Project 1: Where are you?

16:198:520

You are a bot and you've recently been hired as a space roomba on the deep space salvage vessel *Nebulist*. Unfortunately, while you were given a map of the ship, a blast of cosmic radiation accidentally fried your internal sensors, and you have no idea where you are, and cannot even see your surroundings.

The goal of this project is to figure out where you are.

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. *This introduces the possibility of loops and multiple ways of traveling between two points in the ship.*

The remaining blocked squares cannot be entered - you as the bot can move freely between open neighboring cells (in the cardinal directions).

However: because you don't know where you are and can't sense your environment - you can't tell if an attempt to move to a neighboring cell was successful.

Note: In what follows, I am going to ask you to experiment on ships of various sizes.

2 The Task

At the start, you might be anywhere in the ship. Given the map of the ship, your task is to generate a sequence of moves to attempt, after which you will be left in a single, well defined location *regardless of where you started.*

> To see that this is doable (as we saw in that example in class, note): if you consider the set of possible locations the bot is in as a set of cells L, and attempt to move in a given direction, the set of locations L' you might be in after the attempted move cannot be large and is likely to be smaller than the original set L. Your goal is to shrink the set of possible locations down to a single point.

3 Baseline Strategy

I will offer you the following strategy. Your job is to come up with a) a strategy for identifying the most efficient solution to the problem, and b) a strategy for more efficiently identifying a solution to the problem.

Dr. Cowan's Strategy:

- To begin, pick dead end at random as a target cell.
- Iteratively do the following:
  - Pick a location the bot might currently be in.
  - Plan a shortest path from that location to the target cell. (*For instance, using A^* with a Manhattan distance heuristic.*)
  - Execute that sequence of actions, updating the set of possible bot locations as you do.
- This process continues until the only possible bot location is the target cell.
- Return the total number of moves taken.

4 Your Strategies

For this project, you need to develop two strategies for solving the problem.

Optimality: Code a strategy for finding the optimal sequence of actions to take to localize the bot.

Note, one thing you may want to consider here: what kind of heuristics might be useful to apply here, and how can you tell how good those heuristics are (relative to the problem you are trying to solve).

Last thing to consider for this strategy: optimality is going to be harder to achieve the larger the spaceship is. Your goal should be to achieve optimality on as large a spaceship as you can manage.

Efficiency: Note that as the ship gets larger, computing the optimal solution will almost certainly get harder. Code a strategy for finding *a* sequence of actions to localize the bot as fast as possible, even if it is not necessarily the optimal sequence of actions.

What changes from considering the optimality strategy to the efficiency strategy?

Note: the entire point of considering this strategy is to be able to apply it to even larger spaceships than what you can apply your optimality strategy to.

5 Data, Analysis, and Writeup

In addition to coding up your solutions for the two strategies described above, you will also need to submit a writeup analyzing their performance.

1) Explain the design choices for your Optimality Strategy. In particular, be clear and explicit about your reasoning for any heuristics or informed search you perform. Justify why you know this strategy will return an optimal solution.

2) Explain the design choices for your Efficiency Strategy. In particular, be clear and explicit about any sacrifices you make to achieve speed over performance of the final solution.

3) Compare the performance of my strategy to your Optimality Strategy. Generate a graph of the performance, plotting size of ship on the x axis, and 'average number of moves to localize' on the y axis. Your graph should have two curves, one for the Optimality Strategy, one for my strategy. You should aim to have a large a range of ship sizes as you can. You do not need to have every ship size over that range, but you need to have enough data that any trends are clearly presented.

   Note: You will need to generate multiple ships at each size and solve them, averaging the results together to get the data for your graph. There will be some variance in the values you generate, which means that you will need to generate as much data as you can to get good values for your averages.

   Additionally, note that my strategy has no guarantee of optimality, and yours *should*. This serves as a sanity check for you - if you see my strategy outperforming yours (ever), then there is an issue you need to address.

4) Compare the performance of my strategy to your Efficiency Strategy. This will require two graphs:

   – In the first, graph 'average moves to localize' as a function of ship size. Compare the performance of my strategy to your Efficiency Strategy here. Note, you should be able to collect data over a wider range of ship sizes than in the previous graph. How does your strategy compare to mine in performance?

   – In the second, graph 'time to compute solution' as a function of ship size. Compare the performance of my strategy, the Optimality Strategy, and the Effiency Strategy. This graph should be over a larger range of ship sizes than you did with the Optimality Strategy originally - but it is all right with me if you do not collect data for the Optimality Strategy for all ship sizes. You should see a very clear trend even with truncated data, that your Efficiency Strategy is more efficient.

     Note: If your Effiency Strategy is not more efficient - this suggests you should revisit your design of the Effiency Strategy.

5) Last thing I am interested in is this: in all previous analysis, I wanted you to start from assuming the bot could be anywhere in the ship. We could also start from a smaller, restricted set of locations. For this final problem, I would like you to develop algorithms to find:

   – The smallest set of locations you can (for a given ship) that forces the Optimality Strategy to take as many moves as possible to localize the bot.

   – The smallest set of locations you can (for a given ship) that forces the Efficiency Strategy to work as long as possible to find localization strategy.

   Be clear about your algorithmic approach here, and display the solutions you find for a number of ships of various sizes. Are there any patterns or trends that emerge? What is it that makes this localization problem 'hard' (both in the optimality sense, and in the effiency sense)?