NWERC 2020 presentation of solutions

NWERC 2020 Jury

- **Arnar Bjarni Arnarson**
  Reykjavík University
- **Per Austrin**
  KTH Royal Institute of Technology
- **Jeroen Bransen**
  Chordify
- **Alexander Dietsch**
  FAU Erlangen-Nürnberg
- **Ragnar Groot Koerkamp**
  ETH Zürich
- **Bjarki Ágúst Guðmundsson**
  Google
- **Nils Gustafsson**
  KTH Royal Institute of Technology
- **Timon Knigge**
  ETH Zürich
- **Robin Lee**
  Google
- **Pehr Söderman**
  Kattis
- **Jorke de Vlas**
  Utrecht University
- **Mees de Vries**
  University of Amsterdam
- **Paul Wild**
  FAU Erlangen-Nürnberg

Big thanks to our test solvers

- **Bernhard Linn Hilmarsson**
  ETH Zürich
- **Tómas Ken Magnússon**
  Google
- **Ludo Pulles**
  Leiden University
- **Bergur Snorrason**
  University of Iceland
- **Tobias Werth**
  Google

K: Keyboardd

Problem Author: Pehr Söderman

Problem

Given are two strings, where some characters are duplicated in the second string. Find the duplicated characters.

Solution

- For each of the possible 27 characters, count how often they appear in both strings.
- Output all characters where the counts differ.

Python solution

```
A = input()
B = input()
print(''.join(x for x in map(chr, range(32, 127)) if A.count(x) < B.count(x)))
```

Statistics: 200 submissions, 118 + ? accepted

C: Contest Struggles

Problem Author: Ragnar Groot Koerkamp

Problem

For n numbers between 0 and 100 you are given the average of all numbers (d), and the average of a subset of k of those numbers (s). Compute the average of the remaining numbers.

Solution

- The sum of all numbers is $d \cdot n$.
- So the sum of the remaining numbers is $d \cdot n - s \cdot k$.
- That parts contains $n - k$ numbers, so the average of those numbers is $(d \cdot n - s \cdot k)/(n - k)$.
- When the average is < 0 or > 100, print impossible.

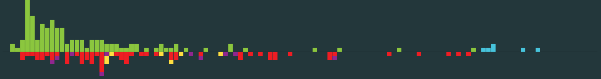
Gotchas

- Precision issues, e.g. answers just below 0 or just above 100

Statistics: 180 submissions, 118 + ? accepted

H: Hot Springs

Problem Author: Timon Knigge



Problem

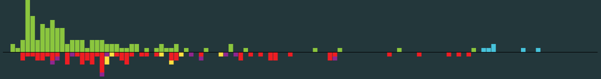
Permute a list of n integers ($n \leq 10^5$) such that for each $2 \leq i \leq n-1$ it holds that $|t'_{i-1} - t'_i| \leq |t'_i - t'_{i+1}|$.

Solution

- Sort the array.
- The largest possible value of $|t_x - t_y|$ is $\max(t) - \min(t)$.
- Put $\max(t)$ in the nth place and $\min(t)$ in the $n-1$th place. It is guaranteed that no other difference will be larger.
- Repeat the same logic with the last two elements fixed and t' as the remaining elements.
- Now the largest value of $|t_x - t_y|$ is $\max(t') - \min(t)$. Put $\max(t')$ in the $n-2$nd place.
- Continue, alternating between min and max of the remaining elements.

H: Hot Springs

Problem Author: Timon Knigge



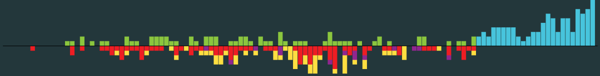
Gotchas

- Not sorting the array in advance.

Statistics: 199 submissions, 114 + ? accepted

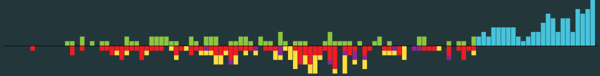
D: Dragon Balls

Problem Author: Paul Wild



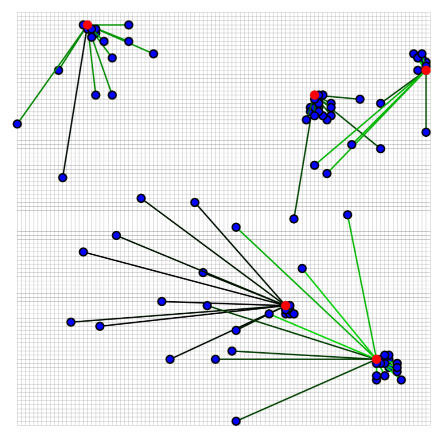
Problem

Find the seven Dragon Balls in the 2D plane. A radar interactively tells you the distances from query points to the closest balls. Balls disappear once found. You may use the radar at most 1 000 times.



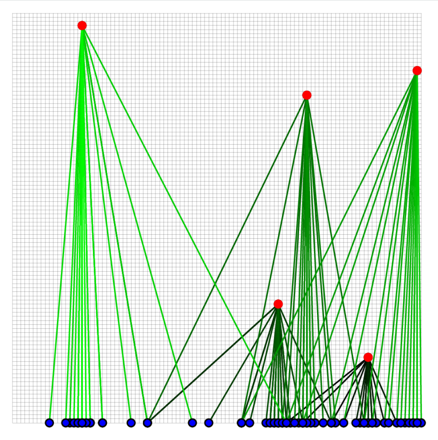
Solution Type 1 – Local Search

Pick a random starting point and home in on one of the balls. Repeat.



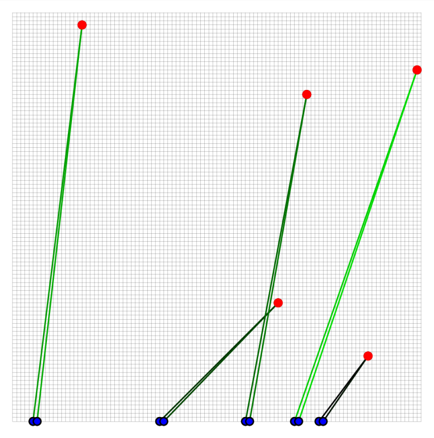
Solution Type 2 – Search Space Partitioning

Use some kind of binary search / ternary search / quadtree.



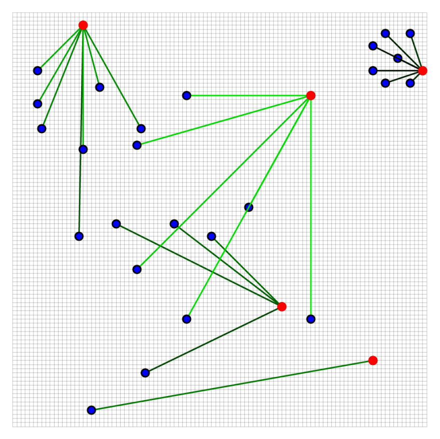
Solution Type 3 – Circle Intersections

Any two adjacent points will have the same closest ball with high probability. Query the two points, then query the intersection point of the two circles.



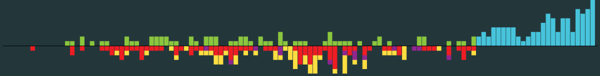
Solution Type 4 – Sum of Squares

Query a random point. Then try all integer points at the given distance.



D: Dragon Balls

Problem Author: Paul Wild



Gotchas

- Asking more queries after all balls have been found.

Statistics: 337 submissions, 70 + ? accepted

A: Atomic Energy

Problem Author: Jorke de Vlas

Problem

Given are the '*explodification*' rules for an atom with a certain amount of neutrons:

- An atom with $k \leq n$ neutrons will be converted into a_k units of energy.
- An atom with $k > n$ will be decomposed into parts $i, j \geq 1$ with $i + j = k$, which are then recursively *explodificated*.

Given an atom with a fixed number of neutrons, what is the minimum energy released?

Observations

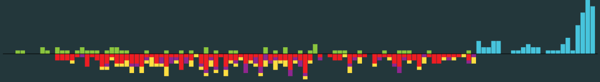
Since the decomposition is arbitrary, we have to assume the worst case – for $k > n$ define:

$$a_k := \min_{1 \leq i \leq k-1} a_i + a_{k-i}.$$

There are upto 10^5 queries with k upto 10^9, so we cannot naively compute all values a_i upto this maximum. Naive computation requires $O(k^2)$ time for the first k values.

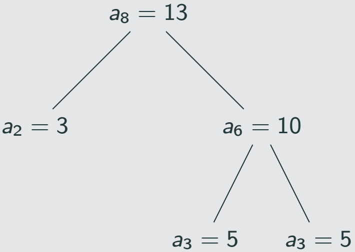
A: Atomic Energy

Problem Author: Jorke de Vlas



Observation 1

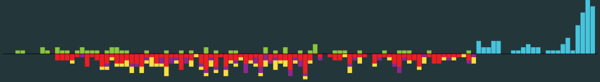
Our first crucial observation is that optimal solutions have a recursive structure. We can write any explodification sequence as a binary tree. This is the first sample, $k = 8$:



Recall this sample had $a_{1,\dots,4} = \{2, 3, 5, 7\}$.

A: Atomic Energy

Problem Author: Jorke de Vlas



Observation 1

For a given query k, imagine recursively following the decomposition $a_k = a_i + a_{k-i}$ until we end up with a decomposition:

$$a_k = \sum_{j=1}^{m} a_{i_j} \quad \text{subj. to} \quad k = \sum_{j=1}^{m} i_j, \text{ with } i_j \in \{1, \ldots, n\}.$$

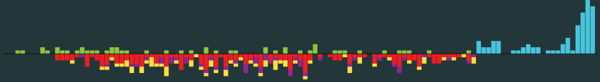
So the leaves of the decomposition tree are a collection of indices i_j that sum to k. Is any decomposition (i_j) satisfying the right hand side realizable?

No – to actually construct this explodification sequence we need to end with some a_x, a_y with $x + y > n$. If $x + y \leq n$, there is no guarantee that $a_{x+y} = a_x + a_y$. (Example: for $n \gg 1$, a sequence of all a_1's is generally impossible.)

A sequence is *realizable* if it contains two x, y with $x + y > n$. After that, we can 'add' new atoms a_{i_j} inductively to construct the explodification tree. In fact any 'prefix' of such a sequence is optimal.

A: Atomic Energy

Problem Author: Jorke de Vlas



Faster computation

Now we can improve the computation of the first k values from $O(k^2)$ to $O(nk)$:

$$a_k = \min_{1 \leq i \leq n} a_i + a_{k-i}.$$

Of course this is still not fast enough with k upto 10^9.

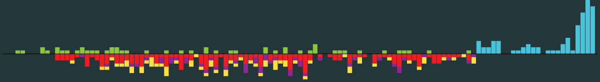
Observation 2

Let $m \in \{1, \ldots, n\}$ minimize a_m/m. When a query k is large enough, most of the terms in the decomposition will be a_m. Indeed, if after removing the two distinguished values a_x, a_y from the sequence we still have m or more values in the tree that are not a_m, by the pigeonhole principle there must be a subset of them that have indices that sum up to a multiple of m, and we can replace them by a_m's to get a decomposition that is not worse.

Hence, any decomposition can be written in such a way that there are at most $m + 1$ terms that are not a_m. In fact we can rearrange the sequence to have these terms in the front, and then fill in the gap with a_m-terms.

A: Atomic Energy

Problem Author: Jorke de Vlas



Full solution

Let m minimize a_i/i over all $i \in \{1, \ldots, n\}$, and use the $O(nk)$ algorithm from earlier to construct the first $(m+1)n$ terms in time $O(n^3)$.

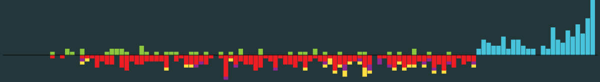
For each query k, find the smallest $j \geq 0$ such that $k - jm \in \{1, \ldots, (m+1)n\}$, and output with $a_{k-jm} + j \cdot a_m$.

Final runtime $O(n^3 + q)$. Efficient implementations of e.g. $O(n^4 + q)$ could also work.

Statistics: 421 submissions, 51 + ? accepted

F: Flight Collision

Problem Author: Jorke de Vlas

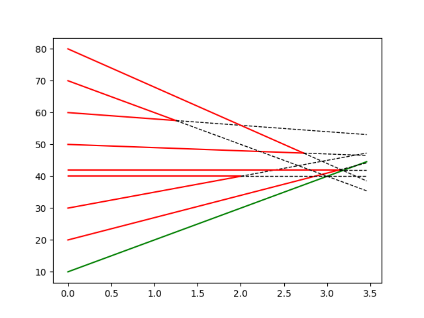


Problem

Some drones are flying along a straight line at constant speed. Simulate the crashes and report the survivors.

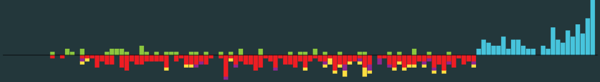
Insight

At any moment, the next crash is going to be between two adjacent drones.



F: Flight Collision

Problem Author: Jorke de Vlas



Solution

- Maintain a set of potential crash events, sorted by time.
- The crash times can be found by solving linear equations.
- When processing a crash, add a new event for the two drones that become adjacent.
- Time complexity: $\mathcal{O}(n \log n)$.

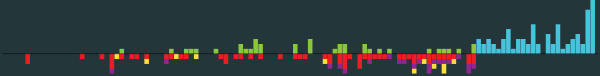
Gotchas

- Use fractions or `long double` to avoid precision errors.
- Only consider crashes at times $t > 0$.

Statistics: 421 submissions, 46 + ? accepted

E: Endgame

Problem Author: Nils Gustafsson



Problem

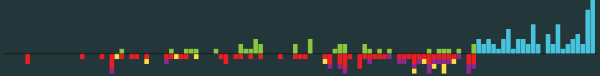
Given the location of a piece on an $n \times n$ playing board and n types of moves ($n \leq 10^5$). Find a position on the board that the piece cannot reach within two moves.

Solution

- Simpler question: Given a specific position, can the piece reach that position within two moves?
  - BFS/DFS will take $O(n^2)$ time, which is too slow.
  - Bidirectional search:
    - F: the set of positions that the piece can reach within one move.
    - B: the set of positions that can reach the target position within one move.
    - F and B intersect iff. the piece can reach the position within two moves.
    - These sets can be constructed and intersected in $O(n \log n)$ time.
- Asking this question for all n^2 positions on the board is way too slow.
  - Do we have to try all of them?

E: Endgame

Problem Author: Nils Gustafsson



Solution

- In the worst case, the piece can reach at most approx. $n^2/2$ positions on the board within two moves.
- If we pick a random position on the board, the piece can reach that position within two moves with probability at most $1/2$.
  - Repeating this k times, the probability that the piece can reach all of them within two moves is at most $1/2^k$, which quickly tends to 0.
- Run bidirectional search on 30 random positions.

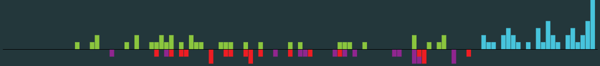
Gotchas

- The piece is not allowed to move off the playing board.
- When $n \in \{2, 3\}$, the piece may be able to reach all the positions within two moves.

Statistics: 210 submissions, 37 + ? accepted

I: Island Tour

Problem Author: Jeroen Bransen



Problem

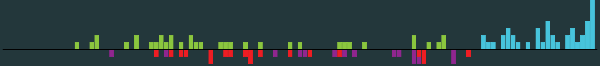
Three people start in three places on a cycle graph and walk around according to a timer. Where can you place them so that they won't ever be in the same place at the same time?

Solution

- A simple solution tries all $O(n^3)$ placements for Tijmen, Annemarie, and Imme and then simulates the $O(n)$ steps recording when each person arrives and departs at the nodes to compare with the others for overlap.
- However, $O(n^4)$ is too slow. We need to do some pre-calculation.
- Conflicts are between two people rather than three. We only need to answer the question does_intersect(a, b, s_a, s_b) for each pair of people a and b.
- So, for each **pair** of people a and b, try all $O(n^2)$ combinations and run the $O(n)$ simulation. Store the result in a table compatible$[a][b][x][y]$ for later.
- Using the table, we can try all $O(n^3)$ possibilities in $O(1)$ time each. This is fast enough.

I: Island Tour

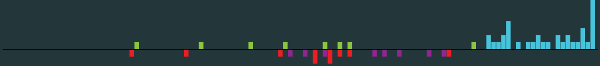
Problem Author: Jeroen Bransen



Statistics: 113 submissions, 36 + ? accepted

G: Great Expectations

Problem Author: Mees de Vries



Problem

Determine the most efficient method to break the record in a speedrun. You may reset at any point.

Insights

During a run, you have $r - n - 1$ time margin to make errors.

Optimally, the only place where you reset is immediately after failing a trick.

Solution attempt

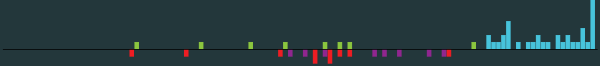
- Use dynamic programming!
- $DP[i,j] :=$ the expected time until a record when you are just before trick i and have used j margin for error. We are interested in $DP[0,0]$.
- When you complete trick i, the rest of the run takes $(t_{i+1} - t_i) + DP[i+1,j]$ time.
- When you fail the trick, you either reset (taking $DP[0,0]$ time) or continue (taking $d_i + (t_{i+1} - t_i) + DP[i+1, j+d_i]$ time).
- This gives a DP relation:

$$DP[i,j] = \begin{matrix} p_i & \cdot & ((t_{i+1} - t_i) + DP[i+1,j]) + \\ (1-p_i) & \cdot & \min(DP[0,0], d_i + (t_{i+1} - t_i) + DP[i+1, j+d_i]) \end{matrix}$$

- We can use $DP[m][j] = 0$ as the base cases for the DP.

G: Great Expectations

Problem Author: Mees de Vries



Catch

We now have a DP relation, but we need to know $DP[0,0]$ in order to use it.

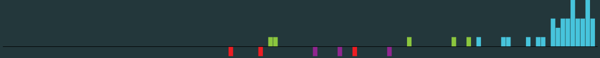
Solution

- Consider making some guess P for the value of $DP[0,0]$. We can use this value to fill the DP table.
- When the resulting $DP[0,0]$ is larger than P, the guess was too low. When $DP[0,0]$ is smaller than P, the guess was too high.
- Use binary search to determine the optimal value of P, and thus the actual value of $DP[0,0]$.

Statistics: 61 submissions, 8 + ? accepted

J: Joint Excavation

Problem Author: Timon Knigge



Problem

A connected graph is to be split into multiple connected components by a non-self-intersecting path. The components are then to be distributed into two groups A and B such that the number of nodes in both groups are the same.

Find a path and distribution that satisfy these requirements.

Solution

- Assign each node to group A.
- Run a Depth–First–Search starting at any node.
- Whenever the *DFS* visits a new node N, remove N from A and add it to the path.
- Whenever the *DFS* backtracks from node N^*, remove N^* from the path and add it to B.
- Repeat until $|A| = |B|$.
- The *DFS* guarantees that A and B never have neighbouring nodes.

B: Bulldozer

Problem Author: Mees de Vries

Problem

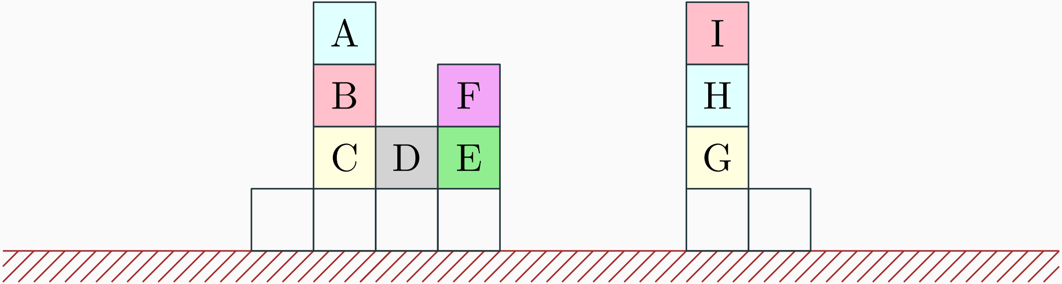
Given a row of stack of blocks, how many 'bulldoze' operations are needed to level all the blocks.

Observations

- Each block can be 'buried' in two moves: push the bottom of the stack right, push the block left.
- It's never worse to do all burying operations at the end.
- All other blocks that start non-grounded end at an initially empty stack.
- Number the non-grounded blocks from left to right, where each stack is numbered bottom to top.
- The final solution has stretches of blocks that move left, stretches of blocks that move right, mixed with stretches of blocks that are buried.
- We have infinite space on the left and right, and the stretches of blocks that go there contain full stacks of blocks only.

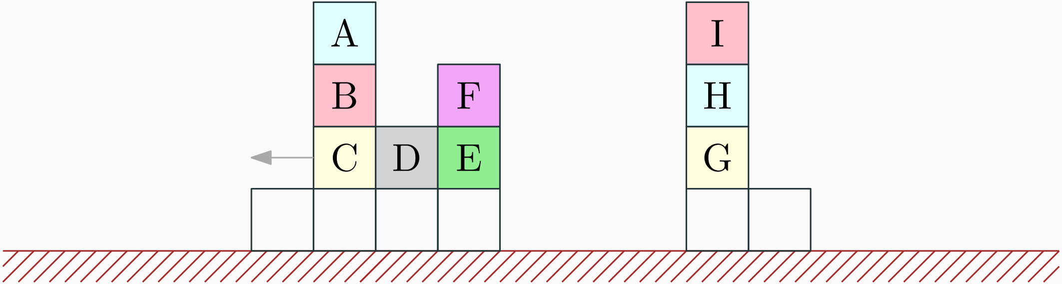
B: Bulldozer

Problem Author: Mees de Vries



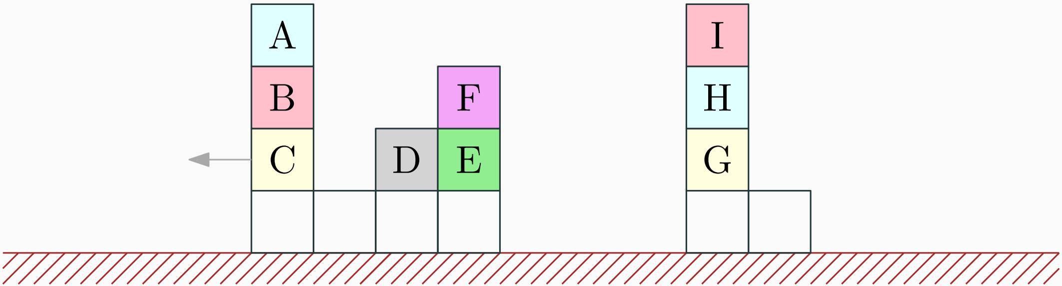
B: Bulldozer

Problem Author: Mees de Vries



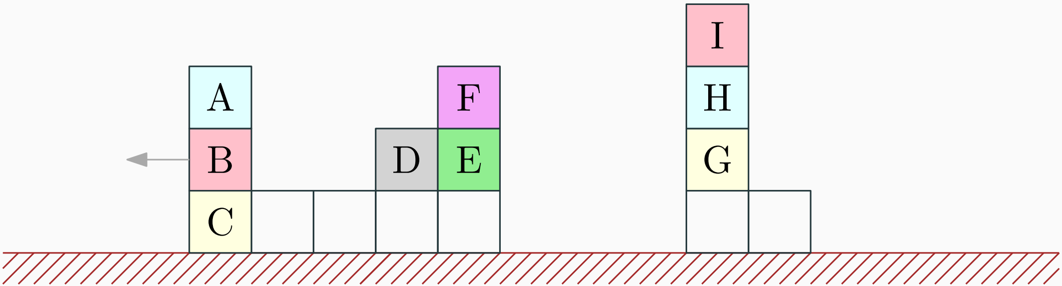
B: Bulldozer

Problem Author: Mees de Vries



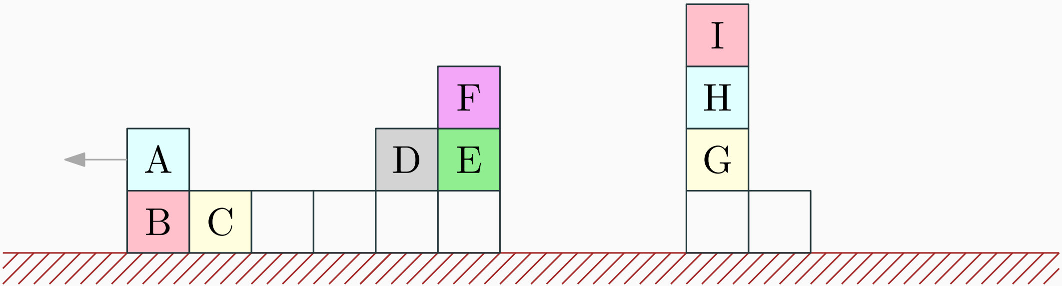
B: Bulldozer

Problem Author: Mees de Vries



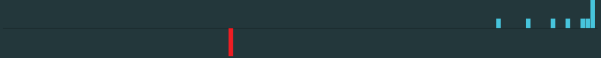
B: Bulldozer

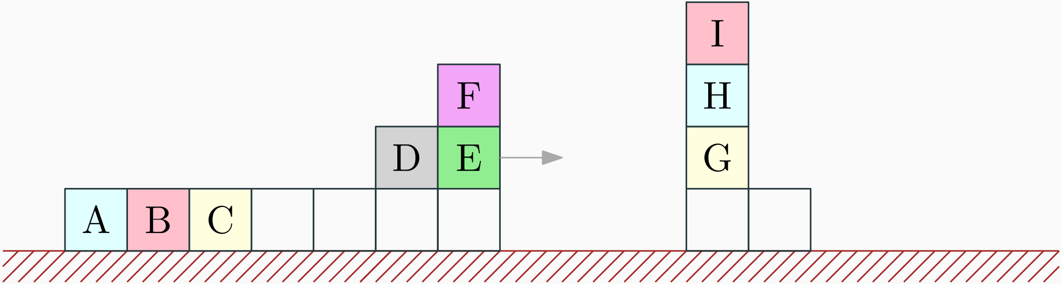
Problem Author: Mees de Vries



B: Bulldozer

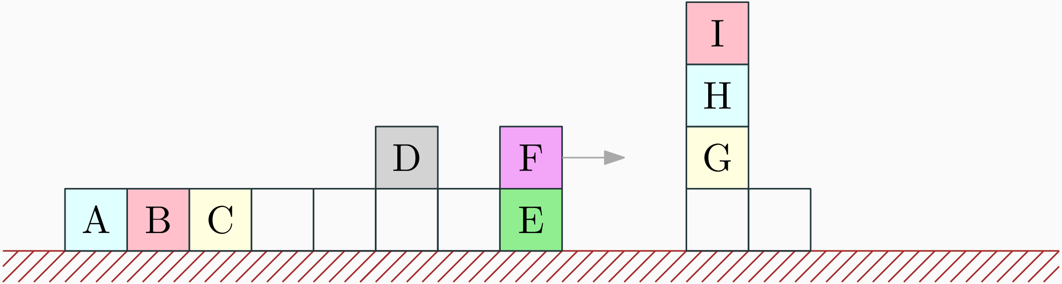
Problem Author: Mees de Vries





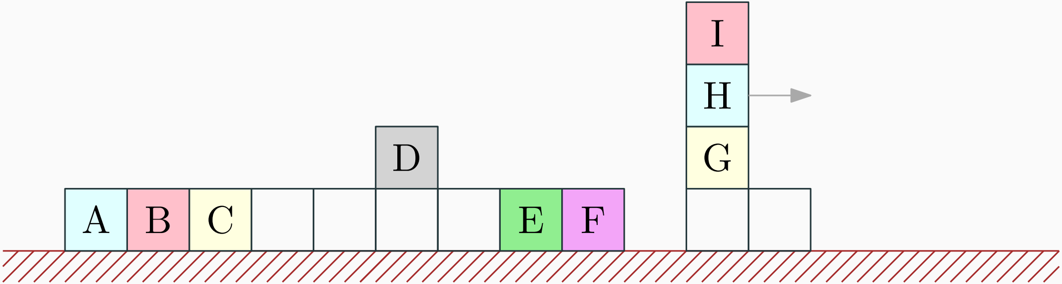
B: Bulldozer

Problem Author: Mees de Vries



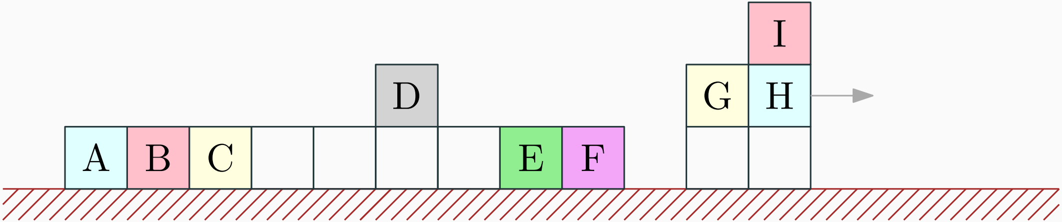
B: Bulldozer

Problem Author: Mees de Vries



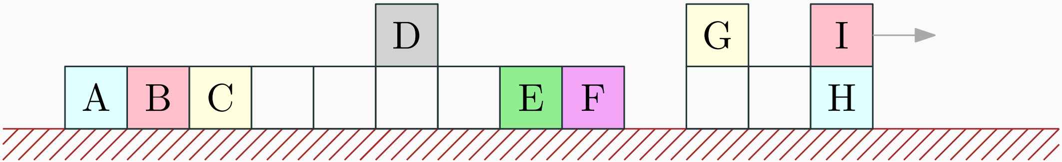
B: Bulldozer

Problem Author: Mees de Vries



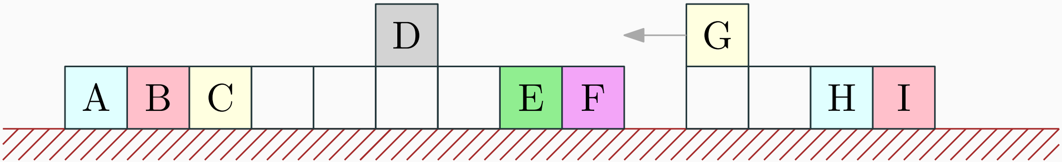
B: Bulldozer

Problem Author: Mees de Vries



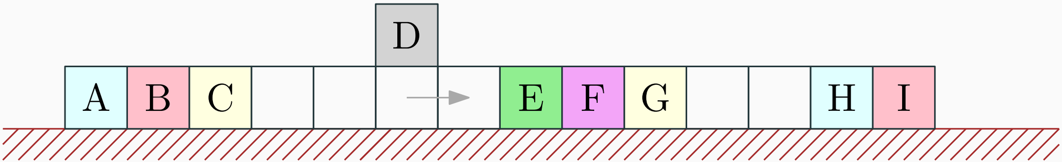
B: Bulldozer

Problem Author: Mees de Vries



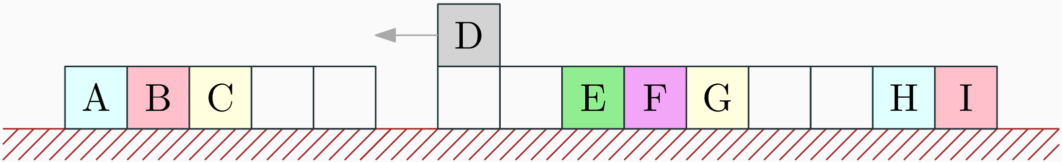
B: Bulldozer

Problem Author: Mees de Vries



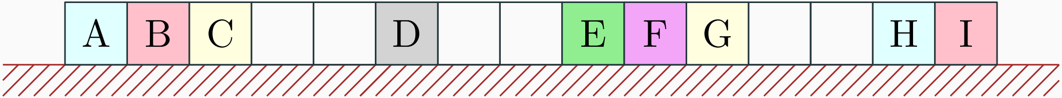
B: Bulldozer

Problem Author: Mees de Vries



B: Bulldozer

Problem Author: Mees de Vries



B: Bulldozer

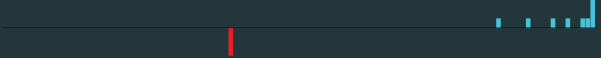
Problem Author: Mees de Vries

Solution

- Make a weighted directed graph on the initial state of the blocks, with a start vertex on the far left and an end vertex on the far right. The shortest path will be the answer.
- For each empty stack S, find the block X that would end there when moving blocks from the left. Add an edge from X to S of cost K, the required number of moves for this.
- Similarly, find the block Y that would end at S when moving blocks from the right. Add an edge from S to Y of cost K.
- When block X ends in empty stack Y after K moves, all blocks in between are already levelled.
- Add an edge from the start vertex to the top of each stack: the cost of moving all in between blocks left.
- Add an edge from the bottom of each stack to the end vertex: the cost of moving all in between blocks right.
- For burying, add an edge between consecutive blocks of cost 2, but merge adjacent edges when possible to prevent adding $2 \cdot 10^{14}$ edges.

B: Bulldozer

Problem Author: Mees de Vries



Statistics: 12 submissions, 0 + ? accepted

Random facts

Jury work

- 616 git commits.
- 252 jury solutions with 11577 lines in total, about 46 lines on average.
- The number of lines the jury needed to solve all problems is

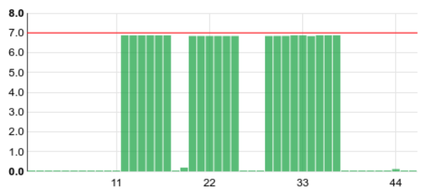
  $$12 + 44 + 5 + 26 + 24 + 33 + 16 + 5 + 20 + 31 + 4 = 220$$

  On average 20 lines per problem.
- 682 test cases, on average 46 per problem.
- Last test cases added yesterday evening, at least 2 submissions failed on only those.

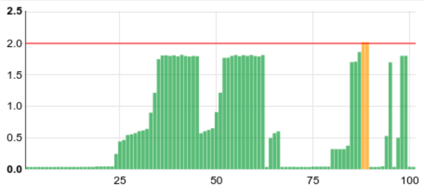
Random facts

Timelimits

- Just lucky:



- Just unlucky (different problem and team):



Language stats