AI-2

import heapq

class Graph:

def \_\_init\_\_(self):

self.graph = {}

def add\_edge(self, u, v, weight):

if u in self.graph:

self.graph[u].append((v, weight))

else:

self.graph[u] = [(v, weight)]

def astar(self, start, goal, heuristic):

open\_set = [(0, start)] # (f-score, node)

came\_from = {}

g\_score = {node: float('inf') for node in self.graph}

g\_score[start] = 0

f\_score = {node: float('inf') for node in self.graph}

f\_score[start] = heuristic(start, goal)

while open\_set:

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

if current == goal:

path = []

while current in came\_from:

path.append(current)

current = came\_from[current]

path.append(start)

return path[::-1]

for neighbor, weight in self.graph[current]:

tentative\_g\_score = g\_score[current] + weight

if tentative\_g\_score < g\_score[neighbor]:

came\_from[neighbor] = current

g\_score[neighbor] = tentative\_g\_score

f\_score[neighbor] = tentative\_g\_score + heuristic(neighbor, goal)

heapq.heappush(open\_set, (f\_score[neighbor], neighbor))

return None # No path found

# Example usage:

def heuristic(state, goal):

return abs(state[0] - goal[0]) + abs(state[1] - goal[1]) # Manhattan distance

game\_graph = Graph()

game\_graph.add\_edge((0, 0), (0, 1), 1)

game\_graph.add\_edge((0, 1), (1, 1), 1)

game\_graph.add\_edge((1, 1), (1, 2), 1)

game\_graph.add\_edge((1, 2), (2, 2), 1)

game\_graph.add\_edge((2, 2), (2, 3), 1)

game\_graph.add\_edge((2, 3), (3, 3), 1)

start = (0, 0)

goal = (3, 3)

path = game\_graph.astar(start, goal, heuristic)

if path:

print("Path found:", path)

else:

print("No path found")

O/P

Path found: [(0, 0), (0, 1), (1, 1), (1, 2), (2, 2), (2, 3), (3, 3)]