COL819 - Assignment 1

Ankit Solanki // 2016CS50401

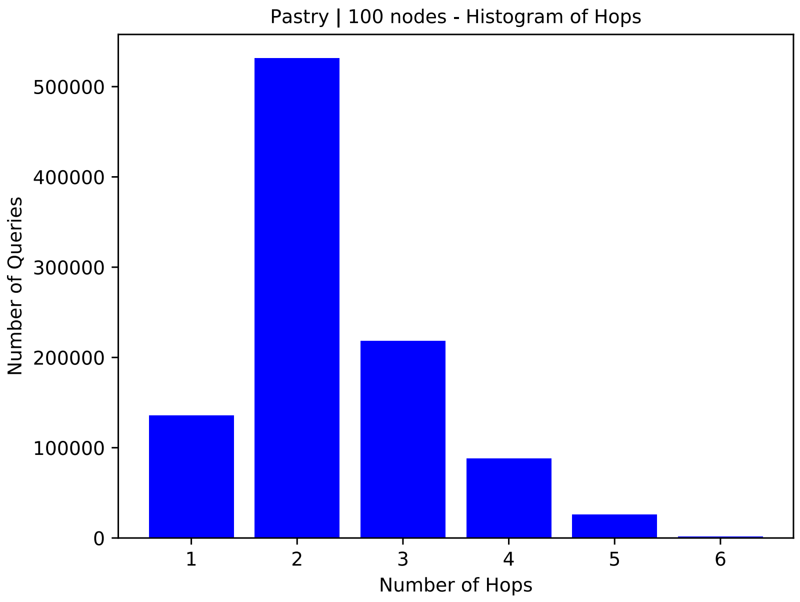
March 4, 2020

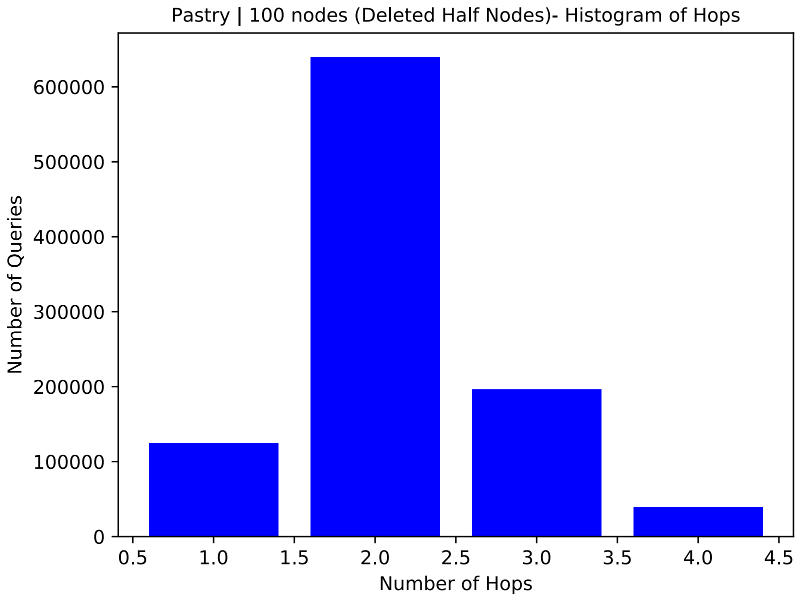
1 Experiments

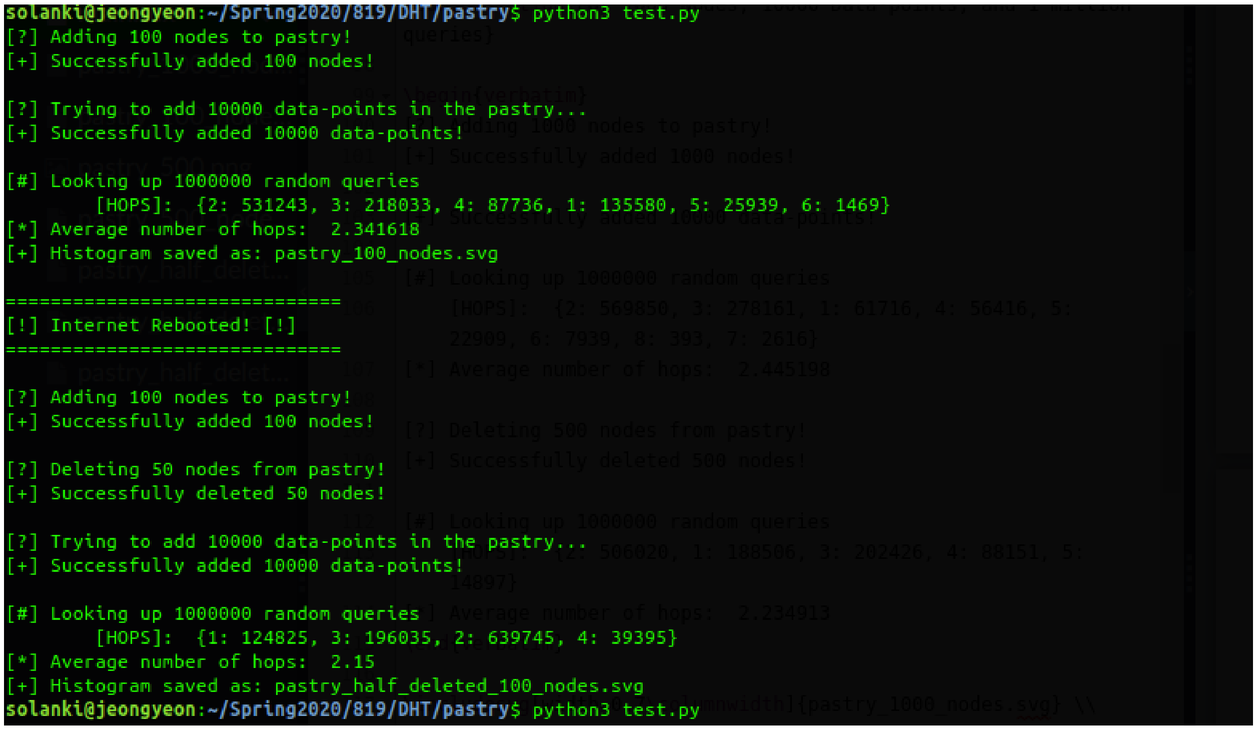
1.1 Pastry

1.1.1 For 100 Nodes, 10000 Data-points, and 1 million queries

```
# verbose output

[?] Adding 100 nodes to pastry!
[+] Successfully added 100 nodes!
[+] Successfully added 10000 data-points!
[#] Looking up 1000000 random queries
[HOPS]:  {2: 531243, 3: 218033, 4: 87736, 1: 135580, 5: 25939, 6: 1469}
[*] Average number of hops:  2.341618

[?] Deleting 50 nodes from pastry!
[+] Successfully deleted 50 nodes!

[#] Looking up 1000000 random queries
[HOPS]:  {1: 124825, 3: 196035, 2: 639745, 4: 39395}
[*] Average number of hops:  2.15
```


Pastry | 100 nodes - Histogram of Hops
Number of Queries
0
100000
200000
300000
400000
500000
Number of Hops
1
2
3
4
5
6

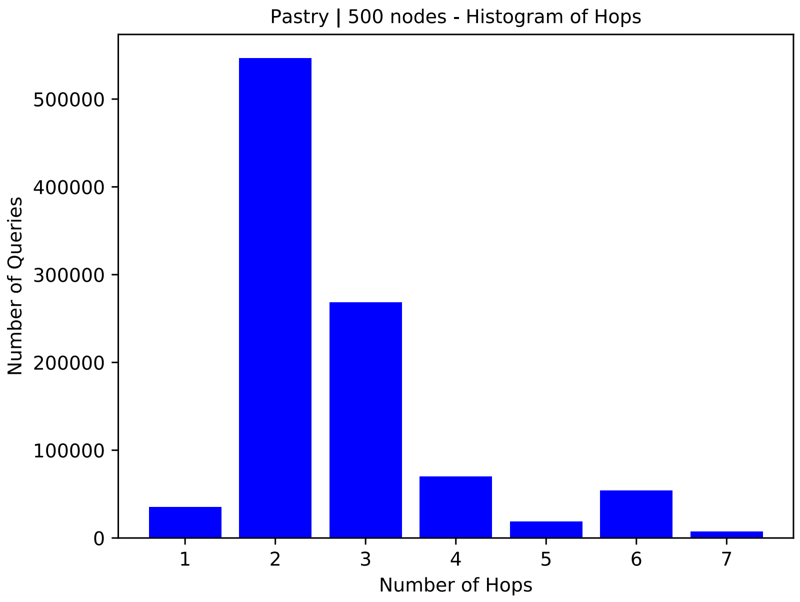

Pastry | 100 nodes (Deleted Half Nodes)- Histogram of Hops
Number of Queries
0
100000
200000
300000
400000
500000
600000
Number of Hops
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

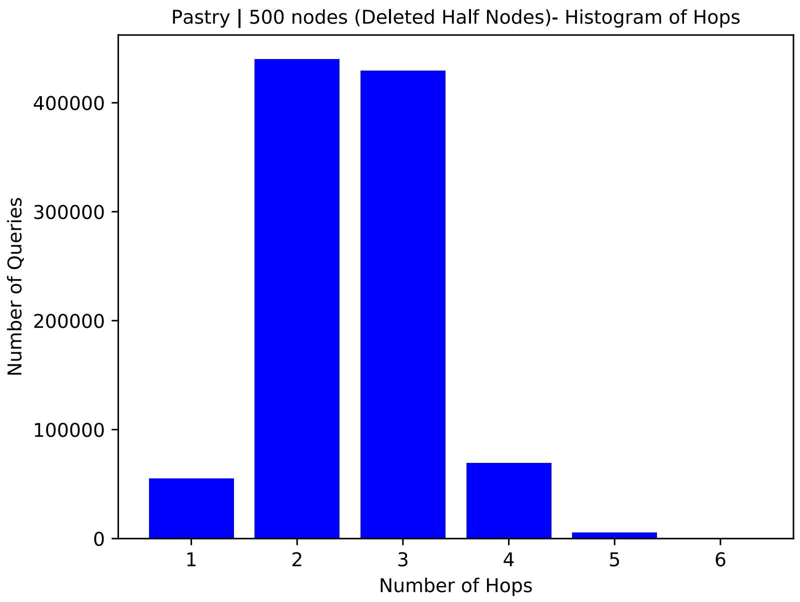


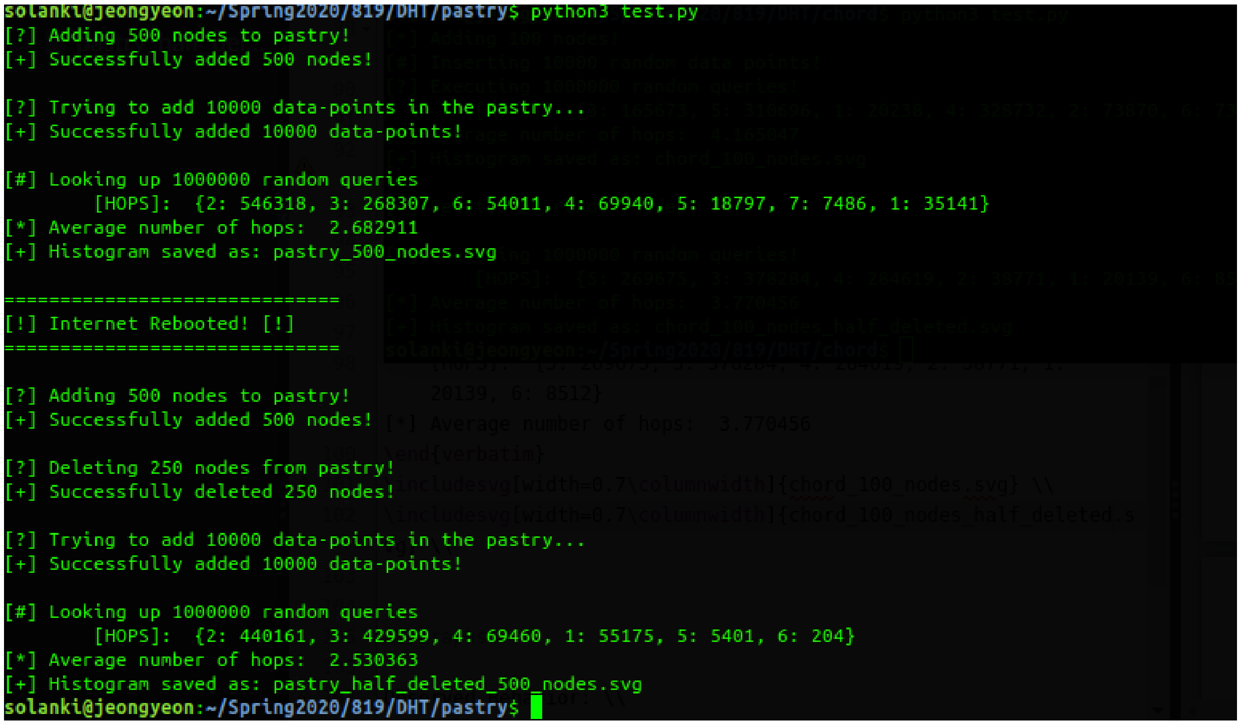
```
solanki@jeongyeon:~/Spring2020/819/DHT/pastry$ python3 test.py
[?] Adding 100 nodes to pastry!
[+] Successfully added 100 nodes!

[?] Trying to add 10000 data-points in the pastry...
[+] Successfully added 10000 data-points!

[#] Looking up 1000000 random queries
        [HOPS]:  {2: 531243, 3: 218033, 4: 87736, 1: 135580, 5: 25939, 6: 1469}
[*] Average number of hops:  2.341618
[+] Histogram saved as: pastry_100_nodes.svg

==============================
[!] Internet Rebooted! [!]
==============================

[?] Adding 100 nodes to pastry!
[+] Successfully added 100 nodes!

[?] Deleting 50 nodes from pastry!
[+] Successfully deleted 50 nodes!

[?] Trying to add 10000 data-points in the pastry...
[+] Successfully added 10000 data-points!

[#] Looking up 1000000 random queries
        [HOPS]:  {1: 124825, 3: 196035, 2: 639745, 4: 39395}
[*] Average number of hops:  2.15
[+] Histogram saved as: pastry_half_deleted_100_nodes.svg
solanki@jeongyeon:~/Spring2020/819/DHT/pastry$ python3 test.py
```

1.1.2 For 500 Nodes, 10000 Data-points, and 1 million queries

```
[?] Adding 500 nodes to pastry!
[+] Successfully added 500 nodes!

[+] Successfully added 10000 data-points!

[#] Looking up 1000000 random queries
[HOPS]:  {2: 546318, 3: 268307, 6: 54011, 4: 69940, 5: 18797, 7: 7486, 1: 35141}
[*] Average number of hops:  2.682911

[?] Deleting 250 nodes from pastry!
[+] Successfully deleted 250 nodes!

[#] Looking up 1000000 random queries
[HOPS]:  {2: 440161, 3: 429599, 4: 69460, 1: 55175, 5: 5401, 6: 204}
[*] Average number of hops:  2.530363
```


Pastry | 500 nodes - Histogram of Hops
Number of Queries
0
100000
200000
300000
400000
500000
1
2
3
4
5
6
7
Number of Hops

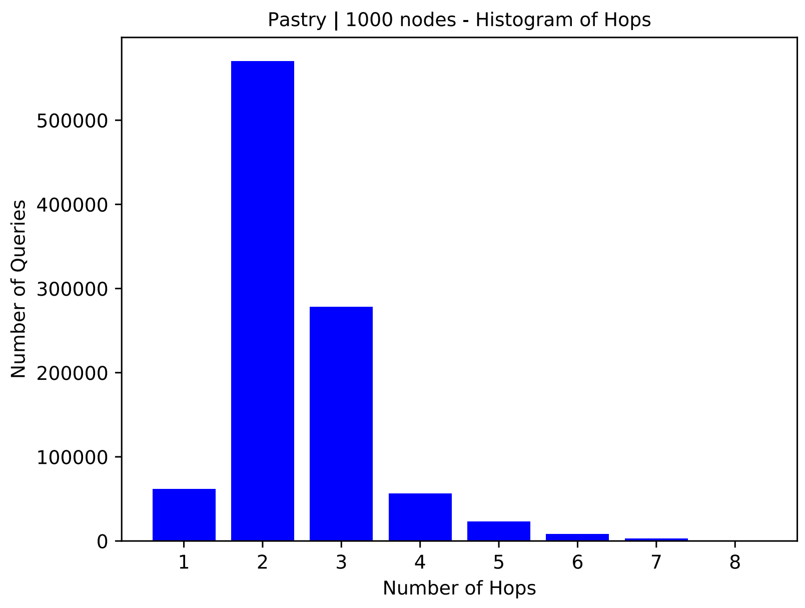

Pastry | 500 nodes (Deleted Half Nodes)- Histogram of Hops
Number of Queries
0
100000
200000
300000
400000
1
2
3
4
5
6
Number of Hops

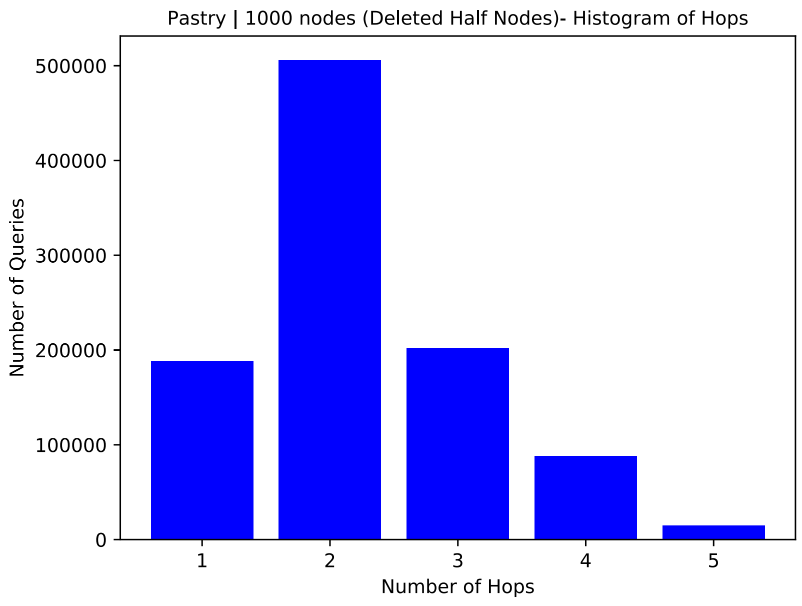


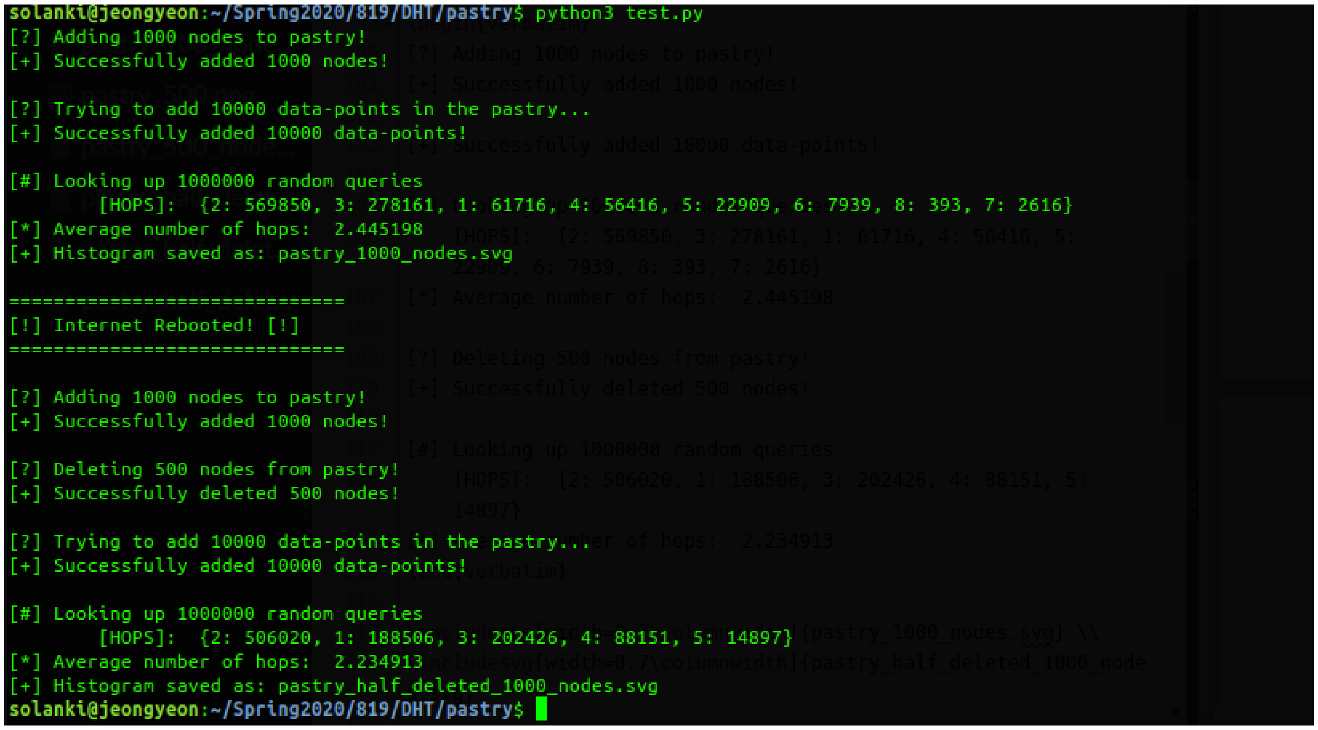
```
solanki@jeongyeon:~/Spring2020/819/DHT/pastry$ python3 test.py
[?] Adding 500 nodes to pastry!
[+] Successfully added 500 nodes!

[?] Trying to add 10000 data-points in the pastry...
[+] Successfully added 10000 data-points!

[#] Looking up 1000000 random queries
        [HOPS]:  {2: 546318, 3: 268307, 6: 54011, 4: 69940, 5: 18797, 7: 7486, 1: 35141}
[*] Average number of hops:  2.682911
[+] Histogram saved as: pastry_500_nodes.svg

==============================
[!] Internet Rebooted! [!]
==============================

[?] Adding 500 nodes to pastry!
[+] Successfully added 500 nodes!

[?] Deleting 250 nodes from pastry!
[+] Successfully deleted 250 nodes!

[?] Trying to add 10000 data-points in the pastry...
[+] Successfully added 10000 data-points!

[#] Looking up 1000000 random queries
        [HOPS]:  {2: 440161, 3: 429599, 4: 69460, 1: 55175, 5: 5401, 6: 204}
[*] Average number of hops:  2.530363
[+] Histogram saved as: pastry_half_deleted_500_nodes.svg
solanki@jeongyeon:~/Spring2020/819/DHT/pastry$
```

1.1.3 For 1000 Nodes, 10000 Data-points, and 1 million queries

```
[?] Adding 1000 nodes to pastry!
[+] Successfully added 1000 nodes!

[+] Successfully added 10000 data-points!

[#] Looking up 1000000 random queries
[HOPS]:  {2: 569850, 3: 278161, 1: 61716, 4: 56416, 5: 22909, 6: 7939, 8: 393, 7: 2616}
[*] Average number of hops:  2.445198

[?] Deleting 500 nodes from pastry!
[+] Successfully deleted 500 nodes!

[#] Looking up 1000000 random queries
[HOPS]:  {2: 506020, 1: 188506, 3: 202426, 4: 88151, 5: 14897}
[*] Average number of hops:  2.234913
```


Pastry | 1000 nodes - Histogram of Hops
Number of Queries
0
100000
200000
300000
400000
500000
Number of Hops
1
2
3
4
5
6
7
8


Pastry | 1000 nodes (Deleted Half Nodes)- Histogram of Hops
Number of Queries
0
100000
200000
300000
400000
500000
Number of Hops
1
2
3
4
5



```
solanki@jeongyeon:~/Spring2020/819/DHT/pastry$ python3 test.py
[?] Adding 1000 nodes to pastry!
[+] Successfully added 1000 nodes!

[?] Trying to add 10000 data-points in the pastry...
[+] Successfully added 10000 data-points!

[#] Looking up 1000000 random queries
        [HOPS]:  {2: 569850, 3: 278161, 1: 61716, 4: 56416, 5: 22909, 6: 7939, 8: 393, 7: 2616}
[*] Average number of hops:  2.445198
[+] Histogram saved as: pastry_1000_nodes.svg

==============================
[!] Internet Rebooted! [!]
==============================

[?] Adding 1000 nodes to pastry!
[+] Successfully added 1000 nodes!

[?] Deleting 500 nodes from pastry!
[+] Successfully deleted 500 nodes!

[?] Trying to add 10000 data-points in the pastry...
[+] Successfully added 10000 data-points!

[#] Looking up 1000000 random queries
        [HOPS]:  {2: 506020, 1: 188506, 3: 202426, 4: 88151, 5: 14897}
[*] Average number of hops:  2.234913
[+] Histogram saved as: pastry_half_deleted_1000_nodes.svg
solanki@jeongyeon:~/Spring2020/819/DHT/pastry$ 
```

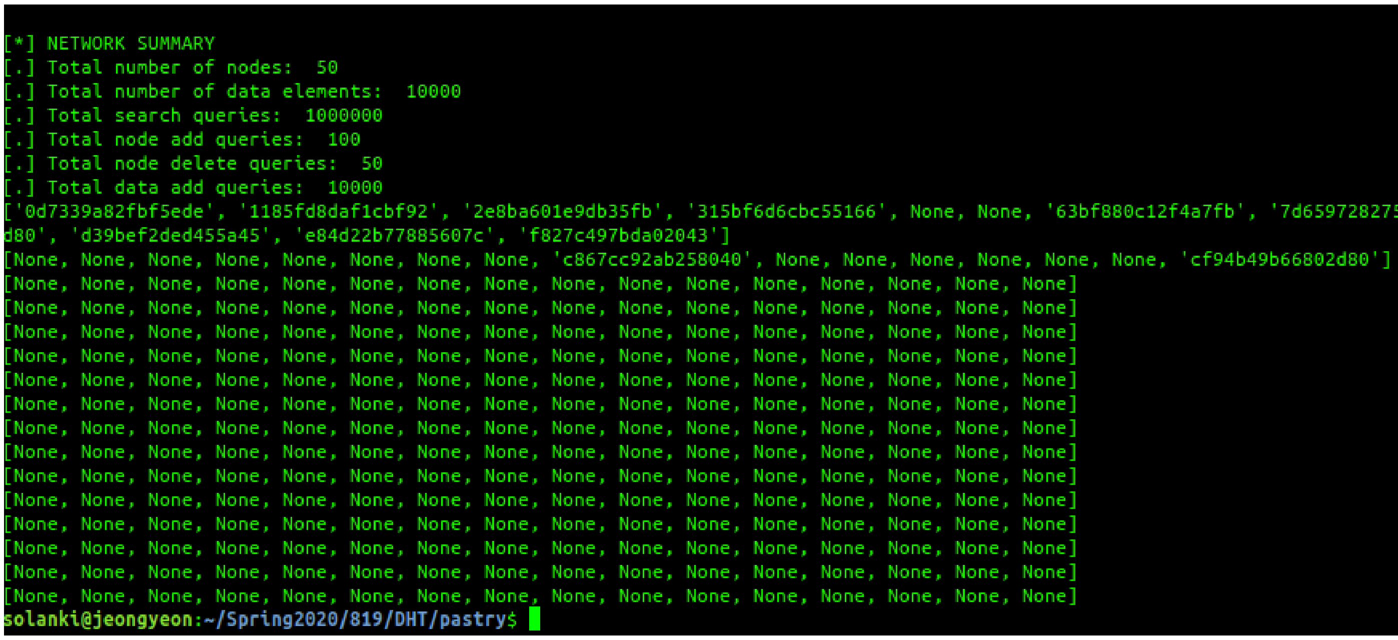
1.1.4 Observed Behavior

- As we increase number of nodes from 100, 500, to 1000. The hops vary as 2.34, 2.68, 2.44.
- Pastry dictates that complexity should vary as $\log_{16}$N. Since 16 raised to a power of 3 is 4096; results vary as intuitive, as the hops comes out to be less than 3.
- Also as we delete half the nodes, the nodes repair themselves and the pastry network gets more compact.
- This results in a decrease in average number of hops:
  - For 100 nodes : 2.34 -> 2.15
  - For 500 nodes : 2.68 -> 2.53
  - For 1000 nodes: 2.44 -> 2.23
- The graphs and so the results corresponds to this idea, which happens exactly how it is said: the rest of the network becomes compact as it repairs itself, thus less hops in average for queries.

1.1.5 Routing Table of a Single Node

```
[*] NETWORK SUMMARY
[.] Total number of nodes:  50
[.] Total number of data elements:  10000
[.] Total search queries:  1000000
[.] Total node add queries:  100
[.] Total node delete queries:  50
[.] Total data add queries:  10000
```

```
['0d7339a82fbf5ede', '1185fd8daf1cbf92', '2e8ba601e9db35fb', '315bf6d6cbc55166',
None, None, '63bf880c12f4a7fb', '7d659728275f7be9', None, None, None,
'bdf4e1d47edcf0e3', 'cf94b49b66802d80', 'd39bef2ded455a45',
'e84d22b77885607c', 'f827c497bda02043']
[None, None, None, None, None, None, None, None, 'c867cc92ab258040', None, None,
None, None, None, None, 'cf94b49b66802d80']
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
```

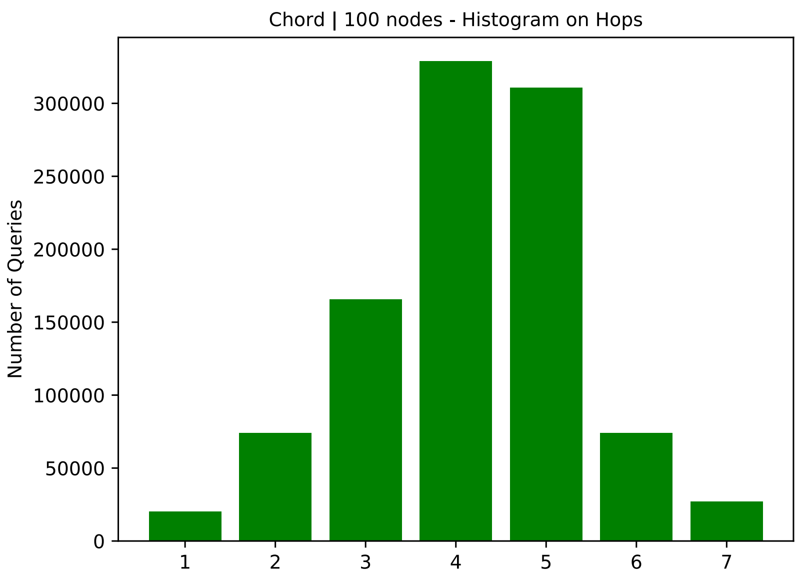


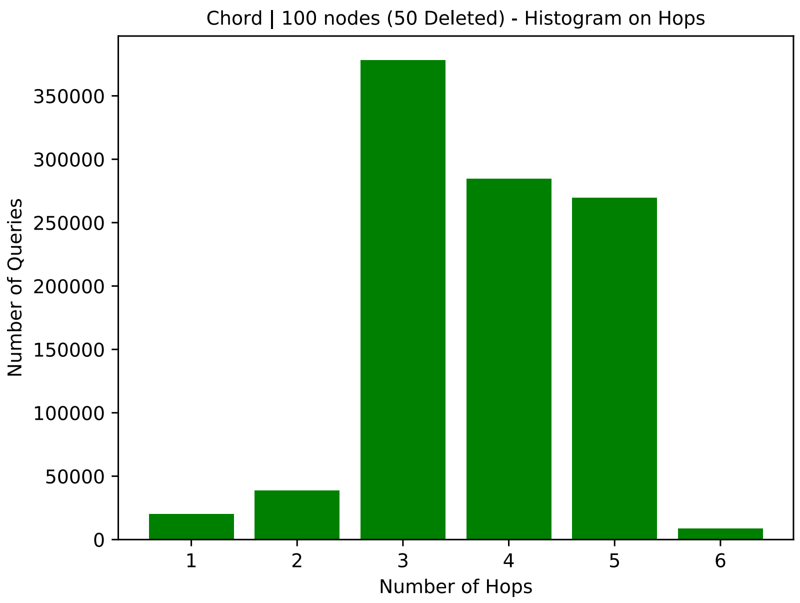
```
[*] NETWORK SUMMARY
[.] Total number of nodes:  50
[.] Total number of data elements:  10000
[.] Total search queries:  1000000
[.] Total node add queries:  100
[.] Total node delete queries:  50
[.] Total data add queries:  10000
['0d7339a82fbf5ede', '1185fd8daf1cbf92', '2e8ba601e9db35fb', '315bf6d6cbc55166', None, None, '63bf880c12f4a7fb', '7d65972827
d80', 'd39bef2ded455a45', 'e84d22b77885607c', 'f827c497bda02043']
[None, None, None, None, None, None, None, None, 'c867cc92ab258040', None, None, None, None, None, None, 'cf94b49b66802d80']
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
solanki@jeongyeon:~/Spring2020/819/DHT/pastry$
```

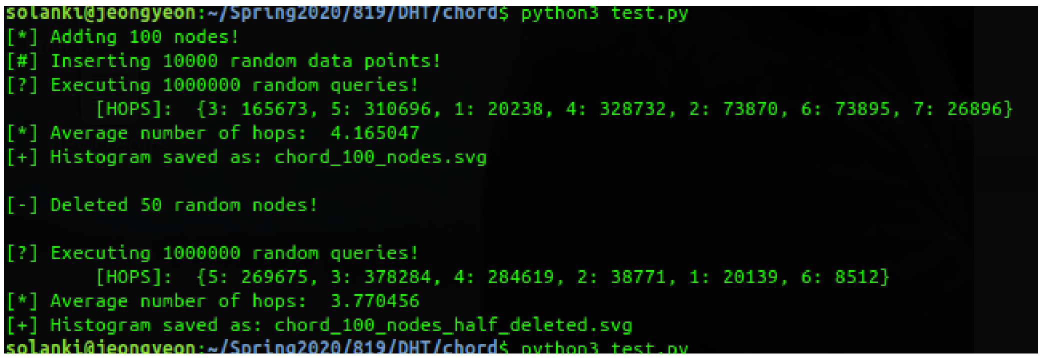
1.2 Chord

1.2.1 For 100 Nodes, 10000 Data-points, and 1 million queries

```
[*] Adding 100 nodes!
[#] Inserting 10000 random data points!
[?] Executing 1000000 random queries!
[HOPS]:  {3: 165673, 5: 310696, 1: 20238, 4: 328732, 2: 73870, 6: 73895, 7: 26896}
[*] Average number of hops:  4.165047

[-] Deleted 50 random nodes!

[?] Executing 1000000 random queries!
[HOPS]:  {5: 269675, 3: 378284, 4: 284619, 2: 38771, 1: 20139, 6: 8512}
[*] Average number of hops:  3.770456
```



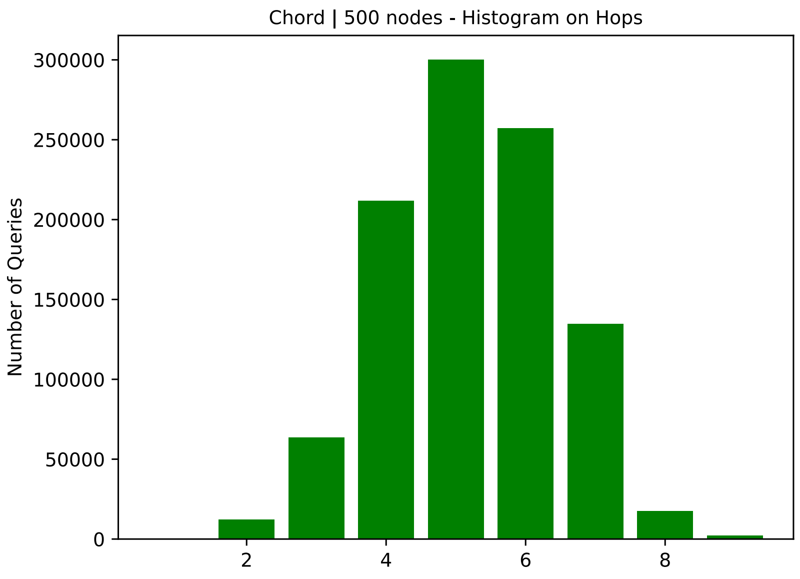


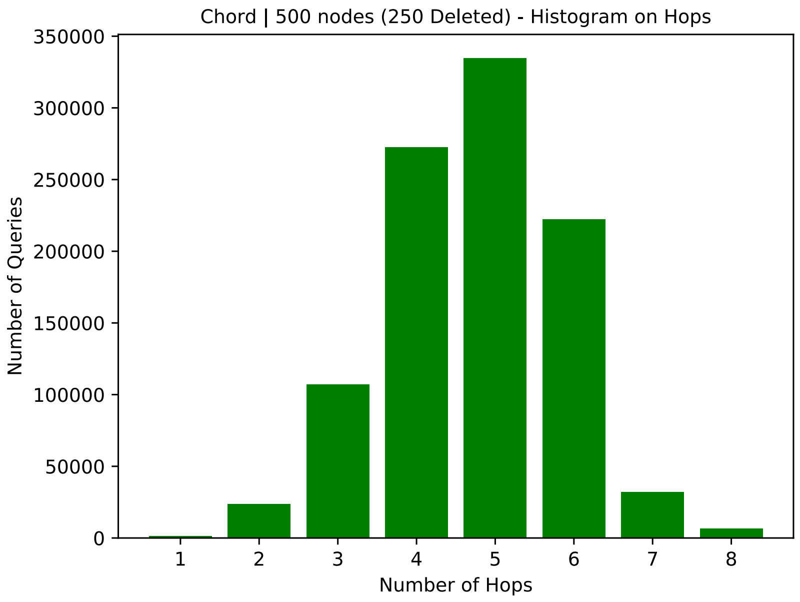


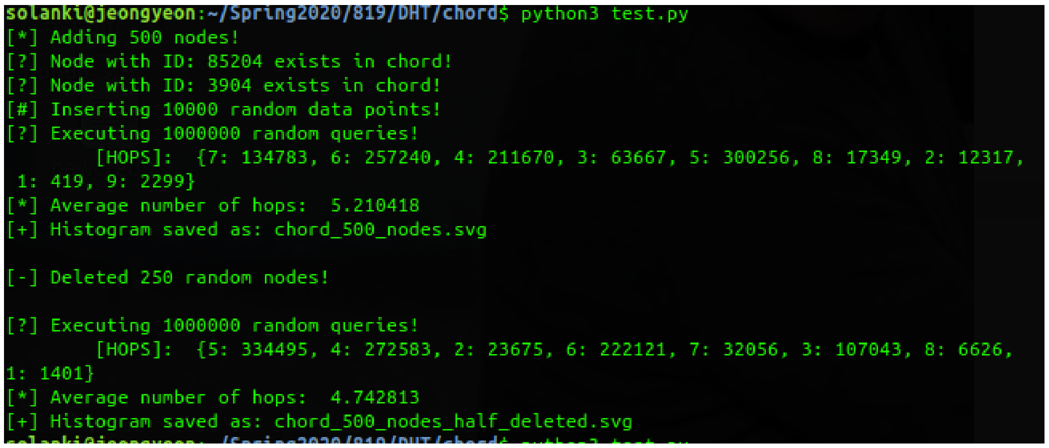
```
solanki@jeongyeon:~/Spring2020/819/DHT/chord$ python3 test.py
[*] Adding 100 nodes!
[#] Inserting 10000 random data points!
[?] Executing 1000000 random queries!
        [HOPS]:  {3: 165673, 5: 310696, 1: 20238, 4: 328732, 2: 73870, 6: 73895, 7: 26896}
[*] Average number of hops:  4.165047
[+] Histogram saved as: chord_100_nodes.svg

[-] Deleted 50 random nodes!

[?] Executing 1000000 random queries!
        [HOPS]:  {5: 269675, 3: 378284, 4: 284619, 2: 38771, 1: 20139, 6: 8512}
[*] Average number of hops:  3.770456
[+] Histogram saved as: chord_100_nodes_half_deleted.svg
solanki@jeongyeon:~/Spring2020/819/DHT/chord$ python3 test.py
```

1.2.2 For 500 Nodes, 10000 Data-points, and 1 million queries

```
[*] Adding 500 nodes!
[#] Inserting 10000 random data points!
[?] Executing 1000000 random queries!
[HOPS]:  {7: 134783, 6: 257240, 4: 211670, 3: 63667, 5: 300256, 8: 17349, 2: 12317,
1: 419, 9: 2299}
[*] Average number of hops:  5.210418

[-] Deleted 250 random nodes!

[?] Executing 1000000 random queries!
[HOPS]:  {5: 334495, 4: 272583, 2: 23675, 6: 222121, 7: 32056, 3: 107043, 8: 6626,
1: 1401}
[*] Average number of hops:  4.742813
```



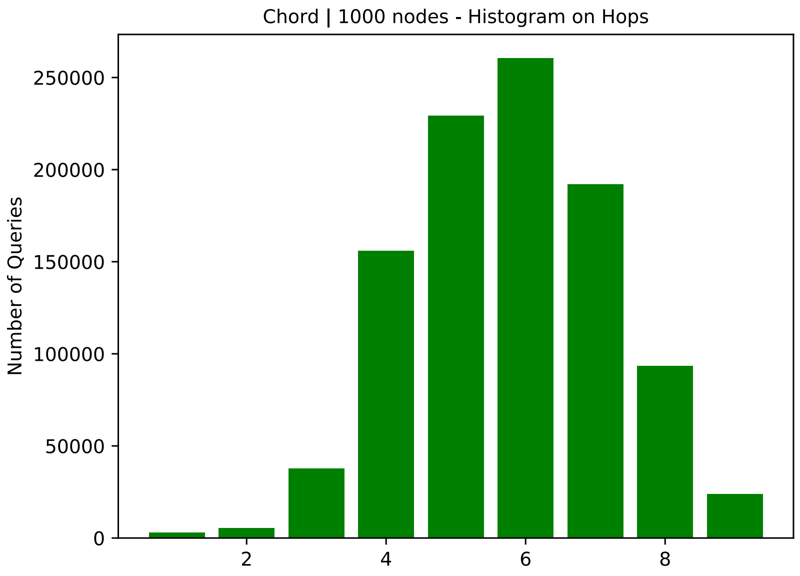


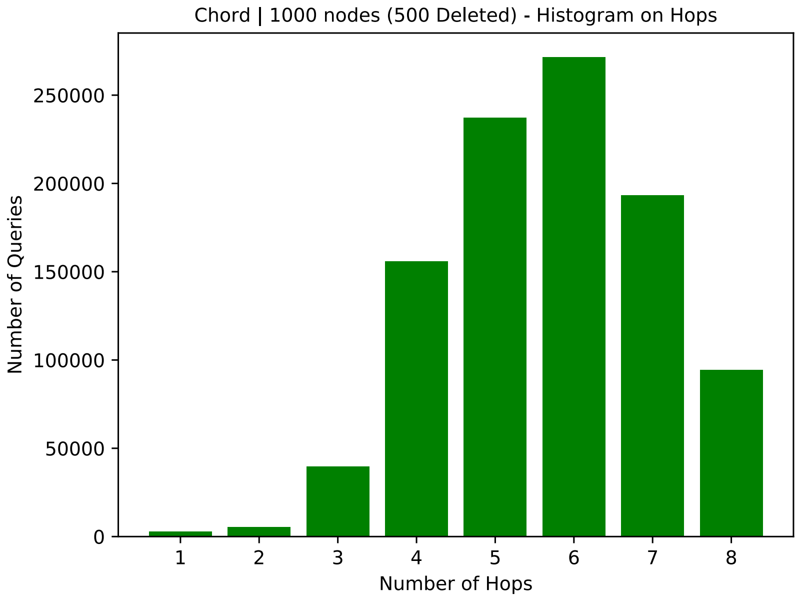


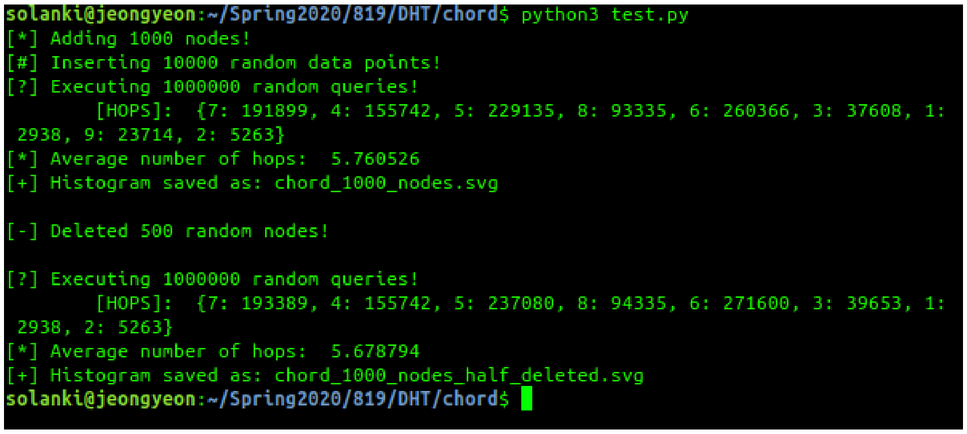
```
solanki@jeongyeon:~/Spring2020/819/DHT/chord$ python3 test.py
[*] Adding 500 nodes!
[?] Node with ID: 85204 exists in chord!
[?] Node with ID: 3904 exists in chord!
[#] Inserting 10000 random data points!
[?] Executing 1000000 random queries!
        [HOPS]:  {7: 134783, 6: 257240, 4: 211670, 3: 63667, 5: 300256, 8: 17349, 2: 12317,
 1: 419, 9: 2299}
[*] Average number of hops:  5.210418
[+] Histogram saved as: chord_500_nodes.svg

[-] Deleted 250 random nodes!

[?] Executing 1000000 random queries!
        [HOPS]:  {5: 334495, 4: 272583, 2: 23675, 6: 222121, 7: 32056, 3: 107043, 8: 6626,
1: 1401}
[*] Average number of hops:  4.742813
[+] Histogram saved as: chord_500_nodes_half_deleted.svg
```

1.2.3 For 1000 Nodes, 10000 Data-points, and 1 million queries

```
[*] Adding 1000 nodes!
[#] Inserting 10000 random data points!
[?] Executing 1000000 random queries!
[HOPS]:  {7: 191899, 4: 155742, 5: 229135, 8: 93335, 6: 260366, 3: 37608, 1: 2938, 9: 23
[*] Average number of hops:  5.760526

[-] Deleted 500 random nodes!

[?] Executing 1000000 random queries!
[HOPS]:  {7: 193389, 4: 155742, 5: 237080, 8: 94335, 6: 271600, 3: 39653, 1: 2938, 2: 52

[*] Average number of hops:  5.678794
```


Chord | 1000 nodes - Histogram on Hops
Number of Queries
250000
200000
150000
100000
50000
0
2
4
6
8


Chord | 1000 nodes (500 Deleted) - Histogram on Hops
Number of Queries
250000
200000
150000
100000
50000
0
1
2
3
4
5
6
7
8
Number of Hops

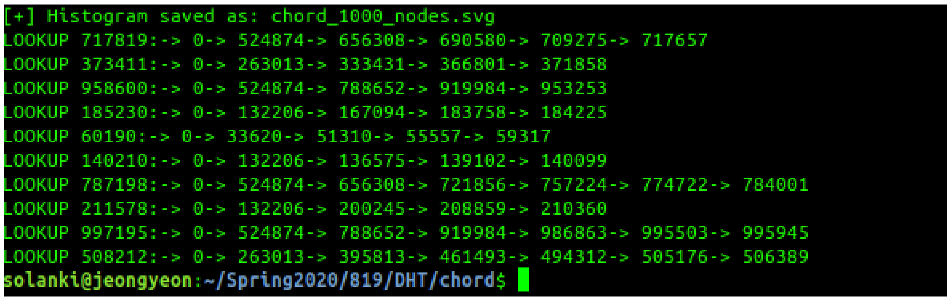

```
solanki@jeongyeon:~/Spring2020/819/DHT/chord$ python3 test.py
[*] Adding 1000 nodes!
[#] Inserting 10000 random data points!
[?] Executing 1000000 random queries!
        [HOPS]:  {7: 191899, 4: 155742, 5: 229135, 8: 93335, 6: 260366, 3: 37608, 1:
 2938, 9: 23714, 2: 5263}
[*] Average number of hops:  5.760526
[+] Histogram saved as: chord_1000_nodes.svg

[-] Deleted 500 random nodes!

[?] Executing 1000000 random queries!
        [HOPS]:  {7: 193389, 4: 155742, 5: 237080, 8: 94335, 6: 271600, 3: 39653, 1:
 2938, 2: 5263}
[*] Average number of hops:  5.678794
[+] Histogram saved as: chord_1000_nodes_half_deleted.svg
solanki@jeongyeon:~/Spring2020/819/DHT/chord$
```

1.2.4 Observed Behavior

- As we increase number of nodes from 100, 500, to 1000. The hops vary as 4.16, 5.21, 5.76.
- As we delete half the nodes, the nodes fix their own finger tables and the next set of queries take comparatively less hops.
- This results in a decrease in average number of hops:
  - For 100 nodes : 4.16 -> 3.77
  - For 500 nodes : 5.21 -> 4.74
  - For 1000 nodes: 5.76 -> 5.67

1.2.5 Path Taken by 10 Requests

```
LOOKUP 717819:-> 0-> 524874-> 656308-> 690580-> 709275-> 717657
LOOKUP 373411:-> 0-> 263013-> 333431-> 366801-> 371858
LOOKUP 958600:-> 0-> 524874-> 788652-> 919984-> 953253
LOOKUP 185230:-> 0-> 132206-> 167094-> 183758-> 184225
LOOKUP 60190:-> 0-> 33620-> 51310-> 55557-> 59317
LOOKUP 140210:-> 0-> 132206-> 136575-> 139102-> 140099
LOOKUP 787198:-> 0-> 524874-> 656308-> 721856-> 757224-> 774722-> 784001
LOOKUP 211578:-> 0-> 132206-> 200245-> 208859-> 210360
LOOKUP 997195:-> 0-> 524874-> 788652-> 919984-> 986863-> 995503-> 995945
LOOKUP 508212:-> 0-> 263013-> 395813-> 461493-> 494312-> 505176-> 506389
```


```
[+] Histogram saved as: chord_1000_nodes.svg
LOOKUP 717819:-> 0-> 524874-> 656308-> 690580-> 709275-> 717657
LOOKUP 373411:-> 0-> 263013-> 333431-> 366801-> 371858
LOOKUP 958600:-> 0-> 524874-> 788652-> 919984-> 953253
LOOKUP 185230:-> 0-> 132206-> 167094-> 183758-> 184225
LOOKUP 60190:-> 0-> 33620-> 51310-> 55557-> 59317
LOOKUP 140210:-> 0-> 132206-> 136575-> 139102-> 140099
LOOKUP 787198:-> 0-> 524874-> 656308-> 721856-> 757224-> 774722-> 784001
LOOKUP 211578:-> 0-> 132206-> 200245-> 208859-> 210360
LOOKUP 997195:-> 0-> 524874-> 788652-> 919984-> 986863-> 995503-> 995945
LOOKUP 508212:-> 0-> 263013-> 395813-> 461493-> 494312-> 505176-> 506389
solanki@jeongyeon:~/Spring2020/819/DHT/chord$
```

2 Performance Comparison

- From the values of hops it looks like pastry works faster than chord.
- Because of the extensive routing table, neighbourhood sets, and leaf sets stored at every node of pastry, it takes an average of less hops for it.
- While chord takes more time to delete a data filled node, and also more hops as shown by the numbers.
- I will prefer using pastry in this sense, but implementing pastry was harder than chord. Though the end result is more quick.

3 How to Run the Code

```
# zip name: 2016CS50401.zip

# contains:
pastry // pastry scripts folder
chord // chord scripts folder
report.pdf // use this for further instructions
```

3.1 Pastry

```
# Directory Structure

'node.py': node class
'internet.py': methods and calls analogous to internet layer
'network.py': network class
'constats.py': declared constants here
'helper.py': helper functions
'graphs': saved graphs
'test.py': testing pastry // use this for experiment

# How to run pastry

1. Go into `test.py`
2. Change `data` dictionary with suitable parameters
    For e.g: data = {"nodes": 100, "data": 10000, "queries": 1000000}
3. Run `python3 ./pastry/test.py`
```

3.2 Chord

```
# Directory Structure
```

```
'node.py': node class
'dht.py': chord class and main methods
'graphs': saved graphs
'helper.py': helper functions
'test.py': testing chord // use this for experiment

# How to run chord

1. Go into `./chord/test.py`
2. Change `data` dictionary with suitable parameters
    For e.g: data = {"nodes": 100, "data": 10000, "queries": 1000000}
3. Run `python3 test.py`
```