Bài thuyết trình giữa kỳ môn Nhập môn trí tuệ nhân tạo

Nhóm 8:
Mã Trường Quang – 52100925
Huỳnh Tấn Đạt – 52200152
Lê Như Đạt – 52200160
Ngô Đức Anh Tuấn - 51800951

Câu 1. 8-Puzzle

Biểu diễn bài toán A* trong không gian trạng thái:

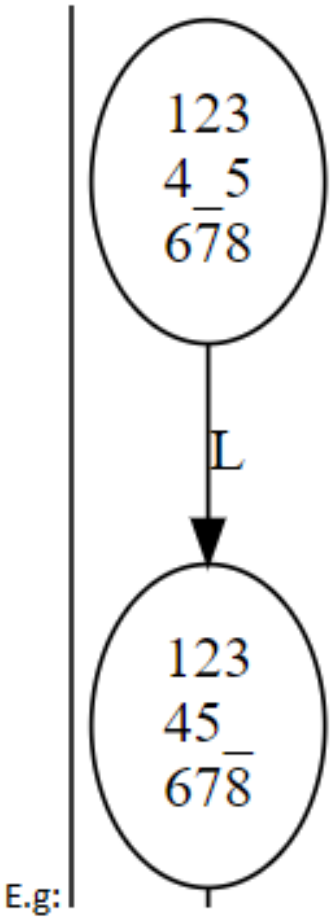
- Initial state: user's input
- Actions: the possible actions available (L, R, U, D)
- Transition model: what each action does
- Successor: a state reachable from a given state by a single action (các nước đi hợp lệ mà trạng thái cha có thể đi)
- Total_cost: len(node) in explored set
- Goal_Test:

Trạng thái đích

1	2	3
4	5	6
7	8	

hoặc

	1	2
3	4	5
6	7	8



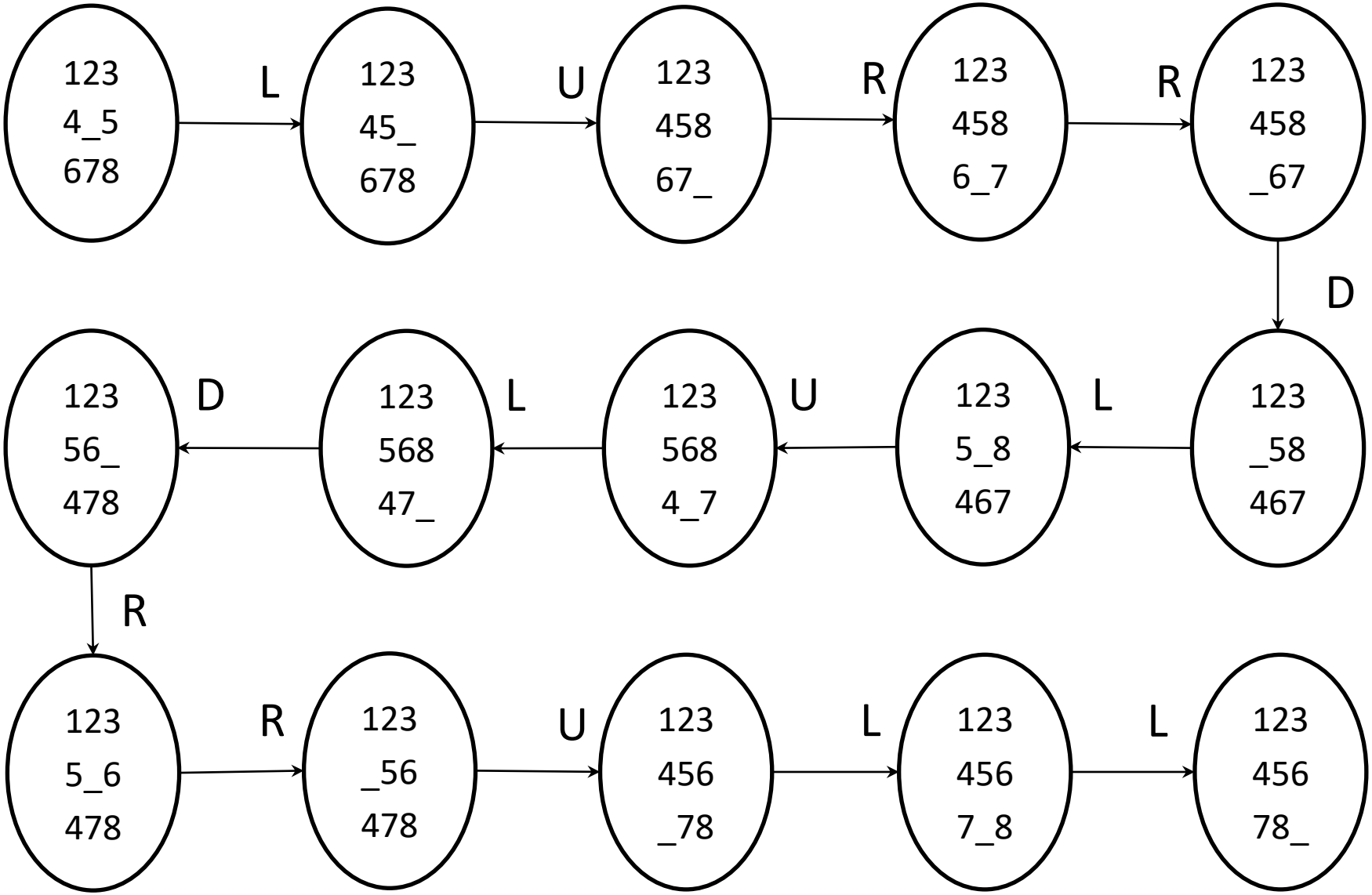
Câu 1. 8-Puzzle (mã giả BFS)

- BFS(graph, start_node):

```
create an empty queue Q
create a set visited to keep track of visited nodes
create a set infrontier to know if nodes were into frontier or not
enqueue start_node into Q
add start_node to visited set

while Q is not empty
        current_node = dequeue from Q
        visit current_node for each successor of current_node:
                if successor is not in visited set:
                        if successor is a goal:
                                return successor
                        if successor not in frontier:
                                add neighbor to visited set
                                enqueue neighbor into Q
```

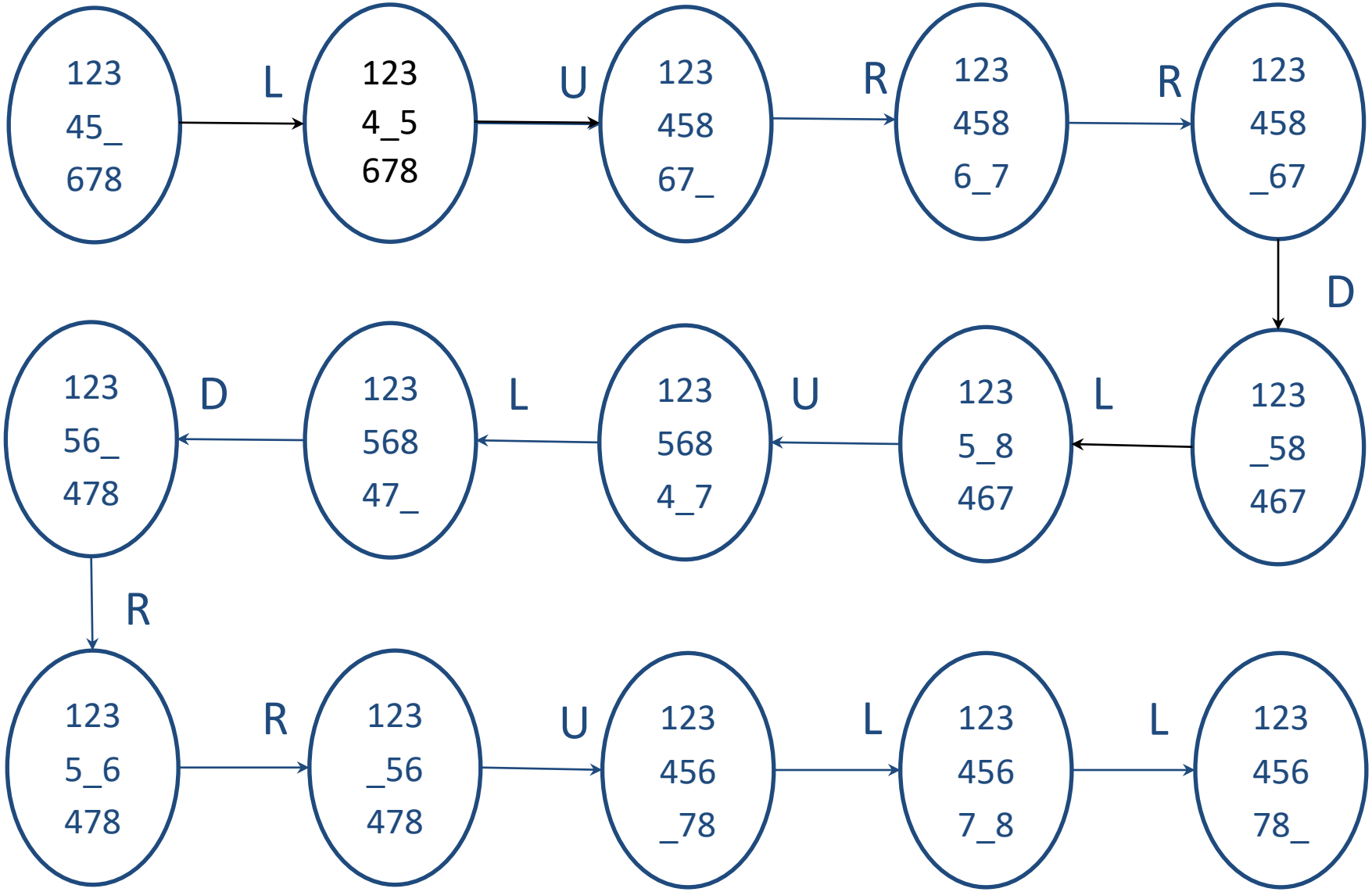
Câu 1. 8-Puzzle (BFS)



Câu 1. 8-Puzzle (mã giả A*)

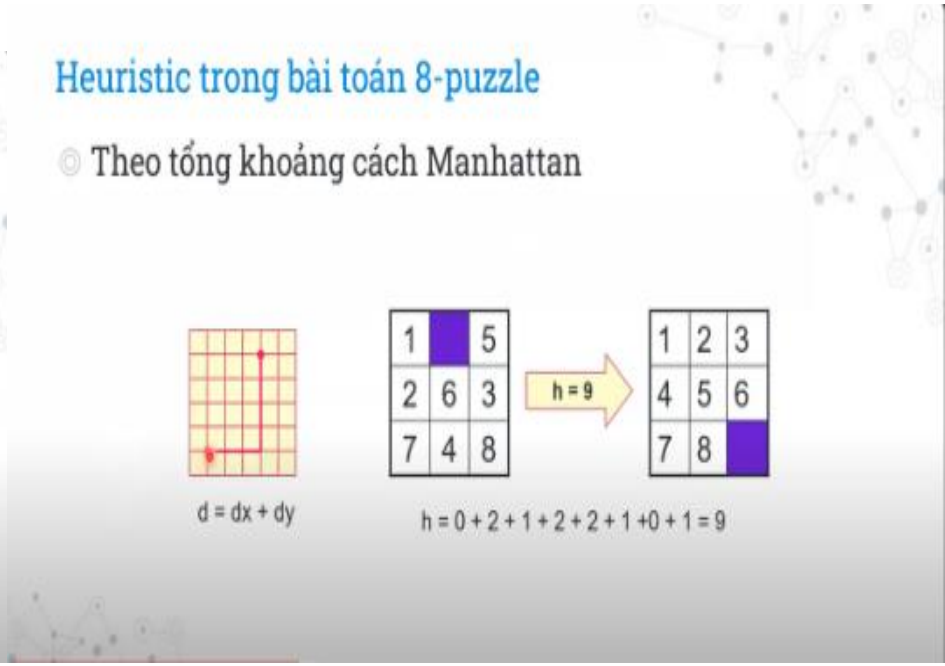
```
function A-STAR-SEARCH(problem) returns a solution or failure
        node ← a node with STATE = problem.INITIAL-STATE
        path ← save the path from start state to goal
        frontier ← a priority queue with node as the only element, ordered by f(n) = g(n) + h(n)
        explored ← an empty set
        parent ← trace the path from start state to goal
        loop do
                if EMPTY?(frontier) then return failure
                node ← POP(frontier)
                if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
                add node.STATE to explored
                for each action in problem.ACTIONS(node.STATE) do
                        child ← CHILD-NODE(problem, node, action)
                        if child.STATE is not in explored or frontier then
                                frontier ← INSERT(child, frontier)
                                update parent for CHILD-NODE
```

Câu 1. 8-Puzzle (A*)



Câu 1. 8-Puzzle (A*)

Hàm heuristic đề xuất:



How to check if an instance of 8 puzzle is solvable?

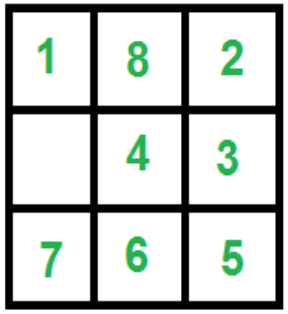
simple rule to check if a 8 puzzle is solvable.

It is not possible to solve an instance of 8 puzzle if number of inversions is odd in the input state.

What is inversion?

Một cặp ô tạo thành một sự đảo ngược nếu các giá trị trên các ô đó được sắp xếp theo thứ tự ngược lại so với vị trí mục tiêu của chúng. Ví dụ, trạng thái bắt đầu của trò chơi 8 ô dưới đây có hai sự đảo ngược, (8, 6) và (8, 7)

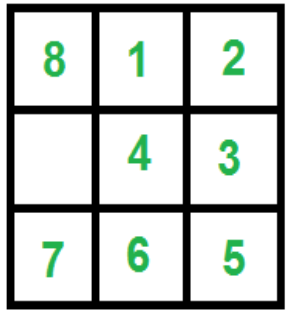
$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & \_ & 5 \\ 8 & 6 & 7 \end{pmatrix}$$



Given State

Solvable

We can reach goal state by sliding tiles using blank space.



Given State

Not Solvable

We can not reach goal state by sliding tiles using blank space.

Câu 2. pacman

Mục tiêu: giúp pacman ăn hết các điểm mồi trong bản đồ và đi qua bốn góc của bản đồ theo thứ tự bất kỳ

Trong đó:

% là vật cản, không thể qua

P là pacman

. là vị trí điểm mồi

Câu 2. pacman

Ý tưởng xây dựng cấu trúc giải bài toán, gồm 4 file chính:

- fringes.py: định nghĩa các tài nguyên giải bài toán như Queue, Stack, Node
- searchAgents.py: định nghĩa các thuật toán tìm kiếm cho bài toán
- problems.py: định nghĩa class problems để giải quyết bài toán
- pacman.py: main

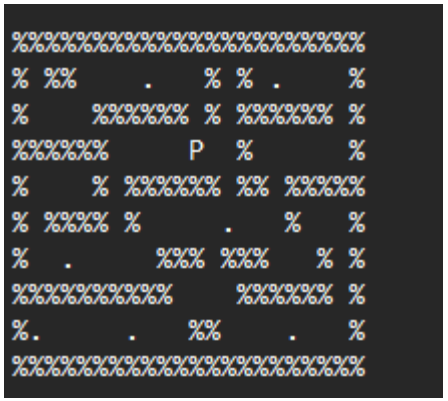
Câu 2. pacman

Class problems trong file problems.py với các thuộc tính *InitialState*, *GoalList* và các hàm để giải quyết bài toán readMaze, PathCost, getSuccessor, …

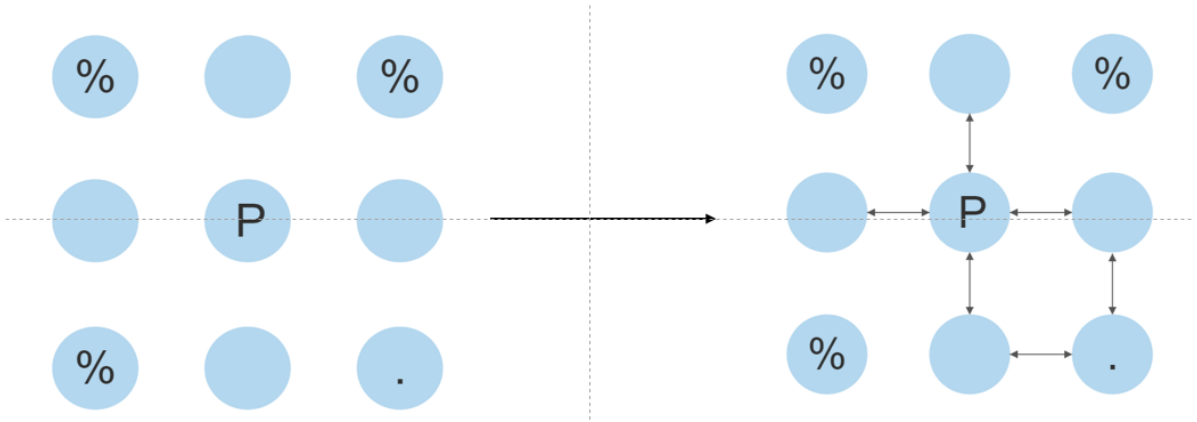
Câu 2. pacman

Tiền xử lý:

- Đọc file mê cung(.lay), có định dạng:



- Tạo node cho mỗi vị trí và link chúng lại:



Câu 2. pacman

Tiền xử lý:

- Tìm Initial State và Goal List từ Node List:

```
#Tìm Initial State:
For i to len( NodeList): // theo hàng
    For j to len( NodeList[i]): // theo cột
        if NodeList[i][j] == 'p':
            InitialState = NodeList[i][j]
                    return None
```

```
#Tìm Goal List:
For i to len( NodeList): // theo hàng
    For j to len( NodeList[i]): // theo cột
        if NodeList[i][j] == '.':
         GoalList.append(NodeList[i][j].Pos )
```

Câu 2. pacman

Cuối cùng ta có một class problems với InitialState và GoalList tìm được từ file maze đầu vào sau đó dùng các thuật toán tìm kiếm(UCS, ASTAR) để tìm đường đi cho pacman.

Câu 2. pacman (mã giả UCS)

```
visited, pqueue = set(), Queue()
Pqueue.add(((problem.InitialState, []),0))

while not pqueue.isEmpty():

    (curPos,actions), cost = pqueue.remove()

     if curPos not in visited:
       visited.add(curPos)
              flag = problem.goalTest(curPos)
              if flag > 0:
                 return actions + ['Stop']
              elif flag < 0:
                 visited, pqueue = set(), Queue()
                 pqueue.add(((curPos, actions,0))
              else:
                 successors = problem.setSuccessors(curPos)
                 for nPos, d in successors:
                          nActions, ncost = actions + [d], cost + 1
                          pqueue.add(((nPos, nActions), ncost))
return actions
```

Câu 2. pacman (mã giả A*)

```
Pq = PriorityQueue () ,Visited = []
Pq.add ((0,problem.InitialState,[])) // (w,Node, actions)
While(Pq is not empty):
    Visited.apped((curNode, actions))
    goalTest =  0 return actions + ['Stop']
    goal Test < 0
       Visited.clear(), Pq.clear(), Pq.add (0, curNode, actions)
    else:
            for d, nNode in successor:
            already_explored = False
            if successor in visited: already_explored = True
            if not already_explored:
                nActions = actions + [d]
                w = len(actions) + Heuristic(nNode)
                Pq.add ((w, nNode, actions))
                Visited.apped (( nNode, len(actions)))
```

Câu 2. pacman (A*)

- Với thuật toán A*, tạo hàm heuristic dựa trên khoảng cách Manhattan như sau:

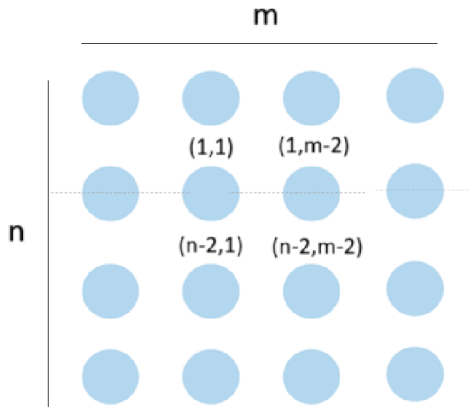
```
Def manhattanDistance(Pos, Goal):
        minHeuristic = 999999
        for g in Goal:
                h = abs( Pos[0] – g[0]) + abs( Pos[1] – g[1])
                if h < minHeuristic = h
        return minHeuristic
```

- Tính admissibility: hàm Manhattan Distance giữa hai điểm trên dưới ô vuông là tổng của sự chênh lệch giữa các thành phần x và y của hai điểm. Do đó nó không bao giờ vượt quá chi phí thực tế nếu chỉ di chuyển theo hướng ngang hoặc dọc.
- Tính consistency: thể hiện ở việc khi chúng ta di chuyển giữa các ô liên tiếp thì giá trị chênh lệch Manhattan Distance giữa các ô đó không bị giảm đi.

Câu 2. pacman

- Xác định 4 góc của ma trận:

Với ma trận có n dòng m cột đều lớn hơn 2(ma trận có vách), ta có thể xác định index 4 góc của ma trận(bỏ thành ma trận) như sau:



So đó kiểm tra nếu chúng là vị trí trống(“ “) thì thêm vào GoalList. Do nếu bằng “%” thì không thể đi qua, còn bằng “P” hoặc “.” chắc chắn sẽ đi qua.

Phân công công việc

Họ và tên	MSSV	Email	Phần phụ trách	Mức độ hoàn thành
Mã Trường Quang	52100925	Matruongquang1@gmail.com	Pacman(70%)	100%
Huỳnh Tấn Đạt	52200152	huynhtandat184@gmail.com	8-Puzzle (BFS)	90%
Lê Như Đạt	52200160	lenhudat181104@gmail.com	8-Puzzle (A*)	100%
Ngô Đức Anh Tuấn	51800951	anhtuan.11020@gmail.com	Pacman(30%), soạn slide	90%

Thuận lợi và khó khăn

Thuận lợi:

Tài liệu phong phú: Có nhiều tài liệu và nguồn thông tin dễ dàng truy cập để nghiên cứu và trích dẫn về các thuật toán khác nhau.

Có sẵn ví dụ: Có nhiều ví dụ cụ thể và minh họa về cách hoạt động của các thuật toán, giúp việc diễn giải trở nên dễ dàng hơn

Khó khăn:

Sự phức tạp: Có những thuật toán phức tạp và khó hiểu, đòi hỏi sự hiểu biết sâu sắc về toán học và lý thuyết.

Tài liệu tham khảo

- geeksforgeeks.org. “How to check if an instance of 8 puzzle solvable”. Available: https://www.geeksforgeeks.org/check-instance-8-puzzle-solvable/, 26/7/2022, [Đã truy cập: 13/3/2024]

- youtube.com. “Solvability of N puzzle problem”. Available: https://www.youtube.com/watch?v=bhmCmbj9VAg&t=326s/, [Đã truy cập: 13/3/2024]

- Thầy Nguyễn Thành An. Available:https://colab.research.google.com/drive/1FZSectdg6fQ-0GW7XNZXcMuuHyyQ6dSU?authuser=1

Thank you