```
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
```