Surname : Pendyala
Firstname : Praveen Kumar
Matrikel # : 2919474

Problem 2.1

a) Assumptions:
   - → b is 4 (typical value for Pastry)
   - → Available peers shown in figure are distributed over the address space.
   - → l is not 128 (typical value for Pastry) - Solution would be trivial in that case.

   With the above assumptions:
   Values of 2^x-1 >= 15, for x in natural numbers, are 15, 31, ….

   That implies:
   - largest possible address = 15, 31, ….
   - Bits in identifier = 4, 5, ….

b) Let's stick to a max. address value of 15 (from previous solution)
   Digits used for Ids: 2
   Current Node Ids would be:
   - → 2 - 2
   - → 5 - 11
   - → 8 - 20
   - → 9 - 21
   - → 11 - 23
   - → 15 - 33

Problem 2.2

Suppose that we are using l-bit identifiers for ids. Now, each row of routing table is filled such that row K has at least K digits prefix common with current nodeId.

Maximum value of N = 2^l-1
Digits to represent max. value = Roof($\log_{2^b}(2^l-1)$) >= Roof($\log_{2^b}N$)
So, max. number of rows = $\lceil \log_{2^b} N \rceil$

Consider the routing for a random key at a random node among the N nodes:
The key entered the network at Node 1 and continues to Node 2, 3 ….

Node 1: Considering worst case, has no prefix match with Key. Picks an entry from Row 1
Node 2: Matches 1 digit prefix and picks entry from Row 2
Node 3: Matches 2 digits prefix and picks entry from Row 3
...
…….

This can continues, increasing Key matching by a digit at each hop. Since the max. number of rows is $\lceil \log_{2^b} N \rceil$, in worst case, this process terminates after reaching $\lceil \log_{2^b} N \rceil - 1$ hops. It can then be routed to the responsible node by looking at the neighbours table of the last node in the above process.

Problem 2.3

a. Fingers needed : 7
   Formula for i-th finger of node n : f(n(i)) = Successor(n + 2^i)
   Responsibility area of node 82 : 73 - 82
   Fingers table for Node 82:

Finger	Target ID	Node ID
0	83	86
1	84	86
2	86	86
3	90	1
4	98	1
5	14	32
6	46	67

b. Steps needed : 2
   Path traced : Node 82 will forward the query to node 1, since it is the closest peer not exceeding the lookup value. Then, Node 1 will forward the query to Node 8, which is responsible for the id.

Problem 2.4

A) Formula for i-th finger of node n : f(n(i)) = Successor(n + 2^i)

B) Finger needed : 7

C) Finger table for Node 50:

Finger	Target ID	Node ID
0	51	52
1	52	52
2	54	63
3	58	63
4	66	75
5	82	2
6	14	15

D) Responsible area for node 50: 31 -50

E) Steps needed : 2
Path traced : Node 50 will forward the query to node 15, since it is the closest peer not exceeding the lookup value. Then, Node 15 will forward the query to Node 30, which is responsible for the id.

Problem 2.5

A. Answers
  → True
  → False
  → True
  → False

B. Small - Replication
Large - Redundancy

C. **Power of Two choices** is an example of such DHT mechanisms
We can use:

$h_1(.) = SHA1(h_0(.))$

$h_2(.) = SHA1(h_1(.))$

D. We could use a hash function like SHA-1, SHA-2 or MD5 on the nodes information like Ip address or other unique identifiers.

   Arbitrary NodeID creation would mean, a node is identified by giving a random id at the time of bootstrapping, instead of hash on a unique identifier of the node.

   So, any malicious node may falsely claim to be some other node. Since there is no way of deriving or verifying this ID using the malicious node’s unique identifiers, the network is more vulnerable compared to the former choice.

E. Using pointers with the *Power of Two Choices* would require Additional administration of pointers - to make sure that the pointers are consistent with the changes in the network. This causes an increase in overall load of the system

F. Let’s say, Previous Successor(A) = B and the newly joined Node is C

   Stabilization:

   → A asks B for it’s predecessor. i.e., Prodecessor(B)
   → A receives C. Not equal to the expected value of A
   → A updates its successor to the newly joined Node C.