Livelock

A livelock is when two or more threads are continuously reacting, each responding to the other's actions. And they can never successfully complete.

You can find different examples of this problem on the internet.

Simple 'Maze'

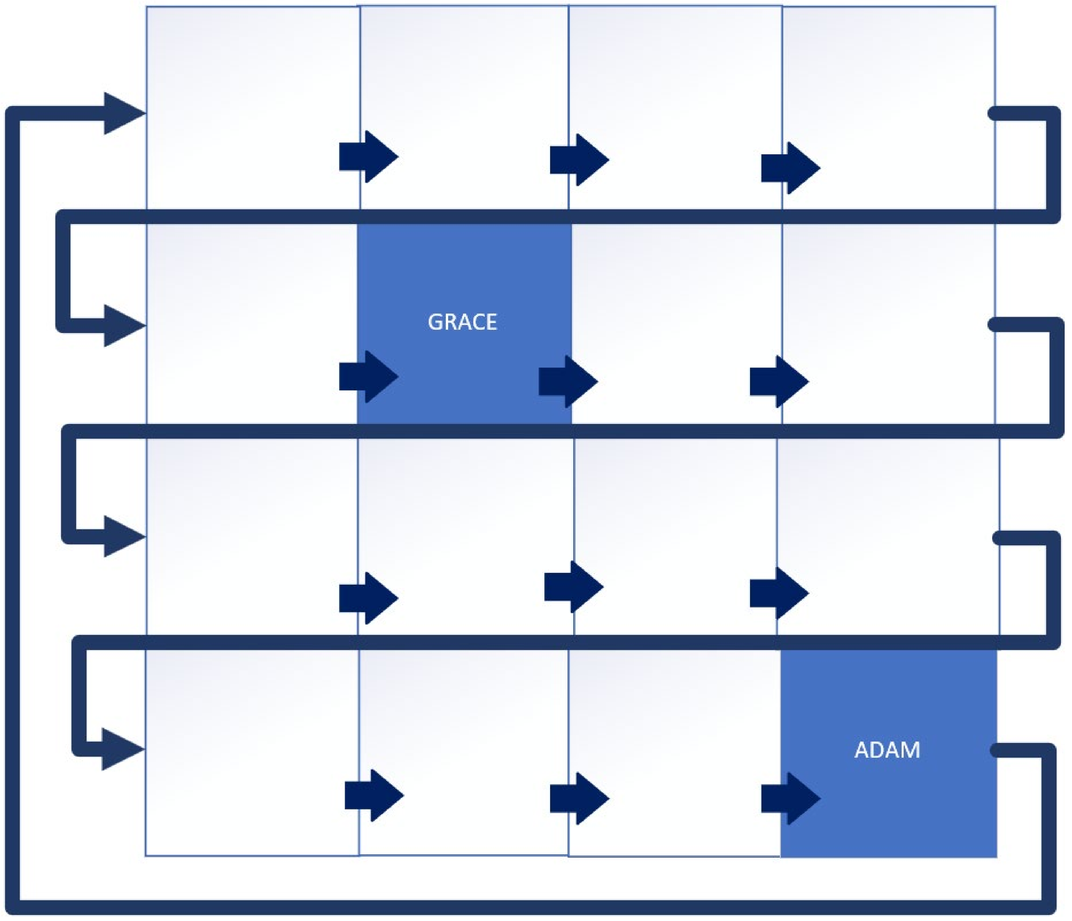
In this video, I'll use a simple maze to demonstrate how you and your friend, in the escape room's maze, might be caught in a live lock scenario.

I'll be creating a 4 by 4 maze, and the paths are shown by the arrows.

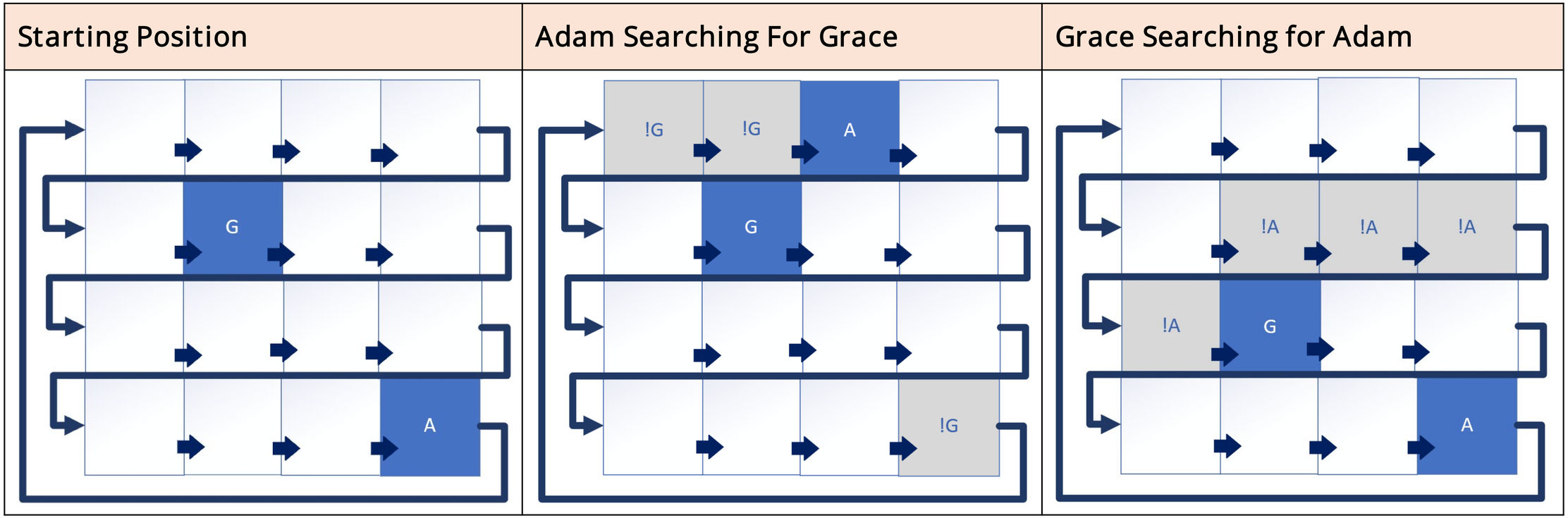
I'll have two people in the maze searching for each other, Grace and Adam.

In this case, Grace is at position (1 1), and Adam is at position (3 3).

Adam, if he moves to the next position, will go to position (0 0), or the top left corner.



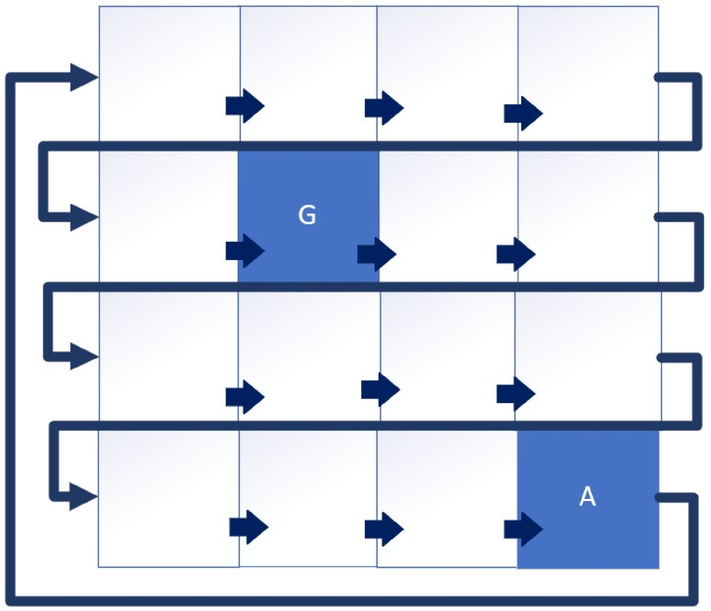
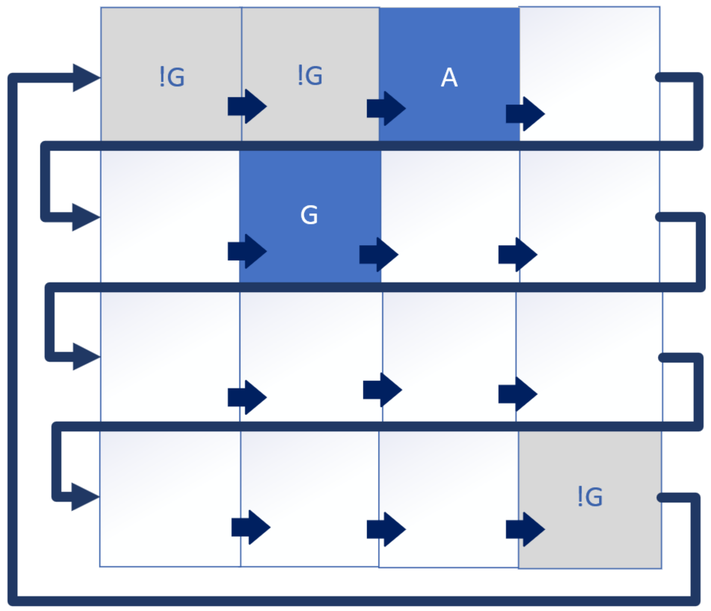
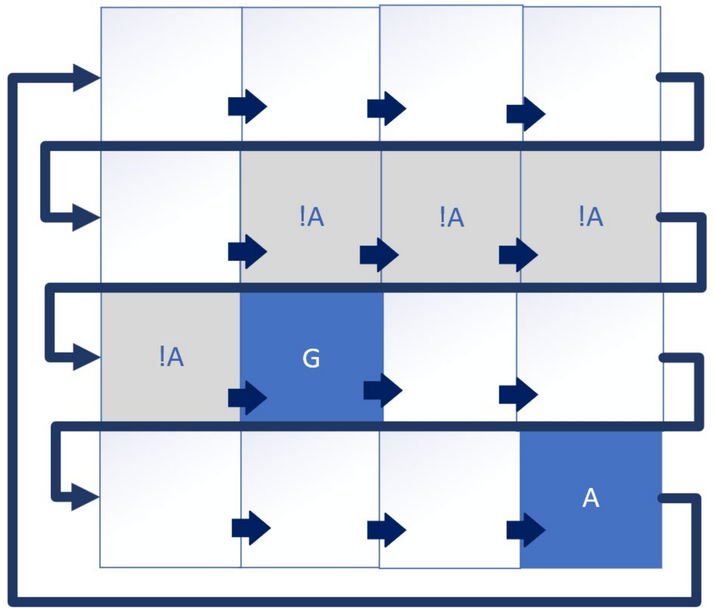
Simple 'Maze'



On this slide, the first grid shows the starting positions of both players.

In the second grid, Adam has moved 4 places from his original position at (3 3), and each cell he searched so far, he hasn't found Grace.

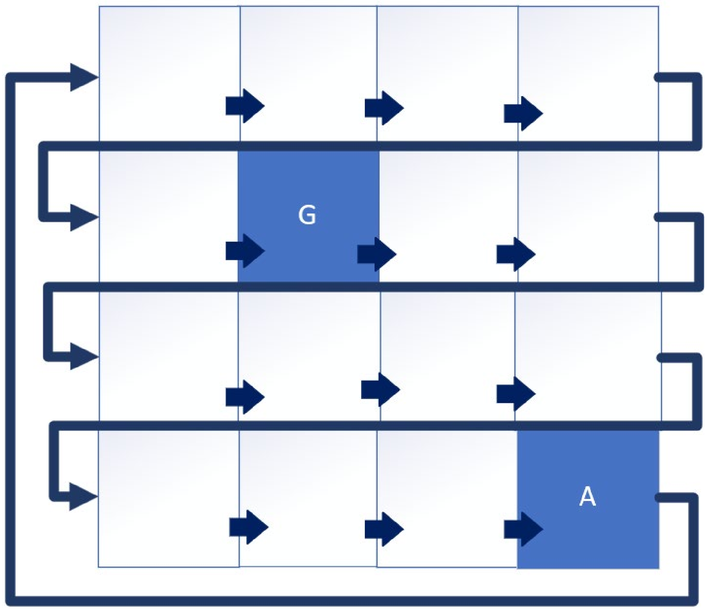
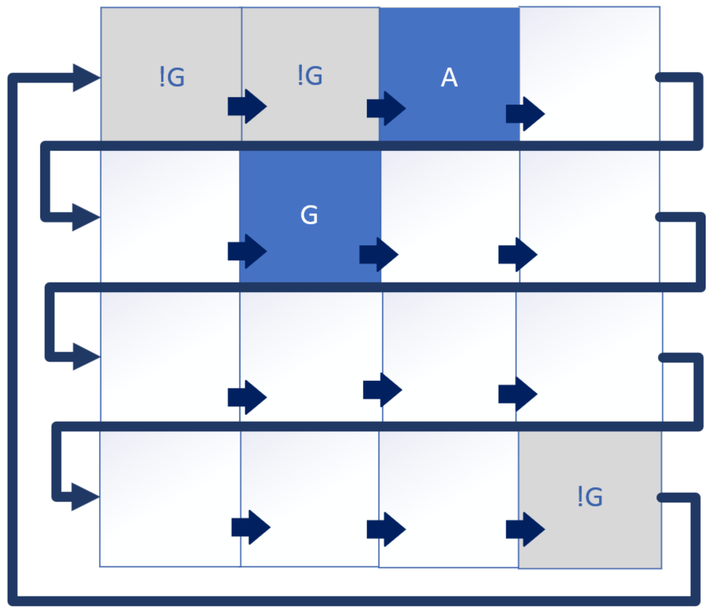
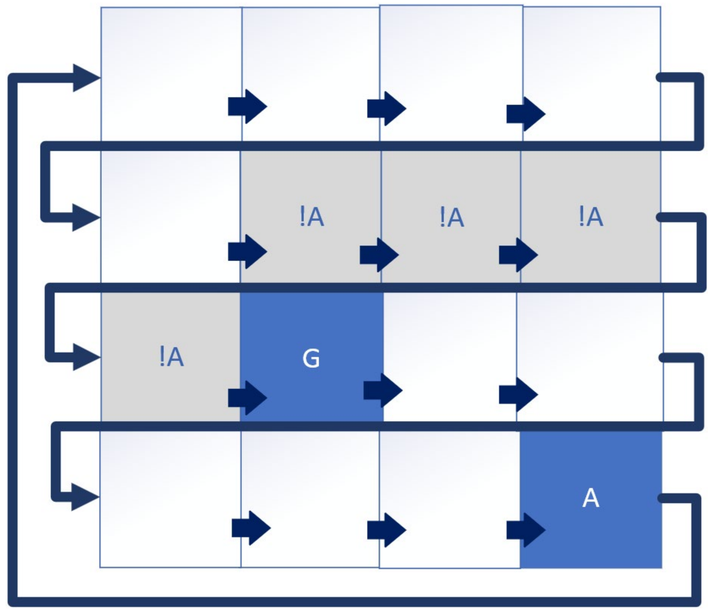
Simple 'Maze'

Starting Position	Adam Searching For Grace	Grace Searching for Adam
		

These are shown as gray cells, and have not G in them, meaning Grace wasn't there.

If Grace is searching, and Adam is standing still, then we have the third grid.

Simple 'Maze'

Starting Position	Adam Searching For Grace	Grace Searching for Adam
		

Grace started in position (1 1), and has moved to (2 1), and all the cells along her path are marked as not A, because Adam wasn't there.

This algorithm is only good, if one player stays put.

Livelocks can be difficult to debug and fix

Livelocks can be difficult to debug and fix.

For this reason, there are a few general things you can do to avoid them:

- Avoid having threads that are constantly checking each other's states.
- Use timeouts to prevent threads from waiting indefinitely for each other.
- Use randomization to break the symmetry between threads.