Vũ Quang Minh
18110150

Bài Báo Cáo Thực Hành Nhập Môn Trí Tuệ Nhân Tạo Tuần 1

I. Đọc file dữ liệu

```
input_file = open('/home/minhvu/Documents/A.I/A.I lab1/BFS.txt', 'r')
f = input_file.read().splitlines()

N = int(f[0])#Number of node
goal = f[1].split()
start = 1#int(goal[0])#Start node
end = 18#int(goal[1])#End node
matrix = [f[i].split() for i in range(2, len(f))]
wt_matrix = [[int(char) for char in line]for line in matrix]
input_file.close()
```

Trong đó, hàm f là được dùng để đặt tên của file DFS/BFS của file txt sau khi được input trong chương trình Python

N là dòng đầu tiên trong hàm f trong đâu N=18. Tức là số node là có tất cả V1,..,V18 tổng cộng là 18 vị trí.

goal là dòng thứ 2 trong hàm f trong đây goad=[‘1’,’18’] tức là phải tìm đường đi ngắn nhất từ vị trí đầu(V1) đến vị trí đích(V18)

start là điểm bắt đầu V1

end là điểm đích V18

matrix là cắt từ dòng thứ 2 của hàm f(V1) đến dòng 20 của hàm f (V18) thành 1 ma trận

wt_matrix làm 1 ma trận hoàn chỉnh từ V1→ V18

II. Cài đặt BFS

```
def bfs(wt_matrix, start, end):

    frontier = Queue()
    list_father = []
    explored = []
    initial_state = start -1
    frontier.put(initial_state)
    while(1):
        if frontier.empty():
            return False, "No way Exception"
        current_node = frontier.get()
        explored.append(current_node)

        # Check if node is goal-node
        if current_node == end - 1:
            result = []
```

```
            result.append(end)

            #Find end
            end_index = 0
            for i in range(-1, -len(list_father)-1, -1):
                sons, father = list_father[i]
                if end -1 in sons:
                    end_index = i
                    break
            #Start tracking
            find = father
            result.append(find + 1)
            for i in range(end_index - 1, -len(list_father)-1, -1):
                sons, father = list_father[i]
                if find in sons:
                    result.append(father + 1)
                    find = father

            result = result[::-1]
            result = [str(num) for num in result]
            return True, '->'.join(result)
        #Expand current node
        temp = []
        for i in range(len(wt_matrix[current_node])):
            if wt_matrix[current_node][i] and i not in explored:
                frontier.put(i)
                temp.append(i)
        list_father.append((temp, current_node))
```

Hàm BFS(ma trận, điểm bắt đầu, điểm kết thúc):
1) Khởi tạo 1 hàng đợi frontier chứa trạng thái ban đầu
2) while(1)
    2.1 if Frontier là rỗng then{
        Thông báo tìm kiếm thất bại;
        stop;
    }
    2.2 Loại trạng thái u ở đầu danh sách frontier
    2.3 if u là trọng thái kết thúc then
    {
        Thông báo tìm kiếm thành công;
        stop;
    }
    2.4 Lấy các trạng thái v kề với u và thêm vào cuối danh sách frontier;
        for mỗi trạng thái v kề u do
            father(v))=u;
end

III. Cài đặt DFS

```
def dfs(graph, start, end):
```

```
    frontier = []#stack
    explored = []
    list_father = []
    initial_state = start -1
    frontier.append(initial_state)
    while(1):
        if len(frontier) == 0:
            return False, "No way Exception"
        current_node = frontier.pop()
        explored.append(current_node)

        if current_node == end - 1:
            result = []
            result.append(end)
            #Find end
            end_index = 0
            for i in range(-1, -len(list_father) - 1, - 1):
                sons, father =list_father[i]
                if end - 1 in sons:
                    end_index = i
                    break
            #Start tracking
            find = father
            result.append(find + 1)
            for i in range(end_index - 1, -len(list_father) - 1, -1):
                sons, father = list_father[i]
                if find in sons:
                    result.append(father + 1)
                    find = father
            #Write result
            result = result[::-1]
            result = [str(num) for num in result]
            return True, '->'.join(result)
        temp = []
        for i in range(len(wt_matrix[current_node])):
            if wt_matrix[current_node][i] and i not in explored:
                frontier.append(i)
                temp.append(i)
        list_father.append((temp, current_node))
```

Hàm DFS(Matrận, start, end):
1. Khởi tạo danh sách L chứa trạng thái ban đầu;
2. While (1)
    2.1 if L rỗng then
    {
        Thông báo tìm kiếm thất bại;
        stop;
    }

2.2 Loại trạng thái u ở đầu danh sách L;
2.3 if u là trạng thái kết thúc then
{
Thông báo tìm kiếm thành công;
stop;
}
2.4 Lấy các trạng thái v kề với u và thêm vào đầu danh sách L;
for mỗi trạng thái v kề u do
father(v) = u;
end

IV Cài đặt UCS

```
def ucs(graph, start, end):

    frontier =PriorityQueue()
    explored = []
    history = []
    result = []
    frontier.put((0, 0, start -1))
    while(1):
        if frontier.empty():
            return False, "No way Exception"
        cur_cost, father, current = frontier.get()
        explored.append(current)

        if current == end - 1:
            #Find dst in history
            dst_index = -1
            for i in range(-1, -len(history) -1, -1):
                tcur_cost, tfather, tcurrent = history[i]
                if tcur_cost == cur_cost:
                    end_index = i
                    result.append(history[i])
                    break
            fcur_cost, ffather, fcurrent = history[end_index]
            #Track path through history, return a list of node
            for i in range(dst_index - 1, -len(history) - 1, -1):
                tcur_cost, tfather, tcurrent = history[i]
                if tcurrent == ffather:
                    path_index = i
                    min_cost = tcur_cost
                    for j in range(i - 1, -len(history), -1):
                        t1cur_cost, t1father, t1current = history[j]
                            if t1current == tcurrent and min_cost >
t1cur_cost:
                            min_cost = t1cur_cost
                            path_index = j
                                    fcur_cost,  ffather,  fcurrent  =
history[path_index]
                    result.append(history[path_index])
```

```
            #Write path
            path = str(start)
            for i in range(-1, -len(result)-1, -1):
                _, _, p = result[i]
                path += "->" + str(p + 1)
            return True, path, cur_cost

        #Expand current node
        for i in range(len(wt_matrix[current])):
            if wt_matrix[current][i] and i not in explored:
                node = (wt_matrix[current][i] + cur_cost, current, i)
                frontier.put(node)
                history.append(node)
```

function Tìm_kiếm_UCS(bài_toán, ngăn_chứa) return lời giải hoặc thất bại.
ngăn_chứa ← Tạo_Hàng_Đợi_Rỗng()
ngăn_chứa ← Thêm(TẠO_NÚT(Trạng_Thái_Đầu[bài_toán]), ngăn_chứa)
loop do
    if Là_Rỗng(ngăn_chứa) then return thất bại.
    Nút← Lấy_Chi_phí_Nhỏ_nhất(ngăn chứa)
    if Kiểm_tra_Câu_hỏi_đích[bài_toán] trên Trạng_thái[nút] đúng.
        then return Lời_giải(nút).
Lg← Mở(nút, bài_toán) //lg tập các nút con mới
ngăn_chứa← Thêm_Tất_cả(lg, ngăn_chứa)

V. Kết quả của 3 bài toán

```
BFS :  True 1->4->5->14->11->15->16->17->18
DFS :  True 1->4->5->14->11->15->16->17->18
UCS :  True 1->3->8->9->11->15->16->17->18 with lowest cost :  6910
```