# ECE 454 Assignment 2

Amir Benham 20393292, Andrew Svoboda 20369388

1. As we can see in the image below we have a network with 5 peers and we want to go from node 0 to node 7. [image q1 here]

   In this network the finger table of node 0 would look like: $FT[0] = 4, FT[1] = 4, FT[2] = 0$ which means our next hop would be to node 4. This means that we would have the following situation.

   $$p = 0, q = 4, k = 7$$
   $$q - p = 0 \; and \; k - p = 3$$
   $$q - p \geq \frac{k - d}{2}$$
   $$0 \geq \frac{3}{2}$$

   Therefore the equality does not hold.

2. (a)
   (b) Only the predecessor to the item that is being inserted needs to be updated, not including the joining node. For a total of 1 node.

3. The worst case scenario for Chord DHT is when the node that is being looked up is the successor of the nodding doing the look up. In that situation regardless of what m is or regardless of the orientation of the nodes the number of hops will be linear relative to m.

4. For this solution we need to introduce a few new functions. First we will introduce a$itob()$ function that converts an integer to its binary representation. We will need the reverse of that function $btoi()$. We will also need a $concat()$ function that will combine two binary values ie. $Concat(100, 010)$ should result in 100010. We will also need the reverse of the concat function, we will call it $split()$ which takes one binary value and splits it down the middle and returns the two tokens ie. $[b1, b2] = split(someBinaryString)$, where $b1$ and $b2$ are the most significant and least significant bits respectively.

   $(i, j) \rightarrow n$

   $$n = btoi(concat(itob(i), itob(j))$$

   $n \rightarrow (i, j)$

   $$[b1, b2] = split(n)$$
   $$i = btoi(b1)$$
   $$j = btoi(b2)$$

5. There are two possible ways that a route can be created between Q and P either Q chooses P or P chooses Q. The total number of routes are $\binom{n-1}{c-1}$

   $$\therefore P = \frac{2}{\binom{n-1}{c-1}}$$

6. Using the bully election system the worst case scenario would be if the node with the lowest id sends out a message to $n - 1$ nodes and revives $n - 1$ messages back. Then the node with the second lowest node sends out $n - 2$ messages out and gets $n - 2$ messages back and so on. The end result would be $2 * (n - 1) + 2(n - 2) + 2(n - 3)...2 * 1 = n * n - 1 = O(n^2)$).

   The worst case scenario for Ring election would be if every node started an election at the same time. So that is $n$ messages that get passed around $n - 1$ times and each message grows to a size of $n$. After that goes around they will all send out the victory message with a list of every living process in it. So that is $n$ messages of size $n$ being sent out. In the end we get a total of $n*(n-1)*n+n*n-1 = O(n^3)$

7. One way to get rid of duplicate messages as soon as possible would be to add a timestamp to the election message. After a node sees the message they will keep track of it and if another message gets past to them that has a new a time stamp the node will not forward it. The node will reset the timestamp after it receives the response from the initiator. If we dont want to modify the election message by adding a timestamp to it the same concept can be applied by counting how large the list in the message is but this method does not work as well.