Homework 3

Due Monday by 11:59pm **Points** 15 **Available** Feb 15 at 8am - Mar 1 at 11:59pm 15 days

Your task this week is to find more efficient ways to guide Sammy the Spartan on the optimal path in his quest in bigger and more complex mazes.

The starter code is similar to that of homework 2 and is available in HW3.zip.

In addition to *dfs*, *bfs* and *ucs*, the main program (spartanquest.py) now allows you to specify *astar* as the search algorithm.

python spartanquest.py sjsu.txt **astar**

Note that if you do not specify a heuristic, the program defaults to the trivial heuristic, *null_heuristic*.

null_heuristic is implemented for you in informed_search.py.

You can also specify a heuristic:

python spartanquest.py sjsu.txt astar gen_heuristic

Your task is to help Sammy the Spartan find the optimal path more efficiently by implementing the following:

1. **A* graph search algorithm.** Fill in your code in the function ***astar*** in the Python module informed_search.py.
   Note that if you run A* with the null heuristic, you should get the same results as with the uniform cost search:
   python spartanquest.py sjsu.txt astar null_heuristic
   Path length: 52
   Path cost: 130
   Number of nodes expanded: 539

2. A simple heuristic, based on the Manhattan distance, that can be used when we know for sure that there is only one medal in the quest. Fill in in your code in the function ***single_heuristic*** in the Python module informed_search.py. A* with this simple heuristic should find the optimal solution faster than uniform cost search for the quest described in questE.txt (108 nodes expanded vs 326 nodes for ucs). To get any credit on this question, your heuristic must be admissible and consistent.

python spartanquest.py questE.txt astar single_heuristic
Path length: 12
Path cost: 28
Number of nodes expanded: 108

3. A better heuristic to be used when we know for sure that there is only one medal in the quest. Fill in in your code in the function ***better_heuristic*** in the Python module informed_search.py. A* with this better heuristic should find the optimal solution even faster for the quest described in questE.txt: less than 40 nodes expanded vs 326 nodes for ucs and 108 for single_heuristic. My implementation expands 23 nodes. Can you beat it? To get any credit on this question, your heuristic must be admissible and consistent.
python spartanquest.py questE.txt astar better_heuristic
Path length: 12
Path cost: 28
Number of nodes expanded: 23

4. A more general heuristic to be used when the maze contains an arbitrary number of medals and these medals can be anywhere in the maze. Fill in your code in the function ***gen_heuristic*** in the Python module informed_search.py. A* with this general heuristic should find the optimal solution faster for the quest described in questF.txt: ucf expands 31,852 nodes. You will be graded based on how many nodes your heuristic expands. My implementation expands 2,335 nodes. Can you beat it?

Number of nodes expanded:	Grade
> 20,000	0
<= 20,000	1
<= 15,000	3
	5

<= 2,500	

If you implement *gen_heuristic* correctly, you should be able to use it to solve the quest described in questG.txt within a reasonable time frame and with less than 100,000 nodes expanded. Note that the ucs solution to that same maze expands 1,641,588 nodes and takes over 2 minutes on my laptop.

You should test your heuristic with ALL the quests provided (SJSU, questA through questI and noway). Make sure that you are getting the optimal solution in all these cases. **If you get a solution that is non optimal, this could be an indication that your heuristic is not admissible or not consistent.**

Please note that if your heuristic is not admissible or non-consistent, you will not get any credit, regardless of the number of nodes expanded.

Make sure that your heuristic is reducing the total compute time and not increasing it!

Make sure you fill in the docstrings for the heuristics to explain what each of them does.

Feel free to implement and use any helper functions to keep your code readable.

Even though you need all the files to test your program, you only need to upload your solution in informed_search.py.

Please make sure you read and follow the grading rubric to ensure full credit.

Homework 3 Grading Rubric

Criteria	Ratings		Pts
A* Graph search is implemented correctly	3.0 pts Full Marks	0.0 pts No Marks	3.0 pts
A* implementation uses the PriorityQueue class	1.0 pts Full Marks	0.0 pts No Marks	1.0 pts
A* implementation uses the Node class and its solution method	1.0 pts Full Marks	0.0 pts No Marks	1.0 pts
single_heuristic *The heuristic is admissible and consistent and expands 108 nodes for questE. The function docstring explains what the heuristic does. python spartanquest.py questE.txt astar single_heuristic Path length: 12 Path cost: 28 Number of nodes expanded: 108*	2.0 pts Full Marks	0.0 pts No Marks	2.0 pts
better_heuristic *The heuristic is admissible and consistent and expands fewer than 40 nodes. The function docstring explains what the heuristic does. python spartanquest.py questE.txt astar better_heuristic Path length: 12 Path cost: 28 Number of nodes expanded: 23*	3.0 pts Full Marks	0.0 pts No Marks	3.0 pts

Criteria	Ratings			Pts
The gen_heuristic function *The heuristic is consistent and admissible and expands fewer than 2,500 nodes for questF. The function docstring explains what the heuristic does. python spartanquest.py questF.txt astar gen_heuristic Path length: 73 Path cost: 141*	5.0 pts Admissible and consistent AND Number of Nodes expanded <= 2,500	3.0 pts Admissible and consistent AND Number of Nodes expanded <= 15,000	1.0 pts Admissible and consistent AND Number of Nodes expanded <= 20,000	5.0 pts
				Total Points: 15.0