AI – Pacman

Thành viên:

Nguyễn Công Tuấn Anh

Trương Việt Anh

Nguyễn Quốc Cường

Bùi Quang Hà

4 câu đầu trả về list actions

State: tọa độ x,y

Câu 1: Depth first search

Util.py: stack push(self,item), truyền biến item

getSuccessors trả về list 3 biến (successor, action, stepCost), chọn Item = state, action

Code:

Tạo stack lưu các node (state, action)

Tạo mảng array lưu visited state

Đẩy state, action rỗng đầu vào stack

Khi stack không rỗng

Loại trạng thái đầu danh sách lấy ra state, action

Nếu state đã visit thì continue đến state tiếp theo trong stack

Đánh dấu state hiện tại visited

Kiểm tra nếu state hiện tại là goal không, nếu có trả về list các action

Duyệt các successor, nếu chưa visit thì push vào stack. actions(successor) = actions(predecessor) + action tới nó.

Câu 2: Breadth first search

Tương tự như câu 1, thay stack thành queue, cú pháp dùng queue giống stack

Câu 3: Uniform cost search

Tương tự như câu 1, nhưng dùng priority queue ưu tiên cost thấp hơn

Util.py: Priority queue push(self, item, priority)

Chọn (Item = state, action), (priority = cost)

Code:

Tạo Priority queue lưu các node ((state, action), cost)

Tạo mảng array lưu visited state

Đẩy state, action rỗng đầu vào stack (cost vị trí đầu là 0)

Khi stack không rỗng

Loại trạng thái đầu danh sách lấy ra state, action

Nếu state đã visit thì continue đến state tiếp theo trong stack

Đánh dấu state hiện tại visited

Kiểm tra nếu state hiện tại là goal không, nếu có trả về list các action

Duyệt các successor, nếu chưa visit thì push vào priorityqueue. actions(successor) = actions(predecessor) + action tới nó. cost = cost(successor) + cost các action trước đó

Câu 4: A\* search

Hàm heuristic trong searchagents.py có dạng nameHeuristic(position, problem, info={}) ước lượng cost từ position đến đích

Tương tự câu 3, nhưng cost + thêm ước lượng tới đích heuristic(successor)

cost = cost(successor) + cost các action trước đó + heuristic cost của successor tới đích

Câu 5:

Function getsuccessors trả về list successors

Với mỗi action trong 4 hướng

Chuyển hướng hiện tại thành toạ độ vector dx,dy với hàm directionToVector trong game.py

Toạ độ tiếp theo với hướng hiện tại nextx, nexty = x+dx,y+dy

Biến next\_node là toạ độ tiếp theo

Nếu bước tiếp theo không va tường

Tạo list successorVisitedCorner copy các visited corner hiện tại

Nếu node tiếp theo là góc và không trong successorVisitedCorner thì thêm node đó vào

Truyền successorVisitedCorner vào state để biết đã đi qua góc nào nên successor state = nextnode, successorVisitedCorner

Thêm successor vào list successors

Trả về list successors

State = position, visited corner

Function init không cần sửa

Function getstart state trả về vị trí start, list rỗng lưu successorVisitedCorner

Fuction isgoalstate

Lấy vị trí là state[0], visited corner là state[1]

Kiểm tra nếu đủ 4 corner đã visit trả về true

Câu 6:

State = position, visited corner

Từ vị trí hiện tại lấy manhattandistance tới unvisit corner, chọn manhattandistance gần nhất đi tới rồi lặp lại đến hết unvisit corner

Code:

Tổng ước lượng = 0

Unvisited corner = corner not in visited corner

While còn unvisited corner

Tính manhattan distance tới tất cả đỉnh chưa visit, chọn distance, đỉnh có hdistance nhỏ nhất

Tổng ước lượng += hdistance nhỏ nhất

Vị trí hiện tại là corner đã chọn

Loại corner đã chọn ra unvisit corner

Trả về tổng ước lượng