

BÁO CÁO KẾT QUẢ CÁC THUẬT TOÁN

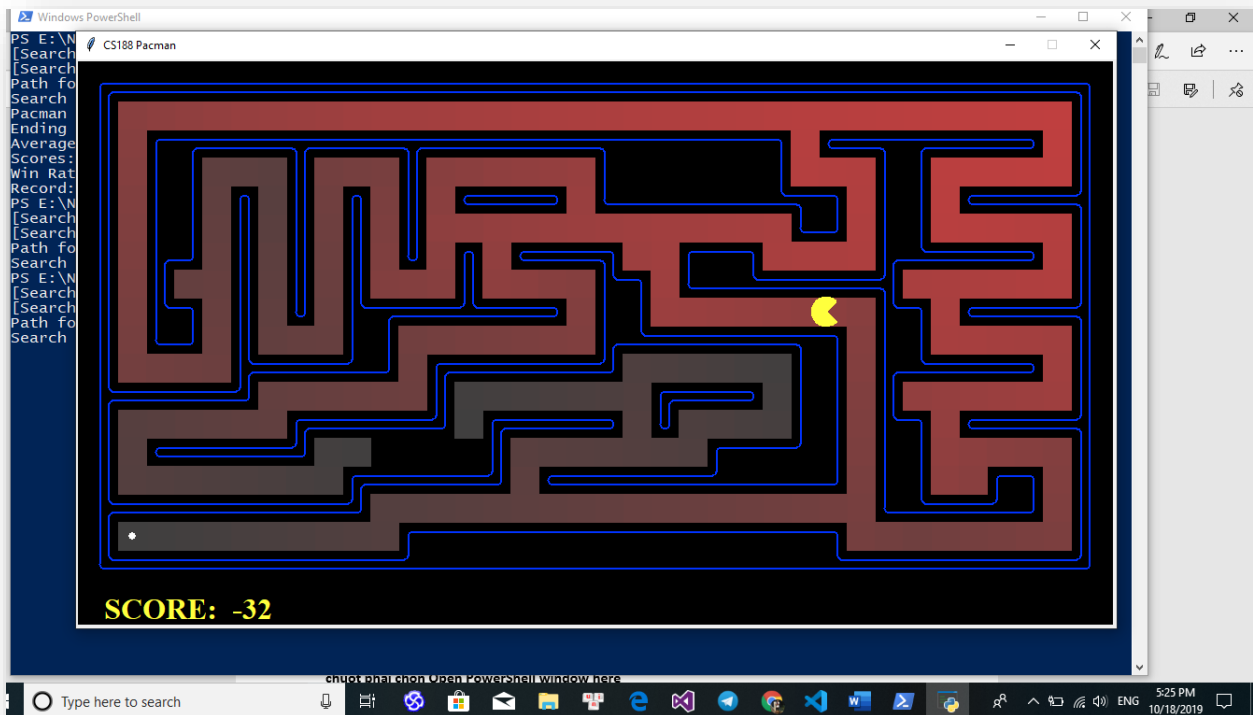
1) THUẬT TOÁN TIỀM KIỂM BFS

-Trong thuật toán này mình sẽ sử dụng hàng đợi (QUEUE) để giải quyết vấn đề tìm đường đi đến đích cho PacMan(Không quan tâm chi phí).Thuật toán này khá giống DFS , chỉ khác mỗi chỗ là bên thuật toán DFS sử dụng ngăn xếp để lấy ra các điểm, hành động và hướng đi cho PacMan.Còn với BFS thì sử dụng ngăn xếp để tìm kiếm (tìm kiếm theo chiều rộng xung quanh các điểm lân cận).

-Demo thuật toán:

```
def breadthFirstSearch(problem):  
    """Search the shallowest nodes in the search tree first."""  
    """ YOUR CODE HERE """  
    # trong thuật toán này thì chỉ khác DFS cái cấu trúc dữ liệu QUEUE thôi , thay Stack bằng Queue  
    # gọi problem.getStartState() để lấy trạng thái bắt đầu, mình sẽ lấy 1 bộ các điểm và hướng đi  
    start = problem.getStartState()  
    # Lúc ban đầu thì vị trí pacman chưa có và các điểm cũng vậy  
    exploredState = []  
    # bắt đầu thêm các điểm  
    exploredState.append(start)  
    states = util.Queue()  
    stateTuple = (start, [])  
    states.push(stateTuple)  
    while not states.isEmpty():  
        state, action = states.pop()  
        # problem.isGoalState(state) để kiểm tra trạng thái state có phải là trạng thái đích không ,lấy 1 điểm  
        if problem.isGoalState(state):  
            return action  
        # problem.getSuccessors(state) để mở trạng thái state, trả về 1 danh sách gồm hướng đi qua các list hành động  
        successor = problem.getSuccessors(state)  
        for i in successor:  
            coordinates = i[0]  
            if not coordinates in exploredState:  
                direction = i[1]  
                exploredState.append(coordinates)  
                states.push((coordinates, action + [direction]))  
    return action  
    util.raiseNotDefined()
```

-Kết quả chạy thuật toán:



Cuối cùng, bạn gõ **python autograder.py -q q2** để kiểm tra phần cài đặt của bạn với các bộ test khác nhau.

```
PS E:\NĂM 3- HK1\Trí tuệ nhân tạo\ThucHanh\1760197\Source> python autograder.py -q q2
autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's document
n for alternative uses
import imp
Starting on 10-18 at 17:26:22

Question q2
=====
*** PASS: test_cases\q2\graph_backtrack.test
*** solution: ['1:A->C', '0:C->G']
*** expanded_states: ['A', 'B', 'C', 'D']
*** PASS: test_cases\q2\graph_bfs_vs_dfs.test
*** solution: ['1:A->G']
*** expanded_states: ['A', 'B']
*** PASS: test_cases\q2\graph_infinite.test
*** solution: ['0:A->B', '1:B->C', '1:C->G']
*** expanded_states: ['A', 'B', 'C']
*** PASS: test_cases\q2\graph_manypaths.test
*** solution: ['1:A->C', '0:C->D', '1:D->F', '0:F->G']
*** expanded_states: ['A', 'B1', 'C', 'B2', 'D', 'E1', 'F', 'E2']
*** PASS: test_cases\q2\pacman_1.test
*** pacman layout: mediumMaze
*** solution length: 68
*** nodes expanded: 269

### Question q2: 3/3 ###

Finished at 17:26:22

Provisional grades
=====
Question q2: 3/3
-----
Total: 3/3
```

Bạn cũng có thể kiểm tra hàm BFS của bạn với bài toán 8-puzzle bằng cách gõ **python eightpuzzle.py**

```
Windows PowerShell
PS E:\NAM 3- HK1\Trí tuệ nhân tạo\ThucHanh\1760197\Source> python eightpuzzle.py
A random puzzle:
-----
| 1 | 4 | 2 |
| 3 | 8 | 7 |
| 6 |   | 5 |
-----
BFS found a path of 7 moves: ['up', 'right', 'down', 'left', 'up', 'up', 'left']
After 1 move: up
-----
| 1 | 4 | 2 |
| 3 |   | 7 |
| 6 | 8 | 5 |
-----
Press return for the next state...
After 2 moves: right
-----
| 1 | 4 | 2 |
| 3 | 7 |   |
| 6 | 8 | 5 |
-----
Press return for the next state...
After 3 moves: down
-----
| 1 | 4 | 2 |
| 3 | 7 | 5 |
| 6 | 8 |   |
-----
```

```

After 4 moves: left
-----
| 1 | 4 | 2 |
-----
| 3 | 7 | 5 |
-----
| 6 |   | 8 |
-----
Press return for the next state...
After 5 moves: up
-----
| 1 | 4 | 2 |
-----
| 3 |   | 5 |
-----
| 6 | 7 | 8 |
-----
Press return for the next state...
After 6 moves: up
-----
| 1 |   | 2 |
-----
| 3 | 4 | 5 |
-----
| 6 | 7 | 8 |
-----
Press return for the next state...
After 7 moves: left
-----
|   | 1 | 2 |
-----
| 3 | 4 | 5 |
-----
| 6 | 7 | 8 |
-----

```

2) THUẬT TOÁN TIỀM KIỂM UCS

-Trong thuật toán này mình sử dụng cấu trúc ***hàng đợi ưu tiên*** đã khai báo trong file Util.py .

Tiếp đến thuật toán này sẽ tìm đường đi đi đến đích với chi phí ít nhất($F=G$).

Mình sẽ tìm đường đi cho Pacman với các điểm chưa xét có chi phí thấp nhất. Mỗi hành động của Pacman đi tìm đường đi sẽ được thêm vào 1 bộ trong khi lấy trạng thái ban đầu. Và chọn ra 1 điểm để kiểm tra nó có phải là trạng thái đích và trả về 1 danh sách khi mở trạng thái.

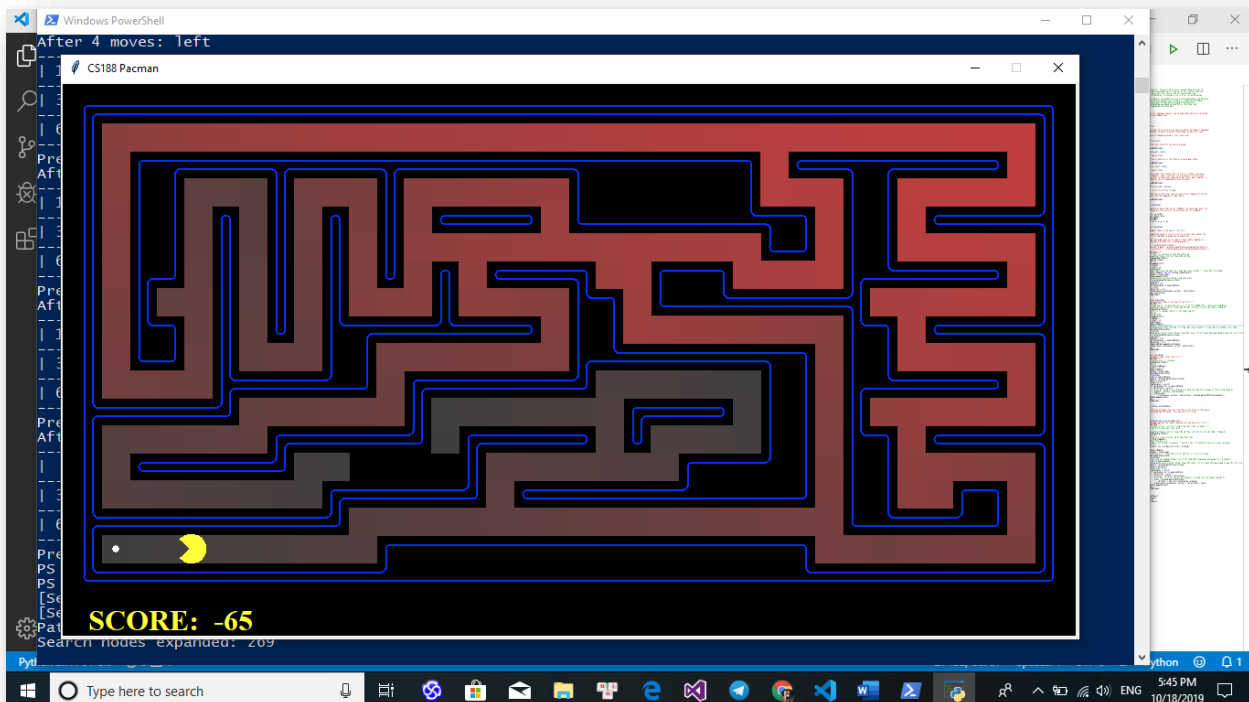
-Hình ảnh demo thuật toán:

```

7
8 def uniformCostSearch(problem):
9     """Search the node of least total cost first."""
10    """** YOUR CODE HERE """
11    # UCS là sd hàng đợi ưu tiên, sd Queue
12    start = problem.getStartState()
13    exploredState = []
14    states = util.PriorityQueue()
15    states.push((start, []), 0)
16    while not states.isEmpty():
17        state, actions = states.pop()
18        if problem.isGoalState(state):
19            return actions
20        if state not in exploredState:
21            successors = problem.getSuccessors(state)
22            for succ in successors:
23                # gán tọa độ
24                coordinates = succ[0]
25                if coordinates not in exploredState:
26                    directions = succ[1]
27                    # chi phí đường đi mới sẽ bằng hành động tại điểm đó mà pacman có thể đi cộng hướng đi
28                    newCost = actions + [directions]
29                    states.push(
30                        (coordinates, actions + [directions]), problem.getCostOfActions(newCost))
31            exploredState.append(state)
32    return actions
33    util.raiseNotDefined()
34
35

```

-Kết quả chạy thuật toán:



```

PS E:\NĂM 3- HK1\Trí tuệ nhân tạo\ThucHanh\1760197\Source> 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)

```

-Lưu ý: với mỗi thuật toán thì mình sẽ có mỗi bộ test khác nhau

```

import imp
Starting on 10-18 at 17:46:55
Question q3
=====
*** PASS: test_cases\q3\graph_backtrack.test
***   solution:      ['1:A->C', '0:C->G']
***   expanded_states: ['A', 'B', 'C', 'D']
*** PASS: test_cases\q3\graph_bfs_vs_dfs.test
***   solution:      ['1:A->G']
***   expanded_states: ['A', 'B']
*** PASS: test_cases\q3\graph_infinite.test
***   solution:      ['0:A->B', '1:B->C', '1:C->G']
***   expanded_states: ['A', 'B', 'C']
*** PASS: test_cases\q3\graph_manypaths.test
***   solution:      ['1:A->C', '0:C->D', '1:D->F', '0:F->G']
***   expanded_states: ['A', 'B1', 'C', 'B2', 'D', 'E1', 'F', 'E2']
*** PASS: test_cases\q3\ucs_0_graph.test
***   solution:      ['right', 'Down', 'Down']
***   expanded_states: ['A', 'B', 'D', 'C', 'G']
*** PASS: test_cases\q3\ucs_1_problemC.test
***   pacman layout: mediumMaze
***   solution length: 68
***   nodes expanded: 269
*** PASS: test_cases\q3\ucs_2_problemE.test
***   pacman layout: mediumMaze
***   solution length: 74
***   nodes expanded: 260
*** PASS: test_cases\q3\ucs_3_problemW.test
***   pacman layout: mediumMaze
***   solution length: 152
***   nodes expanded: 173
*** PASS: test_cases\q3\ucs_4_testSearch.test
***   pacman layout: testSearch
***   solution length: 7
***   nodes expanded: 14
*** PASS: test_cases\q3\ucs_5_goalAtDequeue.test
***   solution:      ['1:A->B', '0:B->C', '0:C->G']
***   expanded_states: ['A', 'B', 'C']
### Question q3: 3/3 ###

```

3) THUẬT TOÁN TIỀM KIỂM A*

-Là một những thuật toán tối ưu trong 2 thuật toán trên. Thuật toán này mình có thể biết khả năng có điểm kết thúc(đích) trước và với chi phí ước lượng của mình mà Pacman có thể qua hành động đi với các hướng qua khung chương trình tìm kiếm.

$F=G+H$ (Trạng thái đích chính bằng chi phí ước lượng + chi phí thực sự ở mỗi điểm trong khung chương trình pacman đi qua để tìm kiếm điểm đích gần nhất.). Thuật toán này khá giống UCS

nhưng tối ưu hơn qua chi phí ước lượng và sử dụng *Heurictis đo khoảng cách Manhattan* không như tìm kiếm điểm mù (DFS, BFS, UCS).

-Demo thuật toán:

```
def aStarSearch(problem, heuristic=nullHeuristic):
    """Search the node that has the lowest combined cost and heuristic first."""
    """ YOUR CODE HERE """
    # Đây là thuật toán tối ưu và tốt nhất trong những thuật toán tìm kiếm trên.
    # Mình sẽ sd các hàm đã khai báo trước để sd

    # gọi problem.getStartState() để lấy trạng thái bắt đầu, mình sẽ lấy 1 bộ các điểm và hướng đi
    start = problem.getStartState()
    exploredState = []
    # sd hàng đợi ưu tiên trong file Util.py đã khai báo trước
    states = util.PriorityQueue()
    # sd tìm kiếm a* theo Heuristic
    # Trong đây thay vì chỉ dùng g, t dùng g+h. Trong đó h được tính bằng hàm heuristic(state, problem).
    # thêm 1 điểm vào
    states.push((start, []), nullHeuristic(start, problem))
    nCost = 0
    while not states.isEmpty():
        state, actions = states.pop()
        # Nếu mà điểm đó có là trạng thái đích thì kết thúc và trả về hành động
        if problem.isGoalState(state):
            return actions
    # Nếu điểm mà k nằm trong cái khung chương trình có các điểm khác xung quanh khi pacman đi tìm đường đi
    if state not in exploredState:
        # problem.getSuccessors(state) để mở trạng thái state, trả về 1 danh sách gồm hướng đi qua các list hành động
        successors = problem.getSuccessors(state)
        for succ in successors:
            # gán biến tọa độ
            coordinates = succ[0]
            if coordinates not in exploredState:
                directions = succ[1]
                nActions = actions + [directions]
```

```

    if problem.isGoalState(state):
        return actions
    # Nếu điểm mà k nằm trong cái khuôn chương trình có các điểm khác xung quanh khi pacman đi tìm đường đi
    if state not in exploredState:
        # problem.getSuccessors(state) để mở trạng thái state, trả về 1 danh sách gồm hướng đi qua các list hành động
        successors = problem.getSuccessors(state)
        for succ in successors:
            # gán biến tọa độ
            coordinates = succ[0]
            if coordinates not in exploredState:
                directions = succ[1]
                nActions = actions + [directions]
                # tại đây, mình sẽ tính chi phí đường đi tìm kiếm cho hành động mà pacman đi
                nCost = problem.getCostOfActions(
                    nActions) + heuristic(coordinates, problem)
                states.push((coordinates, actions + [directions]), nCost)
        exploredState.append(state)
    return actions
    util.raiseNotDefined()

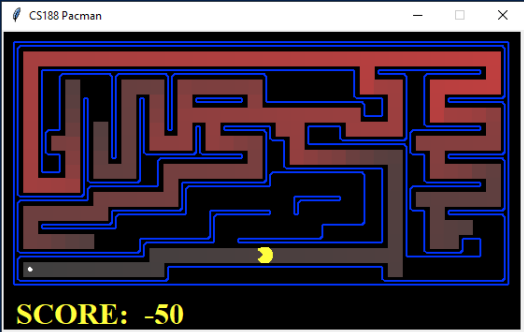
```

-Demo chạy chương trình:

```

PS E:\NĂM 3- HK1\Tri tu nhân tạo\ThucHanh\1760197\Source> python pacman.py -l mediumMaze -z .5 -p SearchAgent -a fn=ast
ar,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

```



Xét bộ test q4 cho thuật toán tìm kiếm A*


```

PS E:\NĂM 3- HK1\Tri tu nhân t\o\ThucHanh\1760197\Source> python autograder.py -q q4
autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentatio
n for alternative uses
  import imp
Starting on 10-18 at 18:03:37

Question q4
=====
*** PASS: test_cases\q4\astar_0.test
*** solution: ['Right', 'Down', 'Down']
*** expanded_states: ['A', 'B', 'D', 'C', 'G']
*** PASS: test_cases\q4\astar_1_graph_heuristic.test
*** solution: ['0', '0', '2']
*** expanded_states: ['S', 'A', 'D', 'C']
*** PASS: test_cases\q4\astar_2_manhattan.test
*** pacman layout: mediumMaze
*** solution length: 68
*** nodes expanded: 221
*** PASS: test_cases\q4\astar_3_goalAtDequeue.test
*** solution: ['1:A->B', '0:B->C', '0:C->G']
*** expanded_states: ['A', 'B', 'C']
*** PASS: test_cases\q4\graph_backtrack.test
*** solution: ['1:A->C', '0:C->G']
*** expanded_states: ['A', 'B', 'C', 'D']
*** PASS: test_cases\q4\graph_manypaths.test
*** solution: ['1:A->C', '0:C->D', '1:D->F', '0:F->G']
*** expanded_states: ['A', 'B1', 'C', 'B2', 'D', 'E1', 'F', 'E2']

### Question q4: 3/3 ###

Finished at 18:03:37

Provisional grades
=====
Question q4: 3/3
-----
Total: 3/3

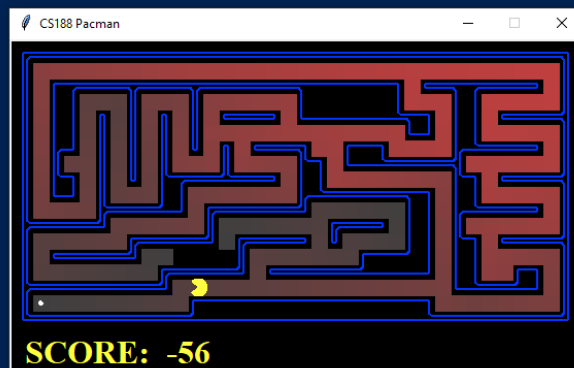
```

Nếu so sánh với thuật toán UCS thì bạn sẽ thấy số node mà thuật toán A* phải mở sẽ ít hơn.

```

PS E:\NĂM 3- HK1\Tri tu nhân t\o\ThucHanh\1760197\Source> python pacman.py -l mediumMaze -z .5 -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

```



THANK YOU!!!