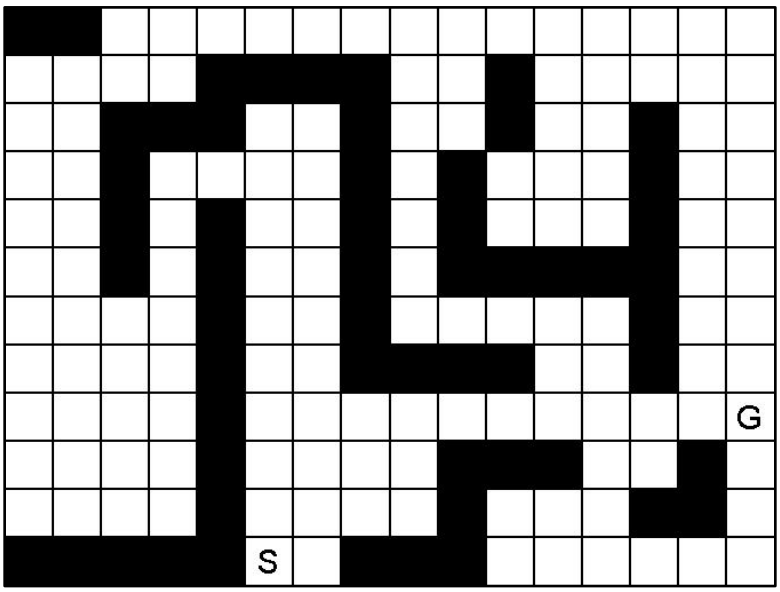
24-787 Artificial Intelligence and Machine Learning for Engineering Design
Homework 7
Search

Maze Search

1. *(30 points)*



Consider the maze shown above. The goal is to travel from the start cell (S) to the goal cell (G). For each of the following cases, number the cells in the order they are visited using the specified search algorithm. Stop numbering after you reach the goal cell (i.e. G should contain a number). You may assume the start cell has a label of zero and that loops are prohibited.

****For your convenience, the maze is included as a jpeg with this assignment****

(a) Depth-first search using North-East-South-West order.

(b) Depth-first search using West-East-South-North order.

(c) Breadth-first search using North-East-South-West order.

DFS v. BFS

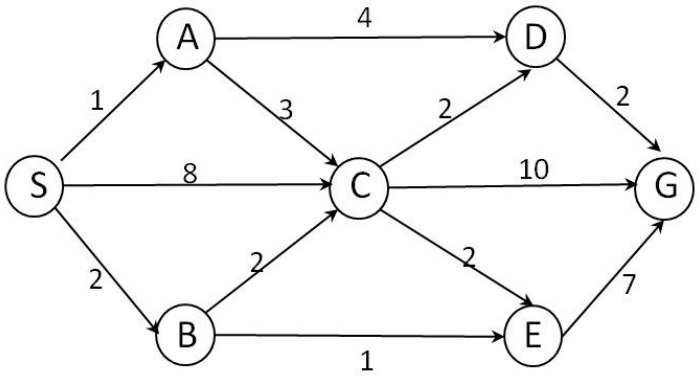
2. *(10 points)*

Explain the advantages and disadvantages of depth-first search compared to breadth-first search (3-5 sentences should suffice).

Uniform-Cost Search

3. *(30 points)*

   Perform uniform-cost search (UCS) on the graph below. The start and goal nodes are labeled S and G, respectively. You must include the priority queue (showing when nodes are inserted/expanded), a cost-to-node table, and a list of backpointers. Consult the lecture slides for an example solution to a similar problem. Clearly specify the final path.



Solving the 8-Puzzle

4. *(30 points)*

   In this programming exercise, you will implement A* search to solve the 8-puzzle. The 8-puzzle is a classic problem consisting of numbered tiles 1-8 arranged in random order on a 3x3 grid; one cell in the grid is open. The objective is to slide tiles into the open space until the numbers are arranged in the configuration shown below.

1	2	3
4	5	6
7	8	

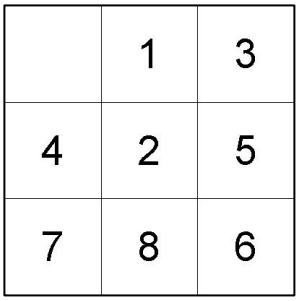
   Sample code has been provided to get you started. Your task is to complete the function called `Astar`, which implements the A* search algorithm. For the heuristic, use the number of misplaced tiles. To test your function, use the following line of code:

```
>> P = Astar(S);
```

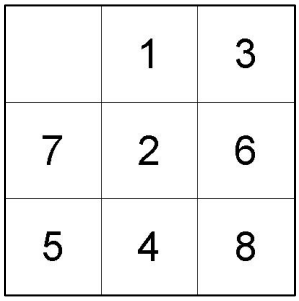
   where `S` is a 3x3 array containing the initial puzzle state, and `P` is the sequence of moves to get from `S` to the goal state. Each move represents the direction that a tile moves to fill the empty cell and is encoded as an integer according to the following scheme: LEFT = 1, RIGHT = 2, UP = 3, DOWN = 4. For instance, a move of 4 on the board above indicates the 6 should slide down into the bottom corner. After completing your code, you may visualize the solution path as follows:

```
>> show8puzzle(S,P);
```

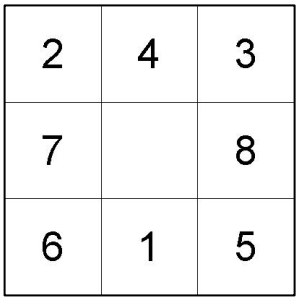
Use your code to solve the 8-puzzle for each of the starting configurations shown below. In your report, clearly state the resulting path (a vector of integers) and give the total number of moves required (the length of the path).



(a)



(b)



(c)

Extra credit: *(10 points max)* Repeat the problem using the sum of Manhattan distances for each number from their goal position as the heuristic. For each puzzle, include the resulting paths and total number of moves in your report.