5. Write a program that, given a graph with costs and two vertices, finds a lowest cost walk between the given vertices, or prints a message if there are negative cost cycles accessible from the starting vertex. The program will use the Ford's algorithm.

def fords\_algorithm(self, start, end):

dist = {}

prev = {}

maximum\_value = float("inf")

for i in self.\_\_vertices:

dist[i] = maximum\_value

dist[start] = 0

for i in range(len(self.\_\_vertices)):

changed = False

for (x, y) in self.\_\_cost:

if dist[y] > dist[x] + self.get\_edge\_cost(x, y):

dist[y] = dist[x] + self.get\_edge\_cost(x, y)

prev[y] = x

changed = True

if not changed:

break

else:

return None, None

if end not in prev:

return None, None

path = [end]

while path[-1] != start:

path.append(prev[path[-1]])

path.reverse()

return path, dist[end]

Input Parameters:

self: the instance of the class that the function belongs to

start: the starting vertex for the algorithm

end: the ending vertex for the algorithm

Output:

A tuple containing the path (list) from start to end and the distance (float) between them.

If the end vertex is not reachable from the start vertex, return (None, None)

If there is a negative cycle in the graph, return (None, None)

Functionality:

This function uses Ford's algorithm to find the shortest path between a starting vertex and an ending vertex in a graph.

It initializes a dictionary of distances and a dictionary of previous vertices, and sets the initial distance of the starting vertex to 0.

It then iteratively updates the distances until no further improvement is possible.

If the end vertex is reachable, it constructs the path and returns it along with the distance.

If there is a negative cycle in the graph, the function returns (None, None).

• the lowest cost walk and the length from 1 to 100 in graph1k:

The walk has cost: 141

The path is: [1, 5, 487, 175, 714, 799, 222, 561, 100]

• the lowest cost walk and the length from 1 to 100 in graph10k:

The walk has cost: 344

The path is: [1, 7317, 460, 6010, 5295, 4560, 5513, 8467, 3517, 99, 9159, 6840, 5177, 7133, 288, 100]

• the lowest cost walk and the length from 1 to 100 in graph100k:

The walk has cost: 304

The path is: [1, 99842, 59480, 5210, 19068, 66428, 33692, 97073, 23675, 73057, 100]

• the lowest cost walk and the length from 100 to 1 in graph1k:

The walk has cost: 196

The path is: [100, 259, 229, 641, 538, 854, 1]

• the lowest cost walk and the length from 100 to 1 in graph10k:

The walk has cost: 238

The path is: [100, 4442, 3980, 1974, 407, 4489, 5162, 2008, 3631, 2305, 8336, 1]

• the lowest cost walk and the length from 100 to 1 in graph100k:

The walk has cost: 361

The path is: [100, 85636, 77467, 52472, 38155, 40962, 34650, 29215, 35260, 1]