**Issafe**

This method checks if it's safe to add vertex **v** to the current Hamiltonian path (**path**) at position **pos**. It returns **True** if adding **v** at **pos** is safe, otherwise **False**. It checks two conditions:

* Whether there is an edge between the last vertex in the path (**path[pos-1]**) and the vertex **v**.
* Whether **v** is already present in the path.

**Hamcycle util**

This method is a recursive utility function used to find a Hamiltonian Cycle. It checks if all vertices have been visited (**pos == self.V**). If so, it checks if there's an edge from the last vertex to the starting vertex, forming a cycle.

class Graph():

def \_\_init\_\_(self, vertices):

self.graph = [[0 for column in range(vertices)]

for row in range(vertices)]

self.V = vertices

def isSafe(self, v, pos, path):

if self.graph[ path[pos-1] ][v] == 0:

return False

for vertex in path:

if vertex == v:

return False

return True

def hamCycleUtil(self, path, pos):

if pos == self.V:

if self.graph[ path[pos-1] ][ path[0] ] == 1:

return True

else:

return False

for v in range(1,self.V):

if self.isSafe(v, pos, path) == True:

path[pos] = v

if self.hamCycleUtil(path, pos+1) == True:

return True

path[pos] = -1

return False

def hamCycle(self):

path = [-1] \* self.V

path[0] = 0

if self.hamCycleUtil(path,1) == False:

print ("Solution does not exist\n")

return False

self.printSolution(path)

return True

def printSolution(self, path):

print ("Solution Exists: Following",

"is one Hamiltonian Cycle")

for vertex in path:

print (vertex, end = " ")

print (path[0], "\n")

g1 = Graph(5)

g1.graph = [ [0, 1, 0, 1, 0], [1, 0, 1, 1, 1],

[0, 1, 0, 0, 1,],[1, 1, 0, 0, 1],

[0, 1, 1, 1, 0], ]

g1.hamCycle();