Name: Phan Tran Thanh Huy
ID: ITCSIU22056

IT159: Intro to Artificial Intelligence
Lab#3/Assignment#3
Informed Search in Pac-Man

There are three exercises in this lab:

1. Best-first search
2. A* Search

Introduction

In this assignment, your Pac-Man agent will find paths through its maze world to reach a particular location. You will build general search algorithms and apply them to many Pac-Man scenarios.

Files you'll edit:

`search.py`	Where all of your search algorithms will reside.
`searchAgents.py`	Where all your search-based agents will reside. [ONLY for Ex: 3]

Files you should look at but NOT edit:

`util.py`	Useful data structures for implementing search algorithms.
`pacman.py`	The main file that runs Pac-Man games. This file describes a Pac-Man `GameState` type, which you use in this lab.
`game.py`	The logic behind how the Pac-Man world works. This file describes several supporting types like `AgentState`, `Agent`, `Direction`, and `Grid`.

Finding a fixed food dot using Informed Search

Exercise 1: Implement the Best-First Search (BFS) algorithm in the `bestFirstSearch` function in `search.py`. Test your code the same way you did for other search algorithms.

```
python pacman.py -l tinyMaze -p SearchAgent -a fn=befs
```

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l  tinyMaze -p SearchAgent -a fn=befs
[SearchAgent] using function befs and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 8 in 0.0 seconds
Search nodes expanded: 15
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
python pacman.py -l mediumMaze -p SearchAgent -a fn=befs
```

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l  mediumMaze -p SearchAgent -a fn=befs
[SearchAgent] using function befs and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 269
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
python pacman.py -l bigMaze -p SearchAgent -a fn=befs -z .5
```

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l  bigMaze -p SearchAgent -a fn=befs
[SearchAgent] using function befs and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.1 seconds
Search nodes expanded: 620
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Does BFS find a least cost solution? How many nodes are expanded?

- Yes, because BFS finds the least-cost solution only if all step costs are equal. But if step costs are different, UCS is the only algorithm that guarantees the least-cost path.
- In tinyMaze, it expands 15 nodes. In mediumMaze, it expands 269 nodes. In bigMaze, it expands 620 nodes.

Exercise 2: Implement the A* Search algorithm in the `aStarSearch` function in `search.py`. Use the same algorithm as shown in your text (or class). `aStarSearch` function takes an optional heuristic function as an argument. The heuristic function itself takes two arguments (a state in the search problem, and the problem itself). `search.py` provides a `nullHeuristic` function that

you can look at. Also, in the `searchAgents.py` a Manhattan heuristic as well as Euclidian heuristic function is defined. Test your code the same way you did for other search algorithms.

```
python pacman.py -l tinyMaze -p SearchAgent -a fn=astar
```

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l  tinyMaze -p SearchAgent -a fn=astar
[SearchAgent] using function astar and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 8 in 0.0 seconds
Search nodes expanded: 15
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
python pacman.py -l mediumMaze -p SearchAgent -a fn=astar
```

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l  mediumMaze -p SearchAgent -a fn=astar
[SearchAgent] using function astar and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 269
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
python pacman.py -l bigMaze -p SearchAgent -a fn=astar -z .5
```

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l  bigMaze -p SearchAgent -a fn=astar
[SearchAgent] using function astar and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.0 seconds
Search nodes expanded: 620
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
```

To specify a heuristic function from `searchAgents.py`, use the following:

```
python pacman.py -l bigMaze -z .5 -p SearchAgent -a
                                  fn=astar,heuristic=manhattanHeuristic
```

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l  bigMaze -z .5 -p SearchAgent -a fn=astar,heurist
ic=manhattanHeuristic
[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.0 seconds
Search nodes expanded: 549
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
```

The function `manhattanHeuristic()` is already written in `searchAgents.py`. Alternatively, you could write your own in `search.py`

What to submit

1. Fill out the table below:

	Best First Search			A* Search		
Maze	**#nodes expanded**	**Solution length**	**Is it optimal?**	**#nodes expanded**	**Solution length**	**Is it optimal?**
tiny	15	8	Yes	15	8	Yes
medium	269	68	Yes	269	68	Yes
big	620	210	Yes	620	210	Yes

2. What happens on `openMaze` for the various search strategies?

```
python pacman.py -l openMaze -p SearchAgent -a fn=befs -z .5
```

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l openMaze -p SearchAgent -a fn=befs
[SearchAgent] using function befs and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.1 seconds
Search nodes expanded: 682
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
python pacman.py -l openMaze -p SearchAgent -a fn=astar -z .5
```

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l openMaze -p SearchAgent -a fn=astar -z .5
[SearchAgent] using function astar and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.0 seconds
Search nodes expanded: 682
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
```

python pacman.py -l openMaze -p SearchAgent -a fn=dfs -z .5

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l openMaze -p SearchAgent -a fn=dfs
 -z .5
[SearchAgent] using function dfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 298 in 0.2 seconds
Search nodes expanded: 806
Pacman emerges victorious! Score: 212
Average Score: 212.0
Scores:        212.0
Win Rate:      1/1 (1.00)
Record:        Win
```

python pacman.py -l openMaze -p SearchAgent -a fn=bfs -z .5

```
PS E:\Homework\AI\AIEXT - edited\search\template>  python pacman.py -l openMaze -p SearchAgent -a fn=bf
s -z .5
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.1 seconds
Search nodes expanded: 682
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
```

python pacman.py -l openMaze -p SearchAgent -a fn=ucs -z .5

```
PS E:\Homework\AI\AIEXT - edited\search\template>  python pacman.py -l openMaze -p SearchAgent -a fn=uc
s -z .5
[SearchAgent] using function ucs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.0 seconds
Search nodes expanded: 682
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
```

	Open Maze		
Search	**#nodes expanded**	**Solution length**	**Is it optimal?**
Best-First Search (BFS)	682	54	Yes
A star	682	54	Yes
Depth-First Search	806	298	No
Breadth-First Search	682	54	Yes
Uniform Cost Search	682	54	Yes

3. For each exercise where a heuristic is used, clearly show/mention the heuristic function.

```
python pacman.py -l openMaze -z .5 -p SearchAgent -a
                                fn=befs,heuristic=manhattanHeuristic
```

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l  bigMaze -z .5 -p SearchAgent -a fn=befs,heuristi
c=manhattanHeuristic
[SearchAgent] using function befs and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.0 seconds
Search nodes expanded: 466
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
python pacman.py -l openMaze -z .5 -p SearchAgent -a
                                    fn=befs,heuristic=euclideanHeuristic
```

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l  bigMaze -z .5 -p SearchAgent -a fn=befs,heuristi
c=euclideanHeuristic
[SearchAgent] using function befs and heuristic euclideanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.1 seconds
Search nodes expanded: 471
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
python pacman.py -l openMaze -z .5 -p SearchAgent -a
                                    fn=astar,heuristic=manhattanHeuristic
```

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l  openMaze -z .5 -p SearchAgent -a fn=astar,heuris
tic=manhattanHeuristic
[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.0 seconds
Search nodes expanded: 535
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
python pacman.py -l openMaze -z .5 -p SearchAgent -a
                                    fn=astar,heuristic=euclideanHeuristic
```

```
PS E:\Homework\AI\AIEXT - edited\search\template> python pacman.py -l  bigMaze -z .5 -p SearchAgent -a fn=astar,heurist
ic=euclideanHeuristic
[SearchAgent] using function astar and heuristic euclideanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.0 seconds
Search nodes expanded: 557
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
```

4. Based on the above, a short discussion/reflection of how the searches compare to each other and to the uninformed searches from Assignment#2.

We can see that the BestFirstSearch and A*Search share the same result in three aspects which are nodes expanded, solution length, and optimal. All of them are run in tiny, medium, and big maze also leading to clearer in testing which is the same.

5. Source code includes `search.py` and `searchAgents.py`. This should include your code for the search node, Best-First Search, and A*.
6. Please create a folder called "yourname_studentID_Lab3" that includes all the required files and generate a zip file called "yourname_ studentID _Lab3.zip".
7. Please submit your work (.zip) to Blackboard.