# Solutions – IDI Open 2022

March 19$^{th}$ 2022

# The Judges

(In alphabetical order)

- Jean Niklas L'orange
- Nils Barlaug
- Sander Land

Want to join us? Send an email to `jeannikl@hypirion.com` or contact me or the organizers after the presentation!

# Game Party

- Sum up all the people in the arriving groups, then add one (yourself)
- Take modulo 3 to find the remaining people:

Solved by 31 teams
First solution after 1 minute

# Another Ancient Cipher

- You're given a variant of the Vigenère cipher, and need to decrypt an encrypted message. Recall that the formula is

$$E_{Aug}(M_i, K_i) = \begin{cases} (M_i + K_i) \mod 26 & \text{if } M_i \text{ is even} \\ (M_i - K_i) \mod 26 & \text{if } M_i \text{ is odd} \end{cases}$$

- If , then we can compute all possibilities and store all combinations in a dictionary:

# Another Ancient Cipher

- You can then use `dec` to decode the message:

# Another Ancient Cipher

- But you can also deduce that $M_i \pm K_i \equiv C_i \pmod{2}$: If the parity of $M_i$ and $K_i$ are the same, then $C_i$ will be even. That gives us

$$D_{Aug}(C_i, K_i) = \begin{cases} (C_i - K_i) \mod 26 & \text{if } C_i \equiv M_i \pmod{2} \\ (C_i + K_i) \mod 26 & \text{if } C_i \not\equiv M_i \pmod{2} \end{cases}$$

Solved by 26 teams
First solution after 10 minutes

# Kattis Completionist

- Sort all numbers in ascending order, then group them so that all groups have at least $G$ points each. The number of groups is the answer
- Complicated by the fact that, $1.3 + 2.3 = 3.5999999999999996$ with floating point numbers
- Solution: Either convert the numbers to ints by removing the decimal point, or use an epsilon comparison instead of exact comparison

Solved by 23 teams
First solution after 5 minutes

# Changing Complexity

- Given a DAG with $V$ nodes, find a topological sorting that alternates between the most and least complex task available at the moment
- Find all tasks with no dependencies, and grow that set as you complete tasks
- We need a data structure that supports efficient insertion and removal of smallest and biggest element: Any kind of balanced search tree should do the job.

# Changing Complexity

- You can use `Set` in C++ or `TreeSet` in Java
- No alternative for Python in the standard library, but it's possible to use a set (to track which you've visited), a max heap and a min heap to do the same thing.
- Total worst case running time is $\mathcal{O}(V \log V)$

Solved by 9 teams
First solution after 29 minutes

# Healthy Headgear

- If you know the number of machines, you can compute the total time $T$ with a priority queue.
- And the time taken will be monotonically non-increasing with the number of machines.
- Use binary search to find the lowest amount of machines that runs within the time limit.
- NB: It's possible to receive in 0 requests, in which case the answer is 0.

Solved by 7 teams
First solution after 18 minutes

# Frugal Ferry Fees

- Shortest path with a lot of queries and over a big graph
- Dijkstra too slow (too many queries) and Floyd-Warshall too slow (too big graph)
- Key insight: All nodes reachable from one another with zero cost can be considered one clique.
- Idea: Solve it by computing a mapping from node to clique, then computing Floyd-Warshall on the cliques

# Frugal Ferry Fees

# Frugal Ferry Fees

- The number of cliques can be at most $F + 1$: A ferry can (but doesn't have to) split two groups of nodes into different cliques
- Floyd-Warshall on $F < 300$ is fast enough, giving us $\mathcal{O}(V + F^3 + Q)$: The time to find the cliques, Floyd-Warshall on the cliques, then computing the requested paths

Solved by 7 teams
First solution after 40 minutes

# Delicious Diet for Ducks

- Dynamic programming
- Multiple ways to run the DP, but only some will work in Python. Here's one:
- If you sort all the shop's duck treats in reverse, the index will represent the number of treats left to evaluate (plus one). For one such state, we can store the probabilities of each different shop.
- We can use remaining treats left as a triple tuple representing the state

# Delicious Diet for Ducks

- Define the base case as a shop with zero treats left for one shop. Those give a 100% chance of using that shop.
- Run the DP by number of treats left in total: (1, 1, 1) is the lowest non-base case, which has 3 treats left
- Then do all valid permutations with 4 treats left ((2, 1, 1), (1, 2, 1), (1, 1, 2)) etc.

# Delicious Diet for Ducks

presoimgs/ddd-3.pdf

# Delicious Diet for Ducks

presoimgs/ddd-4.pdf

# Delicious Diet for Ducks

presoimgs/ddd-5.pdf

# Delicious Diet for Ducks

presoimgs/ddd-6.pdf

# Delicious Diet for Ducks

presoimgs/ddd-7.pdf

# Delicious Diet for Ducks

presoimgs/ddd-8.pdf

# Delicious Diet for Ducks

presoimgs/ddd-9.pdf

# Delicious Diet for Ducks

- Finding all the different permutations efficiently is the hard part (here C/Java makes it easier than Python)

Solved by 7 teams
First solution after 97 minutes

# BBBBBB

- Shortest path with a compound key (number of gravity flips and number of actions taken)
- Dijkstra is sufficient, but a node can be visited twice: One for each gravity direction.
- A lot of different edge cases: Read the pseudocode **carefully**

# BBBBBB

```
4 4
####
#@ #
# X#
#^##
```

Here the player immediately dies.

# BBBBBB

```
4 4
####
#@ #
#X #
#^##
```

Here the player immediately wins.

# BBBBBB

```
4 4
####
#@ #
# X#
##v#
```

Here the player wins in one step: Move right.

# BBBBBB

```
4 5
####
#@ #
# X#
#  #
####
```

Here the player wins in two steps: Move right and gravity flip.

# BBBBBB

Solved by 6 teams
First solution after 86 minutes

# Eyeballing Extraterrestials

- Given two ranges, add $M_i$ to each element in the range which has the least value on average. Well, except for one element, but we can programmatically remove that after we've fetched the range.
- Then perform $Q$ range queries
- Needs fast range *insert* and fast range *update*
- A Fenwick tree performs fast range inserts ($\mathcal{O}(\log n)$), but can only do efficient point updates ($\mathcal{O}(\log n)$)
- One solution: Double Fenwick. Strap on to your seat belts

# Eyeballing Extraterrestials

Let `tree1` and `tree2` be two Fenwick trees of size $N$. Then define `range_add` and `range_sum` as
(Where `tree_add` and `tree_sum` are Fenwick tree adds and sums)

- Why does this work? Maths.
- It's a well known trick explained on multiple different webpages, and they do a better job than I would during in this presentation. One such website would be `https://cp-algorithms.com/data_structures/fenwick.html`
- A segment tree will also do the job here, perhaps more intuitively

Solved by 4 teams
First solution after 94 minutes

# Laser-Linked Lighthouses

- Connect all towers to each other by line of sight
- Minimal spanning tree, where the minimal height is the "distance"/"cost" between two towers
- ...but what is the minimal height of two connected towers?

# Laser-Linked Lighthouses

presoimgs/lll-1.pdf

# Laser-Linked Lighthouses

We use the haversine formula to find the smallest $\theta$ on a sphere between two points[1]:

From here on, we can work on the data as a 2D problem.
Note: Must convert input from degrees to radians first.

[1] It's also possible to use 3D Euclidean geometry to find $\theta$.

# Laser-Linked Lighthouses

presoimgs/lll-2.pdf

# Laser-Linked Lighthouses

We can use the law of sines to get $r + h$:

$$r + h = \frac{r}{\cos(\theta/2)}$$

Since we want input in multiples of radiuses of Europa, we set $r = 1$ for convenience, and solve for that:

$$1 + h = \frac{1}{\cos(\theta/2)}$$

$$h = \frac{1}{\cos(\theta/2)} - 1$$

- Height can then be computed as follows:
- Then use a minimal spanning tree algorithm and pick the maximum height seen.

This problem had an error in the judge answer, causing it pick the last height instead of the maximum height. Therefore, it was solved by 0 teams during the competition. After the contest was over, rechecks found that *TBA* teams had one (or more) correct submission[s], the first submitted after *TBA* minutes.

Do note that several of the teams that got “wrong answer” got it changed into a “run time error” or a “time limit exceeded”. This could have given them information which would have made them eventually able to solve the problem during the competition.