Winter Contest 2024 Presentation of Solutions

The Winter Contest Jury

January 29, 2024

Winter Contest 2024 Jury

- **Philipp Fischbeck**
  Hasso-Plattner-Institute Potsdam
- **Rudolf Fleischer**
  Heinrich-Heine-University Düsseldorf, CPUlm
- **Brutenis Gliwa**
  University of Rostock
- **Niko Hastrich**
  Hasso-Plattner-Institute Potsdam
- **Florian Kothmeier**
  Friedrich-Alexander University
  Erlangen-Nürnberg
- **Felicia Lucke**
  Fribourg University CH, CPUlm
- **Jannik Olbrich**
  Ulm University, CPUlm
- **Erik Sünderhauf**
  Technical University of Munich
- **Christopher Weyand**
  Karlsruhe Institute of Technology, CPUlm
- **Paul Wild**
  Friedrich-Alexander University
  Erlangen-Nürnberg, CPUlm
- **Wendy Yi**
  Karlsruhe Institute of Technology
- **Michael Zündorf**
  Karlsruhe Institute of Technology, CPUlm

Winter Contest 2024 Test Solvers

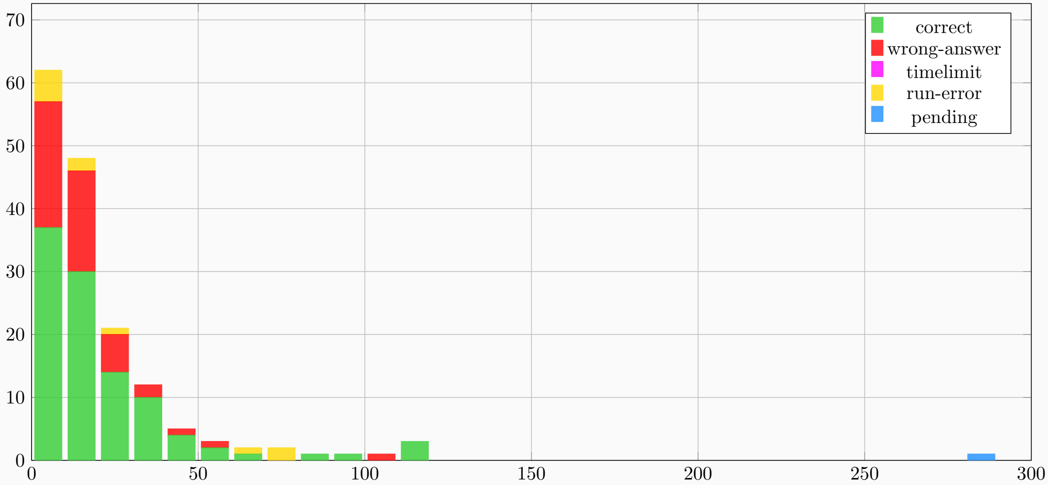
- **Sebastian Angrick**
  Hasso-Plattner-Institute Potsdam
- **Michael Ruderer**
  Augsburg University, CPUlm
- **Jonas Schmidt**
  Hasso-Plattner-Institute Potsdam

Winter Contest 2024 Technical Team

- **Nathan Maier**
  CPUlm
- **Alexander Schmid**
  CPUlm
- **Pascal Weber**
  University of Vienna, CPUlm

A: Alphabetical Athletes

Problem Author: Felicia Lucke



A: Alphabetical Athletes

Problem Author: Felicia Lucke

Problem

Given a German word, check if its letters are lexicographically sorted (increasing or decreasing).

A: Alphabetical Athletes

Problem Author: Felicia Lucke

Problem

Given a German word, check if its letters are lexicographically sorted (increasing or decreasing).

Solution

- Sort the word and check if it is equal to the input or the reversed input.

A: Alphabetical Athletes

Problem Author: Felicia Lucke

Problem

Given a German word, check if its letters are lexicographically sorted (increasing or decreasing).

Solution

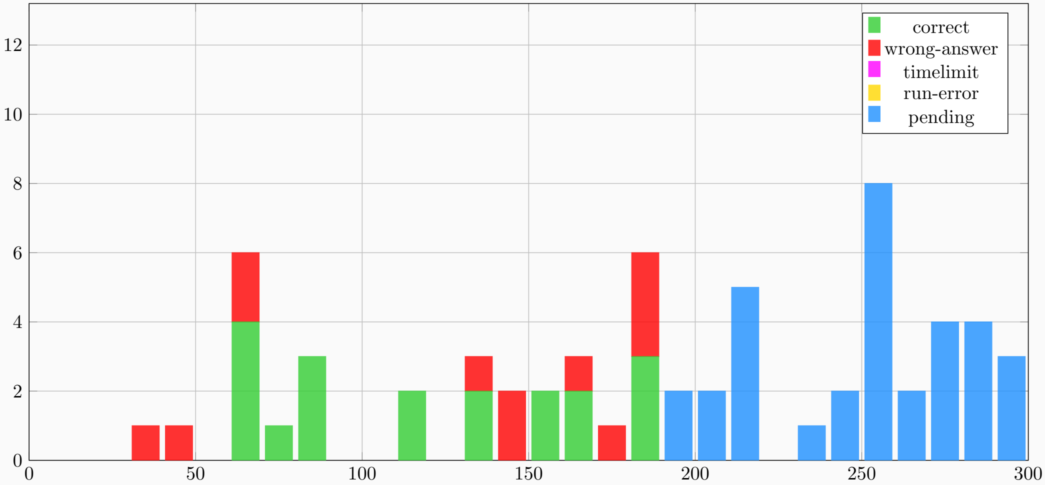
- Sort the word and check if it is equal to the input or the reversed input.

Possible Pitfalls

- The first letter may be capitalized.
- Reversed alphabetical order is considered sorted.
- Did not test all samples.

B: Bright Beacons

Problem Author: Brutenis Gliwa



B: Bright Beacons

Problem Author: Brutenis Gliwa

Problem

Given a grid of mountain heights, what is the shortest path from the top-left to the bottom-right when adjacency is determined by line-of-sight between mountains?

B: Bright Beacons

Problem Author: Brutenis Gliwa

Problem

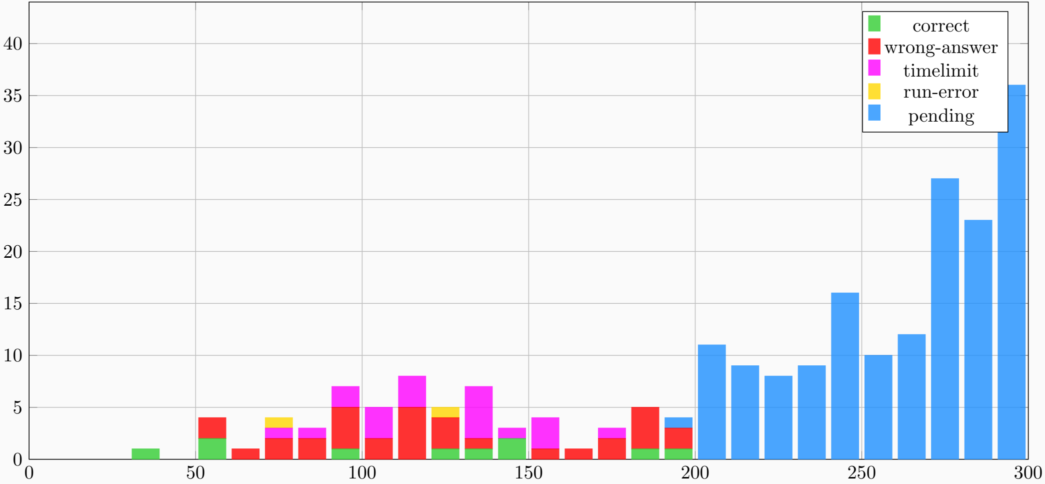
Given a grid of mountain heights, what is the shortest path from the top-left to the bottom-right when adjacency is determined by line-of-sight between mountains?

Solution

- Compute line of sight function $f(x) : ax + b$ for each pair of mountains along the same row or column ($f(x)$ crosses both peaks).
- There is no line of sight if any mountain in between is higher than $f(x)$ at that position.
- Create a graph: each mountain is a node, add edge between mountains if there is a line of sight.
- Traverse graph with breadth-first-search.

C: Chess Challenge

Problem Author: Wendy Yi



C: Chess Challenge

Problem Author: Wendy Yi

Problem

There is a 1D chess board with r rooks. Each rook has a number of captures it can make at most. Find a capture sequence (if possible) such that there is only one rook left in the end.

C: Chess Challenge

Problem Author: Wendy Yi

Problem

There is a 1D chess board with r rooks. Each rook has a number of captures it can make at most. Find a capture sequence (if possible) such that there is only one rook left in the end.

Observations

- It is possible if and only if total number of allowed moves $\geq r - 1$.
- If a rook with 0 moves left can be captured by a neighbour, capturing it does not change solvability.

C: Chess Challenge

Problem Author: Wendy Yi

Problem

There is a 1D chess board with r rooks. Each rook has a number of captures it can make at most. Find a capture sequence (if possible) such that there is only one rook left in the end.

Observations

- It is possible if and only if total number of allowed moves $\geq r - 1$.
- If a rook with 0 moves left can be captured by a neighbour, capturing it does not change solvability.

Solution

- Check total number of moves.

C: Chess Challenge

Problem Author: Wendy Yi

Problem

There is a 1D chess board with r rooks. Each rook has a number of captures it can make at most. Find a capture sequence (if possible) such that there is only one rook left in the end.

Observations

- It is possible if and only if total number of allowed moves $\geq r - 1$.
- If a rook with 0 moves left can be captured by a neighbour, capturing it does not change solvability.

Solution

- Check total number of moves.
- Process rooks from left to right using a stack.

C: Chess Challenge

Problem Author: Wendy Yi

Problem

There is a 1D chess board with r rooks. Each rook has a number of captures it can make at most. Find a capture sequence (if possible) such that there is only one rook left in the end.

Observations

- It is possible if and only if total number of allowed moves $\geq r - 1$.
- If a rook with 0 moves left can be captured by a neighbour, capturing it does not change solvability.

Solution

- Check total number of moves.
- Process rooks from left to right using a stack.
  1. While 0-rook on stack, new non-0-rook: new rook takes rook on stack

C: Chess Challenge

Problem Author: Wendy Yi

Problem

There is a 1D chess board with r rooks. Each rook has a number of captures it can make at most. Find a capture sequence (if possible) such that there is only one rook left in the end.

Observations

- It is possible if and only if total number of allowed moves $\geq r - 1$.
- If a rook with 0 moves left can be captured by a neighbour, capturing it does not change solvability.

Solution

- Check total number of moves.
- Process rooks from left to right using a stack.
  1. While 0-rook on stack, new non-0-rook: new rook takes rook on stack
  2. While non-0-rook on stack, new 0-rook: rook on stack takes new rook

C: Chess Challenge

Problem Author: Wendy Yi

Problem

There is a 1D chess board with r rooks. Each rook has a number of captures it can make at most. Find a capture sequence (if possible) such that there is only one rook left in the end.

Observations

- It is possible if and only if total number of allowed moves $\geq r - 1$.
- If a rook with 0 moves left can be captured by a neighbour, capturing it does not change solvability.

Solution

- Check total number of moves.
- Process rooks from left to right using a stack.
  1. While 0-rook on stack, new non-0-rook: new rook takes rook on stack
  2. While non-0-rook on stack, new 0-rook: rook on stack takes new rook
  3. Else: push new rook on stack

C: Chess Challenge

Problem Author: Wendy Yi

Problem

There is a 1D chess board with r rooks. Each rook has a number of captures it can make at most. Find a capture sequence (if possible) such that there is only one rook left in the end.

Observations

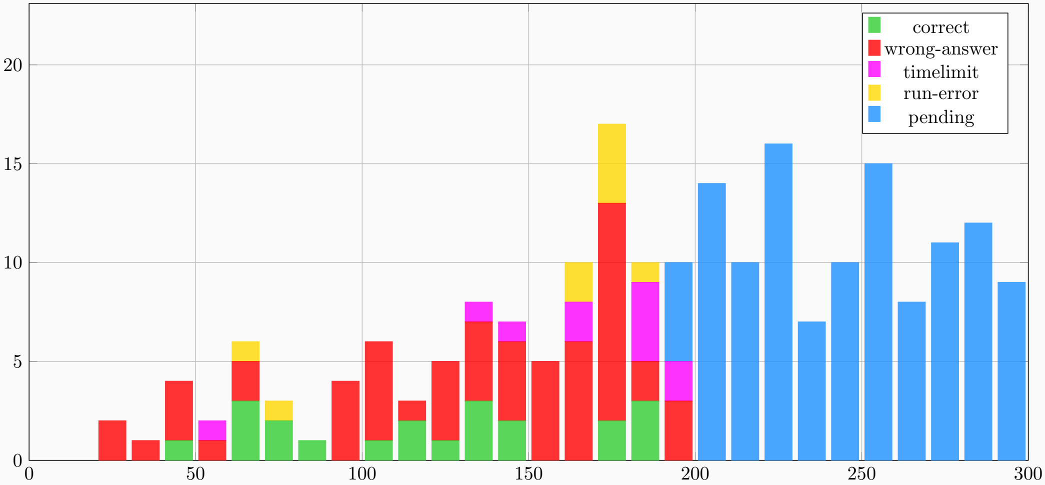
- It is possible if and only if total number of allowed moves $\geq r - 1$.
- If a rook with 0 moves left can be captured by a neighbour, capturing it does not change solvability.

Solution

- Check total number of moves.
- Process rooks from left to right using a stack.
  1. While 0-rook on stack, new non-0-rook: new rook takes rook on stack
  2. While non-0-rook on stack, new 0-rook: rook on stack takes new rook
  3. Else: push new rook on stack
- If no rooks with 0 moves left, repeatedly capture leftmost rook.

D: Devious Dates

Problem Author: Jannik Olbrich



D: Devious Dates

Problem Author: Jannik Olbrich

Problem

Given three integers a, m and k. Find k distinct pairs of integers (a_i, m_i), such that for each i there are x_i, y_i such that

$$a = a_i + x_i \cdot m_i$$
$$a + m = a_i + y_i \cdot m_i$$

D: Devious Dates

Problem Author: Jannik Olbrich

Problem

Given three integers a, m and k. Find k distinct pairs of integers (a_i, m_i), such that for each i there are x_i, y_i such that

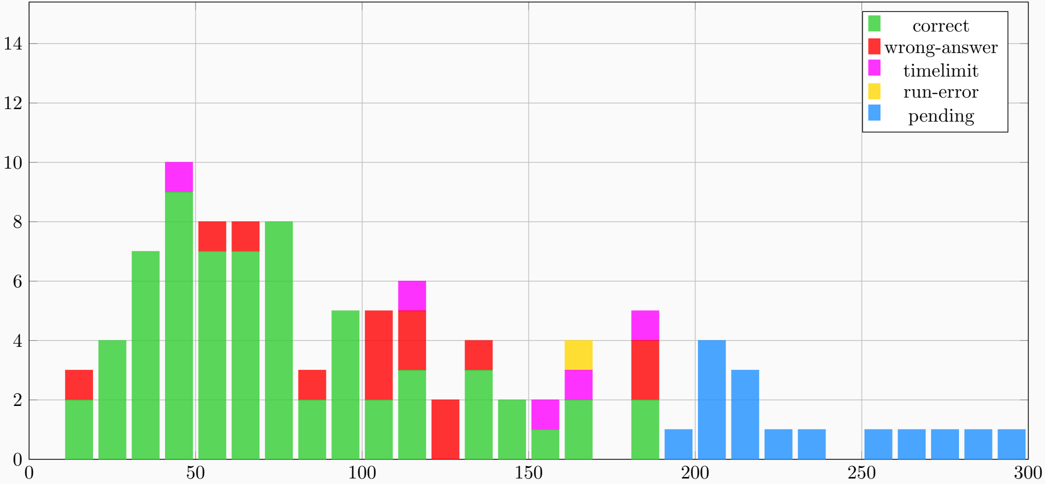
$$a = a_i + x_i \cdot m_i$$
$$a + m = a_i + y_i \cdot m_i$$

Solution

- From subtracting the two equations, we know that m_i must divide $(a + m) - a = m$.
- Once m_i is known, the smallest a_i is $a \bmod m_i$.
- Two schedules $(a_i, m_i), (a_j, m_j)$ are different iff $m_i \neq m_j$.
  $\implies$ There are exactly as many different schedules as there are divisors of m.
  $\implies$ Find all divisors of m, print "`impossible`" if there are fewer than k, otherwise choose k divisors as m_i's (whose lcm is m) and print them.
- Time complexity: $\mathcal{O}(\sqrt{m})$

E: Euroexpress

Problem Author: Michael Zündorf

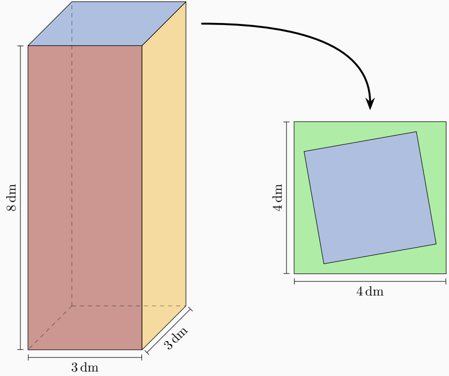


E: Euroexpress

Problem Author: Michael Zündorf

Problem

Given n rectangles (w_i, h_i), find the largest box where each side can be covered by one of the rectangles.



E: Euroexpress

Problem Author: Michael Zündorf

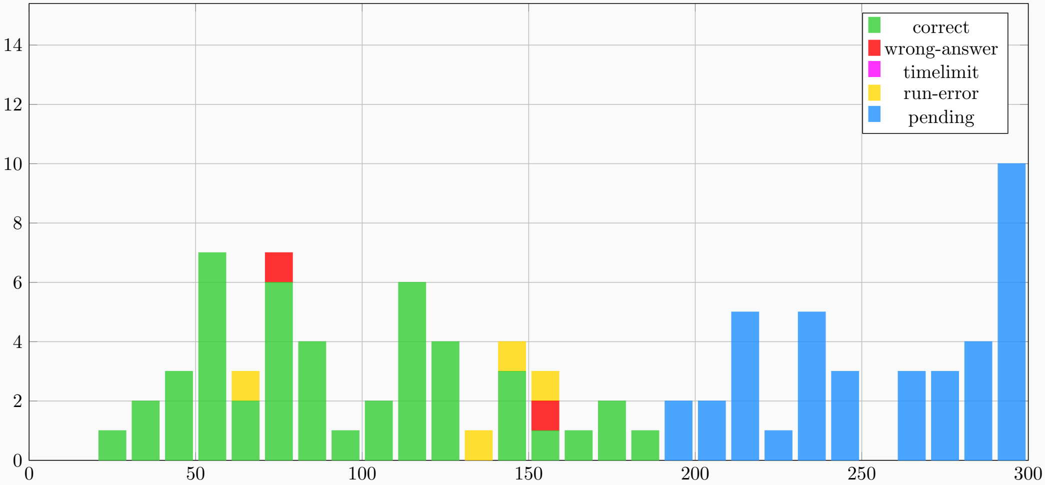
Solution

- All sides of the largest box can always be covered with the same rectangle.
- For a given rectangle, the largest box has size $w \times h \times \min(w, h)$.
- Try all rectangles and take the maximum over all.

$\Rightarrow$ Runtime: $\mathcal{O}(n)$

F: Football Figurines

Problem Author: Rudolf Fleischer



F: Football Figurines

Problem Author: Rudolf Fleischer

Problem

- Given are n floors where stairs go either one or two levels up, and m queries that consist of two floors each.
- For each query, compute the total number of staircases used on all possible different routes between the two queried floors modulo $10^9 + 7$.

F: Football Figurines

Problem Author: Rudolf Fleischer

Problem

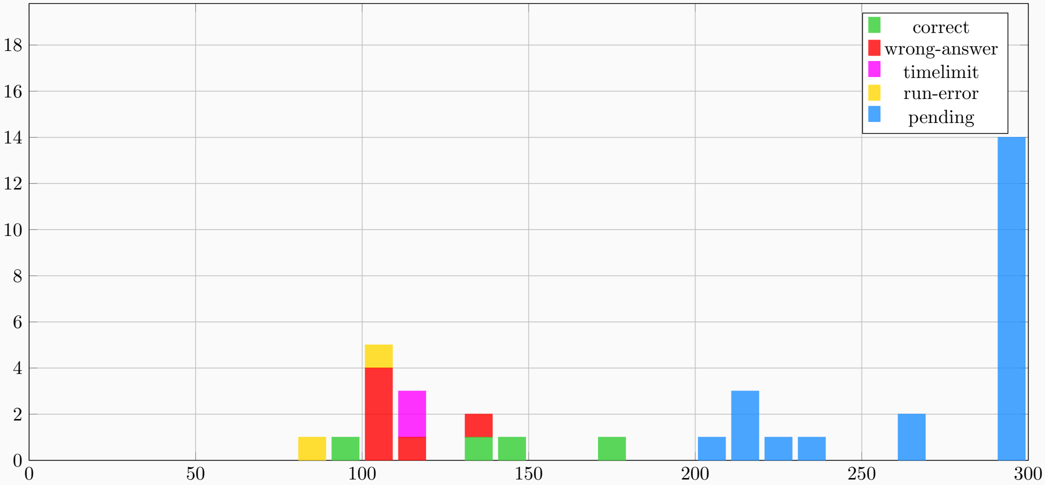
- Given are n floors where stairs go either one or two levels up, and m queries that consist of two floors each.
- For each query, compute the total number of staircases used on all possible different routes between the two queried floors modulo $10^9 + 7$.

Solution

- The number of routes to climb up k floors is the kth Fibonacci number F_k.
- The total number of staircases used is $L_k = L_{k-1} + L_{k-2} + F_k$, where $L_0 = 0$ and $L_1 = 1$.

G: Genius Gamer

Problem Author: Niko Hastrich



G: Genius Gamer

Problem Author: Niko Hastrich

Problem

Given tiles with a color and a numerical value (without duplicates), decide wether they can be partitioned into sets of size at least three that either

- share the same numerical value (group), or
- share the same colour and have consecutive numerical values (run).

G: Genius Gamer

Problem Author: Niko Hastrich

Problem

Given tiles with a color and a numerical value (without duplicates), decide wether they can be partitioned into sets of size at least three that either

- share the same numerical value (group), or
- share the same colour and have consecutive numerical values (run).

Solution

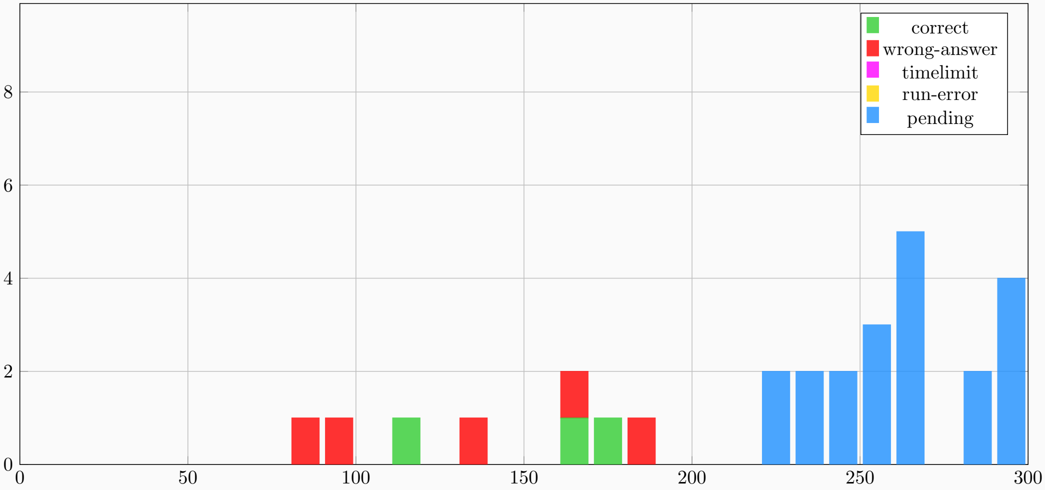
- Solvable via dynamic programming.

$$DP[i][a][b][c][d] = \begin{array}{l}\text{Is it possible to partition the pieces with value at most } i\text{, such that in the}\\ \text{first colour there ends a run of size } a\text{, in the second of size } b\text{, in the third}\\ \text{of size } c\text{, and in the last of size } d \text{ with the tile of value } i.\end{array}$$

- For a, b, c and d only states $\{0, 1, 2, \text{“}\geq 3\text{”}\}$ are interesting.
- Needs $\mathcal{O}(4^4 \max(\text{numerical value}))$ states, with amortized constant time transition.
- Due to small constraints alternative solutions possible (e.g. back-tracking, meet-in-the-middle).

H: Haggling over Hours

Problem Author: Felicia Lucke



H: Haggling over Hours

Problem Author: Felicia Lucke

Problem

Given a set of intervals, what is the smallest number of intervals to delete if you want to reduce the size of the maximum independent set (MIS) by at least 1.

H: Haggling over Hours

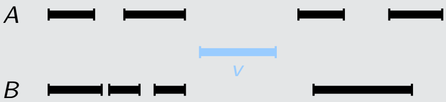
Problem Author: Felicia Lucke

Problem

Given a set of intervals, what is the smallest number of intervals to delete if you want to reduce the size of the maximum independent set (MIS) by at least 1.

Observation

- An inteval v in some MIS has the same number of intervals to the left of it in every MIS.



H: Haggling over Hours

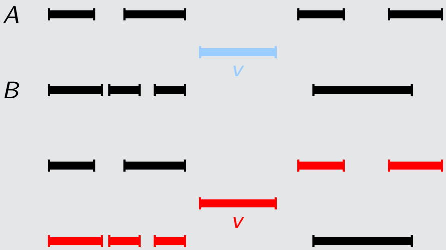
Problem Author: Felicia Lucke

Problem

Given a set of intervals, what is the smallest number of intervals to delete if you want to reduce the size of the maximum independent set (MIS) by at least 1.

Observation

- An inteval v in some MIS has the same number of intervals to the left of it in every MIS.



H: Haggling over Hours

Problem Author: Felicia Lucke

Step 1: Find all intervals contained in some MIS

- For interval v, let left(v) be the size of the MIS to the left of v, similar for right(v).
- Calculate left(v) and right(v) for all intervals using dynamic programming.
- All intervals where left(v)+1+right(v) is maximum are contained in a maximum independent set.

H: Haggling over Hours

Problem Author: Felicia Lucke

Step 1: Find all intervals contained in some MIS

- For interval v, let left(v) be the size of the MIS to the left of v, similar for right(v).
- Calculate left(v) and right(v) for all intervals using dynamic programming.
- All intervals where left(v)$+1+$right(v) is maximum are contained in a maximum independent set.

Observation

- For an interval v in an MIS, we say that pos(v) $=$ left(v) $+1$.
- Two intervals at the same position are always intersecting.

Step 2: construct Digraph

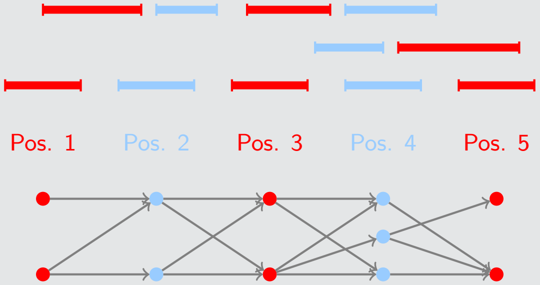
- One vertex per interval contained in some maximum independent set
- Add an arc (u, v) for vertices u and v if their corresponding intervals are at consecutive positions and the intervals do not intersect.
- Add a source s and sink vertex t.

Every maximum independent set corresponds to an (s, t)-path in the graph. The size of a minimum vertex cut is the solution.

Step 2: construct Digraph

- One vertex per interval contained in some maximum independent set
- Add an arc (u, v) for vertices u and v if their corresponding intervals are at consecutive positions and the intervals do not intersect.
- Add a source s and sink vertex t.

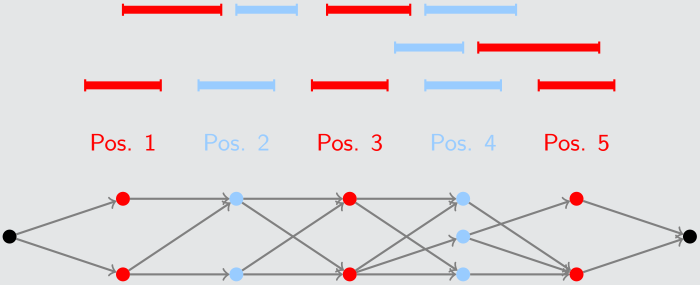
Every maximum independent set corresponds to an (s, t)-path in the graph. The size of a minimum vertex cut is the solution.



Step 2: construct Digraph

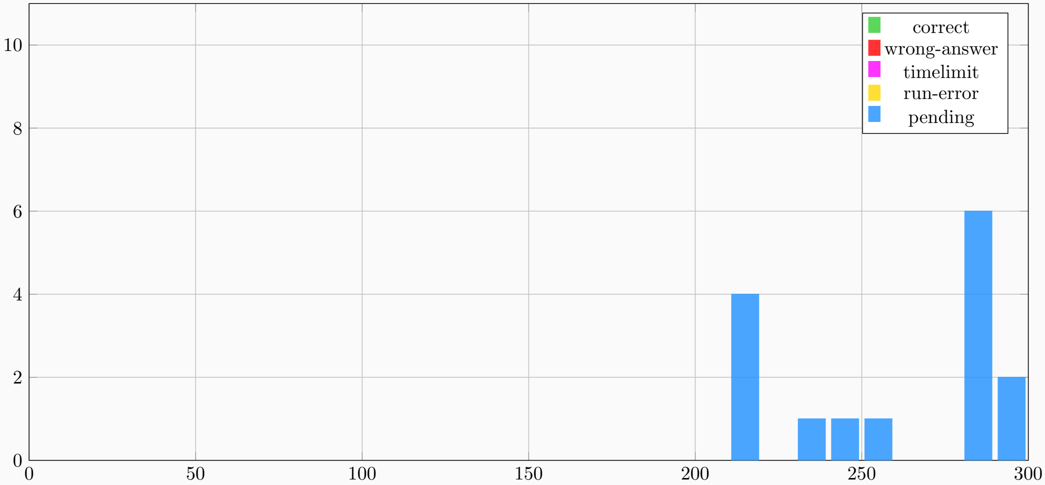
- One vertex per interval contained in some maximum independent set
- Add an arc (u, v) for vertices u and v if their corresponding intervals are at consecutive positions and the intervals do not intersect.
- Add a source s and sink vertex t.

Every maximum independent set corresponds to an (s, t)-path in the graph. The size of a minimum vertex cut is the solution.



I: Impossible Install

Problem Author: Christopher Weyand



I: Impossible Install

Problem Author: Christopher Weyand

Problem

There are software projects and a DAG of dependencies between them. Software projects have versions that specify weakly increasing bounds on the version of each dependency.

Pick a version for each project such that all dependencies are satisfied.

I: Impossible Install

Problem Author: Christopher Weyand

Problem

There are software projects and a DAG of dependencies between them. Software projects have versions that specify weakly increasing bounds on the version of each dependency.

Pick a version for each project such that all dependencies are satisfied.

Solution

- Initialize all versions to 1.
- Repeatedly find a violated dependency and solve it by increasing the version of a project.
- A violation of a dependency where a depends on b is solved like this:
  - $v_b < l_{v_a} \to$ increase v_b
  - $v_b > r_{v_a} \to$ increase v_a
- Runs in $O(W \log n)$ with $W = \sum_p v_p \cdot d_p$ being the amount of input.

I: Impossible Install

Problem Author: Christopher Weyand

Problem

There are software projects and a DAG of dependencies between them. Software projects have versions that specify weakly increasing bounds on the version of each dependency.

Pick a version for each project such that all dependencies are satisfied.

Solution

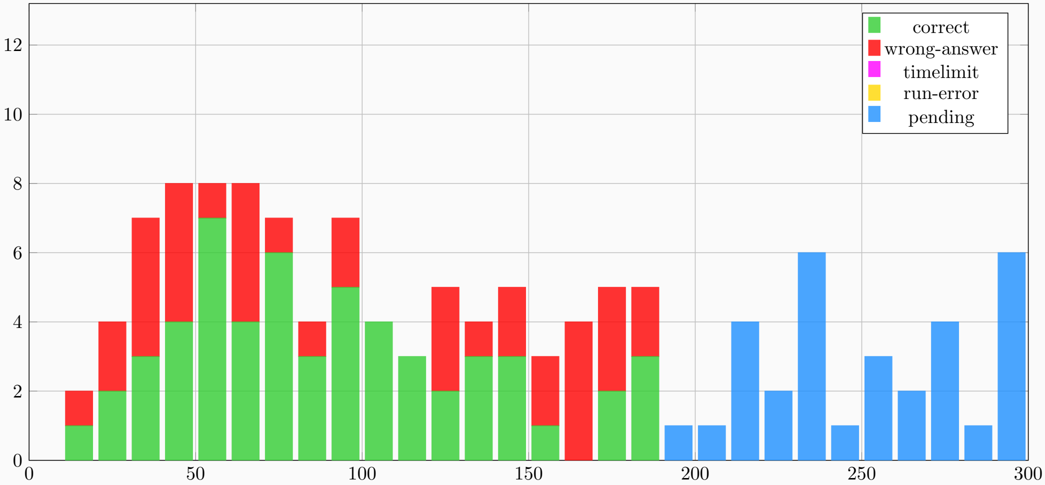
- Initialize all versions to 1.
- Repeatedly find a violated dependency and solve it by increasing the version of a project.
- A violation of a dependency where a depends on b is solved like this:
  - $v_b < l_{v_a} \rightarrow$ increase v_b
  - $v_b > r_{v_a} \rightarrow$ increase v_a
- Runs in $O(W \log n)$ with $W = \sum_p v_p \cdot d_p$ being the amount of input.

Possible Pitfalls

- projects with 10^9 versions and no dependencies

J: Jog in the Fog

Problem Author: Philipp Fischbeck



J: Jog in the Fog

Problem Author: Philipp Fischbeck

Problem

Given an initial position (x, y) and a looping route of n cells (x_i, y_i) on a 2D grid, find the expected time to reach someone running along the route if using the fastest strategy.

J: Jog in the Fog

Problem Author: Philipp Fischbeck

Problem

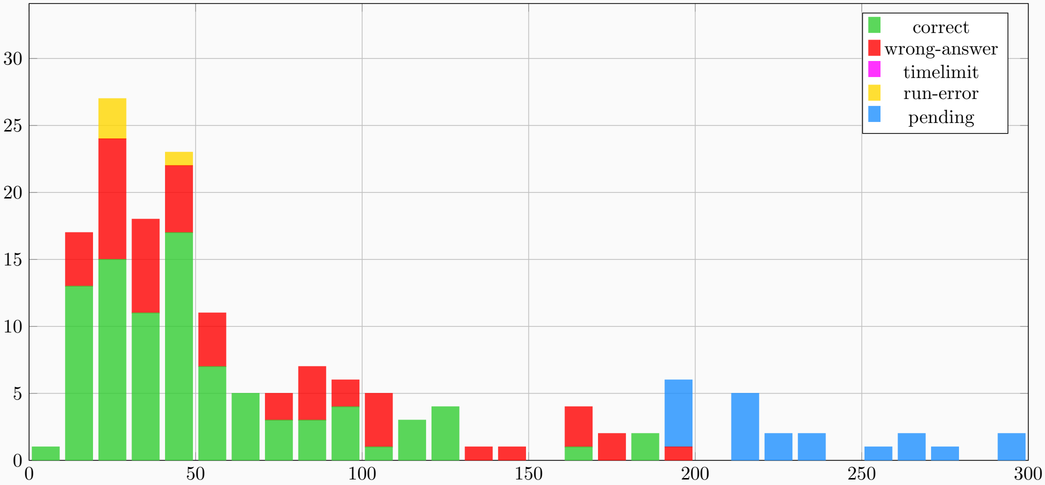
Given an initial position (x, y) and a looping route of n cells (x_i, y_i) on a 2D grid, find the expected time to reach someone running along the route if using the fastest strategy.

Solution

- Optimal strategy: reach the route as fast as possible, then run along the route in opposite direction.
- Reaching the route: $\min_{1 \leq i \leq n} |x - x_i| + |y - y_i|$
- Running along the route: $\frac{1}{n} \sum_{i=1}^{n} \frac{i-1}{2} = \frac{n-1}{4}$

K: Keeping Keys

Problem Author: Brutenis Gliwa



K: Keeping Keys

Problem Author: Brutenis Gliwa

Problem

Pressing a keyboard key costs 1 cent. What is the cost of printing a text consisting of *a-z*, *A-Z* or *spaces* when you are allowed to hold keys?

K: Keeping Keys

Problem Author: Brutenis Gliwa

Problem

Pressing a keyboard key costs 1 cent. What is the cost of printing a text consisting of *a-z*, *A-Z* or *spaces* when you are allowed to hold keys?

Solution

- Handle **SHIFT** and the letters **a-z␣** of the keyboard separately:

K: Keeping Keys

Problem Author: Brutenis Gliwa

Problem

Pressing a keyboard key costs 1 cent. What is the cost of printing a text consisting of *a-z*, *A-Z* or *spaces* when you are allowed to hold keys?

Solution

- Handle **SHIFT** and the letters **a-z␣** of the keyboard separately:
- **SHIFT**: remove spaces, replace a repeating capital letter with a single capital letter

K: Keeping Keys

Problem Author: Brutenis Gliwa

Problem

Pressing a keyboard key costs 1 cent. What is the cost of printing a text consisting of *a-z*, *A-Z* or *spaces* when you are allowed to hold keys?

Solution

- Handle **SHIFT** and the letters **a-z␣** of the keyboard separately:
- **SHIFT**: remove spaces, replace a repeating capital letter with a single capital letter
- **a-z␣**: *.to_lower()* everything, replace repeating letters with a single letter

K: Keeping Keys

Problem Author: Brutenis Gliwa

Problem

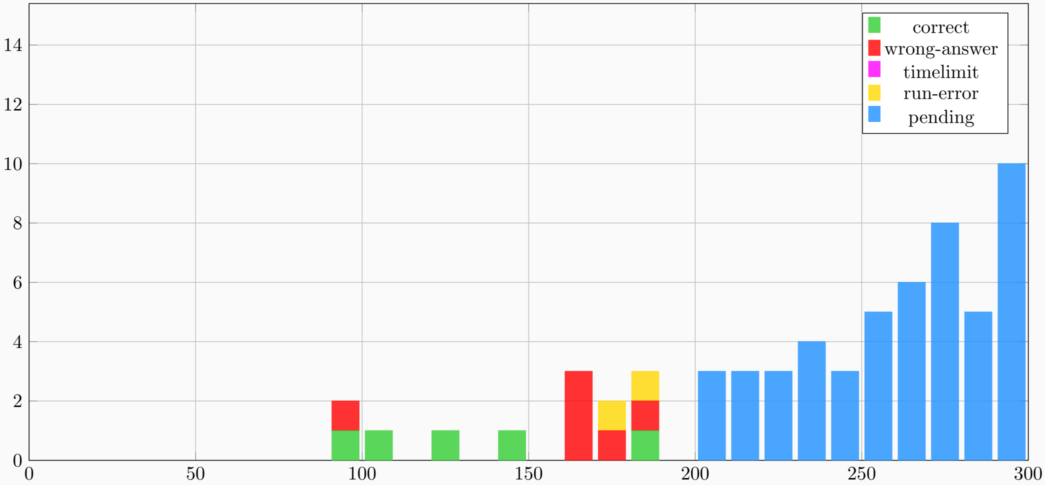
Pressing a keyboard key costs 1 cent. What is the cost of printing a text consisting of *a-z*, *A-Z* or *spaces* when you are allowed to hold keys?

Solution

- Handle **SHIFT** and the letters **a-z␣** of the keyboard separately:
- **SHIFT**: remove spaces, replace a repeating capital letter with a single capital letter
- **a-z␣**: *.to_lower()* everything, replace repeating letters with a single letter
- Print sum of resulting string lengths.

L: Lookup Table Tennis

Problem Author: Paul Wild



L: Lookup Table Tennis

Problem Author: Paul Wild

Problem

Locate a point $p_0 = (x_0, y_0, z_0)$ in the 3D region $[0, n] \times [0, n] \times [0, n]$ using queries of the form

Is p_0 within distance $\sqrt{s}$ of the point $p = (x, y, z)$?

All numbers in the input and output are integers.

L: Lookup Table Tennis

Problem Author: Paul Wild

Problem

Locate a point $p_0 = (x_0, y_0, z_0)$ in the 3D region $[0, n] \times [0, n] \times [0, n]$ using queries of the form

Is p_0 within distance $\sqrt{s}$ of the point $p = (x, y, z)$?

All numbers in the input and output are integers.

Solution 1 – Binary Search

- Pick three arbitrary points and use binary search to find their distances to p_0.
- Intersect the three spheres and query the (at most 2) intersection points.
- Can be made easier by picking suitable points (e.g. three corners of the area).

L: Lookup Table Tennis

Problem Author: Paul Wild

Problem

Locate a point $p_0 = (x_0, y_0, z_0)$ in the 3D region $[0, n] \times [0, n] \times [0, n]$ using queries of the form

Is p_0 within distance $\sqrt{s}$ of the point $p = (x, y, z)$?

All numbers in the input and output are integers.

Solution 1 – Binary Search

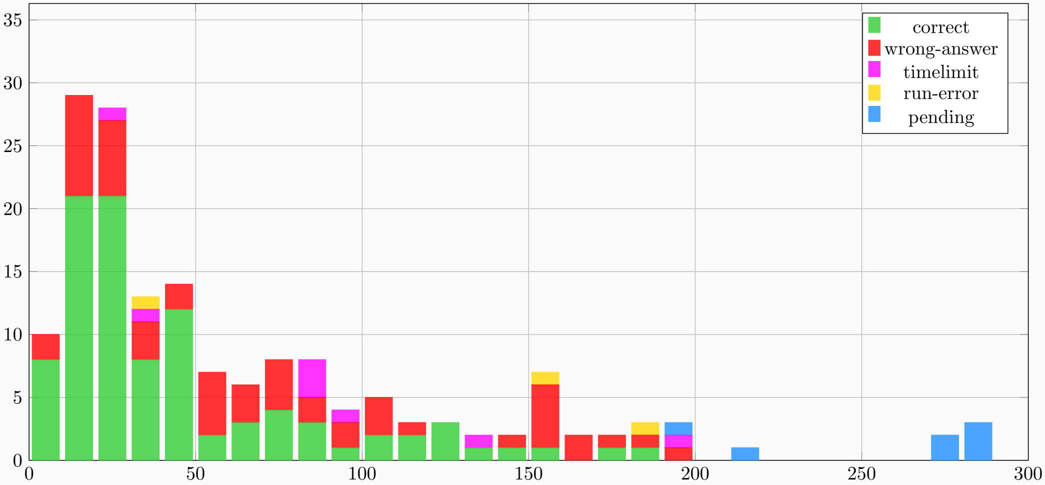
- Pick three arbitrary points and use binary search to find their distances to p_0.
- Intersect the three spheres and query the (at most 2) intersection points.
- Can be made easier by picking suitable points (e.g. three corners of the area).

Solution 2 – Shrinking Bounding Box

- Create a ball whose diameter is half the diameter of the bounding box.
- Place it at random positions in the bounding box until it contains p_0.
- Shrink the bounding box to the query ball. Repeat.

M: Montage Matrix

Problem Author: Florian Kothmeier



M: Montage Matrix

Problem Author: Florian Kothmeier

Problem

Arrange n people in w columns for a photo.

Constraint: Only people with lower height h_i may stand in front of others.

M: Montage Matrix

Problem Author: Florian Kothmeier

Problem

Arrange n people in w columns for a photo.

Constraint: Only people with lower height h_i may stand in front of others.

Solution 1 – Construct Arrangement

- Sort heights from tallest to smallest and rearrange into $w \times \frac{n}{w}$ grid
- For each entry, check that $h_{i,j} > h_{i,j+1}$
- Alternatively: Use only a single row, and replace items when processed

$\Rightarrow$ Runtime $O(n \cdot log(n))$.

M: Montage Matrix

Problem Author: Florian Kothmeier

Problem

Arrange n people in w columns for a photo.

Constraint: Only people with lower height h_i may stand in front of others.

Solution 1 – Construct Arrangement

- Sort heights from tallest to smallest and rearrange into $w \times \frac{n}{w}$ grid
- For each entry, check that $h_{i,j} > h_{i,j+1}$
- Alternatively: Use only a single row, and replace items when processed

$\Rightarrow$ Runtime $O(n \cdot log(n))$.

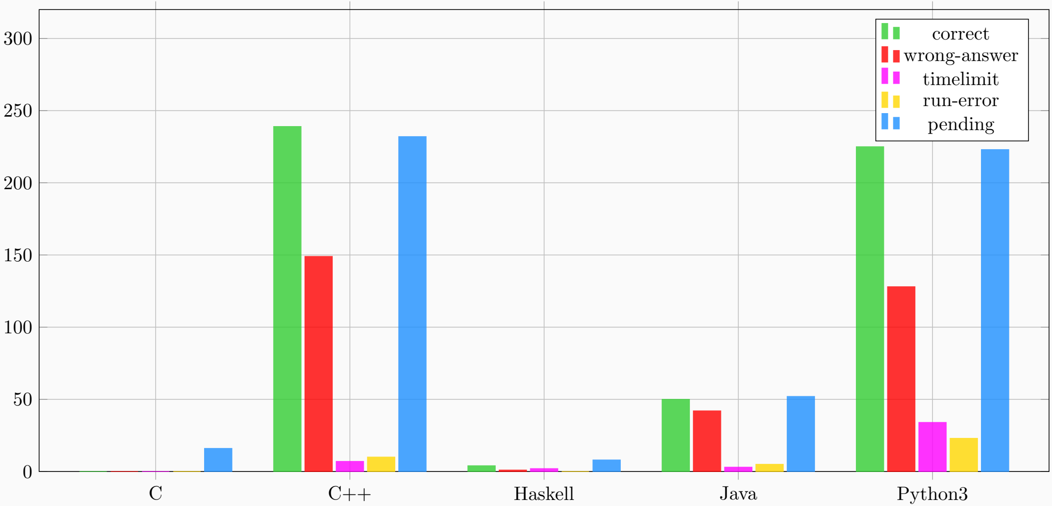
Solution 2 – Count Occurrences

- Constraint only fails if the person standing in front has the same height.
- This is only possible, when there are more than w people with the same height.

$\Rightarrow$ Can be computed in $O(n)$ by using HashMaps.

- Beware of off-by-one errors, e.g. exactly w people with the same height.

Language stats



Random facts

Jury work

- 350 commits

Random facts

Jury work

- 350 commits
- 691 secret test cases ($\approx$ 53 per problem)

Random facts

Jury work

- 350 commits
- 691 secret test cases (≈ 53 per problem)
- 121 jury solutions

Random facts

Jury work

- 350 commits
- 691 secret test cases (≈ 53 per problem)
- 121 jury solutions
- The minimum number of lines the jury needed to solve all problems is

$$1 + 43 + 25 + 5 + 1 + 12 + 20 + 43 + 48 + 6 + 5 + 8 + 3 = 220$$

  On average 16.9 lines per problem