Chord Protocol

Done By:
N Sai Pavan Krishna
AM.EN.U4CSE19347

Question: Implement a chord protocol simulator. You can choose any identifier space > 5-bits. The simulator should start with an initial chord ring of a random set of nodes. The simulator displays the ring and asks input for the functionality you want to perform in the ring,namely the following:
1.Look up a key- You give the key and the initialnode and the simulator gives you the node responsible. The response should include the path that has been followed.
2.Add a node to the ring- You provide the node idfor the node which wants to join the ring and the simulator displays the new ring
3.Remove a node from the ring- You provide the nodeid for the node which wants to leave the ring and the simulator displays the new ring
4.Display the finger table for a given node.
5. Exit the ring- This option ends your simulation. Till then all the above options are available.
Note: The finger tables need to be updated each timea modification is done to the chordring.

In [43]:
```
import random
from bisect import bisect_left,insort,bisect_right

class Node:
    def __init__(self,num,succ=None,pre=None,bitn=6):
        self.num=num
        self.ft=[None]*bitn
        self.succ=succ
        self.pre=pre

class Chord:
    def __init__(self,bit):
        self.bit=bit
        initial_node_values=sorted(random.sample(range(1<<bit), 1<<(bit-3)))
        initial_nodes=[Node(initial_node_values[i],bitn=bit) for i in range(len(initial_node_values))]
        self.nodes=dict()
        for i in range(len(initial_node_values)):
            self.nodes[initial_nodes[i].num]=initial_nodes[i]
            initial_nodes[i].succ=initial_nodes[(i+1)%len(initial_nodes)]
            initial_nodes[i].pre=initial_nodes[(i-1)%len(initial_nodes)]
        self.update_ft()
        print("The following nodes have been initialised randomly: ",*initial_node_values)

    def update_ft(self):
        node_values=list(self.nodes.keys())
        nodes=[self.nodes[i] for i in node_values]
        for i in range(len(node_values)):
            nodes[i].succ=nodes[(i+1)%len(nodes)]
            nodes[i].pre=nodes[(i-1)%len(nodes)]
        for i in node_values:
            #print(i)
            for j in range(self.bit):
                #print(i,j,len(nodes),bisect_left(node_values,(i+(1<<j))%(1<<self.bit))%len(nodes),len(self.node
                self.nodes[i].ft[j]=nodes[bisect_left(node_values,(i+(1<<j))%(1<<self.bit))%len(nodes)]
                #print("ft["+str(j)+"]="+str(self.nodes[i].ft[j].num))
        #display the ring here

    def add_node(self,i):
        node_values=list(self.nodes.keys())
        insort(node_values,i)
        self.nodes[i]=Node(i,bitn=self.bit)
        self.nodes={i:self.nodes[i] for i in node_values}
        self.update_ft()
        print("Node Added Successfully!")

    def lookup(self,start,key):
        q=start
        t=[]
        while 1:
            t.append(q)
            if q==key:
                break
            p=sorted([self.nodes[q].ft[j].num for j in range(self.bit)])
            q=p[(bisect_right(p,key)-1)%len(p)]
        print("Path Followed:",*t)

    def delete_node(self,i):
        del self.nodes[i]
        self.update_ft()
        print("Node Deleted Successfully!")

    def display_ft(self,i):
        print("Finger Table of Node -",i)
        for j in range(self.bit):
            print("ft["+str(j)+"]="+str(self.nodes[i].ft[j].num))

    def exit_ring(self):
        print("Exiting...")
```

In [44]:
```
c=Chord(10)
c.add_node(8)
c.add_node(46)
c.add_node(172)
c.add_node(345)
c.add_node(473)
c.add_node(764)
c.add_node(995)
c.lookup(8,995)
c.lookup(172,46)
```

```
The following nodes have been initialised randomly:  2 4 9 13 25 27 29 33 34 40 51 52 54 59 67 71 73 86 87 99
105 112 115 121 156 166 177 198 231 239 244 248 254 257 267 269 282 288 303 320 325 329 377 388 398 399 402 4
15 419 424 425 428 435 436 445 479 484 486 492 499 502 512 516 517 539 540 542 543 555 563 581 593 598 615 62
2 643 661 678 681 683 684 685 687 698 703 706 717 719 725 743 749 756 757 759 775 776 781 787 793 808 824 825
826 830 847 848 854 860 872 875 876 883 885 888 893 933 943 953 955 958 966 975 987 999 1004 1005 1009 1022
Node Added Successfully!
Node Added Successfully!
Node Added Successfully!
Node Added Successfully!
Node Added Successfully!
Node Added Successfully!
Node Added Successfully!
Path Followed: 8 539 808 943 975 995
Path Followed: 172 684 943 1009 25 46
```

In [*]:
```
n=int(input("Please Enter the Identifier Space Bit-Length: "))
print("Creating Chord with Identifier space =",n)
ch=Chord(n)
while 1:
    k=int(input("Enter the functionality\n1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of No
    if k==1:
        i=int(input("Enter Node Number for Adding: "))
        ch.add_node(i)
    elif k==2:
        i=int(input("Enter Node Number for Deleting: "))
        ch.delete_node(i)
    elif k==3:
        st=int(input("Enter the node from which you want to search: "))
        key=int(input("Enter the node that you want to search: "))
        ch.lookup(st,key)
    elif k==4:
        f=int(input("Enter the node for which you want finger table: "))
        ch.display_ft(f)
    else:
        ch.exit_ring()
        break
```

```
Please Enter the Identifier Space Bit-Length: 8
Creating Chord with Identifier space = 8
The following nodes have been initialised randomly:  7 14 24 30 41 49 50 52 54 59 61 66 69 88 91 97 99 104 11
0 117 124 176 193 197 201 207 227 232 244 248 252 253
Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit1
Enter Node Number for Adding: 36
Node Added Successfully!
Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit1
Enter Node Number for Adding: 47
Node Added Successfully!
Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit1
Enter Node Number for Adding: 153
Node Added Successfully!
Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit1
```

```
Enter Node Number for Adding: 217
Node Added Successfully!
Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit2
Enter Node Number for Deleting: 49
Node Deleted Successfully!
Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit3
Enter the node from which you want to search: 7
Enter the node that you want to search: 253
Path Followed: 7 153 217 252 253
Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit3
Enter the node from which you want to search: 24
Enter the node that you want to search: 47
Path Followed: 24 41 47
Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit3
Enter the node from which you want to search: 52
Enter the node that you want to search: 50
Path Followed: 52 193 7 41 50
Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit4
Enter the node for which you want finger table: 47
Finger Table of Node - 47
ft[0]=50
ft[1]=50
ft[2]=52
ft[3]=59
ft[4]=66
ft[5]=88
ft[6]=117
ft[7]=176
Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit4
Enter the node for which you want finger table: 50
Finger Table of Node - 50
ft[0]=52
ft[1]=52
ft[2]=54
ft[3]=59
ft[4]=66
```

```
ft[5]=88
ft[6]=117
ft[7]=193
Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit4
Enter the node for which you want finger table: 52
Finger Table of Node - 52
ft[0]=54
ft[1]=54
ft[2]=59
ft[3]=61
ft[4]=69
ft[5]=88
ft[6]=117
ft[7]=193
Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit4
Enter the node for which you want finger table: 54
Finger Table of Node - 54
ft[0]=59
ft[1]=59
ft[2]=59
ft[3]=66
ft[4]=88
ft[5]=88
ft[6]=124
ft[7]=193
Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit5
Exiting...

Enter the functionality
1.Add Node  2.Remove Node  3.Lookup Node  4.Display Finger Table of Node  5.Exit
```

```
In [ ]:
```