Exercise 7

0/10 points (graded)

Consider once again our permutations of students in a line. Recall the nodes in the graph represent permutations, and that the edges represent swaps of adjacent students. We want to design a weighted graph, weighting edges higher for moves that are harder to make. Which of these could be easily implemented by simply assigning weights to the edges already in the graph?

A) A large student who is difficult to move around in line. correct

B) A sticky spot on the floor which is difficult to move onto and off of. correct

C) A student who resists movement to the back of the line, but accepts movement toward the front.

**Explanation:**

Answer: A, B

A) is easily implemented by weighting heavily all edges that involve moving the particular student.

B) is implemented by increasing the weight of all edges that involve a swap with that spot in line.

C) cannot be done without weighting two directions of an edge differently. In this case, putting one student behind another is not the same as putting the first student in front of the other, but in our undirected graph, it is.

Write a WeightedEdge class that extends Edge. Its constructor requires a weight parameter, as well as the parameters from Edge. You should additionally include a getWeight method. The string value of a WeightedEdge from node A to B with a weight of 3 should be "A->B (3)".

class WeightedEdge(Edge):

def \_\_init\_\_(self, src, dest, weight):

# Your code here

pass

def getWeight(self):

# Your code here

pass

def \_\_str\_\_(self):

# Your code here

pass

class WeightedEdge(Edge):

def \_\_init\_\_(self, src, dest, weight):

Edge.\_\_init\_\_(self, src, dest)

self.weight = weight

def getWeight(self):

return self.weight

def \_\_str\_\_(self):

return Edge.\_\_str\_\_(self) + " (" + str(self.weight) + ")"