Seminar 6 - Mini (Pseudo)Code Retreat

UCAS Program 2020

Chloe Lau

August 2020

1 What is a Code Retreat?

Code Retreats are free day-long, intensive practice events, focusing on the fundamentals of software development and design. By providing developers the opportunity to take part in focused practice, away from the pressures of "getting things done", the code retreat format has proven itself to be a highly effective means of learning and nurturing software development skills.

2 Introduction

In the following Mini (Pseudo)Code Retreat session, we will be focusing on the task "Conway's Game of Life", which is the most common Code Retreat prompt.

You will form pairs, and based on your ability, to either write pseudo-code for the task, or work on any languages you prefer.

For the first 10 minutes, you will spend the time reading this prompt, and familiarising yourself with the task, subsequently, for 3 sets of 10 minute bursts, (pseudo)code with a different partner each time. The remaining 20 minutes will be for debriefing.

2.1 But wait...

Just one thing that's different from normal rotations, you will have to discard any previous attempts entirely. (It's fine to save the files, but not referring to them during other sessions)

3 Conway's Game of Life

According to the Wikipedia's article: "The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970."

Given a board with m by n cells, each cell has an initial state live (1) or dead (0). Each cell interacts with its eight neighbors (horizontal, vertical, diagonal) using the following four rules (taken from the above Wikipedia article):

- Any live cell with fewer than two live neighbors dies, as if caused by under-population.
- Any live cell with two or three live neighbors lives on to the next generation.
- Any live cell with more than three live neighbors dies, as if by over-population.
- Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

Write a function to compute the next state (after one update) of the board given its current state. The next state is created by applying the above rules simultaneously to every cell in the current state, where births and deaths occur simultaneously.

Example:

Input:

```
[
    [0,1,0],
    [0,0,1],
    [1,1,1],
    [0,0,0]
]
```

Output:

```
[
    [0,0,0],
    [1,0,1],
    [0,1,1],
    [0,1,0]
]
```