CS 440: Assignment 1 - Search and Shortest Path Planning

16:198:440

Due 7/13/2017 by Midnight

The purpose of this assignment is to implement and compare a number of search / shortest path planning algorithms (BFS, DFS, A^*), and compare their various merits.

Generating Environments: In order to properly compare these algorithms, they need to be run multiple times over a variety of environments. A **map** will be a square grid of cells / locations, where each cell is either empty or occupied. An agent wishes to travel from the upper left corner to the lower right corner, along the shortest path possible. The agent can only move from empty cells to neighboring empty cells in the up/down direction, or left/right - each cell has potentially four neighbors.

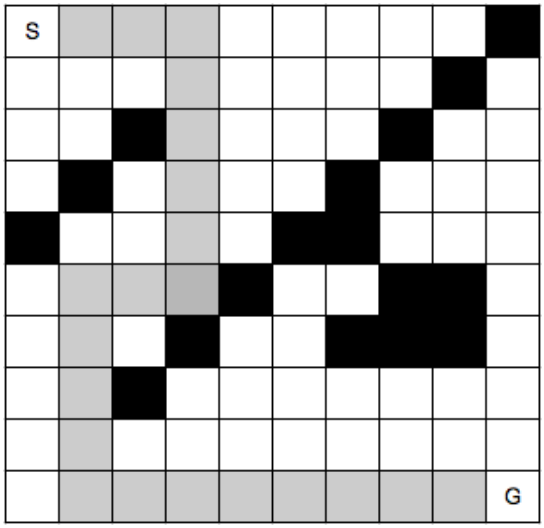

Figure 1: Successful - A path exists from start to finish.

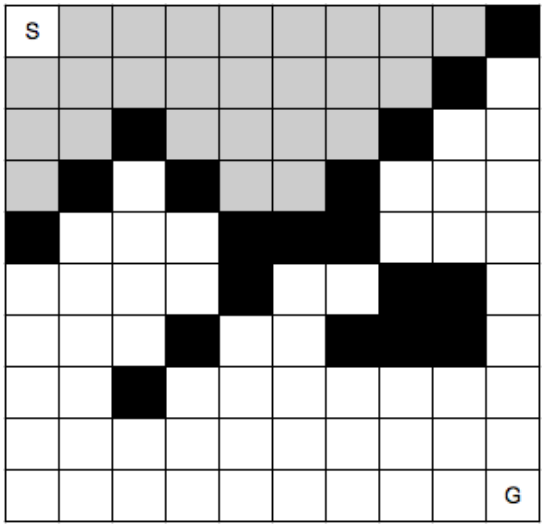

Figure 2: Unsuccessful - No path exists from start to finish.

Maps may be generated in the following way: for a given dimension **dim** construct a **dim x dim** array; given a probability p of a cell being occupied ($0 < p < 1$), read through each cell in the array and determine at random if it should be filled or empty. When filling cells, exclude the upper left and lower right corners (the start and goal, respectively). It is convenient to define a function to generate these maps for a given **dim** and p.

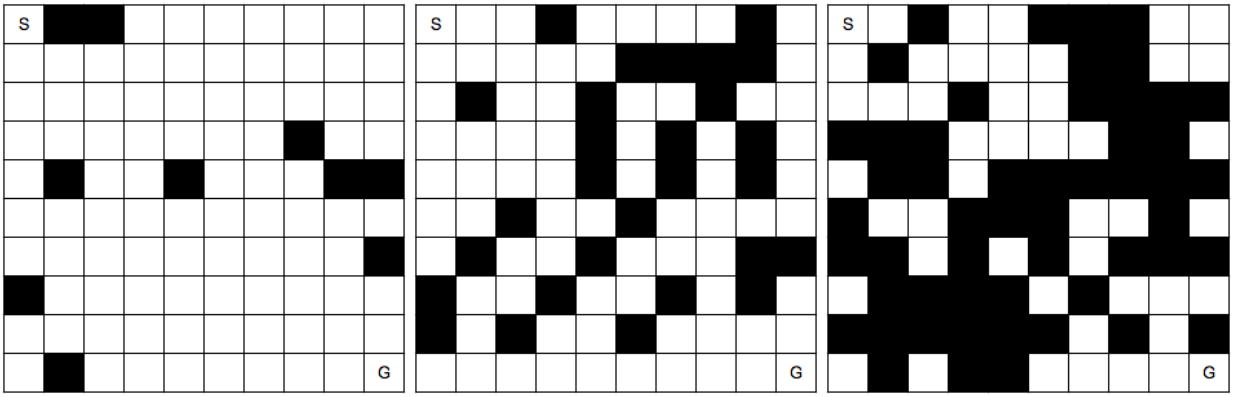

Figure 3: Maps generated with $p = 0.1, 0.3, 0.5$ respectively.

Path Planning: Once you have the ability to generate maps with specified parameters, implement the ability to search for a path from corner to corner, using each of the following algorithms:

- Depth-First (Graph) Search
- Breadth-First (Graph) Search
- A^*: where the heuristic is to estimate the distance remaining via the **Euclidean Distance**

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{1}$$

- A^*: where the heuristic is to estimate the distance remaining via the **Manhattan Distance**

$$d((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|. \tag{2}$$

For any specified map, applying one of these search algorithms should either return failure, or a path from start to goal in terms of a list of cells taken. *(It may be beneficial for some of these questions to return additional information about how the algorithm ran as well.)*

Questions:

1) For each of the implemented algorithms, how large the maps can be (in terms of **dim**) for the algorithm to return an answer in a reasonable amount of time (less than a minute) for a range of possible p values? Select a size so that running the algorithms multiple times will not be a huge time commitment, but that the maps are large enough to be interesting.

2) Find a random map with $p \approx 0.2$ that has a path from corner to corner. Show the paths returned for each algorithm. (Showing maps as ASCII printouts with paths indicated is sufficient.)

3) For a fixed value of **dim** as determined in Question (1), for each $p = 0.1, 0.2, 0.3, \ldots, 0.9$, generate a number of maps and try to find a path from start to goal - estimate the probability that a random map has a complete path from start to goal, for each value of p. Plot your data. Note that for p close to 0, the map is nearly

empty and the path is clear; for p close to 1, the map is mostly filled and there is no clear path. There is some threshhold value p_0 so that for $p < p_0$, there is usually a clear path, and $p > p_0$ there is no path. Estimate p_0. Which path finding algorithm is most useful here, and why?

4) For a range of p values (up to p_0), generate a number of maps and estimate the average or expected length of the shortest path from start to goal. You may discard all maps where no path exists. Plot your data. What path finding algorithm is most useful here?

5) For a range of p values (up to p_0), estimate the average length of the path generated by A^* from start to goal (for either heuristic). Similarly, for the same p values, estimate the average length of the path generated by DFS from start to goal. How do they compare? Plot your data.

6) For a range of p values (up to p_0), estimate the average number of nodes expanded in total for a random map, for A^* using the Euclidean Distance as the heuristic, and using the Manhattan Distance as the heuristic. Plot your data. Which heuristic typically expands fewer nodes? Why? What about for p values above p_0?

7) For a range of p values (up to p_0), estimate the average number of nodes expanded in total for a random map by DFS and by BFS. Plot your data. Which algorithm typically expands fewer nodes? Why? How does either algorithm compare with A^* in Question (6)?

8) Consider the following three heuristics. For a range of p values (up to p_0), estimate the number of nodes expanded by A^* using each of these heuristics.

   - Max Distance:

$$d((x_1, y_1), (x_2, y_2)) = \max(\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}, |x_1 - x_2| + |y_1 - y_2|). \tag{3}$$

   - Min Distance:

$$d((x_1, y_1), (x_2, y_2)) = \min(\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}, |x_1 - x_2| + |y_1 - y_2|). \tag{4}$$

   - α-Weighted Combination: For some constant α in $0 < \alpha < 1$,

$$d((x_1, y_1), (x_2, y_2)) = \alpha\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} + (1 - \alpha)\left(|x_1 - x_2| + |y_1 - y_2|\right) \tag{5}$$

   - β-Norm Distance: For some constant β in $1 < \beta < 2$,

$$d((x_1, y_1), (x_2, y_2)) = \left(|x_1 - x_2|^\beta + |y_1 - y_2|^\beta\right)^{1/\beta}. \tag{6}$$

   For α-weighted combination and β-Norm, look at at least two distinct values of α and β over the indicated ranges. Are there α and β values that seem to work best, over all possible p values?

Bonus 1) Why were you not asked to implement UFCS?

Bonus 2) Argue that the heuristics given in this assignment are admissible and consistent.