Hamiltonian Cycle

def hamil\_recursive(self, path, pos):  
  
 # checking if we have a solution  
 if pos == self.get\_number\_of\_vertices():  
 if self.edge\_exists(path[pos - 1], path[0]):  
 return True  
 else:  
 return False  
  
 # going through all the vertices that have an edge with the previous vertex  
 # seeing which vertex can be put on the current pos and going forward with it  
 for ver in self.parse\_outbound\_edges(path[pos - 1]):  
 if self.is\_okay(path, pos, ver):  
 path[pos] = ver  
  
 if self.hamil\_recursive(path, pos + 1):  
 return True  
  
 path[pos] = -1  
 return False  
  
def is\_okay(self, path, pos, v):  
 # checks if the vertex v can be put on the position pos in the given path  
 # check that there s a path between the last vertex and the given vertex  
 if not self.edge\_exists(path[pos - 1], v):  
 return False  
  
 # check that the vertex doesn't already exist in the path  
 for ver in path:  
 if v == ver:  
 return False  
  
 return True  
  
def check\_hamiltonian(self):  
 path = [-1] \* (self.get\_number\_of\_vertices() + 1)  
  
 path[0] = list(self.\_dictEdges.keys())[0]  
 path[self.get\_number\_of\_vertices()] = list(self.\_dictEdges.keys())[0]  
  
 if not self.hamil\_recursive(path, 1):  
 raise GraphException("There is no hamiltonian cycle in the graph.")  
  
 return path