astar

November 9, 2024

[2]:

print("Name:Vismay Pawar N","USN:1BM22CS331",sep="**\n**")

**import heapq**

**class PuzzleState**:

**def** init (self, board, g=0): self.board = board

self.g = g

self.zero\_pos = board.index(0)

**def** h(self):

**return** sum(1 **for** i **in** range(9) **if** self.board[i] != 0 **and** self.board[i] !

↪= i + 1) *#misplaced tiles*

**def** f(self):

**return** self.g + self.h()

**def** get\_neighbors(self): neighbors = []

x, y = divmod(self.zero\_pos, 3)

directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]

**for** dx, dy **in** directions:

new\_x, new\_y = x + dx, y + dy

**if** 0 <= new\_x < 3 **and** 0 <= new\_y < 3: new\_zero\_pos = new\_x \* 3 + new\_y new\_board = self.board[:]

new\_board[self.zero\_pos], new\_board[new\_zero\_pos] =␣

↪new\_board[new\_zero\_pos], new\_board[self.zero\_pos] neighbors.append(PuzzleState(new\_board, self.g + 1))

**return** neighbors

**def** a\_star(initial\_state, goal\_state): open\_set = []

heapq.heappush(open\_set, (initial\_state.f(), 0, initial\_state)) came\_from = {}

g\_score = {tuple(initial\_state.board): 0}

**while** open\_set:

current\_f, \_, current = heapq.heappop(open\_set)

**if** current.board == goal\_state:

**return** reconstruct\_path(came\_from, current)

**for** neighbor **in** current.get\_neighbors(): neighbor\_tuple = tuple(neighbor.board)

tentative\_g\_score = g\_score[tuple(current.board)] + 1

**if** neighbor\_tuple **not in** g\_score **or** tentative\_g\_score <␣

↪g\_score[neighbor\_tuple]:

came\_from[neighbor\_tuple] = current g\_score[neighbor\_tuple] = tentative\_g\_score

heapq.heappush(open\_set, (neighbor.f(), neighbor.g, neighbor)) ␣

↪*# Use neighbor.g as the tie-breaker*

# return None

**def** reconstruct\_path(came\_from, current): path = []

**while** current **is not None**: path.append(current.board)

current = came\_from.get(tuple(current.board), **None**) **return** path[::-1]

initial\_state = PuzzleState([1, 2, 3, 4, 5, 6, 0, 7, 8])

goal\_state = [1, 2, 3, 4, 5, 6, 7, 8, 0]

solution = a\_star(initial\_state, goal\_state)

**if** solution:

**for** step **in** solution: print(step)

[3]:

**else**:

print("No solution found")

Name:Vismay Pawar N

USN:1BM22CS331

[1, 2, 3, 4, 5, 6, 0, 7, 8]

[1, 2, 3, 4, 5, 6, 7, 0, 8]

[1, 2, 3, 4, 5, 6, 7, 8, 0]

print("Name:Vismay Pawar N","USN:1BM22CS331",sep="**\n**")

**import heapq**

**class PuzzleState**:

**def** init (self, board): self.board = board self.zero\_pos = board.index(0)

**def** h(self):

distance = 0

**for** i **in** range(9):

**if** self.board[i] != 0:

target\_x, target\_y = divmod(self.board[i] - 1, 3) current\_x, current\_y = divmod(i, 3)

distance += abs(target\_x - current\_x) + abs(target\_y -␣

↪current\_y)

**return** distance

**def** f(self):

**return** self.h() *# Just the heuristic value (Manhattan distance)*

**def** get\_neighbors(self): neighbors = []

x, y = divmod(self.zero\_pos, 3)

directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]

**for** dx, dy **in** directions:

new\_x, new\_y = x + dx, y + dy

**if** 0 <= new\_x < 3 **and** 0 <= new\_y < 3: new\_zero\_pos = new\_x \* 3 + new\_y new\_board = self.board[:]

new\_board[self.zero\_pos], new\_board[new\_zero\_pos] =␣

↪new\_board[new\_zero\_pos], new\_board[self.zero\_pos] neighbors.append(PuzzleState(new\_board))

**return** neighbors

**def** a\_star(initial\_state, goal\_state): open\_set = []

heapq.heappush(open\_set, (initial\_state.f(), id(initial\_state),␣

↪initial\_state))

came\_from = {}

g\_score = {tuple(initial\_state.board): 0}

**while** open\_set:

current\_f, \_, current = heapq.heappop(open\_set)

**if** current.board == goal\_state:

**return** reconstruct\_path(came\_from, current)

**for** neighbor **in** current.get\_neighbors(): neighbor\_tuple = tuple(neighbor.board)

tentative\_g\_score = g\_score[tuple(current.board)] + 1 *# All edges*␣

↪*have a cost of 1*

**if** neighbor\_tuple **not in** g\_score **or** tentative\_g\_score <␣

↪g\_score[neighbor\_tuple]:

came\_from[neighbor\_tuple] = current g\_score[neighbor\_tuple] = tentative\_g\_score

heapq.heappush(open\_set, (tentative\_g\_score + neighbor.h(),␣

↪id(neighbor), neighbor))

**return None**

**def** reconstruct\_path(came\_from, current): path = []

**while** current **is not None**: path.append(current.board)

current = came\_from.get(tuple(current.board), **None**) **return** path[::-1]

initial\_state = PuzzleState([1, 2, 3, 4, 5, 6, 0, 7, 8])

goal\_state = [1, 2, 3, 4, 5, 6, 7, 8, 0]

solution = a\_star(initial\_state, goal\_state)

**if** solution:

**for** step **in** solution: print(step)

**else**:

print("No solution found")

Name:Vismay Pawar N

USN:1BM22CS331

[1, 2, 3, 4, 5, 6, 0, 7, 8]

[1, 2, 3, 4, 5, 6, 7, 0, 8]

[1, 2, 3, 4, 5, 6, 7, 8, 0]