def minimum\_cost\_walk\_Floyd\_Warshall(graph: TripleDictGraph, starting\_vertex: int, ending\_vertex: int):  
 nodes = list(graph.dictionary\_in.keys())  
 nodes.sort()  
 dist\_matrix = [[float('inf') for \_ in range(len(nodes))] for \_ in range(len(nodes))]  
 previous\_matrix = [[0 for \_ in range(len(nodes))] for \_ in range(len(nodes))]  
 for i in range(len(nodes)):  
 dist\_matrix[i][i] = 0  
  
 for i in range(len(nodes)):  
 vertex\_i = nodes[i]  
 for j in range(len(nodes)):  
 vertex\_j = nodes[j]  
 if graph.is\_edge(vertex\_i, vertex\_j) is True:  
 dist\_matrix[i][j] = graph.dictionary\_cost[(vertex\_i, vertex\_j)]  
 previous\_matrix[i][j] = nodes[i]  
 list\_of\_intermediate\_matrices = []

for k in range(len(nodes)):  
 list\_of\_intermediate\_matrices.append(copy.deepcopy(dist\_matrix))  
 for i in range(len(nodes)):  
 for j in range(len(nodes)):  
 if dist\_matrix[i][j] > dist\_matrix[i][k] + dist\_matrix[k][j]:  
 dist\_matrix[i][j] = dist\_matrix[i][k] + dist\_matrix[k][j]  
 previous\_matrix[i][j] = previous\_matrix[k][j]  
 if i == j and dist\_matrix[i][j] < 0:  
 raise Exception("There is a negative cost cycle!")

if dist\_matrix[nodes.index(starting\_vertex)][nodes.index(ending\_vertex)] == float('inf'):  
 raise Exception("There is no path between these vertices!")  
 path = [ending\_vertex]  
 end = ending\_vertex  
  
 while starting\_vertex != ending\_vertex:  
 ending\_vertex = previous\_matrix[nodes.index(starting\_vertex)][nodes.index(ending\_vertex)]  
 path.append(ending\_vertex)  
  
 return dist\_matrix[nodes.index(starting\_vertex)][nodes.index(end)], path[::-1], list\_of\_intermediate\_matrices