Planning in a Graph

MEE 4411/5411 Project 1, Phase 3

"Due:" Tuesday, October 17th

For this assignment, you will write a planner which can generate paths from a start position to an end position in a 2D environment. You will use the occupancy grid that you developed in Phase 2 to create the graph which you will use for planning.

1 That Dastardly Dijkstra

Implement Dijkstra's algorithm as described in class. Things to note:

- In addition to returning the path, `dijkstra()` should return the total number of nodes that were visited.
- The algorithm described in class output the costs and the parent nodes. You must use the parent nodes to recreate the path from the start to the goal. To do this, you begin with the goal node. Then find the parent of the goal node, then find the parent of that node, and continue this process until you have reached the start node.
- To find the neighbors of a node, you need to move one grid cell over and then check to make sure that the candidate neighbor is inside of the environment *and* that the cell is empty.
- One of the main considerations of any planner is how fast it is. The Matlab profiler will help you considerably. Also see this Mathworks page. For those of you know how to use C and MEX, please do **not** use those for this assignment. You are also **not** allowed to use any external libraries or implementations on this assignment.

2 A* is Born

Modify `dijkstra()` so that when the `astar` parameter is `true`, it uses distance to the goal as a heuristic to guide the search. Remember that any heuristic must be admissible and valid; if not, the algorithm is no longer guaranteed to return the shortest path.

3 Submission

When you are finished you may submit your code by putting it all into a single `.zip` file and emailing it to pdames@temple.edu with subject line "MEE4411 Project 1.3" (clicking the previous link will automatically start a correctly formatted email for you). Prof. Dames will test your submission and send you a list of items to fix. You are encouraged to update your code to fix any bugs, however each student is limited to 3 submissions per phase of the project.