Lab 2: Uniformed Search in Pac-Man

Name: Nguyễn Đình Khánh Ngân

ID: ITCSIU22236

1. **Implement the depth-first search algorithm in the `depthFirstSearch` function in `search.py`. Although DFS and BFS ignore the costs, you'll need them for later search methods**

   Function Depth first search

```
def depthFirstSearch(problem):  1 usage
    frontier = Stack()
    visited = set()
    frontier.push((problem.getStartState(), []))
    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((successor, path + [action]))
    return []
```

   Implement

   tinyMaze

```
PS C:\Source\AI\Lab2\search> python pacman.py -l tinyMaze -p SearchAgent -a fn=dfs
[SearchAgent] using function dfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 10 in 0.0 seconds
Search nodes expanded: 15
Pacman emerges victorious! Score: 500
Average Score: 500.0
Scores:        500.0
Win Rate:      1/1 (1.00)
Record:        Win
```

   mediumMaze

```
PS C:\Source\AI\Lab2\search> python pacman.py -l mediumMaze -p SearchAgent -a fn=dfs
[SearchAgent] using function dfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 130 in 0.0 seconds
Search nodes expanded: 146
Pacman emerges victorious! Score: 380
Average Score: 380.0
Scores:        380.0
Win Rate:      1/1 (1.00)
Record:        Win
```

bigMaze

```
PS C:\Source\AI\Lab2\search> python pacman.py -l bigMaze -p SearchAgent -a fn=dfs
[SearchAgent] using function dfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.0 seconds
Search nodes expanded: 390
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
```

2. **Implement the breadth-first search algorithm in the `breadthFirstSearch` function in `search.py`. Use the same algorithm as shown in the above pseudocode. Test your code the same way you did for depth-first search**

Function Breadth first search

```
def breadthFirstSearch(problem):  2 usages
    frontier = Queue()
    visited = set()
    start = problem.getStartState()
    frontier.push((start, []))
    visited.add(start)
    while not frontier.isEmpty():
        state, path = frontier.pop()
        if problem.isGoalState(state):
            return path
        for successor, action, stepCost in problem.getSuccessors(state):
            if successor not in visited:
                visited.add(successor)
                frontier.push((successor, path + [action]))
    return []
```

Implement

ttinyMaze

```
PS C:\Source\AI\Lab2\search> python pacman.py -l tinyMaze -p SearchAgent -a fn=bfs
[SearchAgent] using function bfs
[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\Lab2\search> python pacman.py -l mediumMaze -p SearchAgent -a fn=bfs
[SearchAgent] using function bfs
[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\Lab2\search> python pacman.py -l bigMaze -p SearchAgent -a fn=bfs
[SearchAgent] using function bfs
[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
```

3. **Implement the uniform-cost search algorithm in the `uniformCostSearch` function in `search.py`. Does UCS find a least cost solution? How many nodes are expanded?**
Yes, Uniform-Cost Search (UCS) always finds the least-cost solution when all step costs are positive. This is because it expands the lowest-cost node first, ensuring that once a node is expanded, the shortest path to it has been found.
Function Uniformed cost search

```python
def uniformCostSearch(problem):
    frontier = PriorityQueue()
    visited = {}
    start = problem.getStartState()
    frontier.push( item: (start, []),  priority: 0)
    visited[start] = 0
    while not frontier.isEmpty():
        state, path = frontier.pop()
        if problem.isGoalState(state):
            return path
        cost = visited[state]
        for successor, action, stepCost in problem.getSuccessors(state):
            newCost = cost + stepCost
            if successor not in visited or newCost < visited[successor]:
                visited[successor] = newCost
                frontier.push( item: (successor, path + [action]), newCost)
    return []
```

Implement
ttinyMaze

```
PS C:\Source\AI\Lab2\search> python pacman.py -l tinyMaze -p SearchAgent -a fn=ucs
[SearchAgent] using function ucs
[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\Lab2\search> python pacman.py -l mediumMaze -p SearchAgent -a fn=ucs
[SearchAgent] using function ucs
[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\Lab2\search> python pacman.py -l bigMaze -p SearchAgent -a fn=ucs
[SearchAgent] using function ucs
[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
```

4. Compare table

	Depth-First Search			**Breadth-First Search**			**Uniform-Cost Search**		
Maze	Nodes explored	Solution length	Is it optional	Nodes explored	Solution length	Is it optional	Nodes explore	Solution length	Is it optional
Tiny	15	~10	No	15	9	Yes	~15	9	Yes
Medium	~146	130	No	~269	68	Yes	~269	68	Yes

Big	~390	210	No	~620	210	Yes	~620	210	Yes

- **DFS:** Expands fewer nodes quickly in some cases, but can wander around and produce a suboptimal (longer) path
- **BFS:** Guarantees the shortest path in number of actions for uniform step costs, typically expands more nodes than DFS on these mazes but yields an optimal solution
- **UCS:** Identical path length to BFS under uniform cost = 1, but more general (handles varying step costs). It also expands about the same or sometimes slightly more nodes than BFS