Caelan Garrett
6.S078 Assignment 1
09/11/13

Description

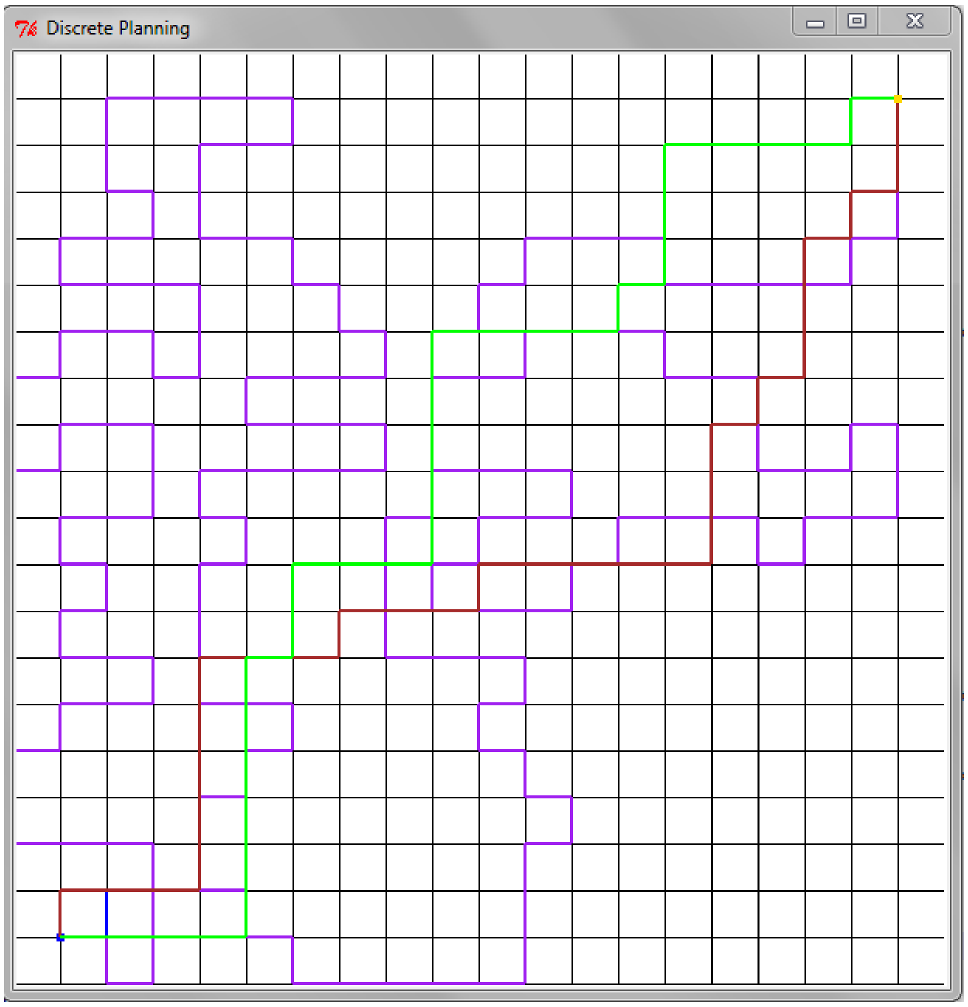
I modeled the obstacles and robot as Objects, a list of convex Polygons that have positions relative to a supplied reference point for the Object. Objects support tests of point containment and collisions with other Objects. Collision tests are performed by testing for line intersections and testing that one point of each Object is outside the other. The lower level geometry classes, Point and Polygon, have several methods to help with these computations. When testing if the robot collides with an object while moving, I first extend the robot between the start and end points to ensure. I use this extended Object in collision tests to ensure that it does not collide between points. This behavior is shown in "Hard world" by the robot moving one unit to the right to avoid a triangle that has a tip between two points.

I chose to consider two touching Objects as being non-colliding. Therefore, optimal paths usually have the robot moving on the wall of an obstacle. The displayed region bounds the robot's movements similarly. The robot is allowed to move along the wall.

I implemented DFS (recursively), BFS, UCS, and A*. DFS and BFS have a worst case complexity of O(m + n). UCS and A* use python's heappush and heappop methods to maintain a priority queue. These use a binary heap which give UCS and A* a worst case complexity of O(m log n). In practice, heuristics and implementation differences caused the differences in runtime. When ordered by increasing average speed, the ranking is DFS, A*, UCS, BFS. I used two different heuristics for A*, both based off the Manhattan distance. The admissible heuristic I used is the Manhattan distance, and the not admissible heuristic is 5 times the Manhattan distance. The not admissible heuristic was faster than the normal heuristic and luckily found an optimal path in each world. Because each edge had equal weight, BFS, UCS, and A* with an admissible heuristic are guaranteed to find an optimal path if it exists.

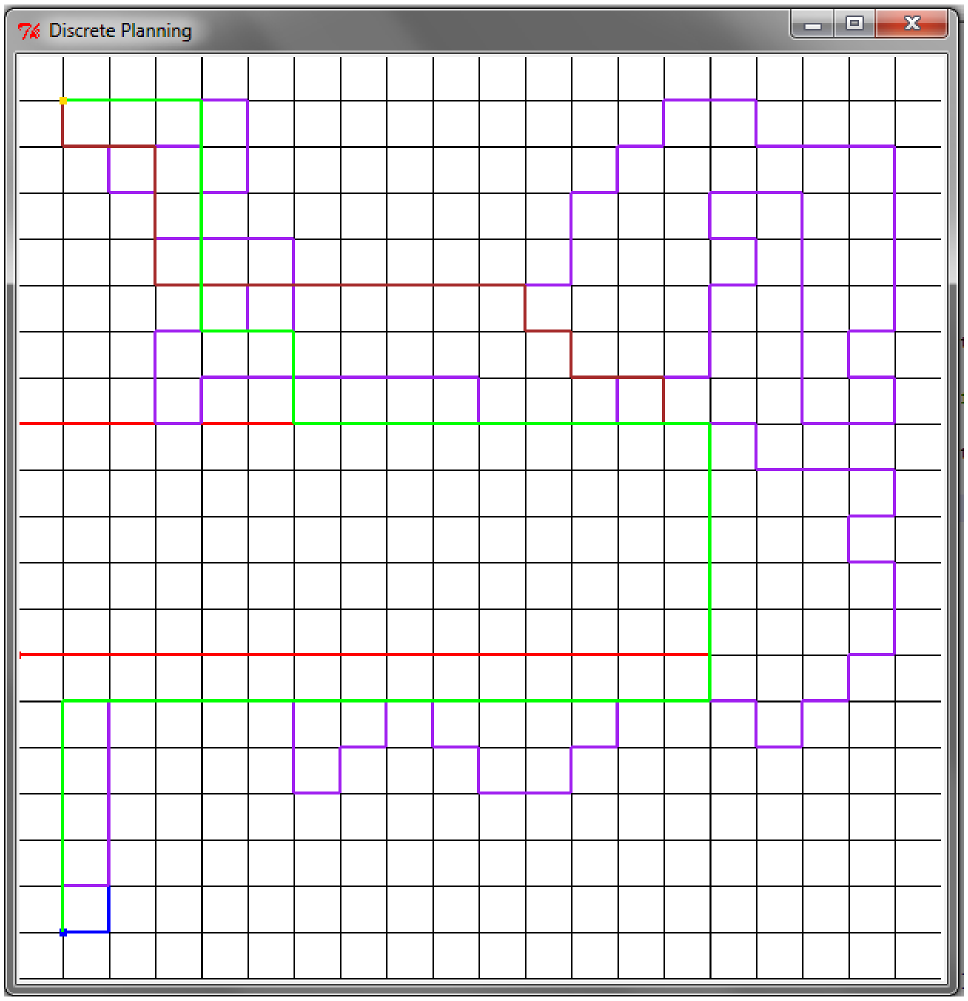
The grid is drawn in black on the canvas. The robot is blue, and obstacles are red. Reference points are drawn on the Objects. The goal is gold. A path determined by DFS is shown in purple. An optimal path generated by UCS is shown in green. Finally, a path generated by A* with a not admissible heuristic is drawn in brown.

Easy World - A 20 x 20 grid where the robot is a 1 x 1 square that starts at (1, 1), there are no obstacles, and the goal is (19, 19).



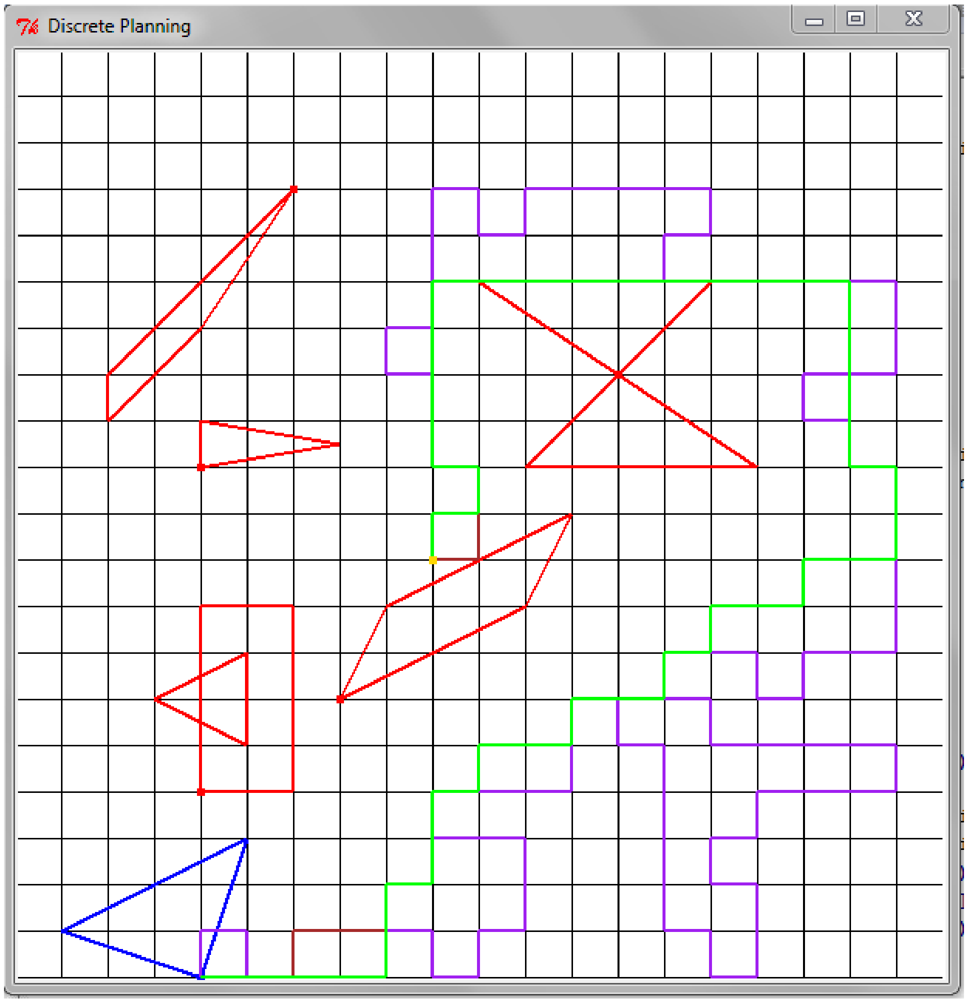
Search Algorithm	Run time (s)	Path length (steps)
DFS	0.118	180
BFS	0.176	36
UCS	0.212	36
A* with Not Admissible Heuristic	0.035	36
A* with Manhattan Distance	0.207	36

Medium World - A 20 x 20 grid where the robot is a 1 x 1 square that starts at (1, 1), the only obstacle is a rectangle that blocks the left side of the world, and the goal is (1, 19).



Search Algorithm	Run time (s)	Path length (steps)
DFS	0.638	126
BFS	1.293	46
UCS	1.304	46
A* with Not Admissible Heuristic	0.641	46
A* with Manhattan Distance	1.041	46

Hard world - A 20 x 20 grid where the robot is a triangle that starts at (4, 0), there are 5 complex obstacles that prevent direct paths, and the goal is (9, 9).



Search Algorithm	Run time (s)	Path length (steps)
DFS	2.180	96
BFS	3.945	48
UCS	3.958	48
A* with Not Admissible Heuristic	3.111	48
A* with Manhattan Distance	3.329	48