ITCS 3153: Introduction to Artificial Intelligence
In-class assignment
Informed Search

Edit your previous program to solve text-based grids (using DFS and BFS) to solve the grids using A* and greedy search algorithms.

Write the final path to a file (as in the previous assignment), and print the following data to console:

1) If a solution could be found or local minima
2) Number of states expanded
3) Final path (list the locations), if one can be found
4) What heuristic function you used - e.g., “Manhattan distance”

Edit your program to toggle between using A* and greedy-search without separate functions.

A big difference in this problem is that the values on the grids will have any value from 0-9. **Any nonzero value represents the step cost to move to that location from any of its neighbors.** A value of 0 indicates that the location cannot be traversed. An example grid can be seen below.

1 5 2 4 1 5 1 0 4 0
1 5 0 2 4 1 0 2 5 0
1 0 5 5 1 2 1 4 4 3
2 1 2 2 0 0 2 4 4 0
2 0 4 2 2 1 1 5 1 2
0 3 4 1 3 3 4 2 1 3
1 0 3 5 2 5 1 0 4 3
2 4 0 1 0 2 4 2 4 0
0 4 5 5 2 5 3 1 4 3
3 0 3 3 5 5 3 4 1 1

Things you may need or want to change

1. Use a normal Python list for the open list, and use the heapq Python module to add things to it. https://docs.python.org/3/library/heapq.html

   ```
   openList = []
   heapq.heappush(openList, current) # where current is a Node object
   ```

   a. Using heapq may require you to overload a comparison operator for your Node class. Here are some resources on overloading operators in Python.
      i. https://www.programiz.com/python-programming/operator-overloading
      ii. https://www.tutorialspoint.com/How-to-overload-Python-comparison-operators
   b. You can use other priority queue implementations if you want.
2. You may want separate functions for checking if something is in the open list or in the closed list so that you can return different information based on which one you're checking.
   a. For the closed list, we may only be concerned with if a Node is in there. For the open list, we may want more information if a Node is in there.
3. The expandNode function will need to be more involved to handle the different behaviors of the informed searches.

*** This is not an exhaustive list. You may need to change other parts of your code, and you are welcome to change any other aspects of your code that you see fit. ***

Bonus:
Create a new program to run A* on the 8-puzzle problem. Refer to the textbook for specific details on the problem. Try testing your algorithm with the following start and goal state pairs. The ‘0’ character represents the blank tile.

Initial:
2 8 3
1 6 4
7 0 5

Goal:
1 2 3
8 6 4
7 5 0

Initial:
7 2 4
5 0 6
8 3 1

Goal:
0 1 2
3 4 5
6 7 8

Report on the same data as above. Display the final path as a series of puzzle states (it will be long).