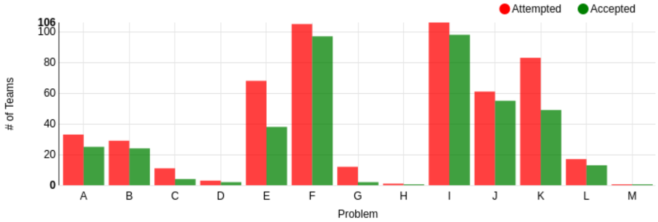
Problem Analysis Session

SWERC judges

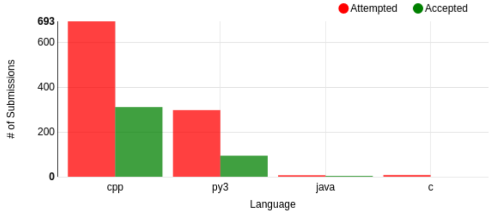
28/01/2024

Statistics

Number of submissions: about 1424

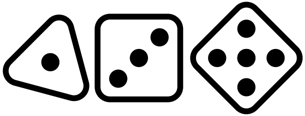


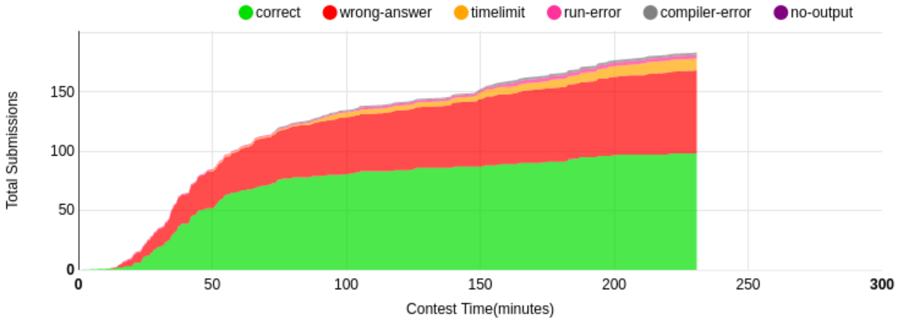
Number of clarification requests: 52 (about 42 answered “No comment.”)



I: Throwing dice

Solved by 98 teams before freeze.
First solved after 11 min by
STM32G431CB (Artois University).





I: Throwing dice

Problem

Compare two probabilities

Solution – Linear time & constant space

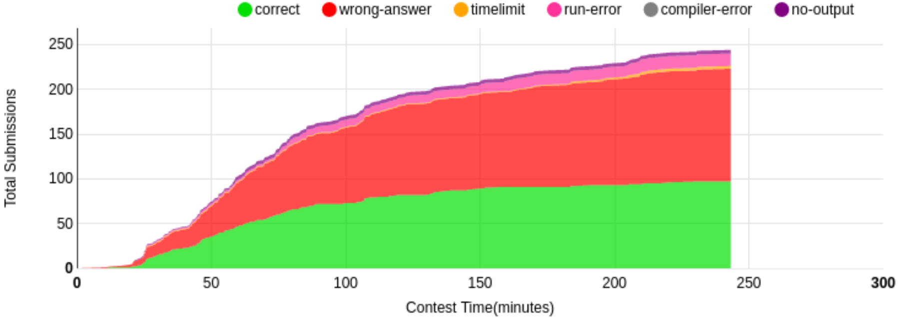
- The score of player X is distributed symmetrically around $\mathbb{E}[X]$, and takes the values $\mathbb{E}[X]$ if $\mathbb{E}[X]$ is integer, or $\mathbb{E}[X] \pm 1/2$ otherwise.
- The expected score for an S-sided die is $(1+S)/2$.

Consequently,

$$\begin{aligned} \mathbb{P}_A > \mathbb{P}_B &\Leftrightarrow \mathbb{E}[A] > \mathbb{E}[B] \\ &\Leftrightarrow A_1 + A_2 + \cdots + A_M + M > B_1 + B_2 + \cdots + B_N + N. \end{aligned}$$

F: Programming-trampoline-athlon!

Solved by 97 teams before freeze.
First solved after 12 min by
EPFL Polympiads 1 (EPFL).



F: Programming-trampoline-athlon!

Problem

Find the medalists of Programming-trampoline-athlon.

Solution – Linear time & space

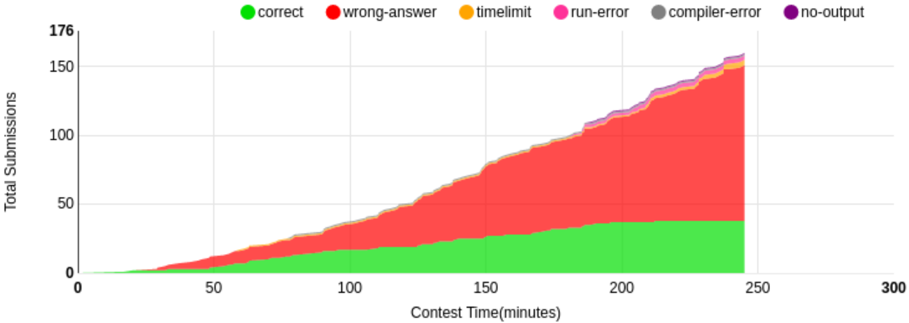
- The score of a team is given by

$$P \cdot 10 + E_1 + \cdots + E_6 - \min(E_1, \ldots, E_6) - \max(E_1, \ldots, E_6)$$

- Compute the score for each team, store the three best scores, and output the information about the teams reaching these scores.
- Quasilinear time solution also accepted: Compute the score for each team, sort, and output at least 3 scores until reaching a different score.

E: Nicest view

Solved by 38 teams before freeze.
First solved after 17 min by
Heroes of the C (Universidade do Porto).



E: Nicest view

Problem

Find the longest horizontal line between two points on a path.

Solution – Linear time & space

- The nicest view is obtained either at a milestone or looking at a milestone.
- At each step k, remember those integers $\ell < k$ such that $H_i < H_\ell$ whenever $\ell < i \leqslant k$.
- You can see at distance $d_k = k - \ell_{\max} - (H_{\ell_{\max}} - H_k)/(H_{\ell_{\max}} - H_{\ell_{\max}+1})$.
- Do not forget to look on you right too! (i.e., go backwards)

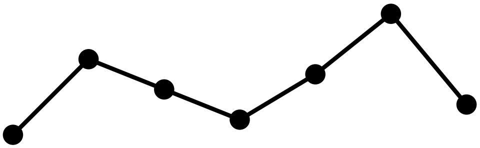
E: Nicest view

Problem

Find the longest horizontal line between two points on a path.

Solution – Linear time & space

- The nicest view is obtained either at a milestone or looking at a milestone.
- At each step k, remember those integers $\ell < k$ such that $H_i < H_\ell$ whenever $\ell < i \leqslant k$.
- You can see at distance $d_k = k - \ell_{\max} - (H_{\ell_{\max}} - H_k)/(H_{\ell_{\max}} - H_{\ell_{\max}+1})$.
- Do not forget to look on you right too! (i.e., go backwards)



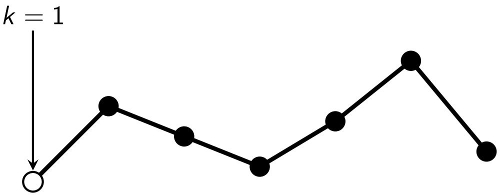
E: Nicest view

Problem

Find the longest horizontal line between two points on a path.

Solution – Linear time & space

- The nicest view is obtained either at a milestone or looking at a milestone.
- At each step k, remember those integers $\ell < k$ such that $H_i < H_\ell$ whenever $\ell < i \leqslant k$.
- You can see at distance $d_k = k - \ell_{\text{max}} - (H_{\ell_{\text{max}}} - H_k)/(H_{\ell_{\text{max}}} - H_{\ell_{\text{max}}+1})$.
- Do not forget to look on you right too! (i.e., go backwards)



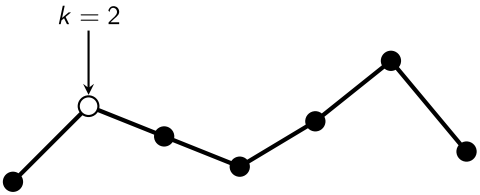
E: Nicest view

Problem

Find the longest horizontal line between two points on a path.

Solution – Linear time & space

- The nicest view is obtained either at a milestone or looking at a milestone.
- At each step k, remember those integers $\ell < k$ such that $H_i < H_\ell$ whenever $\ell < i \leqslant k$.
- You can see at distance $d_k = k - \ell_{\max} - (H_{\ell_{\max}} - H_k)/(H_{\ell_{\max}} - H_{\ell_{\max}+1})$.
- Do not forget to look on you right too! (i.e., go backwards)



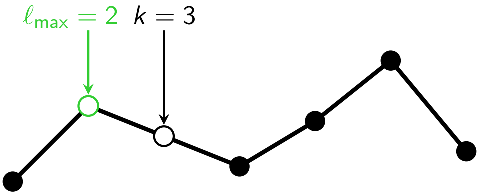
E: Nicest view

Problem

Find the longest horizontal line between two points on a path.

Solution – Linear time & space

- The nicest view is obtained either at a milestone or looking at a milestone.
- At each step k, remember those integers $\ell < k$ such that $H_i < H_\ell$ whenever $\ell < i \leqslant k$.
- You can see at distance $d_k = k - \ell_{\max} - (H_{\ell_{\max}} - H_k)/(H_{\ell_{\max}} - H_{\ell_{\max}+1})$.
- Do not forget to look on you right too! (i.e., go backwards)



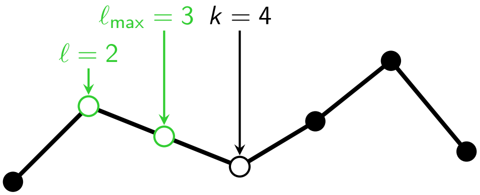
E: Nicest view

Problem

Find the longest horizontal line between two points on a path.

Solution – Linear time & space

- The nicest view is obtained either at a milestone or looking at a milestone.
- At each step k, remember those integers $\ell < k$ such that $H_i < H_\ell$ whenever $\ell < i \leqslant k$.
- You can see at distance $d_k = k - \ell_{\max} - (H_{\ell_{\max}} - H_k)/(H_{\ell_{\max}} - H_{\ell_{\max}+1})$.
- Do not forget to look on you right too! (i.e., go backwards)



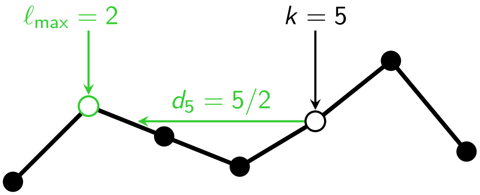
E: Nicest view

Problem

Find the longest horizontal line between two points on a path.

Solution – Linear time & space

- The nicest view is obtained either at a milestone or looking at a milestone.
- At each step k, remember those integers $\ell < k$ such that $H_i < H_\ell$ whenever $\ell < i \leqslant k$.
- You can see at distance $d_k = k - \ell_{\max} - (H_{\ell_{\max}} - H_k)/(H_{\ell_{\max}} - H_{\ell_{\max}+1})$.
- Do not forget to look on you right too! (i.e., go backwards)



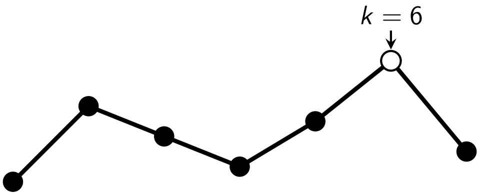
E: Nicest view

Problem

Find the longest horizontal line between two points on a path.

Solution – Linear time & space

- The nicest view is obtained either at a milestone or looking at a milestone.
- At each step k, remember those integers $\ell < k$ such that $H_i < H_\ell$ whenever $\ell < i \leqslant k$.
- You can see at distance $d_k = k - \ell_{\max} - (H_{\ell_{\max}} - H_k)/(H_{\ell_{\max}} - H_{\ell_{\max}+1})$.
- Do not forget to look on you right too! (i.e., go backwards)



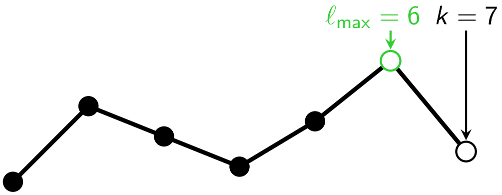
E: Nicest view

Problem

Find the longest horizontal line between two points on a path.

Solution – Linear time & space

- The nicest view is obtained either at a milestone or looking at a milestone.
- At each step k, remember those integers $\ell < k$ such that $H_i < H_\ell$ whenever $\ell < i \leqslant k$.
- You can see at distance $d_k = k - \ell_{\max} - (H_{\ell_{\max}} - H_k)/(H_{\ell_{\max}} - H_{\ell_{\max}+1})$.
- Do not forget to look on you right too! (i.e., go backwards)



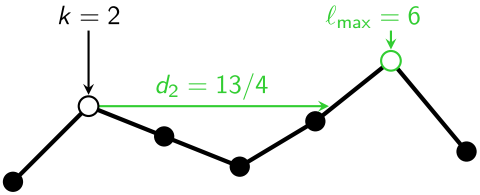
E: Nicest view

Problem

Find the longest horizontal line between two points on a path.

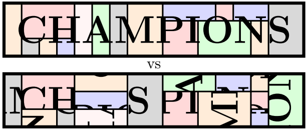
Solution – Linear time & space

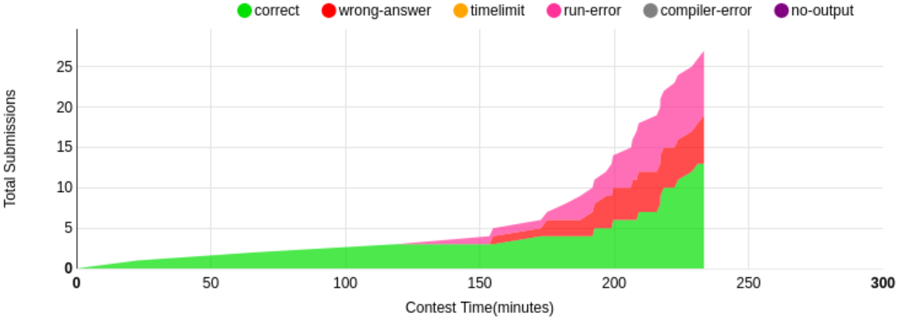
- The nicest view is obtained either at a milestone or looking at a milestone.
- At each step k, remember those integers $\ell < k$ such that $H_i < H_\ell$ whenever $\ell < i \leqslant k$.
- You can see at distance $d_k = k - \ell_{\max} - (H_{\ell_{\max}} - H_k)/(H_{\ell_{\max}} - H_{\ell_{\max}+1})$.
- Do not forget to look on you right too! (i.e., go backwards)



L: Broken trophy

Solved by 13 teams before freeze.
First solved after 22 min by
cnXtv (École Polytechnique).





L: Broken trophy

Problem

Assemble rectangular pieces (tiles) with sides $\in \{1, 2, 3\}$ into a $3 \times N$ rectangle.

This looks like these classic pentomino tilings puzzles

— with simpler tiles (6 different rectangles)

— but with up to $3 \cdot 10^5$ tiles!

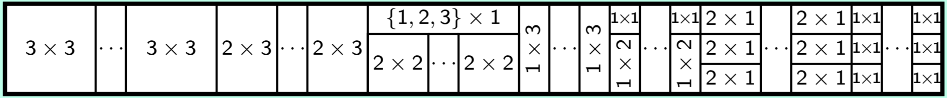
L: Broken trophy

Problem

Assemble rectangular pieces (tiles) with sides $\in \{1, 2, 3\}$ into a $3 \times N$ rectangle.

Solution – Linear time & space

You can always assemble your tiles as follows:

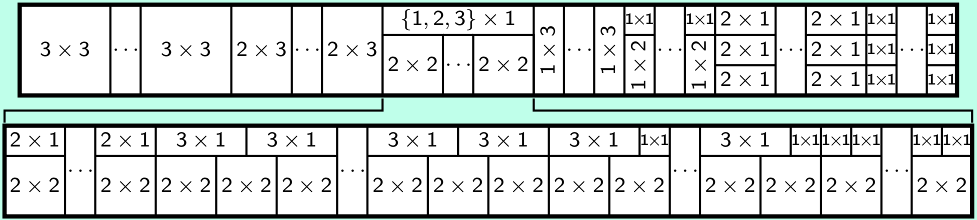


Problem

Assemble rectangular pieces (tiles) with sides $\in \{1, 2, 3\}$ into a $3 \times N$ rectangle.

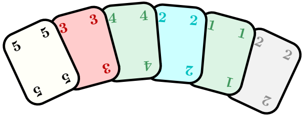
Solution – Linear time & space

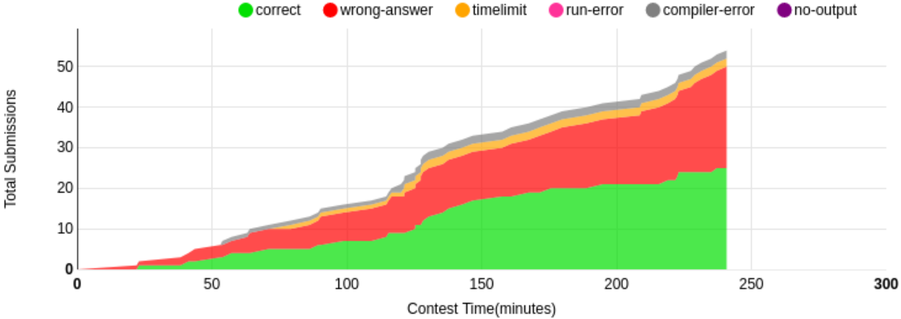
You can always assemble your tiles as follows:



A: Card game

Solved by 25 teams before freeze.
First solved after 22 min by
flag[10] (Università di Pisa).





A: Card game

Problem

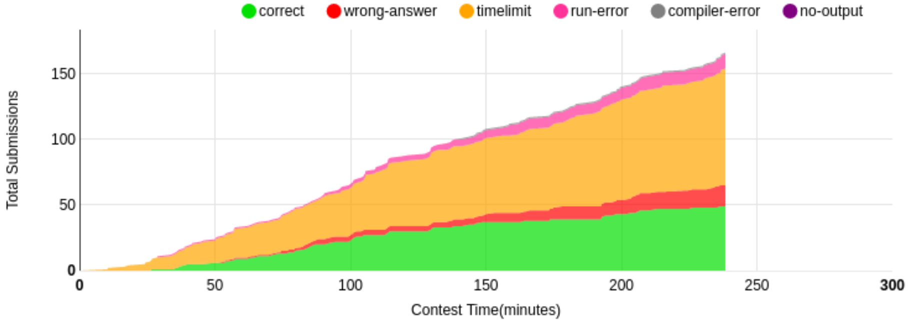
Find the minimum number of single-card moves required to organize the cards in your hand.

Solution – $\mathcal{O}(N \log N)$ time complexity

- First, imagine a total order on the cards is required.
  - Observations:
    - The cards not moved are already in increasing order.
    - One move is required for each moved card.
  - The cost is $N - LIS$, where LIS is the size of a longest increasing subsequence.
  - Use an $\mathcal{O}(N \log N)$ algorithm for finding LIS.
    - Uses an array where position i contains the smallest value that ends an increasing subsequence of length $i + 1$.
- Go over all 4! valid orders (corresponding to suit orderings), and take the minimum.

K: Team selection

Solved by 49 teams before freeze.
First solved after 26 min by
Volterra gng (Università di Pisa).



K: Team selection

Problem

Repeatedly find and extract the k-th element among an ordered list of elements.

Solution – $O(N \times \log(N))$

- For each element, store a 0 or a 1 to denote whether it was picked.
- Maintain a segment tree on these elements.
- Given k, traverse the segment tree efficiently to find the k-th non-zero element in $O(\log(N))$.
- Set it to 0 and update the segment tree in $O(\log(N))$.

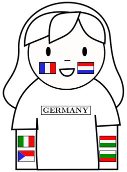
K: Team selection

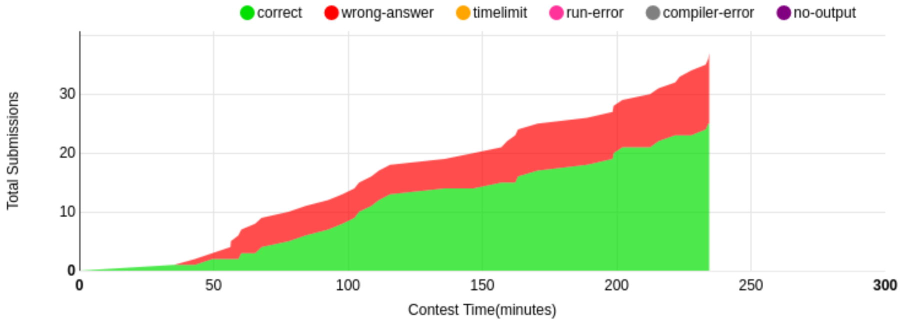
Alternate solutions

- C++'s order_statistics_tree: also $O(N \times \log(N))$ but too slow by a factor 3 in practice.
- Binary search on the segment tree: $O(N \times \log(N)^2)$. Accepted if efficiently implemented.
- Maintain an array of the remaining elements, together with an array of picked indices, update the array once every $O(\sqrt{N})$ queries.
  - Total $O(N^{3/2})$, too slow, but only by a factor 3.
- Naïve $O(N^2)$, too slow.

B: Supporting everyone

Solved by 24 teams before freeze.
First solved after 35 min by
eXotic (École Polytechnique).





B: Supporting everyone

Problem

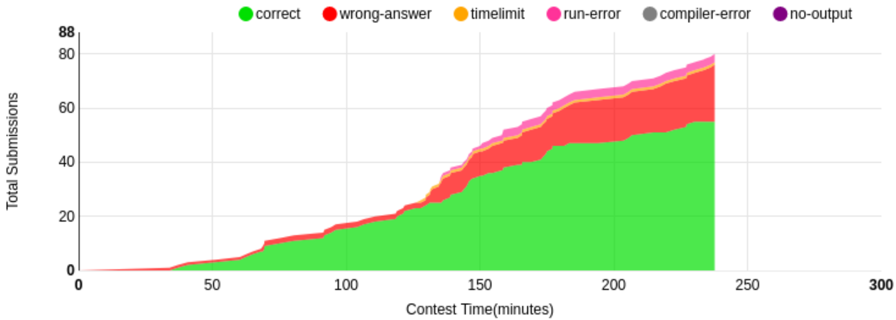
Give the minimum number of crayons and pins required to represent every country.

Solution – Cubic time & quadratic space

- Represent the link between countries and color as a graph.
- This creates a **bipartite** graph with $N + M$ vertex and at most $N * M$ edges.
- Then representing all countries corresponds to the minimum vertex cover problem.
- For bipartite graph, using Koenig's theorem, this equivalent to maximal matching.
- Doable in $O(V * E) = O(NM(N + M))$.

J: Olympic goodies

Solved by 55 teams before freeze.
First solved after 37 min by
doublETHink (ETH Zürich).



J: Olympic goodies

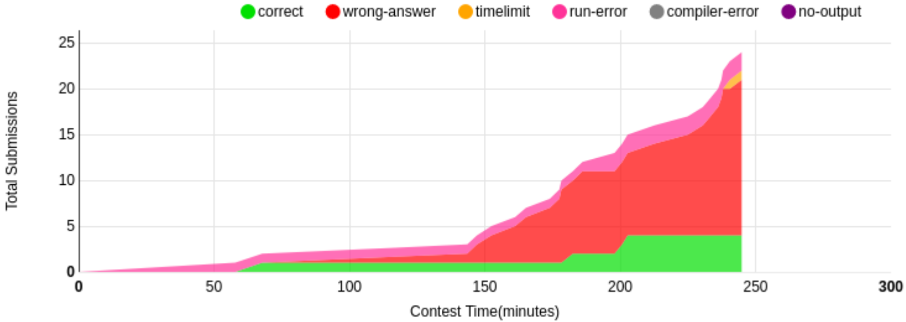
Problem

Place strictly positive integer weights on a subset of the edges of a tree, such that the total sum of edge weights equals P, and the maximum possible weight of a path is minimized.

Solution – Linear time & linear space

- Place weights "as uniformly as possible" on edges incident to leaves
  - Good intuition: longest path should go through leaves
- Maximum weight for a path is:
  - $2 * (P/NumberOfLeaves)$ if $P\%NumberOfLeaves = 0$
  - $2 * (P/NumberOfLeaves) + 1$ if $P\%NumberOfLeaves = 1$
  - $2 * (P/NumberOfLeaves) + 2$ if $P\%NumberOfLeaves \geqslant 2$
- To compute the answer, all you have to do is (read the input and) count leaves.
- Sample was misleading (yeah, well...), but: other solutions? proofs of correctness?

Solved by 4 teams before freeze.
First solved after 67 min by
UNIBOis (University of Bologna).



C: Metro quiz

Problem

A metro map is given. A player thinks of a random line, the second must guess by asking if the line goes through a given station. Find the strategy minimizing the expected (average) number of questions.

Solution – $O((M+N)*M*2^N)$ time

- Dynamic programming
- State is two bitsets, (set of stations with constraints, constraints on these)
- Number of states is $M*2^N$, not 2^N*2^N

C: Metro quiz

Problem

A metro map is given. A player thinks of a random line, the second must guess by asking if the line goes through a given station. Find the strategy minimizing the expected (average) number of questions.

Solution – $O((M+N) * M * 2^N)$ time

- In a state S, choose the best station to ask about.
- Call S_i the state where the line must go through station i and $S_{\neg i}$ the state where the line must not go through station i.

$$M(S) = \min_{0 \leqslant i < N} P(i) * M(S_i) + P(\neg i) * M(S_{\neg i})$$

C: Metro quiz

Problem

A metro map is given. A player thinks of a random line, the second must guess by asking if the line goes through a given station. Find the strategy minimizing the expected (average) number of questions.

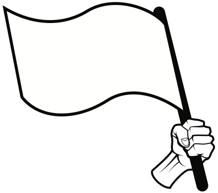
Solution – $O((M+N) * M * 2^N)$ time

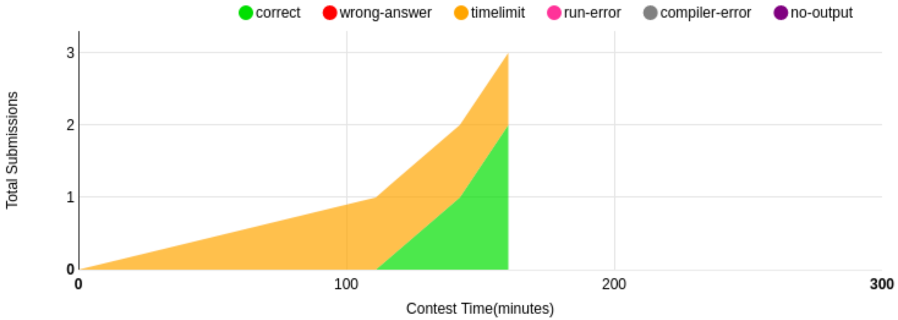
$$M(S) = \min_{0 \leqslant i < N} P(i) * M(S_i) + P(\neg i) * M(S_{\neg i})$$

- To compute the probabilities, need to compute the lines still possible in a given state.
- Precomputing it is slower than doing it on-demand (hash table is too large).

D: Flag performance

Solved by 2 teams before freeze.
First solved after 142 min by
eXotic (École Polytechnique).





D: Flag performance

Problem

Number of ways to sort arrays (permutations) of length N with K swaps (transpositions) ?

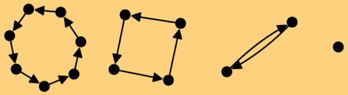
D: Flag performance

Problem

Number of ways to sort arrays (permutations) of length N with K swaps (transpositions) ?

Solution – $O(KN^2p(N))$ insertions/lookups & $O(Np(N))$ space

- Cycle decomposition (without labels) enough to detect identity, e.g.,



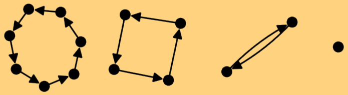
D: Flag performance

Problem

Number of ways to sort arrays (permutations) of length N with K swaps (transpositions) ?

Solution – $O(KN^2 p(N))$ insertions/lookups & $O(Np(N))$ space

- Cycle decomposition (without labels) enough to detect identity, e.g.,



- Dynamic programming over integer partitions of N , e.g., $7 + 4 + 2 + 1 = 15$.
  $\Rightarrow$ number of integer partitions $p(N)$ small, $p(N) \leqslant p(30) = 5604$.

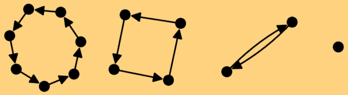
D: Flag performance

Problem

Number of ways to sort arrays (permutations) of length N with K swaps (transpositions) ?

Solution – $O(KN^2p(N))$ insertions/lookups & $O(Np(N))$ space

- Cycle decomposition (without labels) enough to detect identity, e.g.,

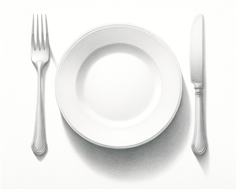


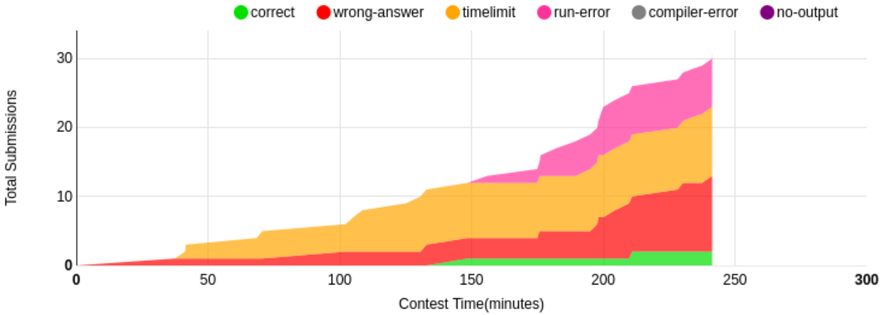
- Dynamic programming over integer partitions of N , e.g., $7+4+2+1=15$.
  $\Rightarrow$ number of integer partitions $p(N)$ small, $p(N) \leqslant p(30) = 5604$.
- All permutations: reverse process, start from identity backwards.
  $\Rightarrow$ normalization required: number of permutations with decomposition $(t_1, \ldots, t_N)$,

$$c(t_1, \ldots, t_N) = N!/\left(t_1!t_2! \ldots t_N! \cdot 1^{t_1} 2^{t_2} \ldots N^{t_N}\right) .$$

G: Favourite dish

Solved by 2 teams before freeze.
First solved after 148 min by
dETHroners (ETH Zürich).





G: Favourite dish

Problem

Calculate each person's favorite dish based on the weights and scores.

Solution – $O(NlogN + MlogM)$ time & linear space

If we calculate the whole score table, it takes O(NM) time which is too high.

We can sort the persons by their weight on taste, like:

Person \Dish	1	2	3	4
1	3.2	3.4	3.2	3.0
3	3.5*	3.5	3.0	3.5
2	4.4	3.8	2.4	5.0*

After sorting, each dish (each column) is a ascending or descending sequence.

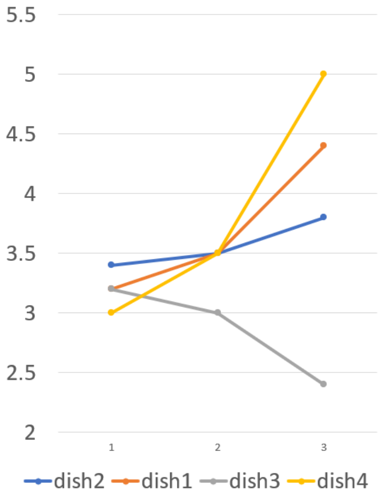
Then sort the dishes by the first person's score, like:

Person \Dish	2	1	3	4
1	3.4*	3.2	3.2	3.0
3	3.5	3.5*	3.0	3.5
2	3.8	4.4	2.4	5.0*

G: Favourite dish

Solution – $O(NlogN + MlogM)$ time & linear space

For each dish, the scores look like an ascending or descending polyline. We may start from the first dish, process the dishes one by one, and maintain a "list of currently highest score" ("h-list") for each person.



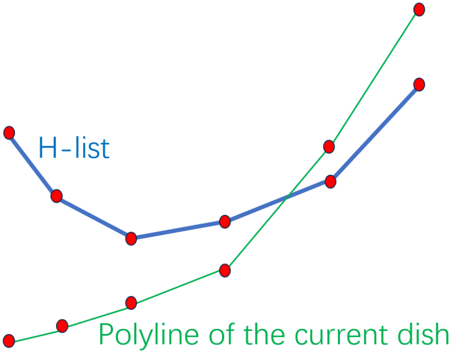
G: Favourite dish

Solution – $O(NlogN + MlogM)$ time & linear space

At any time, the h-list is also a polyline. We can prove that the polyline of dish i and the h-list after processing dish $i-1$ have at most one intersection point.

That intersection point (if exists) can be found by binary search. With this algorithm, the complexity can be reduced to $O(NlogN + MlogM)$.

Alternatively, this it can also be solved with convex hull model.



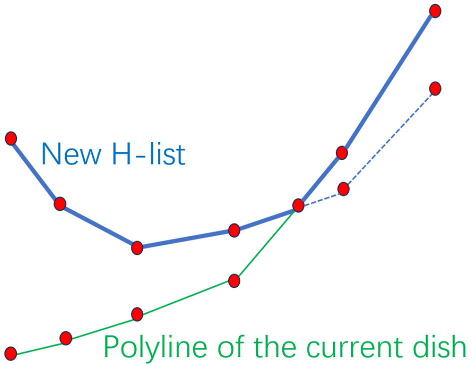
G: Favourite dish

Solution – $O(NlogN + MlogM)$ time & linear space

At any time, the h-list is also a polyline. We can prove that the polyline of dish i and the h-list after processing dish $i - 1$ have at most one intersection point.

That intersection point (if exists) can be found by binary search. With this algorithm, the complexity can be reduced to $O(NlogN + MlogM)$.

Alternatively, this it can also be solved with convex hull model.



H: Break a leg!

Not solved before freeze.

H: Break a leg!

Problem

Given a non-crossing polygon, how many triples of vertices are such that the center of mass of the polygon lies (strictly) inside the triangle formed by these vertices?

Solution – $\mathcal{O}(N \log(N))$ time complexity

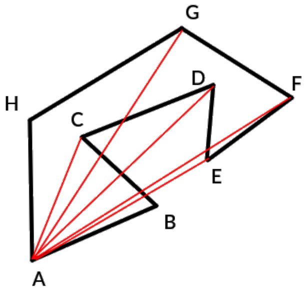
1. Compute center of mass
2. Count every such triples
3. Fix bugs.

H: Break a leg!

Solution – $\mathcal{O}(N \log(N))$ time complexity

1. Compute center of mass
2. Count every such triples
3. Fix bugs.

- easy given a triangulation of the polygon,
- but this is not needed: a signed triangulation suffices.

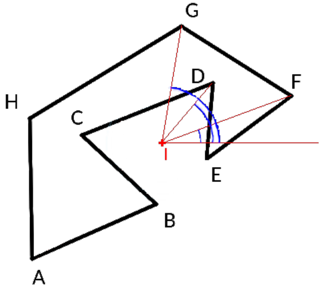


H: Break a leg!

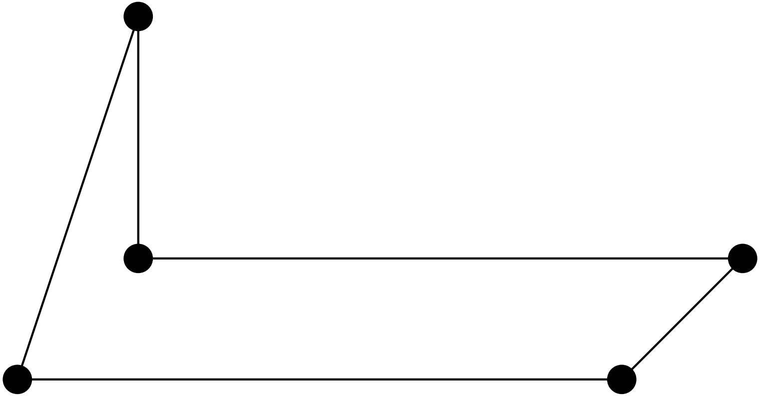
Solution – $\mathcal{O}(N\log(N))$ time complexity

1. Compute center of mass
2. Count every such triples
3. Fix bugs.

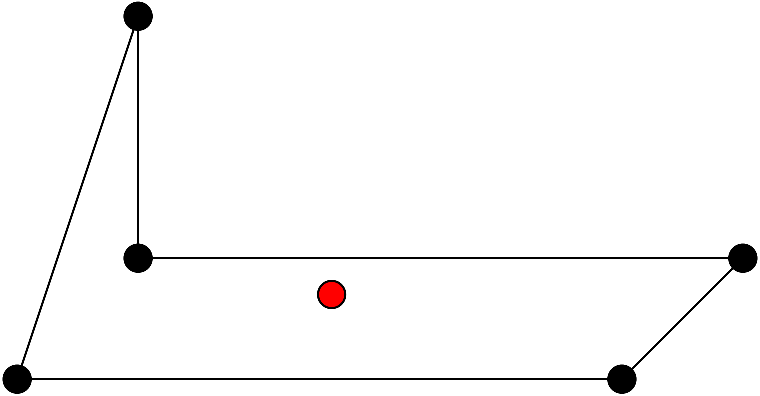
The only thing that matters is the angle formed with the center of mass (and any fixed line)



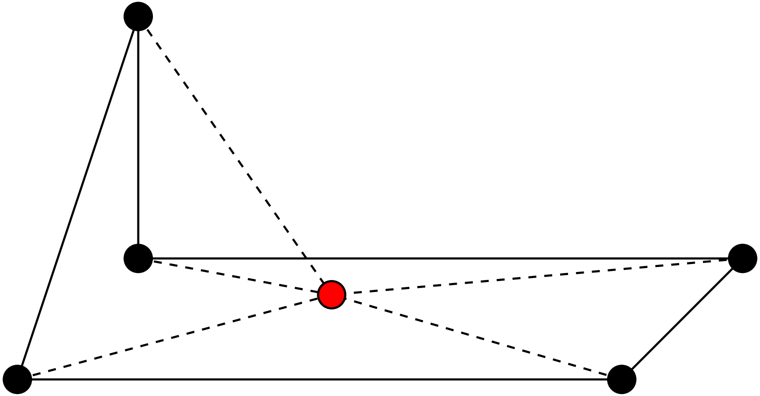
H: Break a leg!



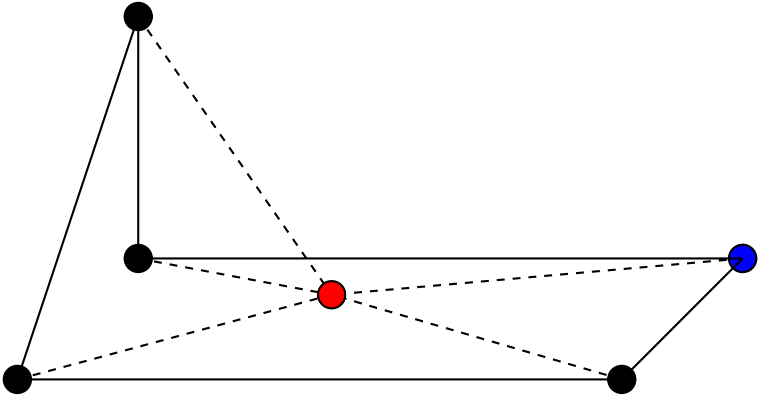
H: Break a leg!



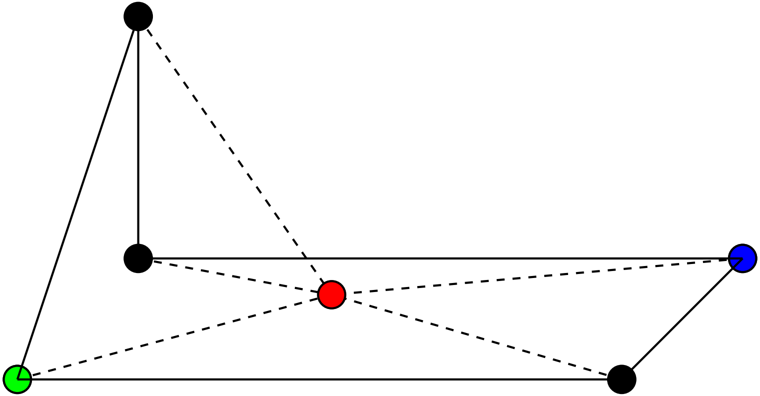
H: Break a leg!



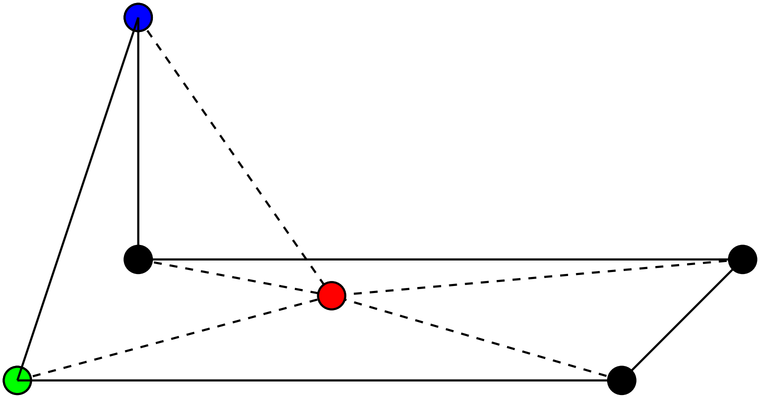
H: Break a leg!



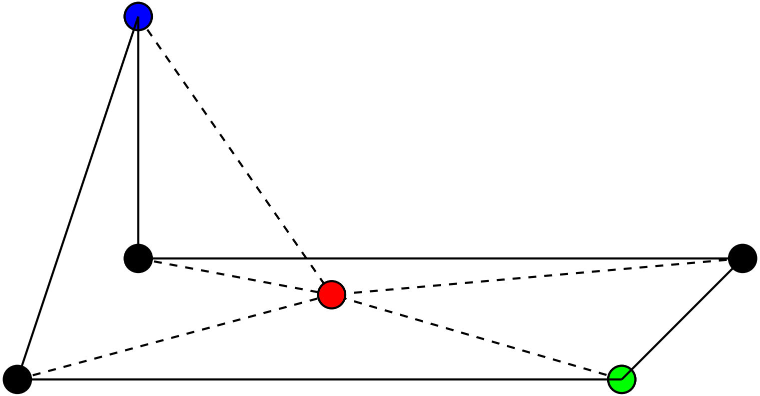
H: Break a leg!



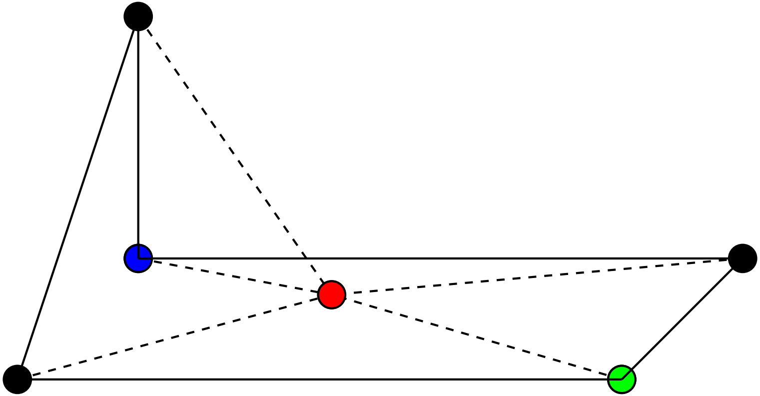
H: Break a leg!



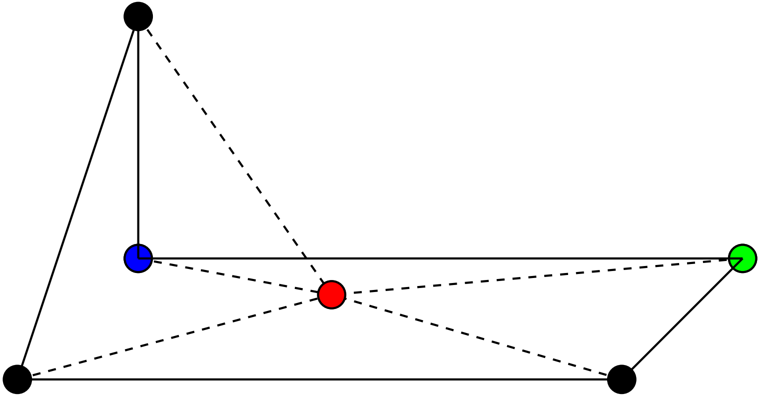
H: Break a leg!



H: Break a leg!



H: Break a leg!



H: Break a leg!

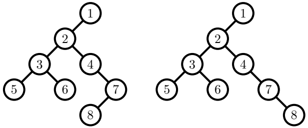
Solution – $\mathcal{O}(N \log(N))$ time complexity

1. Compute center of mass
2. Count every such triples
3. Fix bugs.

- Floats are too imprecise
- Exact representation of rationals may go out of bounds
- The center of mass may be on an edge
- The center of mass can be one of the vertices
- Two vertices may have the same angle

M: In-order

Not solved before freeze.



M: In-order

Problem

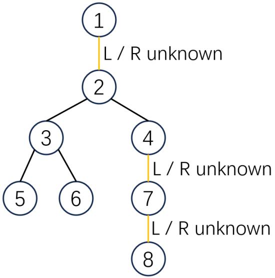
Given pre-order, post-order, a consecutive part of in-order. How many possible in-orders?

Solution – Linear time & linear space

Only knowing the pre-order and post-order cannot determine a binary tree, but we can determine the tree to the following extent –>

(1) Question: if none of the inorder is known, how many possible inorders are there?

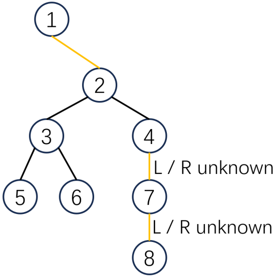
Answer: $2^{\text{number of nodes with 1 child}}$.



Solution – Linear time & linear space

(2) Question: if only root's position is known?

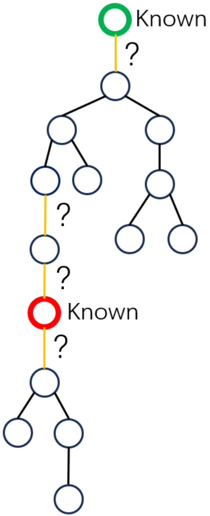
Answer: In addition to (1), we know the root's child (l/r) location (if the root has 1 child).



M: In-order

Solution – Linear time & linear space

(3) Question: if root + one other node X's positions are known?



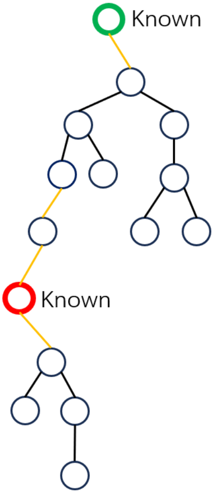
Solution – Linear time & linear space

(3) Question: if root + one other node X's positions are known?

Answer: we can determine the tree to the following extent –>

(4) Question: if root + multiple nodes' positions are known?

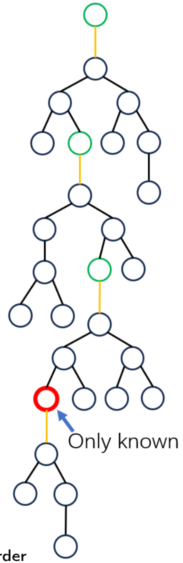
Answer: For all known nodes' all ancestors, if it has 1 child, its child (l/r) location is determinate.



Solution – Linear time & linear space

(5) Question: if only 1 node X's position is known?

Answer: Only X itself's and its ancestors' child (l/r) location (if it has 1 child) are relevant to its position in inorder. Therefore, we need to calculate a combination.



Solution – Linear time & linear space

(6) Question: if multiple nodes' positions are known?

Answer: A continous part in the inorder must contain a single node with the maximum height, we call it X.

Then we combine two independent things:

- the child (l/r) location of X's ancestors
- the child (l/r) location of the nodes in the subtree whose root is X