Programming Homework 9

CS540

July 26, 2019

1 Instruction

Please submit your output files and code on Canvas → Assignments → P9. Please do not put code into zip files and do not submit data files. The homework can be submitted within 2 weeks after the due date on Canvas without penalty (50 percent penalty after that).
Please add a file named "comments.txt", and in the file, you must include the instructions on how to generate the output, for example:

- Data files required: train.csv, test.csv. Run: main.jar.
- Data folder required: data/train1.png ... data/train100.png . Compile and Run: main.java.

2 Details

All the requirements are listed on the course website. The following is only an example workflow to solve the problem.

1. Download the PNG file of a maze. You should decide how to store the information. One possible way is to store a matrix (or vector) of successors for each cell. Suppose the maze is 50 by 50, then create a 2500 by 4 matrix. Let each cell be x pixels by x pixels (white), and each wall be y pixels (black) thick. Please check to make sure $x = 14$ and $y = 2$? To get the successors, start from the center pixel of a cell, check if there is a black pixel $\frac{x}{2}$ or $\frac{x}{2} + 1$ above, below, to the left and to the right pixels away from the center. The center pixel of each cell can be found using x and y too (formula depends on how you index the cells), something like $\left((x+y)\cdot i + \frac{x+y}{2}, (x+y)\cdot j + \frac{x+y}{2}\right)$.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

   Using the small empty matrix above as an example. The successor matrix should be the follow (flattened).

$2, 5$	$1, 3, 6$	$2, 4, 7$	$3, 8$
$1, 6, 9$	$2, 5, 7, 10$	$3, 6, 8, 11$	$4, 7, 12$
$5, 10, 13$	$6, 9, 11, 14$	$7, 10, 12, 15$	$8, 11, 16$
$9, 14$	$10, 13, 15$	$11, 14, 16$	$12, 15$

2. Create a Cell class (object, type, etc.) to store the following information:

   - Index: i (an integer or a pair of integers depending on how you index the cells)
   - Backtracker: i' (an integer or a pair)
   - Cost so far: g
   - Heuristic: h

   For example, suppose you pop or deQueue a cell indexed 123 with $g = 24$, and one of its sucessors according to the sucessor matrix is 124. Then the new Cell object you push or enQueue should be $i = 124, i' = 123, g = 24 + 1 = 25, h =$ distance from 124 to goal.

3. For IDS, you can start with depth 50 (or slightly larger depending on your maze). Create a Stack of Cells. Push the initial Cell. Pop and push its successors (excluding the backtracker) based on the successor matrix. Keep track of how many cells are popped. Repeat until the goal cell is popped. You can set the initial and goal cell indices manually or you can find the cell in the first row that can go up and the cell in the last row that can go down.

4. For $A^\star$, you can use various distance measures as the heuristic function. One possible (and good) heuristic is the Manhattan distance from the cell to the goal. The Manhattan distance from cell $s = (x_i, y_i)$ to $g = (x_{i'}, y_{i'})$ is $h(s) = |y_i - y_{i'}| + |x_i - x_{i'}|$. Create a Priority Queue object of Cells. Define a Comparator of Cell objects based on $g + h$ (or equivalent in non-Java languages). EnQueue the intial cell. DeQueue and enQueue its successors (excluding the backtracker) based on the successor matrix. Repeat until the goal cell is deQueued.

5. The maze should not have the problem with two paths leading to the same cell, so it should not be necessary to deal with repeated elements in the Stack or Queue. Start from the goal state, use the backtrackers to generate the path. You can keep track of the complete backtracking path in a single Cell object or create a separate array to store the backtrackers of all cells.