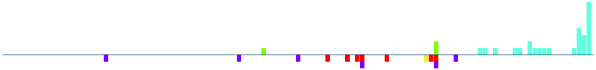
BAPC 2020

Solutions presentation

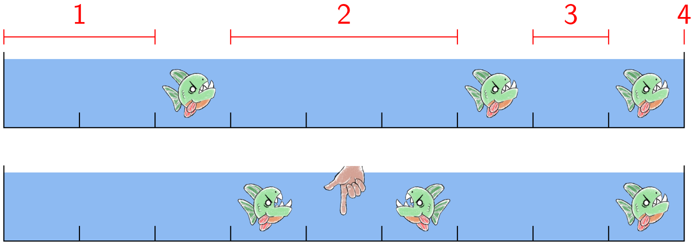
December 15, 2020

A: Aquarium Arrangement

Problem Author: Freek Henstra

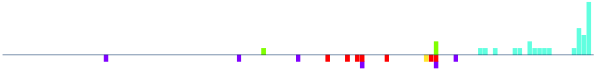


- Problem: how long does it take to move piranhas to the correct positions by luring them with your finger?
- Piranhas cannot pass each other, so we know where each piranha needs to end up.
- Denote by a_i the number of positions piranha i needs to move to the right (negative if the piranha needs to move to the left.)
- There are $k + 1$ intervals between piranhas or between a piranha and a wall.



A: Aquarium Arrangement

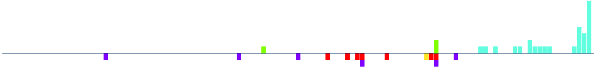
Problem Author: Freek Henstra



- If we spend x seconds in interval 1,
  - we need to spend $x + a_1$ seconds in interval 2,
  - we need to spend $x + \sum_{i=1}^{j-1} a_i$ seconds in interval j.
- Pick x as small as possible such that all these values are non-negative.
- Claim: it is possible with minimal x iff it is possible.
  - Spending 1 second in each interval is equivalent to doing nothing.
  - In a solution where x is not minimal, we can skip the first second in each interval.
  - Consider a step from the original solution in an interval i that is not skipped.
  - Suppose we have spent a seconds in interval $i-1$, b in i and c in $i+1$.
  - In new solution, we have spent $\geq a-1$ in $i-1$, $b-1$ in i and $\geq c-1$ in $i+1$.
  - This is equivalent to having spent $\geq a$ in $i-1$, b in i and $\geq c$ in $i+1$.
  - Hence the step is still possible, if not easier, in the new solution.
- Each $|a_i| \leq n$, so the number of seconds per interval is $O(nk)$, total is $O(nk^2)$.

A: Aquarium Arrangement

Problem Author: Freek Henstra

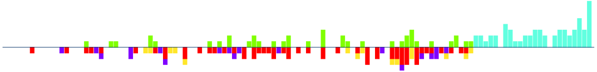


- To test whether it is possible, simulate until you are done or stuck.
- The order of steps does not matter: steps cannot make other steps impossible.
- Put your finger in the leftmost required interval.
- The new leftmost required interval is at most one position to the left.
- We move k more positions to the right than to the left.
- Total time complexity is $O(nk^2)$ amortized.
- Seems too large, but large k makes the problem easier and the constant is small.
- Simulation without precomputing the seconds per interval is also possible.
  - If a piranha needs to move left but cannot, recursively move piranhas left of it left.
  - For the right, replace left with right in the above statement.
  - Repeatedly loop through piranhas moving them left/right until done or stuck.
  - Finding the next step can take long, but in total still $O(nk^2)$ amortized.

Statistics: 44 submissions, 3 accepted, 27 unknown

B: Balanced Breakdown

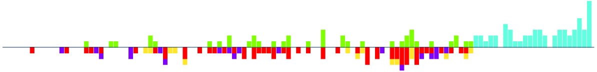
Problem Author: Ludo Pulles and Mike de Vries



- Problem: write a number as sum of ≤ 10 ‘balanced’ (palindrome) numbers.

B: Balanced Breakdown

Problem Author: Ludo Pulles and Mike de Vries

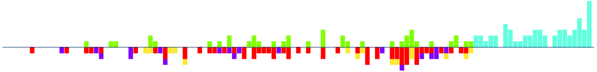


- Problem: write a number as sum of ≤ 10 'balanced' (palindrome) numbers.
- Idea: construct the biggest balanced number less than n greedily.
- Example:

$$
\begin{aligned}
n &= 970\,894\,988\,875\,162\,603 \\
p_1 &= \color{red}{970\,894\,987}\color{black}{\,7}\color{red}{89\,498\,079} \\
n - p_1 &= 1\,085\,664\,524 \\
\hline
p_2 &= \color{blue}{1\,085}\color{black}{\,555\,}\color{blue}{801} \\
n - p_1 - p_2 &= 108\,723 \\
&\ldots.
\end{aligned}
$$

B: Balanced Breakdown

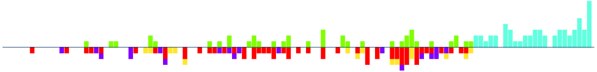
Problem Author: Ludo Pulles and Mike de Vries



- Possible edge case: $n = 10^{\ell}$: $p = n - 1$.

B: Balanced Breakdown

Problem Author: Ludo Pulles and Mike de Vries

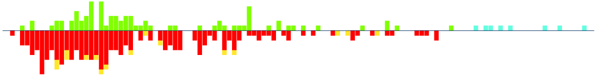


- Possible edge case: $n = 10^\ell$: $p = n - 1$.
- To get $k \leq 5$: run a brute force to express n as sum of three balanced numbers when $n \leq 200\,000$.

Statistics: 192 submissions, 40 accepted, 59 unknown

C: Corrupted Contest

Problem Author: Boas Kluiving



- Given only the time penalties of a valid scoreboard and the total number of problems, can you reconstruct the scoreboard uniquely?
- Idea: starting at the last team fill in the corrupted column conservatively.
- The last team solved $p_n = 1$ problem.
- For $i = n-1, \ldots, 1$

$$p_i = \begin{cases} p_{i+1}, & \text{if } t_i \leq t_{i+1} \\ p_{i+1} + 1, & \text{else.} \end{cases}$$

- If $p_1 = p$, then the scoreboard is non-ambiguous.
- Else $p_1 := p$ gives a different correct scoreboard, so ambiguous.
- Note: a team has a time penalty of 0 if and only if they solved 0 problems.
- Solution: $O(n)$.

- Two important edge cases:
  - It could be that all participants have solved at least 1 problem.

```
4 3
 40            3
 30     ⇒      2
 10            1
 20            1
```

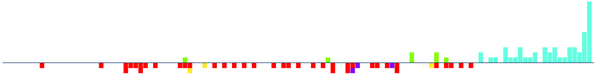
  - If every participant has solved 0 problems, the scoreboard is unambiguous.

```
3 10
  0            0
  0     ⇒      0
  0            0
```

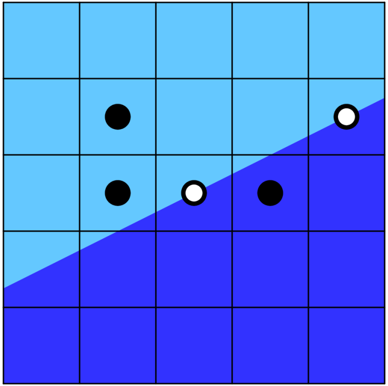
Statistics: 292 submissions, 78 accepted, 8 unknown

D: Destabilized Drone

Problem Author: Ragnar Groot Koerkamp



- Given a $w \times h$ grid and a line going through at least two of the points, find it by querying whether points are above, below, or on the line, using at most 900 queries.



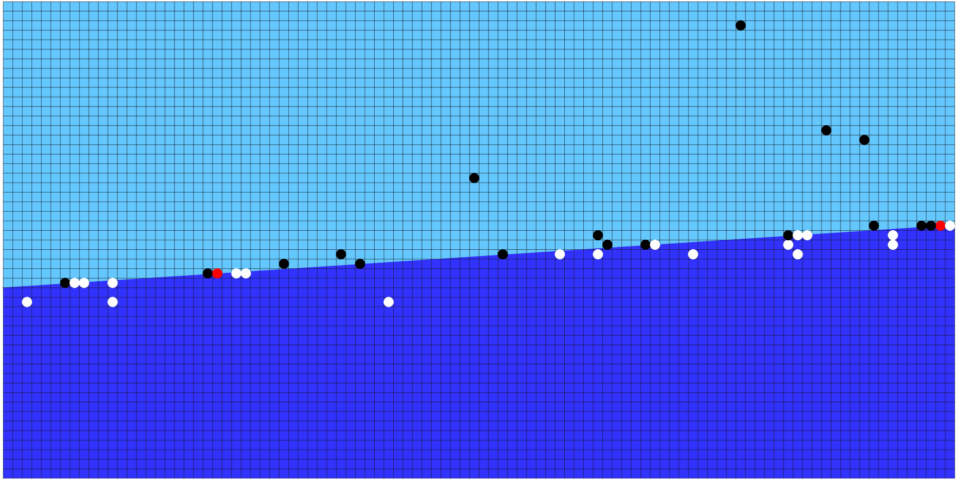
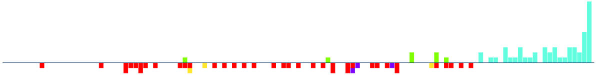


Figure: Generalized Convex Hull: very few queries, but difficult to implement (< 120 queries)

D: Destabilized Drone

Problem Author: Ragnar Groot Koerkamp



- Easier solution: Binary search on the left and right edge, using $2 \cdot \log_2 1000 \approx 20$ queries.
- This leaves a slice of at most 1000 points that must contain the line.
- Querying all 1000 points in (randomized) order uses too many queries!

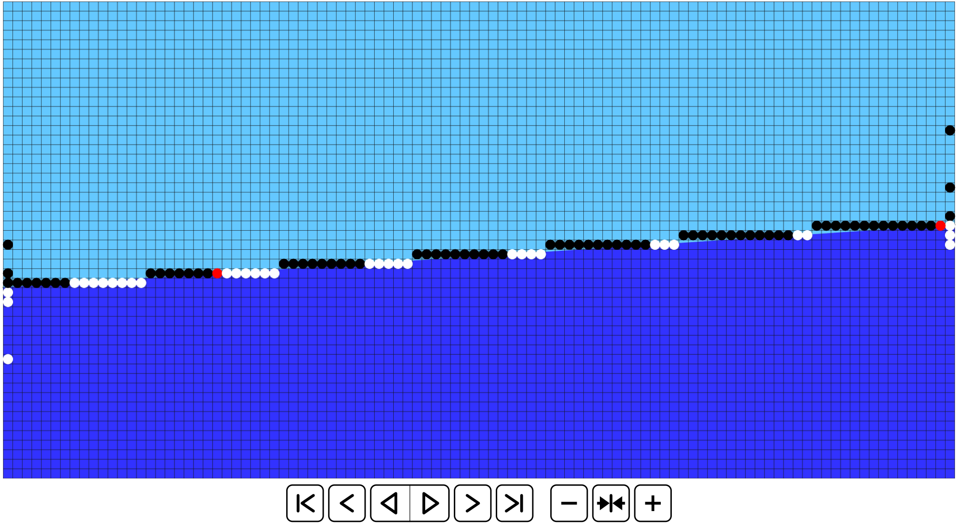


Figure: Binary search + linear scan: WA on large cases ($>$ 900 queries worst case)

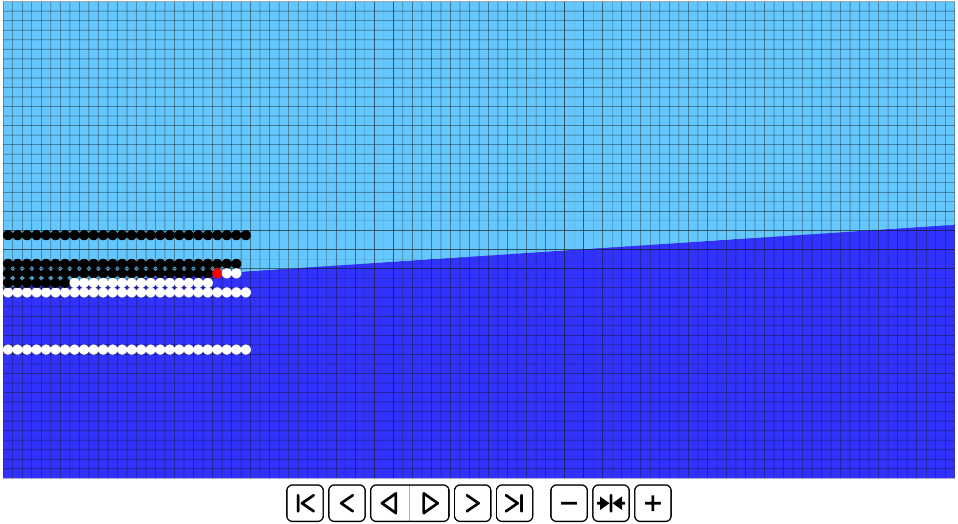
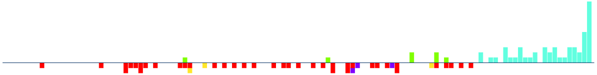


Figure: Binary search + per column: WA on large cases ($>$ 900 queries worst case)

D: Destabilized Drone

Problem Author: Ragnar Groot Koerkamp



- Solution: When P lies above the line, discard any points that lie between P and the upper edge of the slice. And similar for points below the line.

Statistics: 104 submissions, 7 accepted, 52 unknown

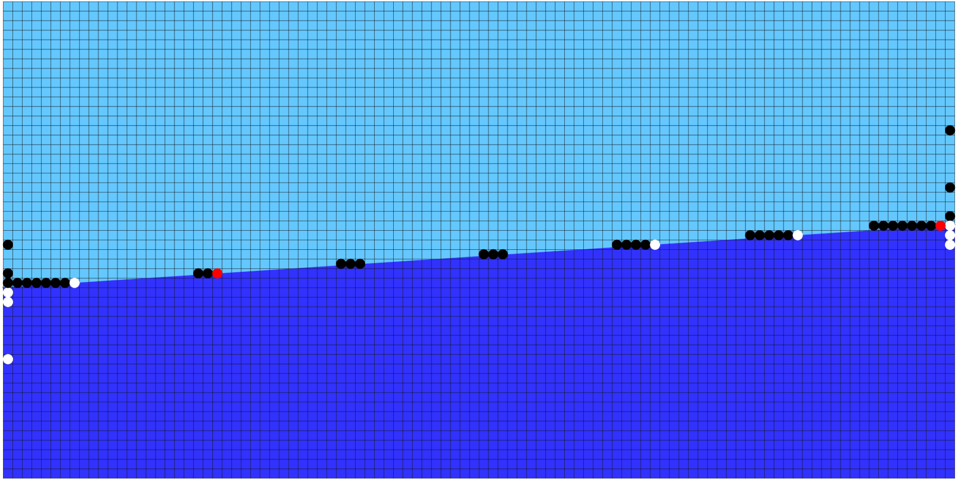


Figure: Binary search + pruning + linear scan: AC ($<$ 800 queries)

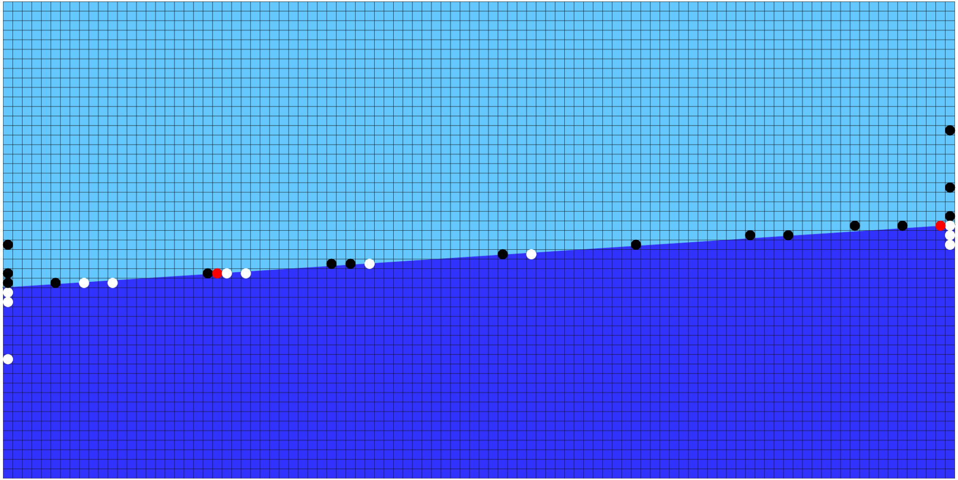


Figure: Binary search + pruning + randomized: AC (< 200 queries)

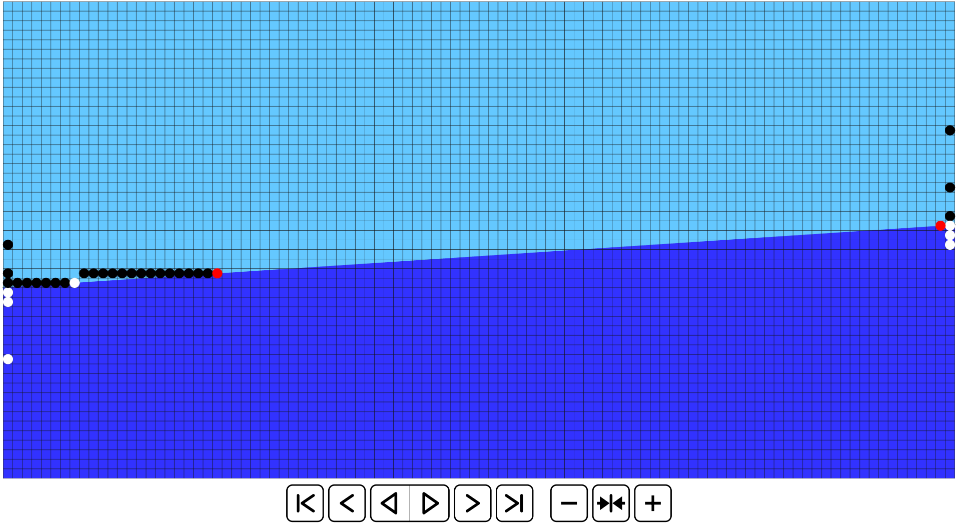
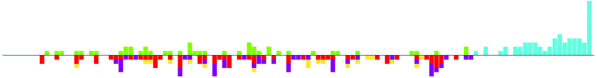


Figure: Binary search + linear scan from left and right: AC ($2/3 \cdot n + 20$ queries)

E: Efficiently Elevated

Problem Author: Mees de Vries



Problem:

- Build the least number of elevators so that all buildings become accessible.
- Need to count 'local maxima' in the floor plan, but only count each maximum once!

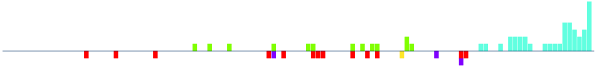
One possible solution:

- Sort all locations in the grid by height (descending).
- In the sorted order go through the locations:
  - If the location is marked 'done', skip it.
  - Do a flood fill/BFS/DFS from that location to all lower/equal location, and mark all those locations 'done'.
- Output the number of flood fills needed.

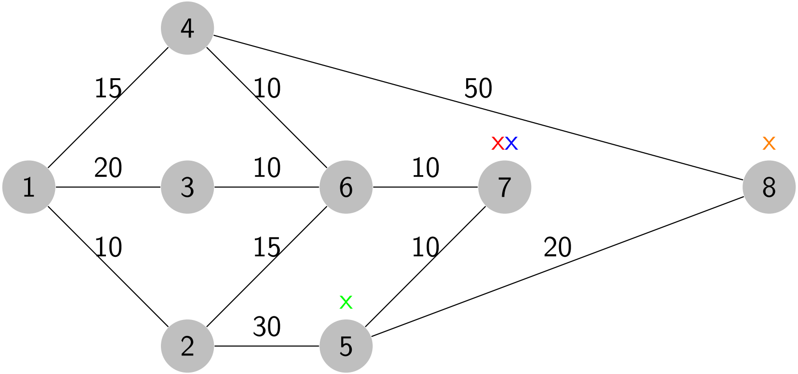
Statistics: 232 submissions, 46 accepted, 64 unknown

F: Family Fares

Problem Author: Boas Kluiving



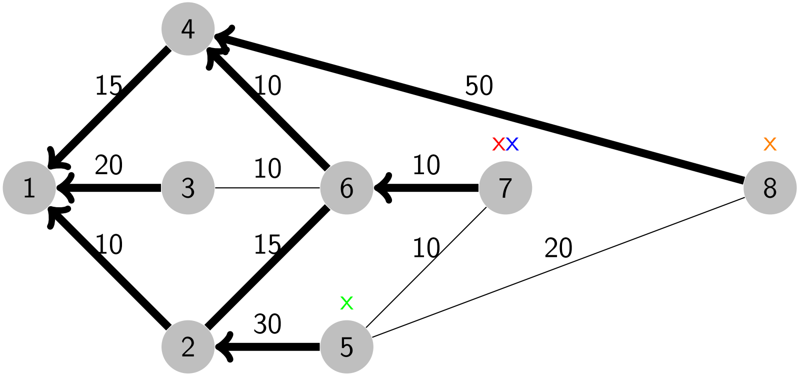
- Problem : From which station to buy the group ticket, so that the sum of the tickets is minimal. NB: you don't have to buy a group ticket.



F: Family Fares

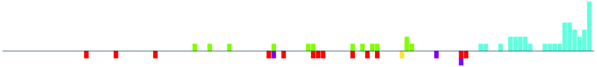
Problem Author: Boas Kluiving

- Step 1: Compute shortest path DAG starting in Delft (station 1).

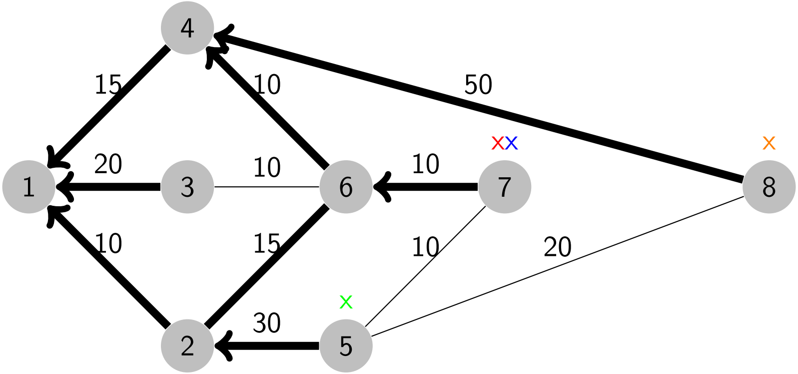


F: Family Fares

Problem Author: Boas Kluiving

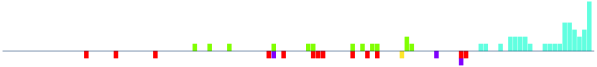


- Step 1: Compute shortest path DAG starting in Delft (station 1).
- Step 2: Compute total cost of tickets without a group ticket.

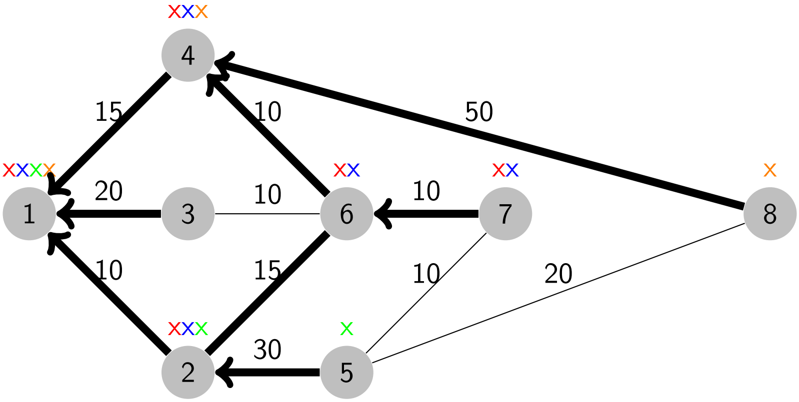


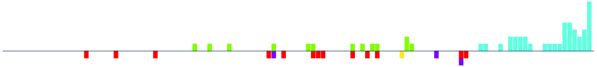
F: Family Fares

Problem Author: Boas Kluiving

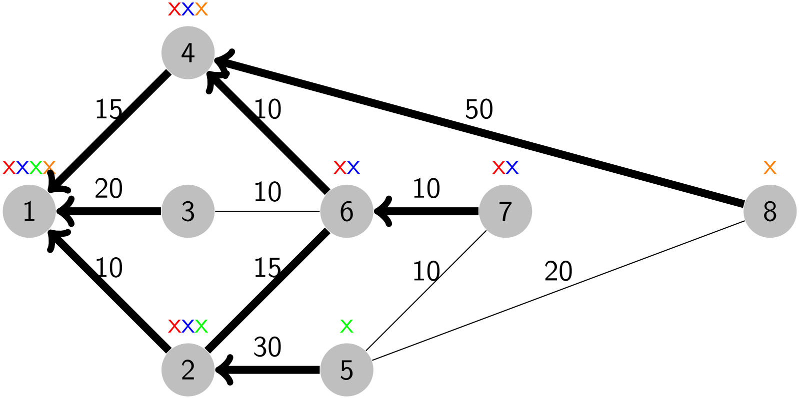


- Step 3: For every family member mark all stations which are on shortest paths from starting point to Delft.





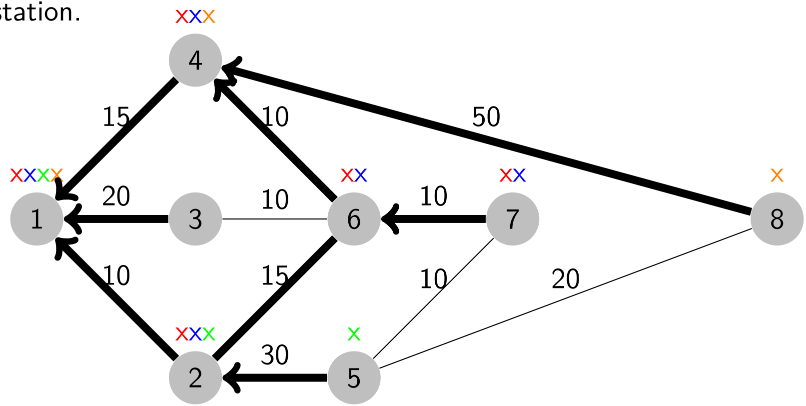
- Step 3: For every family member mark all stations which are on shortest paths from starting point to Delft.
- Step 3 (for higher p): Do this in one sweep by having a bitset at every station.



F: Family Fares

Problem Author: Boas Kluiving

- Step 4: Loop over all stations and compute savings when buying group ticket at that station.

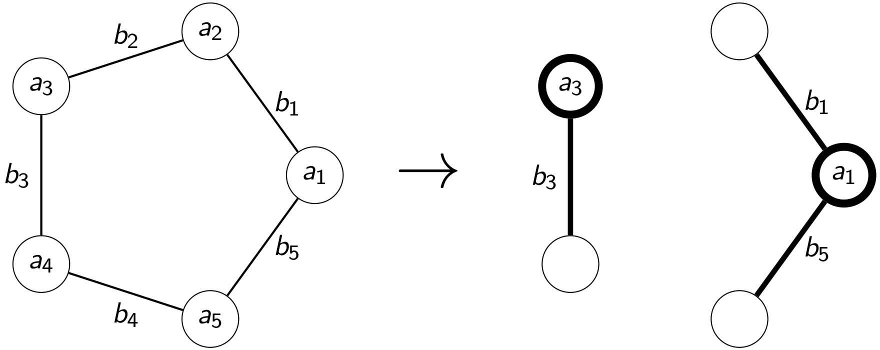


Statistics: 69 submissions, 13 accepted, 39 unknown

G: Generator Grid

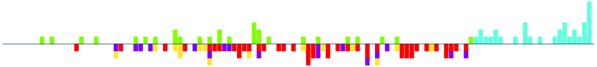
Problem Author: Timon Knigge

Problem: given a weighted cycle, pick some edges and vertices such that each vertex is connected to a marked vertex via a path.

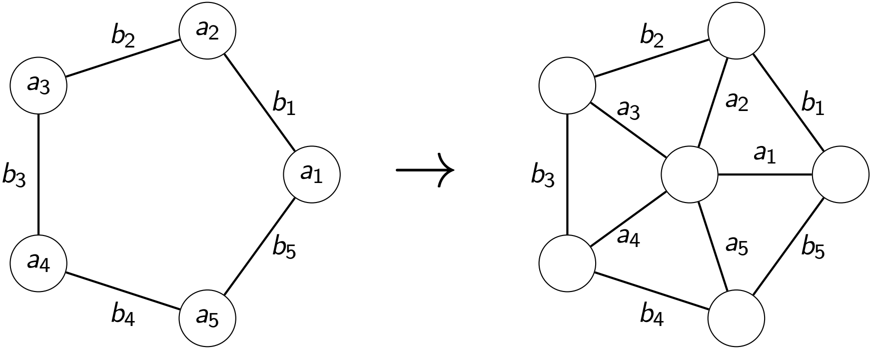


G: Generator Grid

Problem Author: Timon Knigge

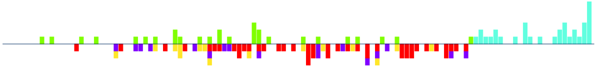


Idea: Add a central node for the concept of ‘power’:

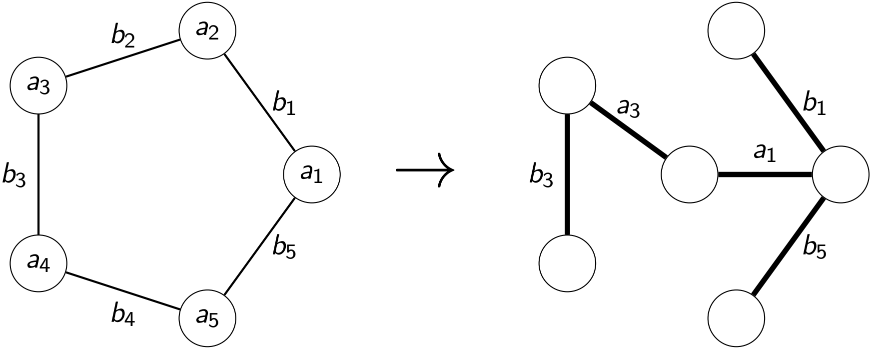


G: Generator Grid

Problem Author: Timon Knigge



Now we just want to find the cheapest way to connect each vertex to the central node.
But this is a classical minimum spanning tree problem $\rightarrow O(n \log n)$.



Note: this works for any graph G, not just cycles!
Statistics: 137 submissions, 28 accepted, 33 unknown

H: Hungry Henk

Problem Author: Pim Spelier, Mike de Vries, and Ragnar Groot Koerkamp

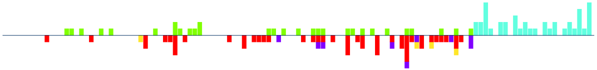
- Problem: help Henk by recommending him exactly one complete meal.
- Given: a list of complete meals, any of which would suffice.
- Solution: just choose any of them!

- Python one-liner: `print((input(), input())[1])`
- Kotlin one-liner: `print(Pair(readLine(), readLine()).second!!)`
- Brainfuck one-liner: `,,>>+[++++++++++>,----------]<[<]>>[.>]`

Statistics: 101 submissions, 86 accepted, 0 unknown

I: Incomplete Implementation

Problem Author: Jorke de Vlas



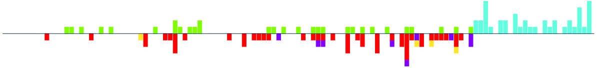
- Problem: sort an array by sorting half of it three times.

3	8	4	7	1	5	2	6

Figure: Unsorted array of the first sample

Statistics: 132 submissions, 30 accepted, 41 unknown

I: Incomplete Implementation

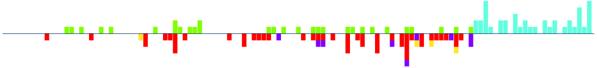


Problem Author: Jorke de Vlas

- Idea: in the first step, make sure that the first quarter is sorted.
- In the second step, make sure that the second quarter is sorted.
- In the final step, sort the remaining numbers.

I: Incomplete Implementation

Problem Author: Jorke de Vlas



- First step: making sure that the first quarter is sorted.
- Choose the first $n/4$ positions and the positions of the first $n/4$ numbers.
- This forces the first $n/4$ numbers into the first $n/4$ positions.

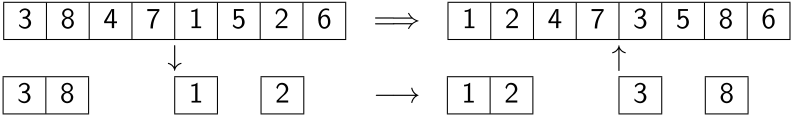
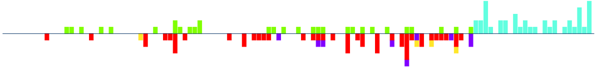


Figure: First sorting step

I: Incomplete Implementation

Problem Author: Jorke de Vlas



- Second step: making sure that the second quarter is sorted.
- Choose the next $n/4$ positions and the positions of the next $n/4$ numbers.
- When some of these overlap, choose arbitrary positions until the subarray is full.

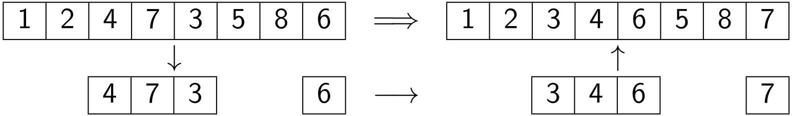
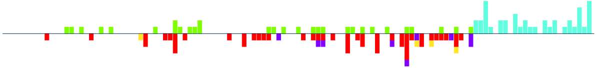


Figure: Second sorting step

I: Incomplete Implementation

Problem Author: Jorke de Vlas



- Final step: sorting the remaining numbers.

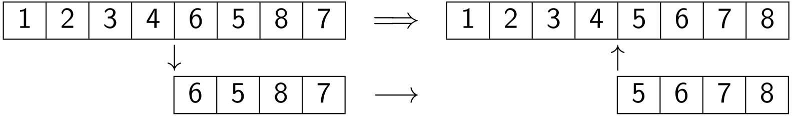
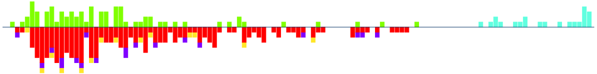


Figure: Final sorting step

Statistics: 132 submissions, 30 accepted, 41 unknown

J: Jigsaw

Problem Author: Mike de Vries

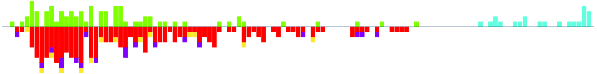


- Problem: find the dimensions of a jigsaw puzzle given the amount of edge, corner and center pieces.

Statistics: 305 submissions, 73 accepted, 22 unknown

J: Jigsaw

Problem Author: Mike de Vries



- Problem: find the dimensions of a jigsaw puzzle given the amount of edge, corner and center pieces.
- A jigsaw puzzle of size $w \cdot h$ contains:
  - 4 corner pieces
  - $2(h-2)+2(w-2)$ edge pieces
  - $(h-2)(w-2)$ center pieces
- This reduces the problem to a simple system of equations.

K: Xortest Path

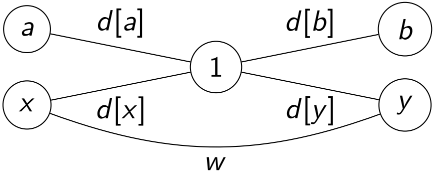
Problem Author: Jorke de Vlas

- Problem: given a connected undirected graph, find a path from a to b that minimizes XOR of the values on the edges.

K: Xortest Path

Problem Author: Jorke de Vlas

- Problem: given a connected undirected graph, find a path from a to b that minimizes XOR of the values on the edges.
- Build tree rooted at 1, and create a 'distance' array with $d[i]$ as XOR of path from 1 to i using tree edges.
- Any edge (x, y) of weight w gives option to XOR $v_{(x,y)} = d[x] \oplus d[y] \oplus w$ with answer.
- Claim: path from a to b has XOR-value $d[a] \oplus d[b]$ XOR-ed with some $v_{(x,y)}$'s.



K: Xortest Path

Problem Author: Jorke de Vlas

- Issue: there are $\approx 10^5$ edges with which the answer can be reduced, but most of them are "redundant".
- Solution: if there are cycles $c_1, c_2, \ldots, c_\ell$ all with a 1 in the ith bit, replace the cycle cost of c_j by

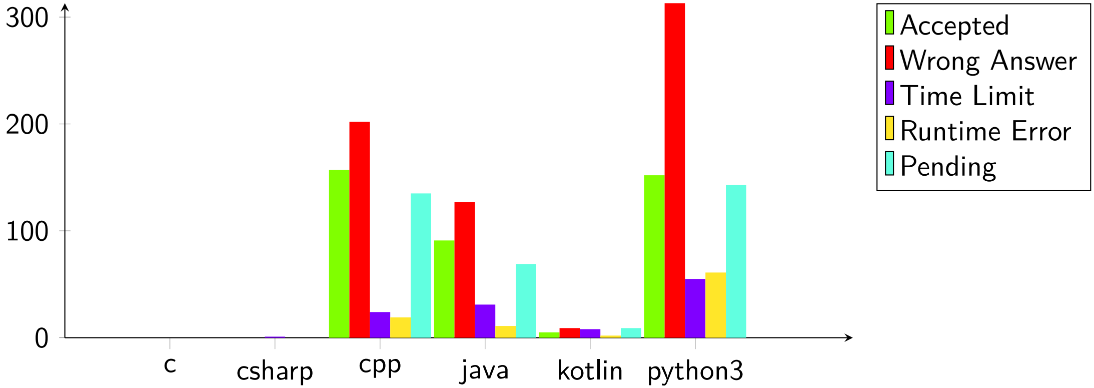
$$c_j \oplus c_1 \quad < 2^i, \qquad (2 \leq j \leq \ell).$$

(For mathematicians: do gaussian elimination over $\mathbb{F}_2$.)

- Removing zeros gives at most 64 non-zero cycles, each with distinct most-significant bit.
- For a query (a, b), try to remove the biggest 1s in the cost of $d[a] \oplus d[b]$, by using these 64 cycles.

Statistics: 15 submissions, 1 accepted, 11 unknown

Language stats



Some stats

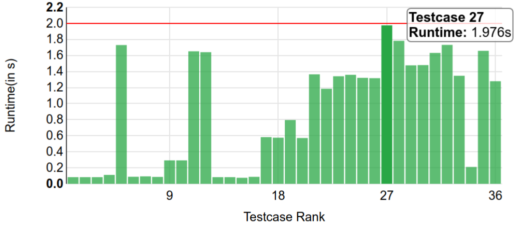
- 1400 commits
- 425 secret testcases
- 204 jury solutions
- The number of lines the jury needed to solve all problems is

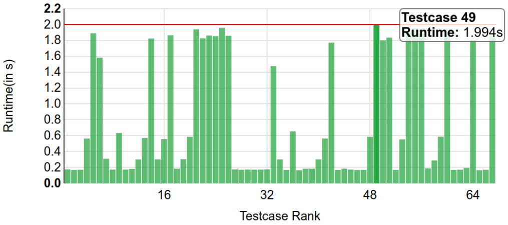
$$36 + 13 + 8 + 49 + 21 + 32 + 31 + 1 + 15 + 11 + 29 = 246$$

On average 22.4 lines per problem, up from 13.9 in the preliminaries!

Tips for next time

- Don't submit code for problems of the preliminaries
- Test your code on the provided samples
- Use the correct testing tool for the correct problem
- Write efficient code:





- Don't submit the testing tool.

While you were coding...

- In DOMjudge, 5 issues/feature requests were created, and 3 were fixed.
- In BAPCtools, 2 issues/FRs was found and 1 fixed.
- In the solve stats, 1 issue was found and fixed.

The Proofreaders

- Nicky Gerritsen
- Ian Pratt-Hartmann
- Michael Vasseur
- Kevin Verbeek
- David Wärn

The Jury

- Ruben Brokkelkamp
- Daan van Gent
- Ragnar Groot Koerkamp
- Joey Haas
- Freek Henstra
- Boas Kluiving
- Timon Knigge
- Ludo Pulles
- Maarten Sijm
- Harry Smit
- Pim Spelier
- Jorke de Vlas
- Mees de Vries
- Mike de Vries
- Wessel van Woerden