**Distributed Systems**

Lab 5 – Chord Protocol

S Abhishek

AM.EN.U4CSE19147

import matplotlib.pyplot as plt  
import networkx as nx  
  
def succ(key, arr, identifier):  
 arr = sorted(arr)  
 for i in range(len(arr)):  
 if arr[i] >= key%pow(2,identifier):  
 return arr[i]  
 return arr[0]  
  
  
class chord:  
 def \_\_init\_\_(self, identifier):  
 self.identifier = identifier  
 self.G = nx.DiGraph()  
 self.nodes = dict()  
   
 def add\_node(self, node\_id):  
 self.nodes[node\_id] = [[]]  
 self.nodes[node\_id] = [[i,succ(node\_id+pow(2,i-1), self.nodes.keys(), self.identifier)] for i in range(1,self.identifier+1)]  
 for j in self.nodes:  
 self.nodes[j] = [[i,succ(j+pow(2,i-1), self.nodes.keys(), self.identifier)] for i in range(1,self.identifier+1)]  
 print("Nodes in the ring :",self.nodes.keys())  
  
   
 def remove\_node(self, node\_id):  
 self.nodes.pop(node\_id)  
 for j in self.nodes:  
 self.nodes[j] = [[i,succ(j+pow(2,i-1), self.nodes.keys(), self.identifier)] for i in range(1,self.identifier+1)]  
 print("Nodes in the ring :",self.nodes.keys())  
   
 def finger\_table(self, node\_id):  
 self.G.add\_nodes\_from(self.nodes.keys())  
 return self.nodes[node\_id]  
   
 def look\_up(self, key, node):  
  
 print("Route : ", end = "")  
 k=0  
 edge = []  
 print(node, "->", end=" ")  
 edge.append(node)  
 if key in [i[1] for i in self.nodes[node]]:  
 print(key)  
 edge.append(key)  
 k=1  
 while True:  
 if k==1:  
 break  
 for i in range(0,self.identifier-1):  
 if node < key and key < self.nodes[node][0][1]:  
 node = self.nodes[node][0][1]  
 print(node)  
 edge.append(node)  
 k=1  
 break  
 if self.nodes[node][i][1] <= key and self.nodes[node][i+1][1] > key:  
 node = self.nodes[node][i][1]  
 print(node, "->",end=" ")  
 edge.append(node)  
 break  
 if (self.nodes[node][0][1] - node < 0) and node<key:  
 node = self.nodes[node][i+1][1]  
 print(node)  
 edge.append(node)  
 k=1  
 break  
 if i == (self.identifier-2):  
 node = self.nodes[node][i+1][1]  
 print(node, "->",end=" ")  
 edge.append(node)  
   
 color\_map = []  
 for n in range(0,pow(2,self.identifier)):  
 if n in edge:  
 color\_map.append('green')  
 elif n in self.nodes.keys():  
 color\_map.append('cyan')  
 else:   
 color\_map.append('orange')   
 edges = [[edge[i],edge[i+1]] for i in range(0,len(edge)-1)]  
 self.G.remove\_nodes\_from(list(self.G.nodes))  
 self.G.add\_nodes\_from(range(0,pow(2,self.identifier)))  
 self.G.add\_edges\_from(edges)  
 print()  
 plt.figure(figsize=(10,7))  
 plt.title('S Abhishek - AM.EN.U4CSE19147')  
 nx.draw\_circular(self.G, with\_labels=True, font\_weight='bold', node\_size=500, node\_color=color\_map)

c = chord(6)

c.add\_node(0)  
c.add\_node(11)  
c.add\_node(18)  
c.add\_node(21)  
c.add\_node(29)  
c.add\_node(40)  
c.add\_node(46)  
c.add\_node(51)  
c.add\_node(58)

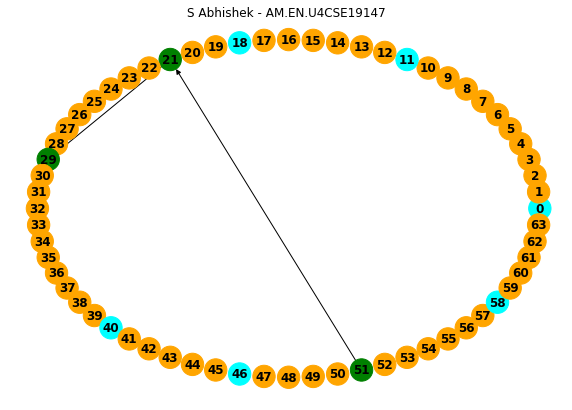
Nodes in the ring : dict\_keys([0])  
Nodes in the ring : dict\_keys([0, 11])  
Nodes in the ring : dict\_keys([0, 11, 18])  
Nodes in the ring : dict\_keys([0, 11, 18, 21])  
Nodes in the ring : dict\_keys([0, 11, 18, 21, 29])  
Nodes in the ring : dict\_keys([0, 11, 18, 21, 29, 40])  
Nodes in the ring : dict\_keys([0, 11, 18, 21, 29, 40, 46])  
Nodes in the ring : dict\_keys([0, 11, 18, 21, 29, 40, 46, 51])  
Nodes in the ring : dict\_keys([0, 11, 18, 21, 29, 40, 46, 51, 58])

c.nodes

{0: [[1, 11], [2, 11], [3, 11], [4, 11], [5, 18], [6, 40]],  
 11: [[1, 18], [2, 18], [3, 18], [4, 21], [5, 29], [6, 46]],  
 18: [[1, 21], [2, 21], [3, 29], [4, 29], [5, 40], [6, 51]],  
 21: [[1, 29], [2, 29], [3, 29], [4, 29], [5, 40], [6, 58]],  
 29: [[1, 40], [2, 40], [3, 40], [4, 40], [5, 46], [6, 0]],  
 40: [[1, 46], [2, 46], [3, 46], [4, 51], [5, 58], [6, 11]],  
 46: [[1, 51], [2, 51], [3, 51], [4, 58], [5, 0], [6, 18]],  
 51: [[1, 58], [2, 58], [3, 58], [4, 0], [5, 11], [6, 21]],  
 58: [[1, 0], [2, 0], [3, 0], [4, 11], [5, 11], [6, 29]]}

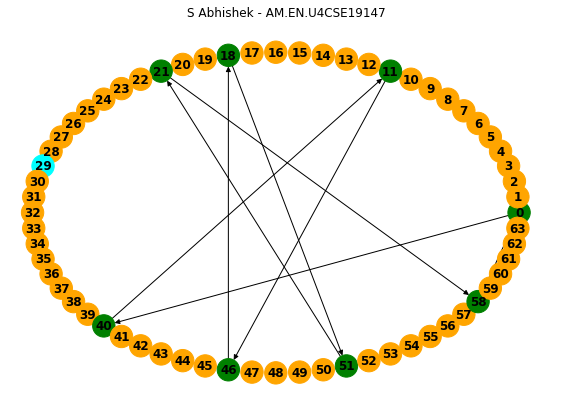
c.look\_up(27, 51)

Route : 51 -> 21 -> 29



c.look\_up(63, 0)

Route : 0 -> 40 -> 11 -> 46 -> 18 -> 51 -> 21 -> 58 -> 0

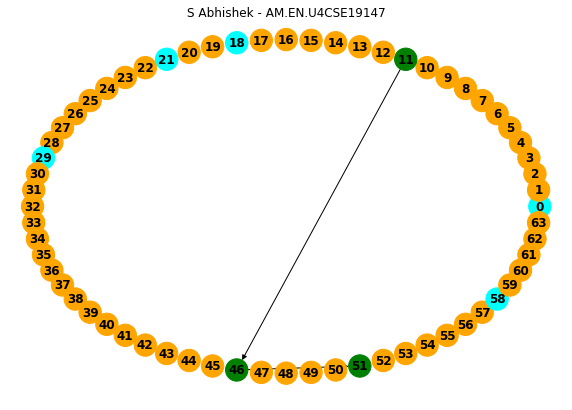


c.remove\_node(40)

Nodes in the ring : dict\_keys([0, 11, 18, 21, 29, 46, 51, 58])

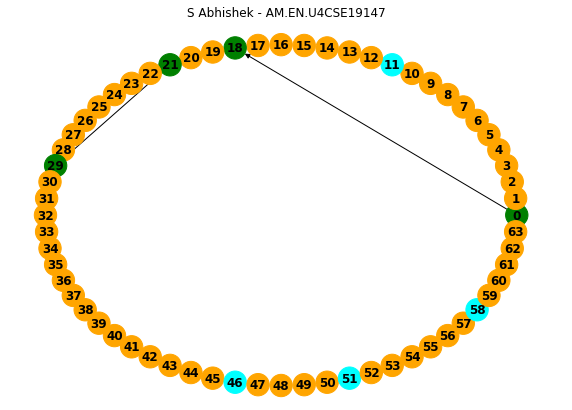
c.look\_up(50, 11)

Route : 11 -> 46 -> 51



c.look\_up(22, 0)

Route : 0 -> 18 -> 21 -> 29

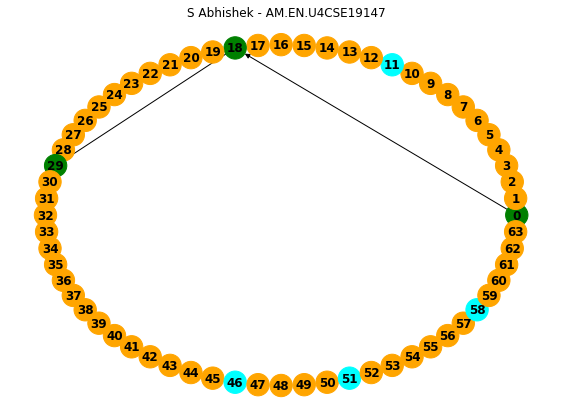


c.remove\_node(21)

Nodes in the ring : dict\_keys([0, 11, 18, 29, 46, 51, 58])

c.look\_up(22, 0)

Route : 0 -> 18 -> 29



Thankyou!!