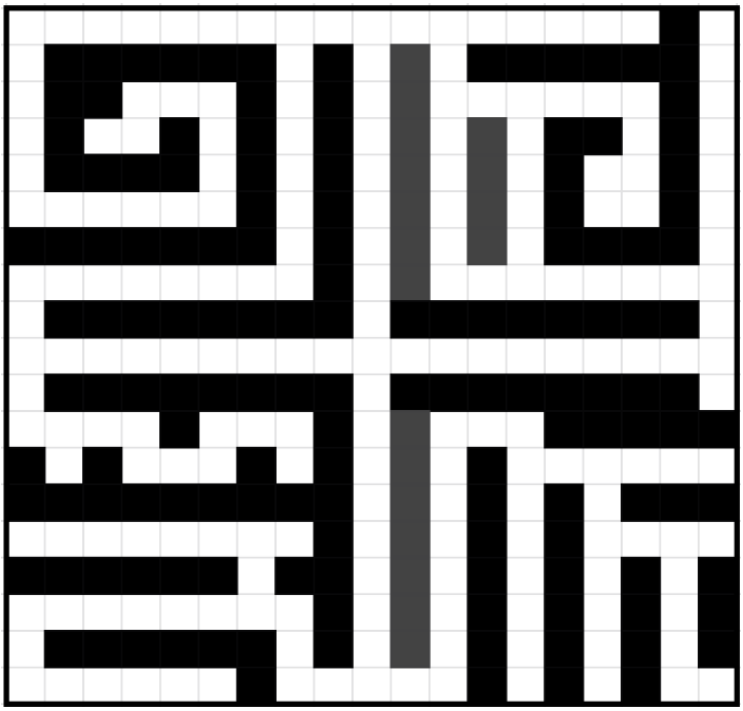
520 : FINAL EXAM – Q.1

A Project Report by:
- Saransh Sharma (ss4368)

Environment

I have used the **NumPy** library to make the *grid(skeleton)* for our 19*19 maze.
I use different numbers as entries in the grid to denote different entities (OPEN=0, WALL=1, DRONE=2) and use mathematical operations to update and manipulate them.



Q.1) Before you do anything, what is the probability that the drone is in the top left corner? Why?

➔ The probability of the drone being in the left corner can be denoted by:

$$P\big(Drone\ in\ (0,0)\big) = \frac{1}{Number\ of\ Unblocked\ Cells}$$

$$= \frac{1}{199}$$

Since, the drone is equally likely to be in any of the unblocked cells.

COMMAND – “DOWN”

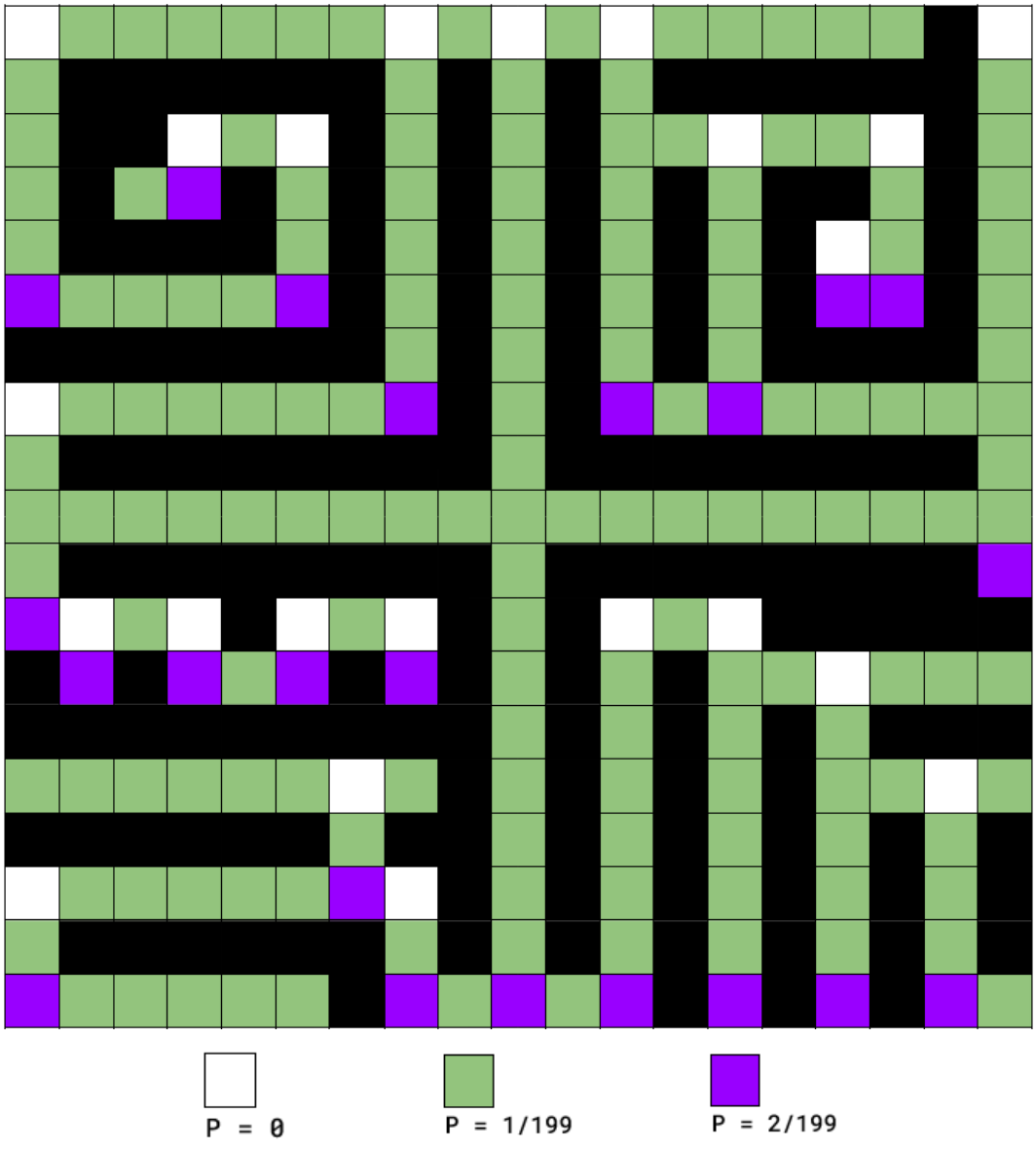
Q.2) What are the locations where the drone is most likely to be? Least likely to be?

How likely is it to be in all the other locations? Indicate your results visually.

➔ The drone is most likely to be in cells that are already close to a wall below them and can’t move down but have room for movement above them. These cells will basically have twice the probability as the others above.

The drone is least likely to be in cells that touch a wall above them and have space to move down.

The drone is equally likely to be in all other cells except the most and least likely ones mentioned above.



Q.3) Write a program that takes a reactor schematic as a text file (see associated file for this reactor) and finds a sequence of commands that, at the end of which, you know exactly what cell the drone is located in. Be clear in your writeup about how you are formulating the problem, and the algorithms you are using to find this sequence.
What is the sequence for this reactor?

➔ I have assumed initially that the drone is present in each open cell of the reactor. I then use below method to perform a sequence of computed steps on all of them till they all merge into 1 drone at a single point.

I have approached this as a value iteration problem, but the state space for merging drones present in this 19*19 grid would be very very large.

To solve the problem of the state space, I tried to split the problem into smaller parts → I perform value iteration on just 2 cells at a time, which essentially reduces the state space to a very manageable size of 19^4. Each state is taken to be a pair of (x, y) co-ordinates.

I used the following equation:

$$U^*_{k+1}(x) = \min_{a \in A(x)} [\, R^a_x \, + \, \beta \sum_{x'} U^*_k(x') \,]$$

As there is just 1 possible neighbouring state for each action – ***UP, DOWN, LEFT, RIGHT*** – and there is no transitional probability between the states either as there is no probabilistic movement. So, the above reduces to……:

$$U^*_{k+1}(x) = \min_{a \in A(x)} [\, R^a_x \, + \, \beta \, U^*_k(x') \,]$$

……in our case.

I have initialised my utilities to be the BFS distance between 2 points and reward is initialised as +1 for each state.

The terminal state is defined as the merging of the 2 locations.

Running the code for above till the error between 2 states converges to a value below 0.0001 results in certain $U^*(s)$ values attached in the file ***"exam_util.txt"*** [This stays the same for any particular reactor schematic]

The code to take a reactor schematic as input has been integrated into the main code file as well.

After we have $U^*(S)$ values for all valid states, we select 2 states with minimum U^* values and move to the neighbouring state with minimum value. I repeat this until all drones have merged into 1 and we get a single tuple in our ***"drone_list"*** which signifies the location of the drone.