VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY

INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

IT159IU

REPORT LAB 3

Instructor:

Dr. Nguyen Trung Ky

Dr. Ly Tu Nga

Nguyen Huynh Ngan Anh - ITDSIU23003

0. Setup:

a. Running befs, astar in tinyMaze, mediumMaze, bigMaze with nullHeuristic and manhattanHeuristic:

bestFirstSearch algorithm

```python
def bestFirstSearch(problem: SearchProblem, heuristic=nullHeuristic):
    """Search the node that has the lowest heuristic cost first."""
    "*** YOUR CODE HERE ***"
    visited_list = []
    my_prioqueue = util.PriorityQueue()
    init_state = problem.getStartState()
    my_prioqueue.push(Superstate(init_state, None, None, heuristic(init_state, problem)), heuristic(init_state, problem))

    while not my_prioqueue.isEmpty():
        current_superstate = my_prioqueue.pop()

        if problem.isGoalState(current_superstate.state):
            re_solution = []
            while current_superstate.move_from_prev is not None:
                re_solution = [current_superstate.move_from_prev] + re_solution
                current_superstate = current_superstate.prev
            return re_solution

        if current_superstate.state not in visited_list:
            visited_list.append(current_superstate.state)
            for successor in problem.getSuccessors(current_superstate.state):
                successor_state, action, step_cost = successor
                if successor_state not in visited_list:
                    my_prioqueue.push(
                        Superstate(successor_state, current_superstate, action, heuristic(successor_state, problem)),
                        heuristic(successor_state, problem)
                    )
    util.raiseNotDefined()
```

With **nullHeuristic**:

- Befs:

```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\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
```

```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\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
```

```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\search\template>  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
```

- A*:

```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\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
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\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
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\search\template>  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
```

With **manhattanHeuristic**:

- Befs:

```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\search\template> python pacman.py -l tinyMaze -p SearchAgent -a fn=befs,heuristic=manhattanHeuristic

[SearchAgent] using function befs and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 8 in 0.0 seconds
Search nodes expanded: 8
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\search\template> python pacman.py -l mediumMaze -p SearchAgent -a fn=befs,heuristic=manhattanHeuristic
[SearchAgent] using function befs and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 74 in 0.0 seconds
Search nodes expanded: 78
Pacman emerges victorious! Score: 436
Average Score: 436.0
Scores:        436.0
Win Rate:      1/1 (1.00)
Record:        Win
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\search\template> python pacman.py -l bigMaze -p SearchAgent -a fn=befs,heuristic=manhattanHeuristic -z
5
[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
```

- A*:

```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\search\template> python pacman.py -l tinyMaze -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 8 in 0.0 seconds
Search nodes expanded: 14
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\search\template> python pacman.py -l mediumMaze -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 68 in 0.0 seconds
Search nodes expanded: 221
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Record:        Win
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\search\template> python pacman.py -l bigMaze -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic -z
.5
[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
```

b. Running befs, astar in openMaze with nullHeuristic and manhattanHeuristic:

With **nullHeuristic**:

```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\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 0.0 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
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\search\template> python pacman.py -l openMaze -p SearchAgent -a fn=astar
[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
```

With **manhattanHeuristic**:

```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\search\template> python pacman.py -l openMaze -p SearchAgent -a fn=befs,heuristic=manhattanHeuristic
[SearchAgent] using function befs and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 89
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\ArtificialIntelligence\Lab3\sourcecode\search\template> python pacman.py -l openMaze -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 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
```

1. Exercise 1:

With **nullHeuristic**:

	Best First Search			A* Search		
Maze	**#nodes expanded**	**Solution length**	**Is it optimal?**	**#nodes expanded**	**Solution length**	**Is it optimal?**
tiny	15	8	No	15	8	No
medium	269	68	No	269	68	No
big	620	210	No	620	210	No

With **manhattanHeuristic**:

	Best First Search			A* Search		
Maze	**#nodes expanded**	**Solution length**	**Is it optimal?**	**#nodes expanded**	**Solution length**	**Is it optimal?**
tiny	8	8	Yes	14	8	No
medium	78	74	Yes	221	68	No
big	466	210	Yes	549	210	No

2. Exercise 2:

	Best First Search			A* Search		
Maze	**#nodes expanded**	**Solution length**	**Is it optimal?**	**#nodes expanded**	**Solution length**	**Is it optimal?**

nullHeuristic	682	54	No	682	54	No
manhattanHeuristic	89	68	Yes	535	54	No

Overall, the performance of each search strategy on OpenMaze depends on the characteristics of the maze and the search problem, as well as the parameters and implementation details of the search algorithm.

- For Null Heuristic, BEFS and A* behave like UCS, they expand many nodes and do not necessarily find the best path.
- For Manhattan Heuristic, BEFS is faster, A* is more optimal.

=> BEFS with Manhattan Heuristic expands fewer nodes but has a longer path → use for efficiency.

=> A* with Manhattan Heuristic finds a shorter path but expands more nodes → use for optimal paths.

3. **Exercise 3:**

nullHeuristic: is straightforward but ineffective because it consistently returns 0, regardless of where the objective or the Pac-Man is. It lacks knowledge of the area. In other words, a null heuristic does not provide any information or guidance to the search algorithm and is equivalent to performing a search algorithm.

manhattanHeuristic: is more intelligent because it makes calculations based on the locations of the Pacman and the objective. It gives the algorithms clues regarding how near or far Pacman is from the objective. When a heuristic function is used in a search algorithm, it is usually incorporated into the cost function or evaluation function that determines the priority of nodes in the search frontier. For example, in A* search, the cost of a node is the sum of the actual cost of the path so far and the estimated cost to the goal based on the heuristic function. The evaluation function for A* is the sum of the cost function and a tie-breaking function that ensures nodes with the same cost are explored in a consistent order. It is knowledgeable about the area.

4. **Exercise 4:**

With ignorant algorithms in Assignment#2, BEFS and A* both perform equally well if combined with the nullHeuristic.

But when combined with ManhattanHeuristic, BEFS, and A* outperform the same uninformed algorithms in Assignment#2 while using fewer enlarged nodes. In my trials, BEFS outperforms A*.