Lab 3: Informed Search in Pac-Man

Name: Nguyễn Đình Khánh Ngân

ID: ITCSIU22236

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**

   Best First Search function

```python
def bestFirstSearch(problem, heuristic=nullHeuristic):
    frontier = PriorityQueue()
    start = problem.getStartState()
    frontier.push( item: (start, []), heuristic(start, problem))
    visited = set()
    while not frontier.isEmpty():
        state, path = frontier.pop()
        if problem.isGoalState(state):
            return path
        if state not in visited:
            visited.add(state)
            for successor, action, stepCost in problem.getSuccessors(state):
                if successor not in visited:
                    frontier.push( item: (successor, path + [action]), heuristic(successor, problem))
    return []
```

   Implement function in different maze layout

   tinyMaze

```
PS C:\Source\AI\Lab3\search> 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
```

   mediumMaze

```
PS C:\Source\AI\Lab3\search> 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
```

bigMaze

```
PS C:\Source\AI\Lab3\search> python pacman.py -l bigMaze -p SearchAgent -a fn=befs -z .5
[SearchAgent] using function befs 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
```

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

Yes, BFS find a least cost solution

tinyMaze: 15 nodes

mediumMaze: 269 nodes

bigMaze: 620 nodes

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**

A* Search function

```
def aStarSearch(problem, heuristic=nullHeuristic):  3 usages
    frontier = PriorityQueue()
    start = problem.getStartState()
    frontier.push( item: (start, []), heuristic(start, problem)) #The priority is g(n) + h(n), at begin it 0
    visited = {}
    visited[start] = 0
    while not frontier.isEmpty():
        state, path = frontier.pop()
        if problem.isGoalState(state):
            return path
        currentCost = visited[state]
        for successor, action, stepCost in problem.getSuccessors(state):
            newCost = currentCost + stepCost
            if successor not in visited or newCost < visited[successor]:
                visited[successor] = newCost
                priority = newCost + heuristic(successor, problem)
                frontier.push( item: (successor, path + [action]), priority)
    return []
```

Implement A* search function (search.py provides a nullHeuristic)

tinyMaze

```
PS C:\Source\AI\Lab3\search> 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
```

mediumMaze

```
PS C:\Source\AI\Lab3\search> 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
```

bigMaze

```
PS C:\Source\AI\Lab3\search> python pacman.py -l bigMaze -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 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
```

Implement A* search function (searchAgents.py a Manhattan heuristic)

```
PS C:\Source\AI\Lab3\search> python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=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
```

3. Compare table

	Best First Search			**A* Search**		
Maze	Nodes explored	Solution length	Is it optional	Nodes explored	Solution length	Is it optional
Tiny	15	8	Yes	15	8	Yes
Medium	269	68	Yes	269	68	Yes
Big	620	210	Yes	620	210	Yes

- What happens on openMaze for the various search strategies?
  BreathFirstSearch, AStarSearch, UniformCostSearch, and BestFirstSearch all display the same path and path cost
  Despite not optimizing, depthFirstSearch searches every inch of an open maze, increasing the path cost
- Based on the above, a short discussion/reflection of how the searches compare to each We can observe that in three areas—nodes expanded, solution length, and optimal—BestFirstSearch and A*Search yield identical results. They are all conducted in small, medium, and large mazes, which also makes the testing process more transparent.