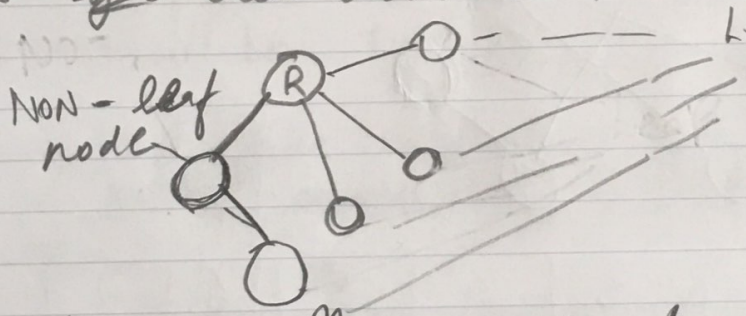
Exam 1

Q1

i. R, will have the maximum height. A height of any node is the number of edges from one node to its leaf. So, if R is the root then all the nodes are connected to it so, R will have the number of edges, hence the maximum height.

ii There is not enough information to prove this. Because to find a depth of a node in a tree you would calculate the length from the Root R, to L. In this case L is a random leaf node and the origin problem states "One root node R & Several leaf and One non leaf"

This tree would look like




According to this graph L is a "Random" leaf node so if is not for certain it connected to "One - non leaf node" or part of one of the "Several" leaf nodes. So we cannot come to the conclusion that 'L' has the maximum depth.

iii The height of a tree can be calculated recursively. We can do this by first calculating the left & right Subtrees heights.

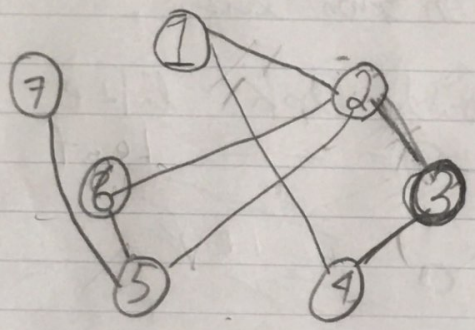
To do this

(a) Size of right subtree
(b) Size of left subtree
(c) The height will be the size of right + left & add it with (1) this will give the height of tree recursively.

the best algorithm to use for this is a stack algorithm.

---

Q2) 1-4, 1-2, 2-3, 2-5, 2-6, 4-3, 5-6, 5-7
Undirected, 10 nodes




1)

| Nodes | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 3 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 4 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 6 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

2) The time complexity for adjacency matrix is $O(n^2)$ since it's a undirected graph.

(2) The time complexity for the matrix is $O(n^2)$
Since this is a undirected graph it still is $O(n^2)$

3) This has the same reasoning as the last
the time complexity will be $O(n^2)$

---

Q3 1 million integers, in array randomly.
Each cell → 1 nano sec

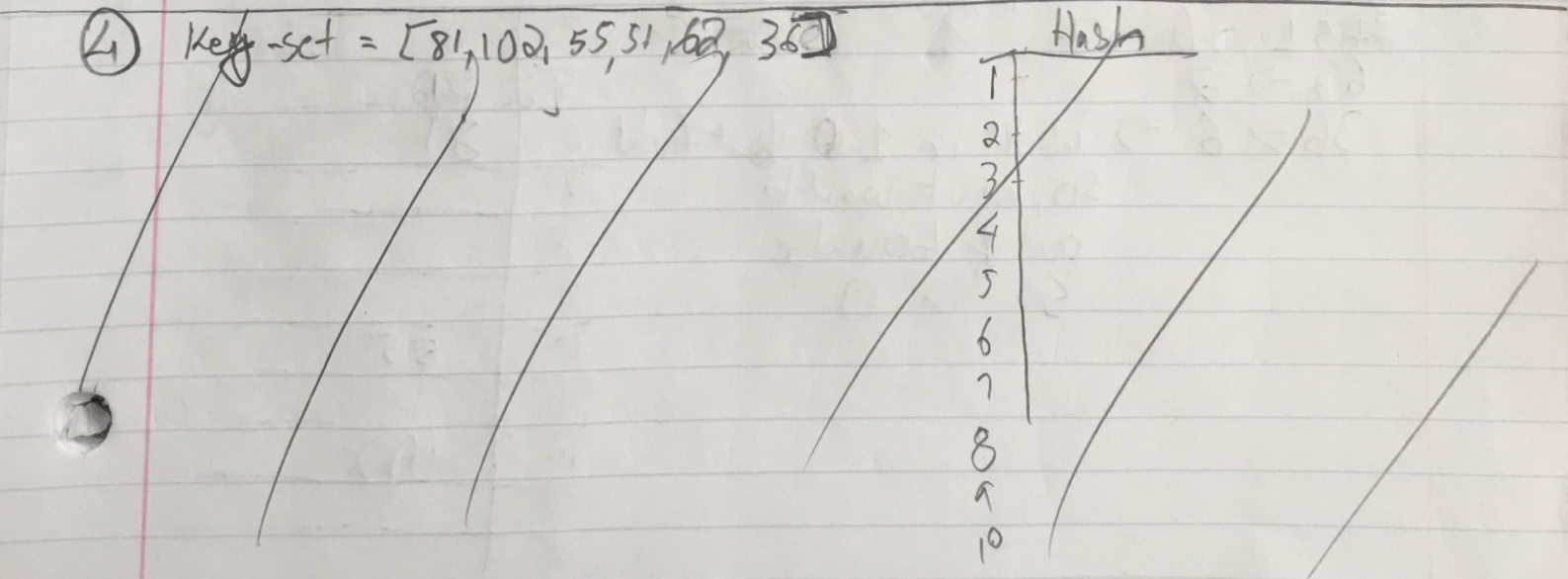
(1) We would need to take each cell an multi by it by 1 million
Worst case is = 1 million * 1 nano sec.

(2) It would not be any different if an arry got sorted as long as the time & no. of integers do not Change it will be 1 million * 1 nano sec.

(3) Yes, just like with most serches the binary search will be better we can improve it by using $O(\log n)$

---

(4) Key-set = [81, 102, 55, 51, 62, 36]



$S(K) = S(K) \% 15$

(4) [81, 102, 55, 51, 62, 36]

(i) Lin Probing

So 81 % 15 → 6
102 % 15 → 12
55 % 15 → 10
51 % 15 → 6 but full so 7
62 % 15 → 2
36 % 15 → 6 but full so 8

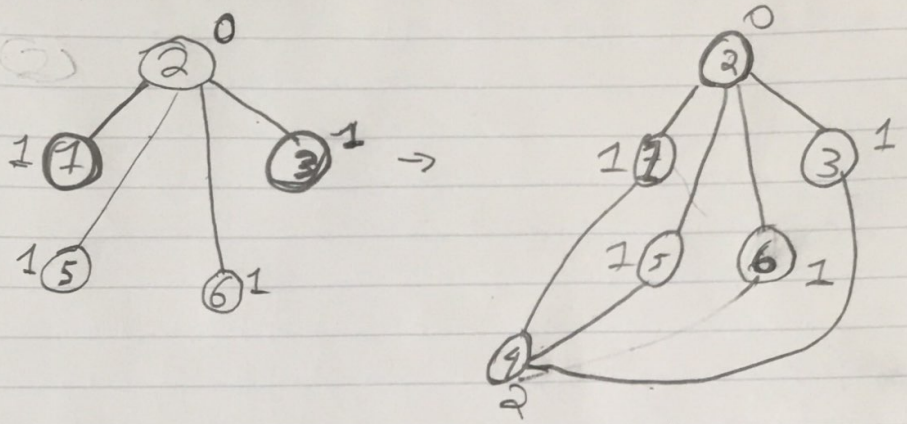
| | Table |
|---|---|
| 1 | |
| 2 | 62 |
| 3 | |
| 4 | |
| 5 | |
| 6 | 81 |
| 7 | 51 |
| 8 | 36 |
| 9 | |
| 10 | 55 |
| 11 | |
| 12 | 100 |
| 13 | |
| 14 | |
| 15 | |

ii) Quad Probing

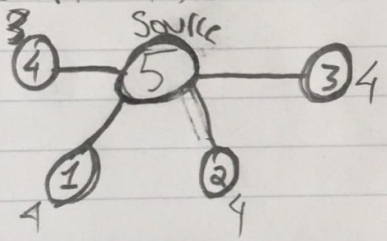
81 → 6
102 → 12
55 → 10
51 → 6 → 7 insert in 7
62 → 2
36 → 6 → insert in 10 but full
so, next would be out of bounds so in 0

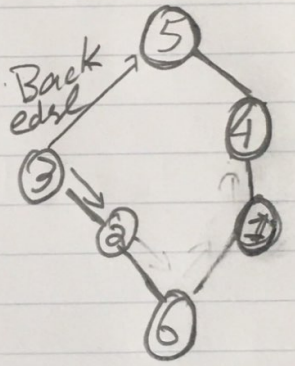
| | Table |
|---|---|
| 1 | 36 |
| 2 | 62 |
| 3 | |
| 4 | |
| 5 | 51 |
| 6 | 81 |
| 7 | |
| 8 | |
| 9 | |
| 10 | 55 |
| 11 | |
| 12 | 102 |
| 13 | |
| 14 | |
| 15 | |

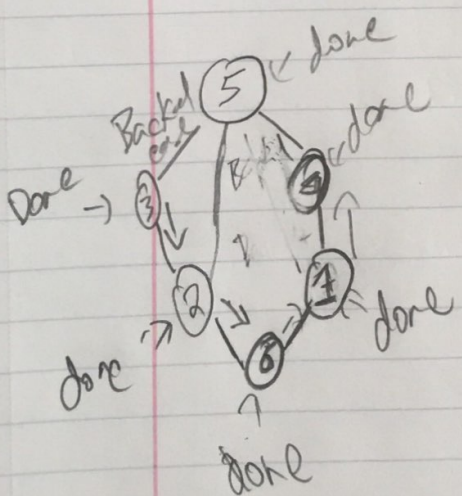
Q5 (1) Breadth First Search



(6) Depth First Search




Back edges = (3,5) (6,3), (1,5), (2,3), (4,3)