Problem Analysis Session

SWERC judges

December 2, 2018

Statistics

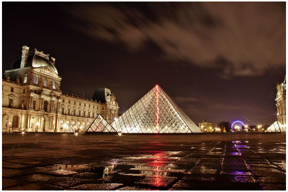
Number of submissions: about 2500
Number of clarification requests: 28 (20 answered “No comment.”)

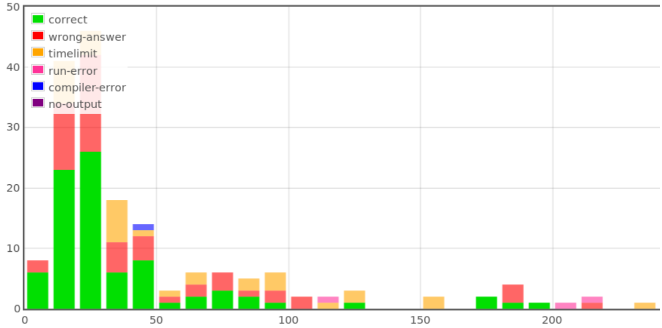
Languages:

- 1533 C++
- 34 C
- 232 Java
- 330 Python 2
- 233 Python 3

A – City of Lights

Solved by 83 teams before freeze.
First solved after 6 min by **Team RockETH**.





A – City of Lights

This was the easiest problem of the contest.

Problem

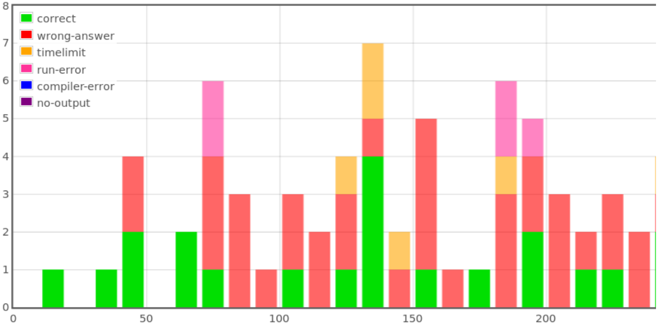
Toggle regularly spaced lights at every step, and print the maximum number of turned-off lights.

Straightforward solution

- Keep an array with the light status (or a bit set).
- Keep the number of currently turned-off lights in a variable.

K – Dishonest Driver

Solved by 18 teams before freeze.
First solved after 17 min by **Team RaclETH**.



K – Dishonest Driver

Problem

Given a string, compute the length of its shortest compressed form.

How to build a compressed form:

- one character c (size: $|c| = 1$),
- concatenation $w_1 w_2$ (size: $|w_1 w_2| = |w_1| + |w_2|$),
- repetition $(w)^n$ (size: $|(w)^n| = |w|$).

K – Dishonest Driver

Solution in time $\mathcal{O}(N^3)$

Dynamic programming on:

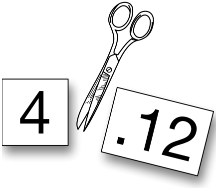
$$F(i,j) = \text{size of compressed form of substring } u_{ij} = u_i \dots u_{j-1}$$

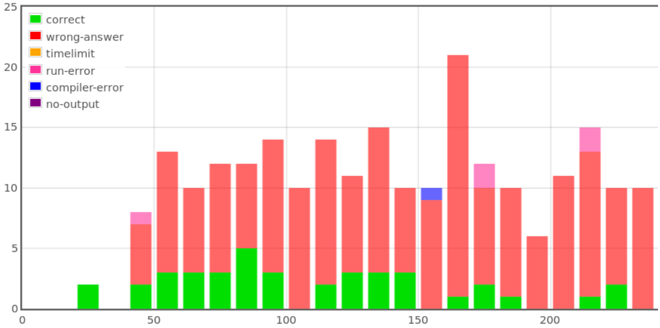
If $j = i+1$, then $F(i,j) = 1$. Otherwise:

- Try splitting $u_{ij} = u_{ik} u_{kj}$ for any position $k \in [i+1, j-1]$;
- Try factorizing u_{ij} into $u_{ij} = u_{ik}^n$:
  - What are the factorizations of u_{ij}?
  - Trick: search second occurence of u_{ij} in $u_{ij} u_{ij}$
  - $\mathcal{O}(N)$ with KMP (e.g., use C++ stdlib `find` function)

Note: we also have a $\mathcal{O}(N^2 \log N)$ algorithm

Solved by 39 teams before freeze.
First solved after 23 min by **SNS 1**.





E – Rounding

First bounds

Each monument **m** with rounded value $\mathbf{round_m}$ had an original value $\mathbf{origin_m}$ such that:

- $\mathbf{origin_m} \geqslant \mathbf{min_m}$, with $\mathbf{min_m} = \max\{0, \mathbf{round_m} - 0.50\}$;
- $\mathbf{origin_m} \leqslant \mathbf{max_m}$, with $\mathbf{max_m} = \min\{100, \mathbf{round_m} + 0.49\}$.

Possible or not?

Possible **if and only if**

$$\sum_{\mathbf{m}} \mathbf{min_m} \leqslant 100 \leqslant \sum_{\mathbf{m}} \mathbf{max_m}.$$

E – Rounding

Solution

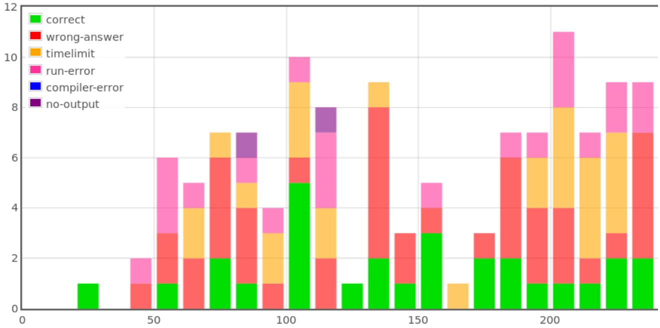
- Compute **minSum** $= \sum_m \mathbf{min_m}$ and **maxSum** $= \sum_m \mathbf{max_m}$
- Return `IMPOSSIBLE` if **minSum** > 100 or **maxSum** < 100
- **Real minimal value** for monument **m**:
  $\mathbf{realMin_m} = \max\{\mathbf{min_m}, \mathbf{max_m} - (\mathbf{maxSum} - 100)\}$
- **Real maximal value** for monument **m**:
  $\mathbf{realMax_m} = \min\{\mathbf{max_m}, \mathbf{min_m} + (100 - \mathbf{minSum})\}$

Main causes for wrong answers

- Allowing original values < 0 or > 100
- Using floating point numbers
- Result formatting issues

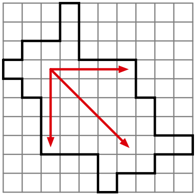
B – Blurred pictures

Solved by 28 teams before freeze.
First solved after 29 min by **UPC-1**.



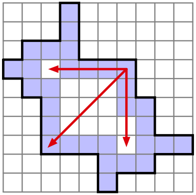
B – Blurred pictures

Dynamic programming **on the grid** would take time $\mathcal{O}(N \times N) \longrightarrow$ time limit exceeded



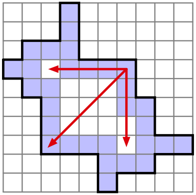
Dynamic programming **on the grid** would take time $\mathcal{O}(N \times N) \longrightarrow$ time limit exceeded

Note that perimeter is in $\mathcal{O}(N)$ and use it to compute only the mandatory extreme values in time $\mathcal{O}(N)$.



B – Blurred pictures

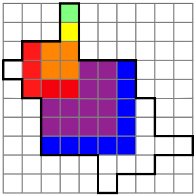
Dynamic programming **on the grid** would take time $\mathcal{O}(N \times N) \longrightarrow$ time limit exceeded



Note that perimeter is in $\mathcal{O}(N)$ and use it to compute only the mandatory extreme values in time $\mathcal{O}(N)$.

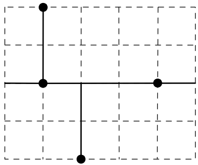
Even simpler:

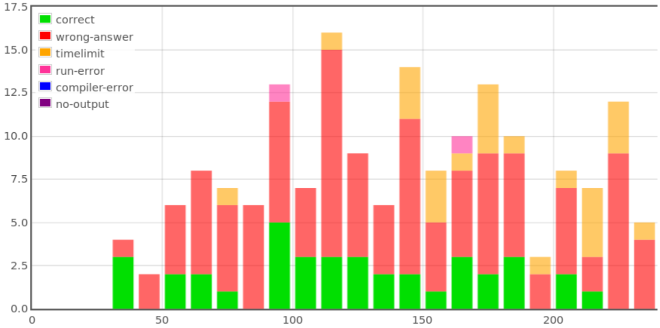
- You only need to keep track of the size of the largest square.
- Start from the first line and grow the maximum square from there, increasing its size at each new line when possible, else changing the starting line.



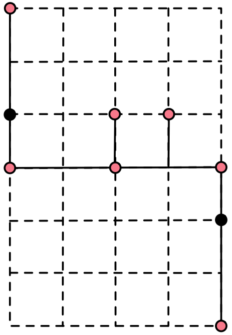
D – Monument Tour

Solved by 38 teams before freeze.
First solved after 37 min by **Blaise1**.





D – Monument Tour



Coordinates

0, 2, 2, 2, 3, 3, 3, 6

Solution

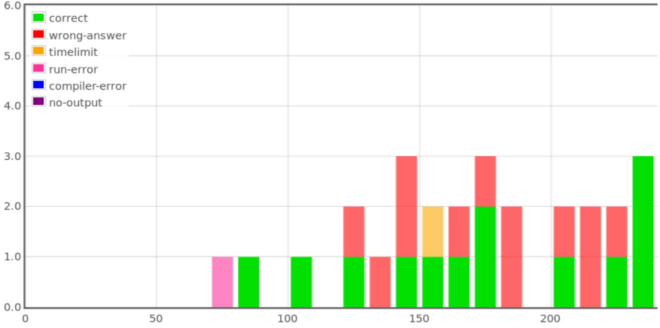
- the main road will always pass through at least one monument
- the best placement is the median of the y coordinates of the extreme points of "monument segments"

Monument Segment

- keep only the extremes of y coordinates corresponding to the same x
- count single points as a segment (i.e., count y coordinate twice)

F – Paris by Night

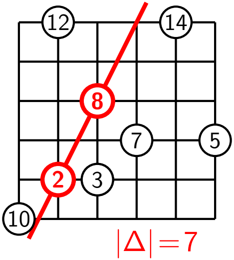
Solved by 13 teams before freeze.
First solved after 83 min by **Team RaclETH**.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

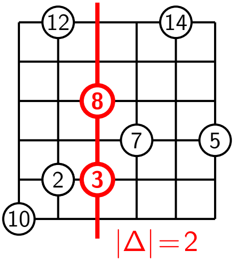
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M}'$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M}'}$ from scratch.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

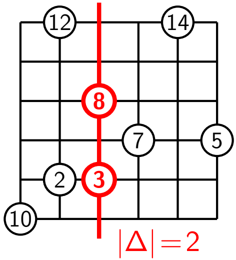
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M}'$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M}'}$ from scratch.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

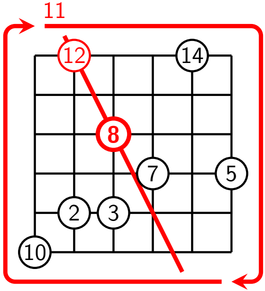
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M'}$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M'}}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

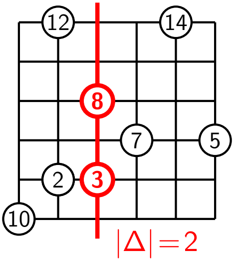
For all limiting monuments $\mathbf{M}$:

- order monuments $\mathbf{M'} \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M'})$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M'}}$ incrementally.



Naive approach in time $\mathcal{O}(N^3)$

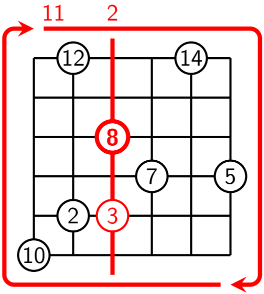
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M'}$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M'}}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

For all limiting monuments $\mathbf{M}$:

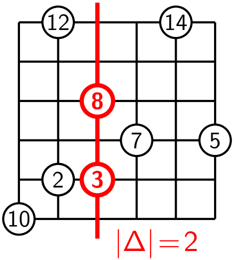
- order monuments $\mathbf{M'} \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M'})$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M'}}$ incrementally.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

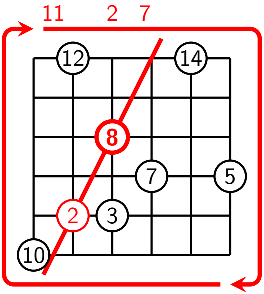
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M}'$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M}'}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

For all limiting monuments $\mathbf{M}$:

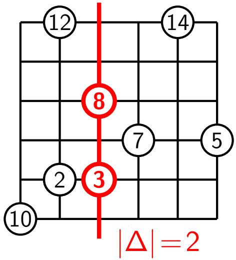
- order monuments $\mathbf{M}' \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M}')$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M}'}$ incrementally.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

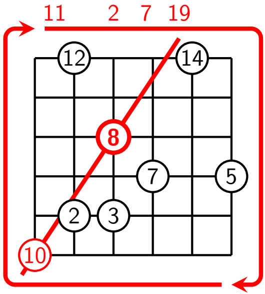
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M'}$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M'}}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

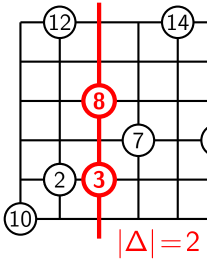
For all limiting monuments $\mathbf{M}$:

- order monuments $\mathbf{M'} \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M'})$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M'}}$ incrementally.



Naive approach in time $\mathcal{O}(N^3)$

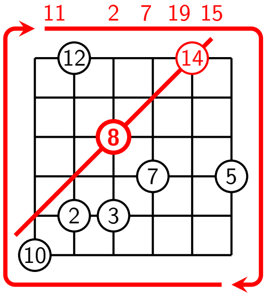
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M'}$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M'}}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

For all limiting monuments $\mathbf{M}$:

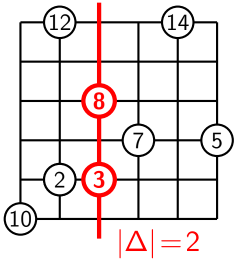
- order monuments $\mathbf{M'} \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M'})$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M'}}$ incrementally.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

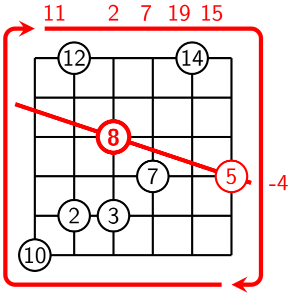
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M}'$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M}'}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

For all limiting monuments $\mathbf{M}$:

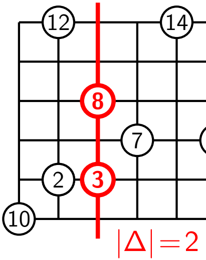
- order monuments $\mathbf{M}' \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M}')$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M}'}$ incrementally.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

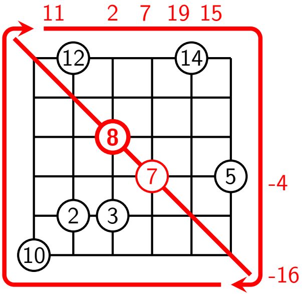
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M'}$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M'}}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

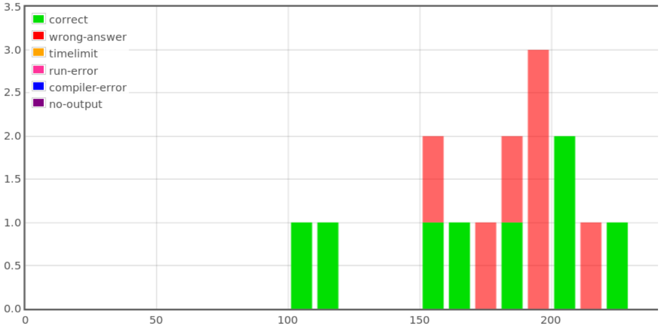
For all limiting monuments $\mathbf{M}$:

- order monuments $\mathbf{M'} \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M'})$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M'}}$ incrementally.



I – Mason's Mark

Solved by 8 teams before freeze.
First solved after 100 min by **ENS Ulm 1**.

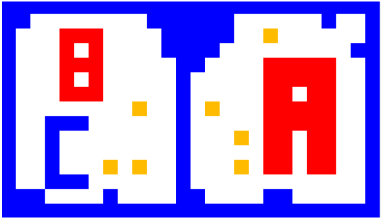


I – Mason's Mark

Many solutions are possible. For example:

Find connected components in a grid

Black dots form connected components, one of them contains the frame, others are single noise dots, and the remaining correspond to marks.

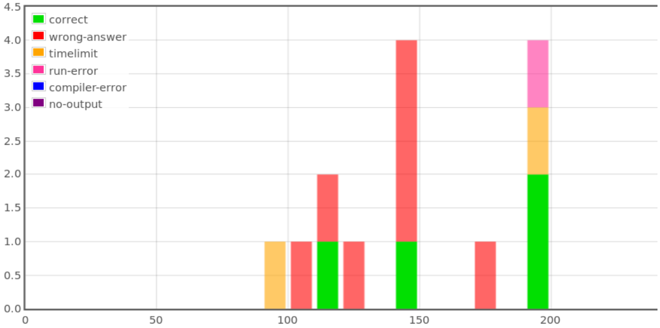


One possibility

Let M be manson's mark. Determining its bounding box. Now either inspect two particular points, or comparing the size of M with a threshold, in order to determine the type of M.

H – Travel Guide

Solved by 4 teams before freeze.
First solved after 118 min by **Team RacIETH**.



H – Travel Guide

Moving from a graph problem towards a vector problem

Three passes of Dijkstra algorithm to compute the distance from each POI to each node. $\mathcal{O}(|E| \times \log(|E|))$

We sort the vectors by lexicographical order.

$$\begin{array}{lll} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ \dots & & \\ x_n & y_n & z_n \end{array}$$

Key observation

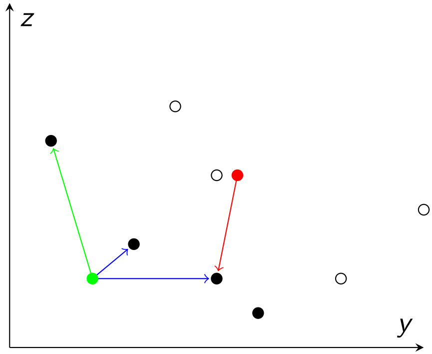
A vector v_i is minimal *iff* it is minimal among the vectors $v_1, \dots, v_i$ without considering the x coordinate.

H – Travel Guide

Idea: Maintain the 2D minimal vectors

Maintain a list of minimal vectors sorted by increasing y with a tree.

Note that it is sorted by decreasing z!



Checking that (y, z) is minimal

Is $z < z'$ for all (y', z') with $y' < y$?

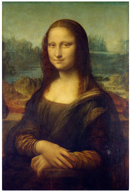
Inserting (y, z) as a minimal

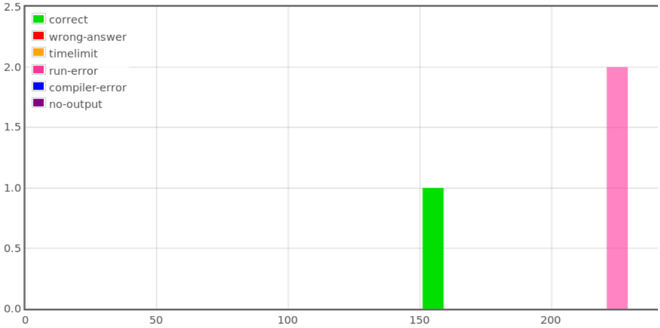
Remove all $z < z'$ and $y' < y$?

Note that you need to deal with duplicates.

J – Mona Lisa

Solved by 1 team before freeze.
First solved after 154 min by **ENS Ulm 1**.





J – Mona Lisa

Problem

Given 4 streams X_1, X_2, X_3, X_4 of pseudo-random n-bit integers, find $x_1 \in X_1, \dots, x_4 \in X_4$ such that $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0$.

Naive solution in $\mathcal{O}(2^{n/2})$ (exceeds time limit)

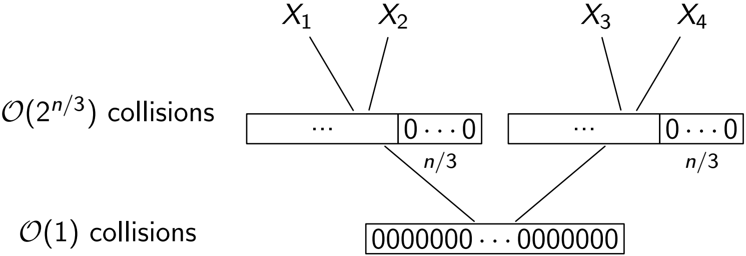
- Store $\mathcal{O}(2^{n/2})$ values from X_1 in a hashmap.
- Pick $x_3 \in X_3$ and $x_4 \in X_4$ arbitrarily.
- Iterate over $x_{2,i} \in X_2$, look for $x_{2,i} \oplus x_3 \oplus x_4$ in the hashmap.
- We expect to find a match after $\mathcal{O}(2^{n/2})$ steps by Birthday Paradox.

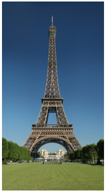
J – Mona Lisa

Solution in $\mathcal{O}(2^{n/3})$ (space and time)

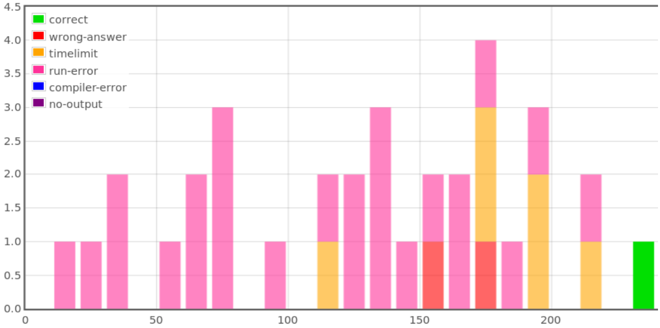
- Build a list of $x_1 \oplus x_2$ when x_1 and x_2 match on their $n/3$ least significant bits. When using $\mathcal{O}(2^{n/3})$ values from X_1 and X_2, the list has $\mathcal{O}(2^{n/3})$ elements by Birthday Paradox.
- Do the same on X_3, X_4.
- The two lists generated have $\mathcal{O}(2^{n/3})$ elements of only $2n/3$ bits. By Birthday paradox, we expect $\mathcal{O}(1)$ matches.

J – Mona Lisa


X_1
X_2
X_3
X_4
$\mathcal{O}(2^{n/3})$ collisions
$\cdots$
$0 \cdots 0$
$\cdots$
$0 \cdots 0$
$n/3$
$n/3$
$\mathcal{O}(1)$ collisions
$0000000 \cdots 0000000$



Solved by 1 team before freeze.
First solved after 235 min by **ENS Ulm 1**.



G – Strings

Source

Ropes: an Alternative to Strings
Boehm, Atkinson, Plass, 1995

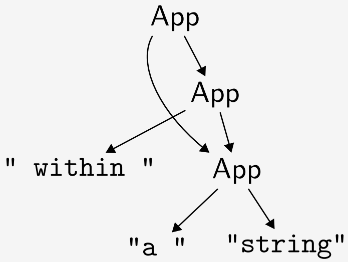
Main ideas

- do not concatenate strings, build binary trees instead
- ropes are immutable, thus sharing is possible

Implementation

- rope length in $\mathcal{O}(1)$
- substring of a leaf in $\mathcal{O}(1)$, else recursively in $\mathcal{O}(N)$

Example

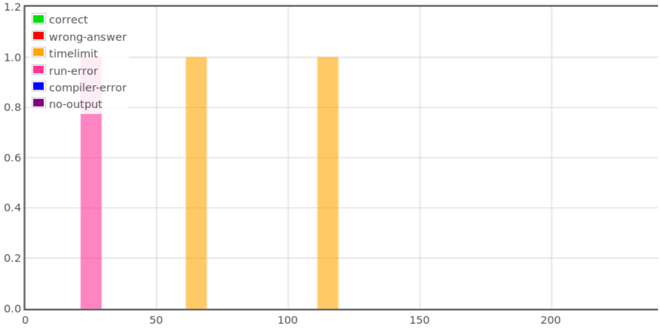


Overall complexity

$\mathcal{O}(N^2)$

C – Crosswords

Not solved before freeze.



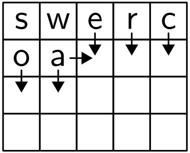
C – Crosswords

Source

Knuth, The Art of Computer Programming
forthcoming volume 4B, pre-fascicle 5b Introduction to Backtracking
Word Rectangles (page 8)

Backtracking Algorithm

- fill the grid, in any order
- +1 when completely filled



Data Structure

- build two tries, for horizontal and vertical words
- maintain pointers into these tries, for the columns and the row
- speed up the lookup at the intersection with sparse, sorted branches in your tries (see ex. 28)