Portal

Titus Cieslewski

Easy: Compute where Heidi lands

Heidi's flight trajectory

$$\begin{aligned} x &= v \cdot t \\ y &= \tfrac{1}{2}t^2 \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} x(y,v) &= v\sqrt{(2y)} \\ y(x,v) &= \tfrac{1}{2}\left(\tfrac{x}{v}\right)^2 \end{aligned}$$

Reverse equations to compute x given y and v

Check:

$y(x_c, v) \in [y_c, y_c + 1]$ ⇒ Heidi will hit cube c

$x(y_c, v) \in]x_c, x_c + 1]$ ⇒ Heidi will land on cube c

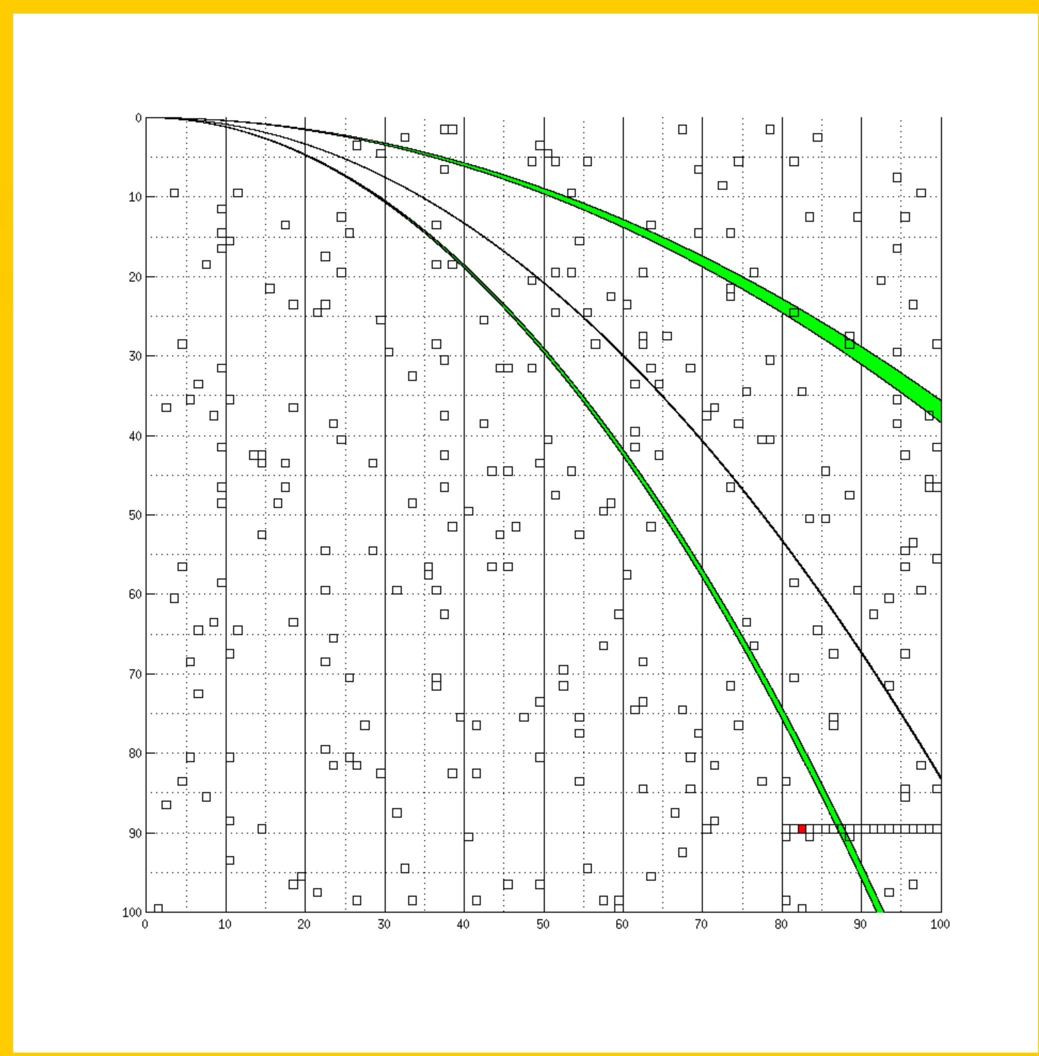
Medium: Can Heidi reach the wormhole?

First: Find all cubes from which Heidi can reach the wormhole.
Call these hittable and landable cubes.

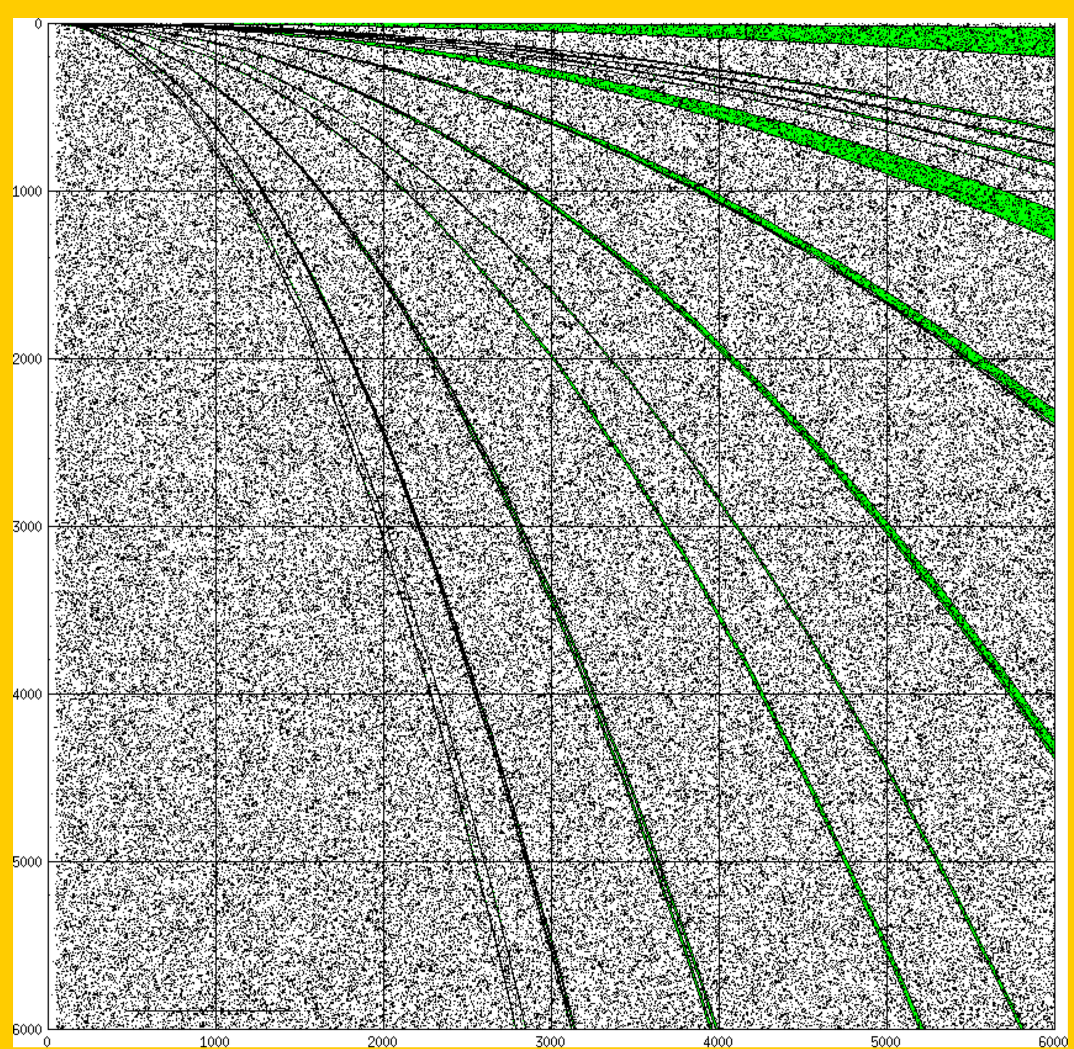
- Use a BFS or DFS, starting from the wormhole
- All cubes from which one can fall on a landable cube are landable
- All cubes just to the right and above a landable cube are hittable

This can be made much faster if cubes are sorted by x and y coordinates.

Given a speed, iterate through all cubes to find the one that Heidi hits first.



Hard: So many obstacles...



Processing all cubes for every speed is too slow :(
⇒ Build a whitelist/blacklist for speeds

Cubes can only be obstructed by cubes with smaller coordinates. Process cubes starting from the top left:

- If cube is landable/hittable
  - extract speed interval
  - subtract existing blacklisted speeds
  - insert speed interval into whitelist
- insert speed interval for non-hittable/non-landable sides into blacklist

Need a smart data structure, s.a. an interval tree.