Flood

CMIMC Optimization Round

April, 2025

> Even in the deepest waters, there is a place for the light to rest.
>
> *René Descartes*

1 Proposal

You control a collection of 64 bots that move around a 512×512 torus grid. Each square has some floating-point height, typically between 0 and 1000.

A flood approaches! The flood begins at height 0. All squares below the flood height is flooded, and any bots on these squares die. On each time step, the bot may move one step, and additionally broadcast a message of 32 bits. The bot may see the terrain within an 17x17 square centered at the bot (thus all squares with a Chebyshev distance at most 8). The bot additionally sees all messages broadcasted by other bots within the square from the previous time step. After the bot makes their action, the flood raises by height 1.

The flood only stops before the highest point, and all bots on the highest point live. The goal of the problem is for as many bots to survive by reaching the highest point.

Bots are each assigned a unique ID at the beginning, and informed of the terrain option. Their strategy may depend on their ID and terrain option. Bots do not know their absolute location on the grid.

2 Implementation

You are to implement a `Bot` class, and two methods, `init` and `step`. `init` takes an integer `index` between 0 and 63, and an integer `difficulty` representing the difficulty of the map they are on as an integer. 0 represents easy, 1 represents medium, and 2 represents hard.

`step` takes a 17x17 numpy array of floats `height` representing the height of the terrain in the field of view, and a list of triplet of ints `neighbors` where each element of `neighbors` is of the form (x, y, m). Here x, y are integers between -8 and 8, and (x,y) represents the relative location of the neighbor, and m is an integer between 0 and $2^{32} - 1$ representing the message broadcasted by the neighbor on the previous step.

3 Terrain

3.1 Terrain Easy

Terrain easy consists of a cubically interpolated terrain from a 16x16 grid of independent random numbers, normalized to range $[0, 1]$. The range $[0, 0.1]$ is mapped to a height $[0, 500]$ while the rage $[0.1, 1.0]$ is mapped to $[500, 680]$. Thus, 90% of the land has height above 500. Additionally, a higher frequency terrain interpolated from a 64x64 grid is added, with norm 50.

3.2 Terrain Medium

Terrain medium consists of piecewise constant height from an 8x8 grid. whose heights between 0 and 700 are independently selected with probability proportional to the height. Additionally, a higher frequency interpolated from a 64x64 grid is added, with norm 200.

3.3 Terrain Hard

God is mad that you are defying his intentions and surviving the flood. He summons bombs from the heavens and shatters the terrain with long narrow cracks.

Terrain easy consists of a cubically interpolated terrain from a 16x16 grid of independent random numbers, normalized to range $[0, 1]$. The range $[0, 0.15]$ is mapped to a height $[0, 750]$ while the rage $[0.15, 1.0]$ is mapped to $[750, 930]$. Thus, 85% of the land has height above 750. Additionally, a higher frequency interpolated from a 64x64 grid is added, with norm 50.

Finally, 8 horizontal cracks are generated along the terrain with random vertically stratified positions, with lengths of 400 blocks and widths of 3 blocks. Vertical distances between cracks mostly lie in the range $[30, 98]$. The heights on the cracks are capped to be $100i - 50$, where i is the i-th crack.

4 Evaluation

The randomness in terrain generation and initial position is seeded. For evaluation during the competition, you code will be evaluated on each terrain 20 times, with numpy seeds 0 to 19 used. For final evaluation, Your code will be evaluated on each terrain 40 times, with secret seeds used.

A Example terrains

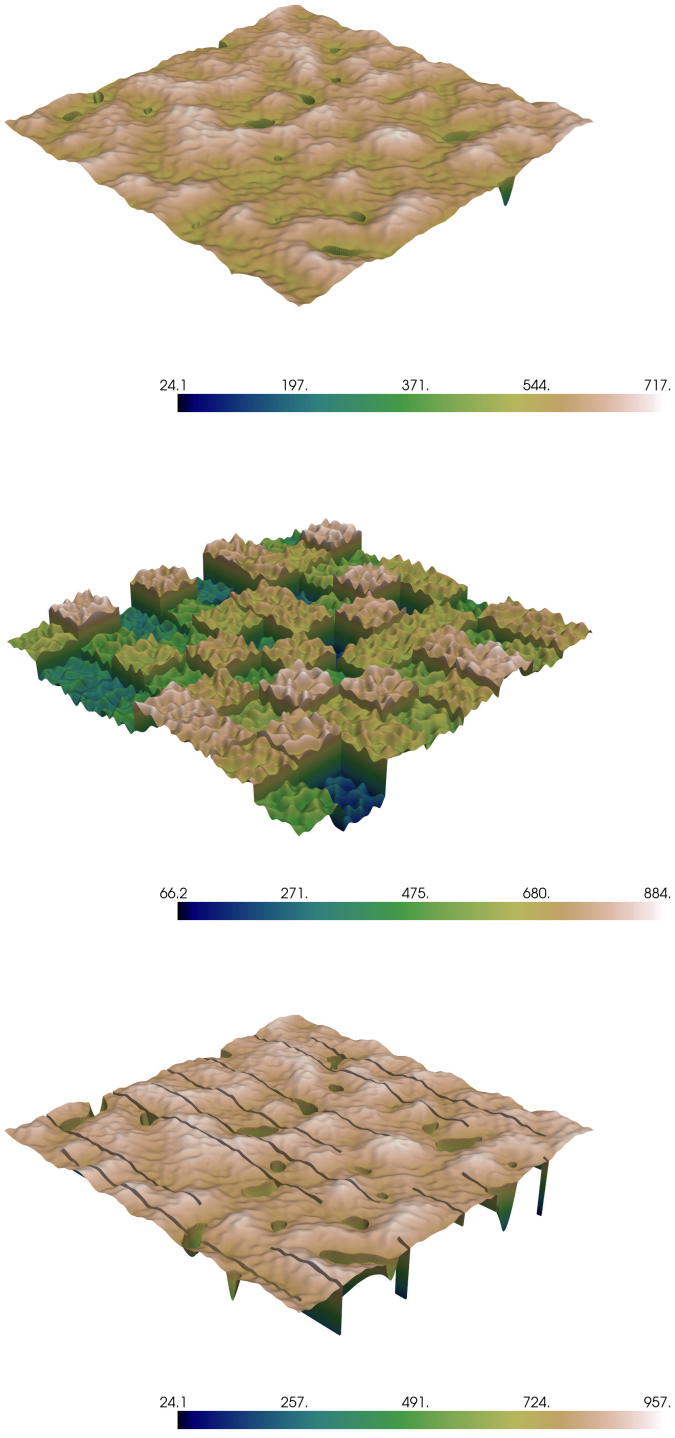

Figure 1: Example of easy, medium, hard terrain