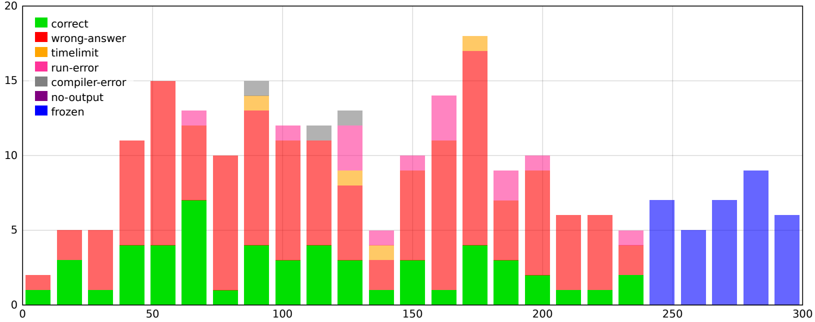
Solution (i) Repetition score

- Simple brute-force approach sufficient.
- Better: simplification, based on the following observations:
  1. A string consists of the same pattern repeated multiple times if and only if the string is a non-trivial rotation of itself.
  2. If x and y are strings of the same length, then x is a rotation of y if and only if x is a substring of yy.
- Pseudocode: $\text{score} = (s + s).\text{find}(s, 1, 2 \cdot \text{len}(s) - 1)$

Solution (ii) Matching

- Use the Hungarian method (also called Munkres algorithm) for optimal matching (rather complex, but $O(n^3)$).
- Better: Sorting and matching yields minimal Euclidean distance in $O(n \log n)$.

M – Move & Meet



correct
wrong-answer
timelimit
run-error
compiler-error
no-output
frozen
0
5
10
15
20
0
50
100
150
200
250
300

M – Move & Meet

Problem

Two players are on a grid, and each has to move a certain number of times. Each move is to an adjacent cell (no diagonals, no staying in place). Determine a cell they can both end up on after their move, or state that there is none.

M – Move & Meet

Problem

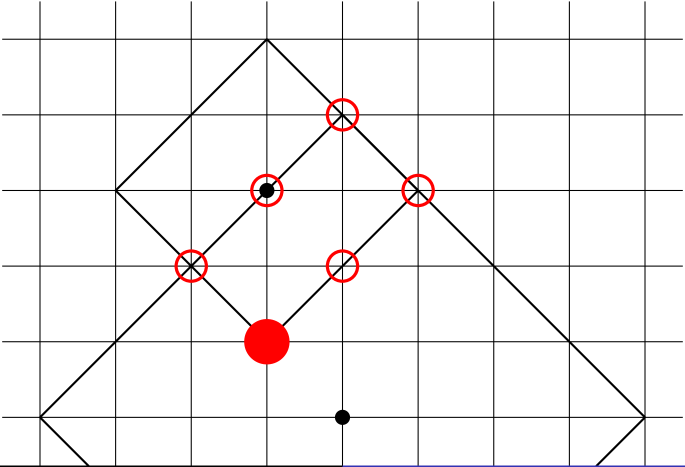
Two players are on a grid, and each has to move a certain number of times. Each move is to an adjacent cell (no diagonals, no staying in place). Determine a cell they can both end up on after their move, or state that there is none.

Solution

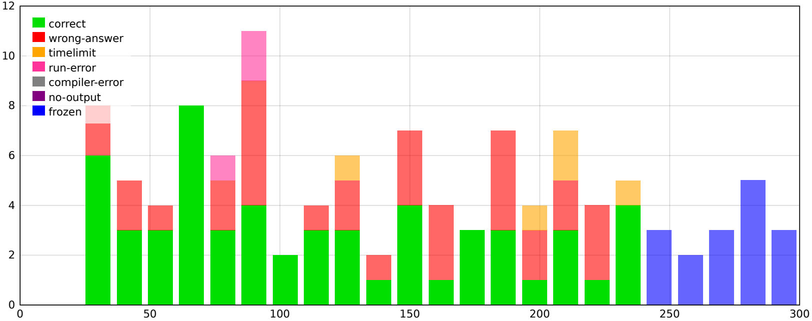
- If $abs(x_1 - x_2) + abs(y_1 - y_2) > d_1 + d_2$, the players are too far away from each other -> `impossible`.
- Each move changes the player's x or y coordinate by 1. Therefore, $x_1 + y_1 + d_1 \equiv x_2 + y_2 + d_2 \ mod \ 2$ must be satisfied, otherwise `impossible`.
- No other condition leads to an `impossible`.

Finding a valid target cell

Consider the smaller of the two ranges. The corner closest to the other player's starting point is a valid cell.



E – Election Meddling



E – Election Meddling

Problem

Given an election with voting districts and majority rule, how many voters have to be bribed such that our party wins a majority of districts?

E – Election Meddling

Problem

Given an election with voting districts and majority rule, how many voters have to be bribed such that our party wins a majority of districts?

Solution

- For each district simulate how many votes are needed to achieve majority by taking one vote at a time from the currently highest voted party and adding it to our total.
- Greedily take the districts that need the fewest votes until our party has won the majority of districts.

E – Election Meddling

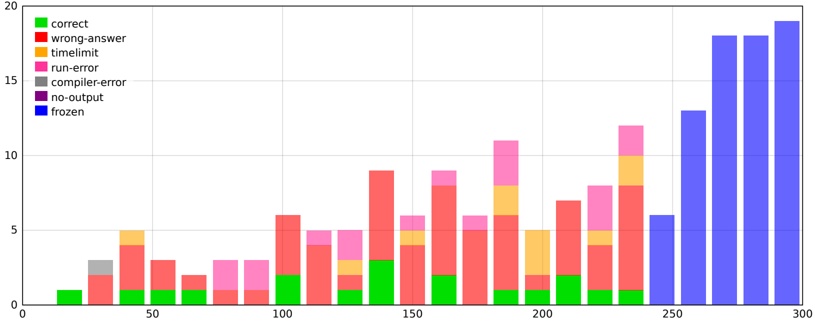
Problem

Given an election with voting districts and majority rule, how many voters have to be bribed such that our party wins a majority of districts?

Solution

- For each district simulate how many votes are needed to achieve majority by taking one vote at a time from the currently highest voted party and adding it to our total.
- Greedily take the districts that need the fewest votes until our party has won the majority of districts.
- Can be sped up by taking enough votes at a time from the currently highest voted parties until they are equal to the next highest party. Repeat until our party has the majority of votes.

I – Insertion Order



I – Insertion Order

Problem

Insert the numbers $\{1, \ldots, n\}$ into an empty binary search tree in such an order that the final tree has height exactly k.

I – Insertion Order

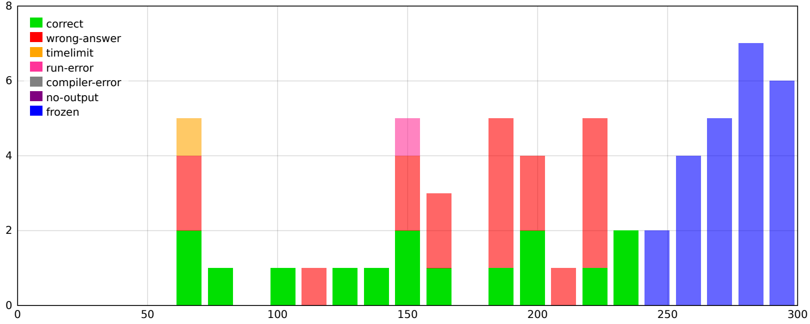
Problem

Insert the numbers $\{1, \ldots, n\}$ into an empty binary search tree in such an order that the final tree has height exactly k.

Solution

- A solution exists if and only if $k \leq n < 2^k$.
- First, build a binary tree with n nodes and height k:
  - Start with a degenerate tree that is just a path of length k.
  - Then add nodes, making sure not to exceed height k.
- Label the tree nodes $1, \ldots, n$ from left to right.
- Output the labels from top to bottom.
- Many other approaches are possible.

G – Game of Falling Blocks



G – Game of Falling Blocks

Problem

Write a Tetris AI that is able to clear at least one row.

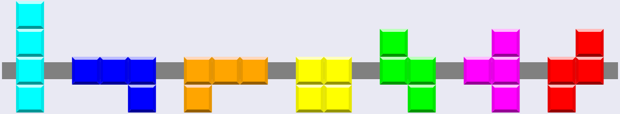
G – Game of Falling Blocks

Problem

Write a Tetris AI that is able to clear at least one row.

Insight

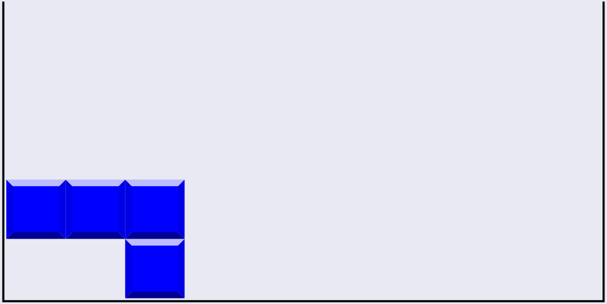
- In this simplified game it is not possible to slide pieces under those already placed.
- So if the first piece is an S or Z, it becomes impossible to clear the first row.
- However, with the right orientation of pieces, it becomes easy to clear the second row instead:



G – Game of Falling Blocks

Solution

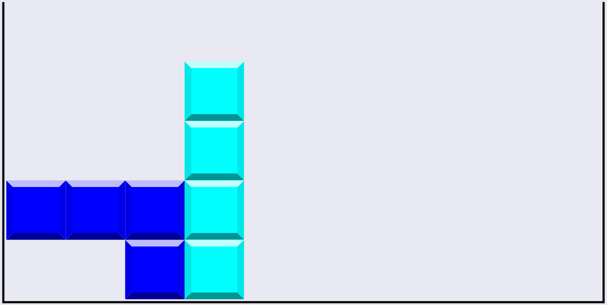
- Build the second row from left to right, always rotating the pieces appropriately.



G – Game of Falling Blocks

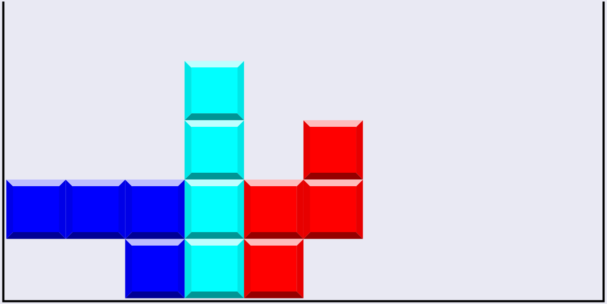
Solution

- Build the second row from left to right, always rotating the pieces appropriately.



Solution

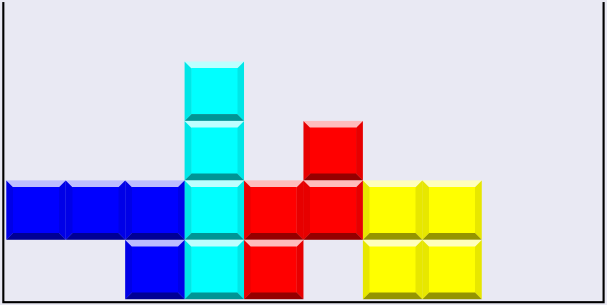
- Build the second row from left to right, always rotating the pieces appropriately.



G – Game of Falling Blocks

Solution

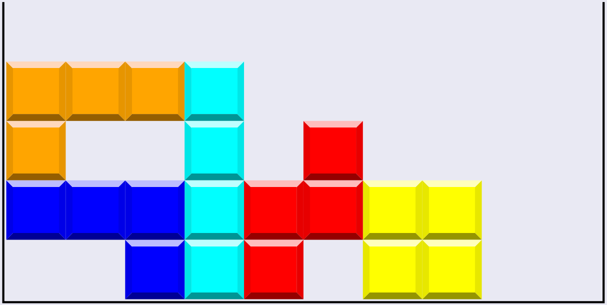
- Build the second row from left to right, always rotating the pieces appropriately.



G – Game of Falling Blocks

Solution

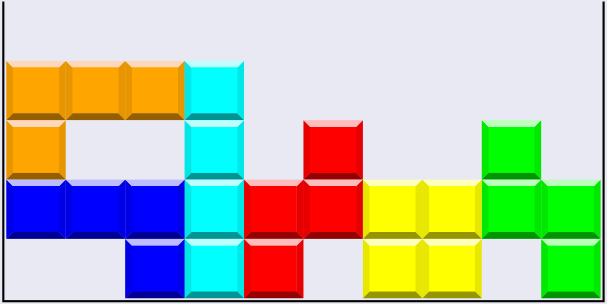
- Build the second row from left to right, always rotating the pieces appropriately.
- If the current piece is too wide, drop it on the left instead.



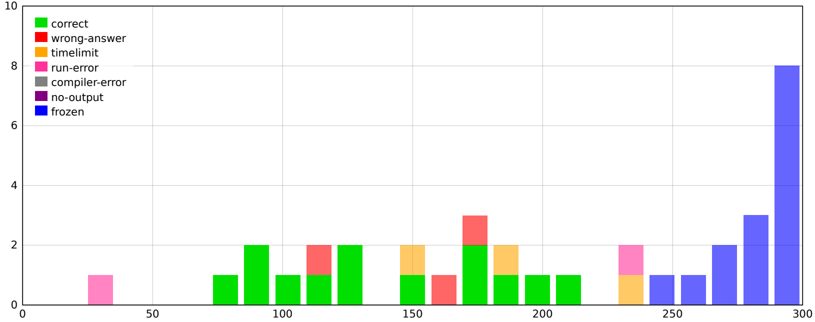
G – Game of Falling Blocks

Solution

- Build the second row from left to right, always rotating the pieces appropriately.
- If the current piece is too wide, drop it on the left instead.



F – Final Standings



F – Final Standings

Problem

Given a frozen contest scoreboard, find the probability that your team won, assuming you know for each pending submission the probability that it was accepted.

F – Final Standings

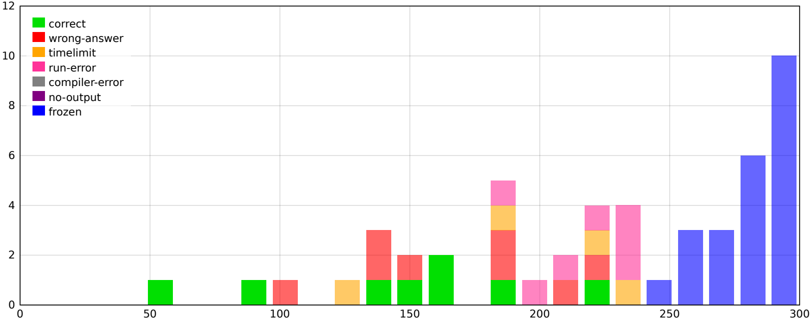
Problem

Given a frozen contest scoreboard, find the probability that your team won, assuming you know for each pending submission the probability that it was accepted.

Solution

- For a team i and number of problems k, let $f(i, k)$ be the probability that the team solved exactly k problems.
- The values of $f(i, k)$ can be computed with dynamic programming (DP): consider the problems one at a time and update the probabilities.
- For each team, find the probability that it does not solve more problems than you.
- The final answer is the product of those probabilities.

B – Bouldering



B – Bouldering

Problem

Scale a bouldering wall without running out of stamina.

B – Bouldering

Problem

Scale a bouldering wall without running out of stamina.

Observation

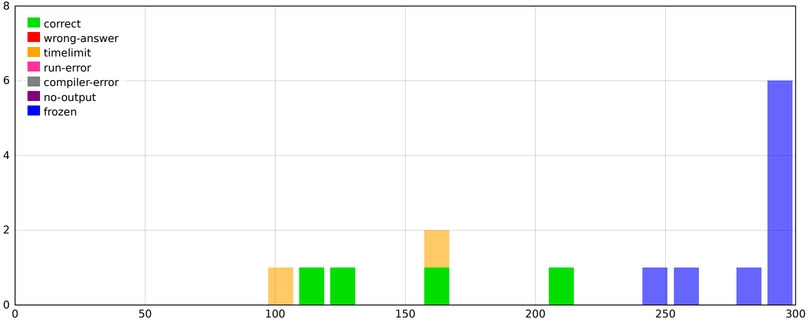
s, the required stamina, is at most $h \times w \times 9 = 5\,625$

B – Bouldering

Solution

- Build graph nodes from the reachable holds of the wall
- Copy every node s times to count the remaining stamina at this hold.
- For an edge from hold u to v taking t units of stamina, insert an edge from any copy of node u to the copy of node v with exactly t less stamina remaining.
- Use Dijkstra's algorithm for shortest paths.

C – Colourful Chameleons



C – Colourful Chameleons

Problem

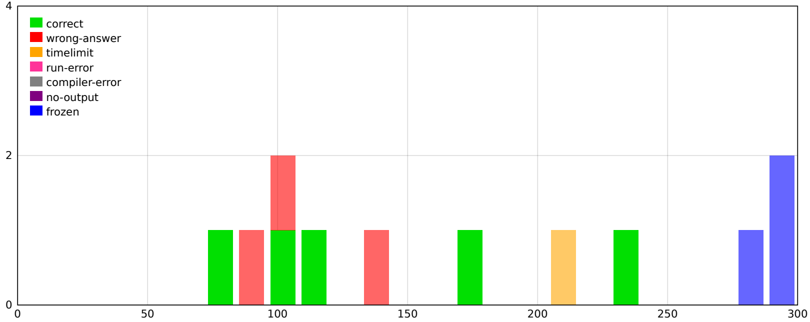
Given chameleons of n different colours, determine the minimal number of breeding steps necessary so that all chameleons have the same colour c. A single breeding step transforms $n-1$ chameleons of pairwise distinct colours to y chameleons of the missing colour.

C – Colourful Chameleons

Solution

- Let $z = x_i - x_j$ be the difference between the number chameleons of the ith and jth colour.
- If colour k is bred, then z does not change.
- If colour i or j is bred, then z changes by $y + 1$.
- Consequence: Initially, all x_i (except for x_c) must leave the same remainder modulo $y + 1$.
- At least $\max_{i \neq c} x_i$ breeding steps necessary.
- Due to the constraints, at most $\max_{i \neq c} x_i$ breeding steps necessary.
- Compute the total number of chameleons in the end using the number of breeding steps.

K – Keeping the Dogs Out



K – Keeping the Dogs Out

Problem

Given quadratic pieces whose side lengths are powers of two. Find a and b so that they can be arranged in a rectangle of size $a \times b$.

K – Keeping the Dogs Out

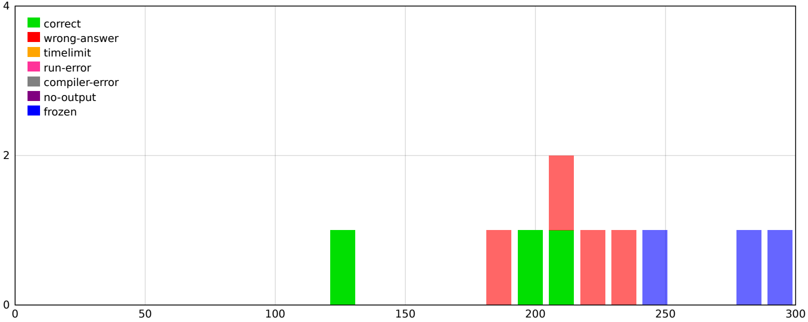
Problem

Given quadratic pieces whose side lengths are powers of two. Find a and b so that they can be arranged in a rectangle of size $a \times b$.

Solution

- Calculate sum of area of all pieces
- Assume $a \leq b$ and try out all possible a with a simple loop.
- Check if all pieces of size $\geq 2^i$ fit, for decreasing i:
  - Round a and b down to multiples of 2^i.
  - Calculate area sum of all pieces of size $\geq 2^i$.
  - If the sum is larger than the rounded-down rectangle, a rectangle of size $a \times b$ is not possible.
- If the previous check did not find any conflict, it is always possible to arrange the pieces in a rectangle of size $a \times b$.

D – Dungeon Crawler



D – Dungeon Crawler

Problem

Check if two edge-labelled graphs (`Map` and `Level`) are identical ($\rightsquigarrow$ Graph isomorphism for special graphs).

D – Dungeon Crawler

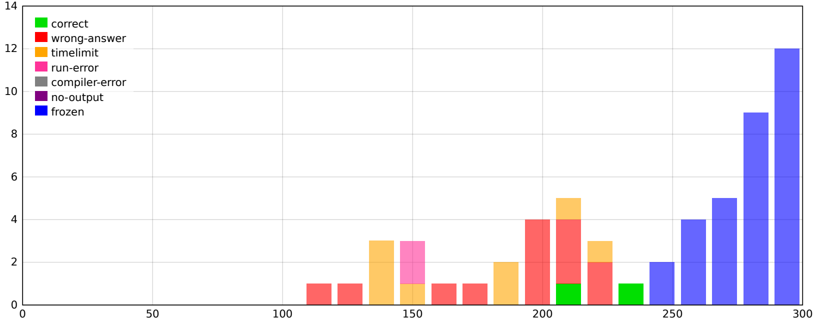
Problem

Check if two edge-labelled graphs (`Map` and `Level`) are identical ($\rightsquigarrow$ Graph isomorphism for special graphs).

Solution

- Traverse `Level` in DFS order to obtain the complete graph (usual backtracking with explicit *Walk* instructions emitted).
- Find a bijective mapping between nodes in `Map` and `Level`:
  - For each node in `Map`, traverse `Map` and `Level` in parallel to construct a mapping.
  - Detect inconsistencies in mappings (different edges for a node, non-bijective mapping, ...).
- Number of remaining mappings determines the result.
- Pitfalls: All nodes have the same outgoing edges, but still a unique mapping is possible; `Level` can be really large.

H – Historical Maths



H – Historical Maths

Problem

Given a multiplication $f_1 \cdot f_2 = p$ of unknown base b, what could a possible b be?

H – Historical Maths

Problem

Given a multiplication $f_1 \cdot f_2 = p$ of unknown base b, what could a possible b be?

Insight

- If f_1 and f_2 are multiplied in a base greater than b, the resulting product is always smaller than p in this base.
- If f_1 and f_2 are multiplied in a base smaller than b, the resulting product is always greater than p in this base.
- Since all digits are smaller than or equal to 2^{30} and we have at most $1\,000$ digits, b must be smaller than 2^{61}.

H – Historical Maths

Solution

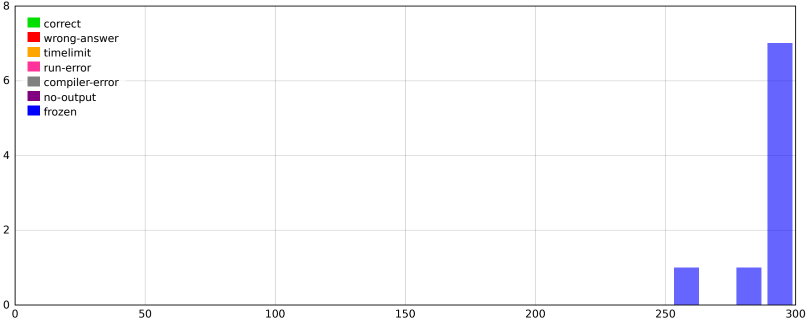
- Write a multiplication routine that can multiply two numbers in a given base.
- Use binary search to determine the correct base.

H – Historical Maths

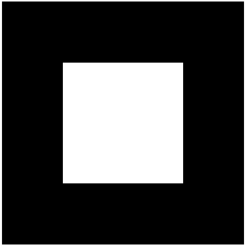
Solution

- Write a multiplication routine that can multiply two numbers in a given base.
- Use binary search to determine the correct base.
- Factorisation approach only leads to accepted solution if a fast factorisation algorithm is used.

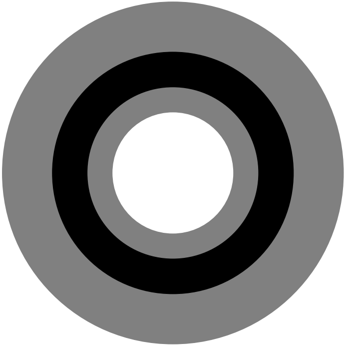
L – Long-Exposure Photography



L – Long-Exposure Photography



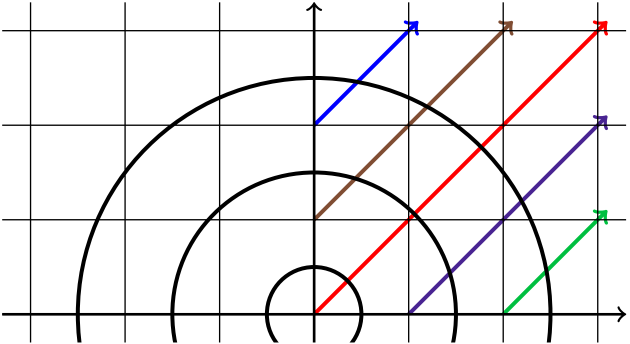
->



Problem

A white surface with n black rectangles is rotated around its origin fast. A photo is taken of the entire rotation. Determine the 'black' surface area of the photo (parts covered by rectangles throughout the entire rotation) as well as the 'grey' surface area (parts covered by rectangles through some but not all of the rotation).

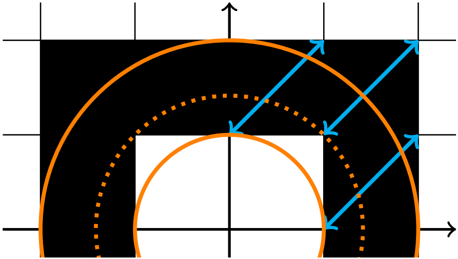
L – Long-Exposure Photography



Solution (Variant 1)

- As a circle's radius increases, the number of 1×1 squares intersected by its outline only increases for every new integer.
- Number of intersected 1×1 squares for radius r is $8 \cdot \lfloor r \rfloor + 4$.
- Squares can be grouped by the 'diagonal' they lie on.

L – Long-Exposure Photography



Solution (Variant 1)

- Each rectangle is essentially a set of intervals, one interval per affected diagonal.
- For every integer radius, use the interval entry and exit points (up to the next integer) to compare the covered diagonals to the required number of 1×1 squares.
- Add surface areas of the resulting black and grey ‘rings’ to the respective total.

L – Long-Exposure Photography

Solution (Variant 2)

- Do coordinate compression, creating an approximately $n \times n$ grid.
- Each rectangle covers some fields in the grid. Mark those as black. All unmarked fields are white. ($O(nwh)$)
- Fields reach minimal/maximal distance to the centre at a vertex or at $x = 0$ or $y = 0$. Thus, we can calculate the distance interval this field covers.
- Sort and merge black and white intervals ($O(n^2 \log(n))$ for n^2 intervals). If black and white intervals overlap, the overlapping part is grey.

L – Long-Exposure Photography

Solution (Variant 2)

- Do coordinate compression, creating an approximately $n \times n$ grid.
- Each rectangle covers some fields in the grid. Mark those as black. All unmarked fields are white. ($O(nwh)$)
- Fields reach minimal/maximal distance to the centre at a vertex or at $x = 0$ or $y = 0$. Thus, we can calculate the distance interval this field covers.
- Sort and merge black and white intervals ($O(n^2 \log(n))$ for n^2 intervals). If black and white intervals overlap, the overlapping part is grey.

Optimization

The second step can be optimized to run in $O(n^2 \log(n))$ using a sweepline approach with a segment tree. This was not required.