Project 1: Fire Extinguisher

16:198:520

It is another day on the deep space vessel *Archaeopteryx*, and you are a lonely bot. You're responsible for the safety and security of the ship while the crew is in deep hibernation. As an example, in the case that there is a fire on the ship, you are responsible for pressing the button to trigger the fire suppression system. Of course, you have to get to it first.

1 The Ship

The layout of the ship (walls, hallways, etc) is on a square grid, generated in the following way:

- Start with a square grid, $D \times D$, of 'blocked' cells. Define the neighbors of cell as the adjacent cells in the up/down/left/right direction. Diagonal cells are not considered neighbors.
- Choose a square in the interior to 'open' at random.
- Iteratively do the following:
  - Identify all currently blocked cells that have exactly one open neighbor.
  - Of these currently blocked cells with exactly one open neighbor, pick one at random.
  - Open the selected cell.
  - Repeat until you can no longer do so.
- Identify all cells that are 'dead ends' - open cells with one open neighbor.
- For approximately half these cells, pick one of their closed neighbors at random and open it.

Note: I want you to try to work on a 40 × 40 ship. If this becomes infeasible based on your code and hardware, talk to me.

> Sanity Check: Before you open the dead ends, I estimate that about 60% of the ship should be opened by this process.

2 The Bot

The bot occupies an open cell somewhere in the ship (to be determined shortly). The bot can move to one adjacent cell every time step (up/down/left/right).

3 The Fire

At a random open cell, a fire starts. Every time step, the fire has the ability to spread to adjacent open cells. The fire cannot spread to blocked cells. The fire spreads according to the following rules: At each timestep, a non-burning cell catches on fire with the probability $1 - (1 - q)^K$:

- q is a parameter between 0 and 1, defining the *flammability* of the ship.

- K is the number of *currently burning neighbors* of this cell.

Every timestep, each cell is updated according to the above rules.

Note: If q is very close to 0, $1-(1-q)^K$ will be close to 0, and the fire spreads with very low probability. If q is close to 1, then $1-q$ is close to 0, and $1-(1-q)^K$ is close to 1 - meaning that the fire spreads with very high probability. Additionally, the more neighbors a cell has that are on fire, the higher K is, and the closer $1-(1-q)^K$ will be to 1 - the faster the fire will spread.

Sanity Check: A common mistake here is to spread the fire to one new cell, update neighbor counts, then spread the fire again, until every cell has been processed. This is incorrect. Think about the following:

- Take the current ship
- Generate a copy
- For each open cell in the original ship (not currently on fire):
  - Count the number of currently on fire neighbors K.
  - Compute the probability this cell will catch on fire, based on K.
  - Pick a random number in [0,1] (uniformly, for instance with random.random() in python).
  - If this random number is less than the probability you calculated, set that cell on fire in the ship copy.
- Replace the current ship with the updated copy.

4 The Button

Located somewhere in the ship, in an empty cell (to be determined shortly), is a button that triggers the fire suppression system, instantly choking the fire of oxygen and putting it out. This must be pressed in order to stop the fire.

5 The Task

At time $t = 0$, place the bot, the button, and the initial fire cell at random open cells in the ship (three distinct open cells). At every time step, $t = 1, 2, 3, 4, \ldots$, the following happens in sequence:

- The bot decides which open neighbor to move to.
- The bot moves to that neighbor.
- If the bot enters the button cell, the button is pressed and the fire is put out - the task is completed.
- Otherwise, the fire advances.
- If at any point the bot and the fire occupy the same cell, the task is failed.

Ideally, the bot navigates to the button, avoids the fire, and puts the fire out. The goal of this assignment is to build and evaluate different strategies for governing how the bot decides where to go.

6 The Strategies

At any time, the bot can choose from the following actions: move to a neighboring open cell, or stay in place. The thing that differentiates the strategies is how the bot decides what to do.

For this assignment, you will implement and compare the performance of the following strategies.

- **Bot 1** - This bot plans the shortest path to the button, avoiding the initial fire cell, and then executes that plan. The spread of the fire is ignored by the bot.
- **Bot 2** - At every time step, the bot re-plans the shortest path to the button, avoiding the current fire cells, and then executes the next step in that plan.
- **Bot 3** - At every time step, the bot re-plans the shortest path to the button, avoiding the current fire cells *and any cells adjacent to current fire cells, if possible*, then executes the next step in that plan. If there is no such path, it plans the shortest path based only on current fire cells, then executes the next step in that plan.
- **Bot 4** - A bot of your own design.

Note: The only restriction I am putting on Bot 4 is that it cannot be a version of Bot 3 that just puts a wider buffer on the fire cells. Do something more interesting. You are welcome and encouraged to talk to the TAs or myself to discuss your ideas or get inspiration.

Each bot will be evaluated on a number of ships, at various values of q, to establish its effectiveness.

7 Data and Analysis

In your writeup, consider and address the following:

1) Explain the design and algorithm for your Bot 4, being as specific as possible as to what your bot is actually doing. How does your bot factor in the available information to make more informed decisions about what to do next?
2) For each bot, repeatedly generate test environments and evaluate the performance of the bot, for many values of q between 0 and 1. Graph the probability (or average frequency, rather) of the bot successfully putting out the fire. Be sure to repeat the experiment enough times to get accurate results for each bot, and each tested value of q.

   > Note: If a bot fails, it doesn't mean a better bot would've succeeded - there may be times when /no/ bot could possibly have succeeded. Additionally plot *the success rate for each bot* ***out of the simulations where success was possible***. This will give you a more realistic comparison between the performances of the bot. But it requires a solution to the following problem: how can you tell if a success was possible for *some* bot in a simulation? If one of the bots succeeds, success was possible. If none of the bots succeed, how could you test whether there was a route the bot could have taken to succeed? Be clear and explicit about your algorithm, and your results.

3) When bots fail or get trapped by the fire, *why* do they fail? Was there a better decision they could have made that would have saved them? Why or why not? Support your conclusions.

4) Speculate on how you might construct the ideal bot. What information would it use, what information would it compute, and how?

> Sanity Check: If you are tempted to answer this question with, 'I would use advanced algorithms,' think twice. What algorithms? How would you process the information you have available? Be specific.

Bonus: *Write an algorithm to find a ship layout that maximizes (as best you are able) the probability that some bot will be able to put out a random fire. Be clear as to your algorithm and results. Show me the actual ship layout you achieve, and data to support your conclusions.*