Exercise 10

from queue import PriorityQueue

def heuristic(a, b):

return abs(a[0] - b[0]) + abs(a[1] - b[1])

def astar(start, goal, graph):

open\_set = PriorityQueue()

open\_set.put((0, start))

came\_from = {}

g\_score = {start: 0}

while not open\_set.empty():

\_, current = open\_set.get()

if current == goal:

path = []

while current in came\_from:

path.append(current)

current = came\_from[current]

path.append(start)

return path[::-1]

for neighbor in graph.get(current, []):

tentative\_g = g\_score[current] + 1

if neighbor not in g\_score or tentative\_g < g\_score[neighbor]:

came\_from[neighbor] = current

g\_score[neighbor] = tentative\_g

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

open\_set.put((f\_score, neighbor))

return None

# Example graph as adjacency list

graph = {

(0, 0): [(0, 1), (1, 0)],

(0, 1): [(0, 0), (1, 1)],

(1, 0): [(0, 0), (1, 1)],

(1, 1): [(1, 0), (0, 1), (2, 1)],

(2, 1): [(1, 1)]

}

start = (0, 0)

goal = (2, 1)

path = astar(start, goal, graph)

print("Path:", path)

