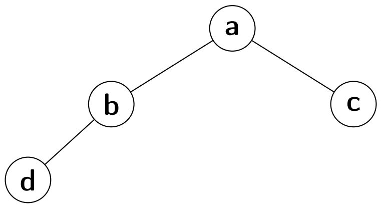
1. (6 points) Prove by induction on h (height) that the number of leaves L in a binary tree satisfy:

$$L \leq 2^{h-1}$$

For example, the following tree statisfies the above relation:



Solution:

Base Case: Base case is already available with the question. It satisfies the stated property. It has two leaf nodes and its height = 2. The total number of leaves = $2^{2-1} = 2^1 = 2$

Induction Hypothesis: Now assume that the property holds for any binary tree of height less than equal to $h - 1$.

Induction Step: Consider a tree of height h. Just delete the root node. It produces two binary subtrees of heights $h1$ and $h2$ respectively. Both $h1 \leq h - 1 < h$ and $h2 \leq h - 1 < h$. So the induction hypothesis can be applied to both subtrees.

Each tree has number of leave nodes $\leq 2^{h-1}$ each. So the original tree has leaves $\leq 2^{h-1} + 2^{h-1} = 2^{h-1}$.

2. (6 points) Simulate the behavior of a hashing set storing the integers in a hash table (closed hashing) with the following conditions:

   - The initial table size is $m = 5$.
   - Quadratic probing is used for resolution of collision.
   - The hash function uses the integer value (plus any probing needed) mod the size of the table, i.e., $h(x, i) = (x + i^2) \mod m$, for $i = 0, 1, \ldots$
   - When the load factor α reaches or exceeds 0.5, the table expands to double the capacity and rehashes the values stored at smaller indices first.
   - An insertion fails if more than half of the table slots have been tried.

   Keys are inserted in the sequence {86,76,16,66,26}

   Find out the table indices for the values in the final hash table?

   (Give precise explanation to earn partial credits.)

Solution (a): First insertion: Initial size of hash table = 5, 86 mod 5 = 1. So, initially 86 will be stored at table index 1.

Second insertion: 76 mod 5 = 1 which leads to a collision, using quadratic probe, with first probe we find 76+1 mod 5 = 2 which inserts 76 in table index 2.

Third insertion: The table is not yet half full because $\alpha = m/n = 5/2 > 1/2$. Now 16 is inserted at index 0 on third attempt. Notice had the third attempt been unsuccessful, insertion of 16 would have failed.

Table Index	0	1	2
Inserted Key	16	86	76

Now load factor $n/m = 3/5 > 0.5$ causing expansion of table size to $5 \times 2 = 10$. A rehashing is carried out with same hash function but now table size is 10. The rehashing should be performed in sequence {16, 86, 76} as corresponding table indices are {0, 1, 2}. After rehashing new indices are

Table Index	6	7	0
Inserted Key	16	86	76

`+4 marks, if correct up to this point`

After table expansion is done, the remaining insertions are performed.

Fourth insertion: 66 is inserted in new table at the position $(66+3^2) \bmod 10 = 5$ on third attempt as it clashes will three previously inserted keys.

Table Index	6	7	0	5
Inserted Key	16	86	76	66

Fifth insertion: Next key is 26, it clashes with all four existing keys 16, 86, 76, 66, so it goes position $(26 + 4^2) \bmod 10 = 2$. Key indices in table are now:

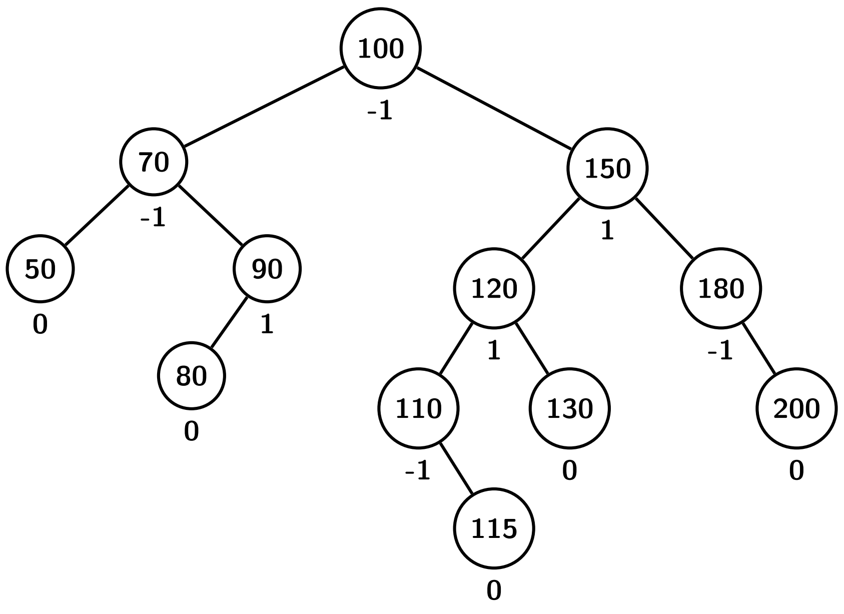
Table Index	6	7	0	5	2
Inserted Key	16	86	76	66	26

`+2 marks for the last two insertions and final results`

3. (6 points) You have studied about rotation in the context of red black tree balancing (insertion/deletion). This question is about the role of rotation in rebalancing.

   - Define balance factor of each node in a tree as the difference between heights of left and right subtrees of the node.
   - A tree is said to be balanced if balance factor of each node in the tree is equal to +1, or 0, or -1.

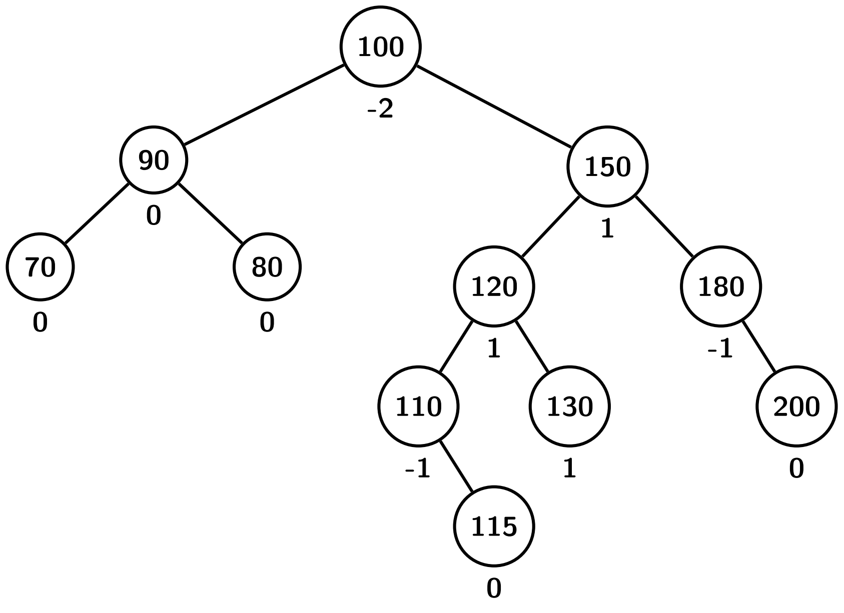
   Now consider the figure of a balanced BST shown below and provide solutions for the problems below.



(a) (3 points) What will be the tree after deleting 50 and then applying a single left rotation on 70-90-80? Is the resultant tree balanced?

(b) (3 points) What will be tree after delete 130 from BST resulting from (a) and applying a single right rotation on 120-110-115 to it. Is the new BST balanced?

Solution (a): After deletion of 50 and applying single left rotation on 70-90-80, it no longer remains a BST. The BST property is violated in subtree rooted at 90, and the tree is also unbalanced.

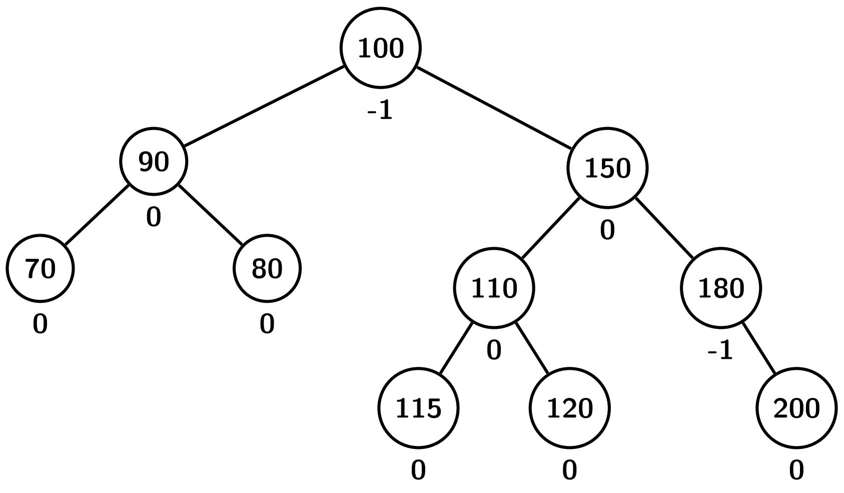


```
No partial credits

If double rotation instead of single roation is used without specifying
the reasons -1 mark

If BST violation not identified with single rotation -1 mark
```

Solution (b): The BST after deletion of 115 and applying right rotation on 120-110-115 is given below.



The new tree violates BST property at two subtrees rooted at 90 and 110. But due to right rotation the tree becomes balanced as balance factor of each node is either 0, or 1, or -1.

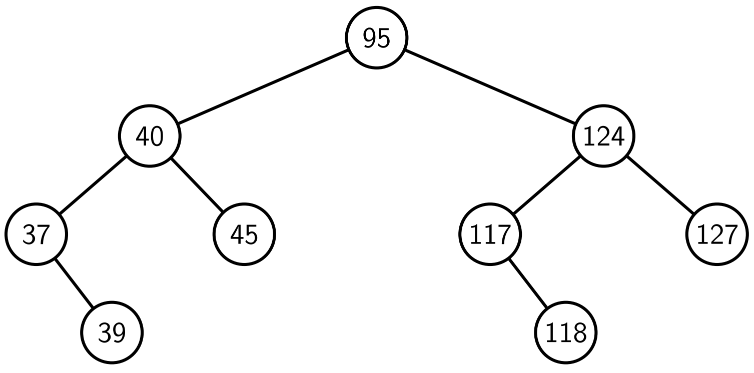
`No partial credits`

`If double rotation instead of single roation is used without specifying the reasons -1 mark`

`If BST violation not identified with single rotation -1 mark`

4. (6 points) This question is about Binary Search Tree (BST).

   (a) (3 points) For a BST, there will be some keys that, if inserted, increase the height of the tree. For the BST below, list out all the integer keys (distinct from those already present in the tree) for which this is true. If no such key exists, explain why not.



   (b) (3 points) Consider the set of elements {3, 12, 14, 15, 21, 35, 45} for insertion into an initially empty BST. Find the three different insertion sequences each of which results in a completely balanced tree.

   (No partial credit for this question.)

Solution (a): 38, 119-123.

`+3 for all correct answer.  No partial marks for this question.`

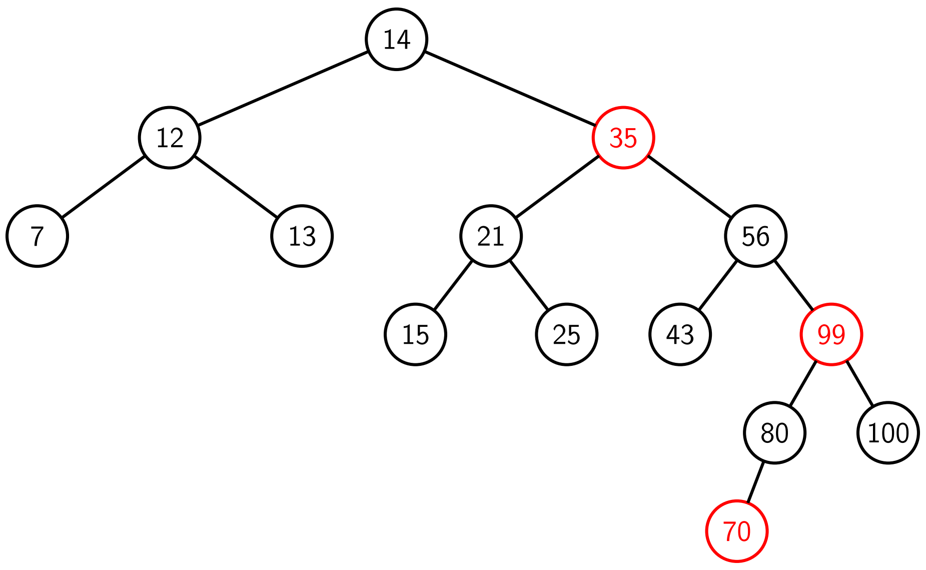
Solution (b): The sequences could be:

1. {15, 12, 35, 14, 3, 42, 21},
2. {15, 35, 12, 14, 21, 42, 3}, and
3. {15, 35, 13, 21, 42, 14, 3}

`+1 mark for each correct sequence.`

5. (6 points) Give an example of a smallest red-black tree with of maximum height having total number of nodes equal to three more than the minimum possible number of nodes. Give correct explanation in support of your solution.

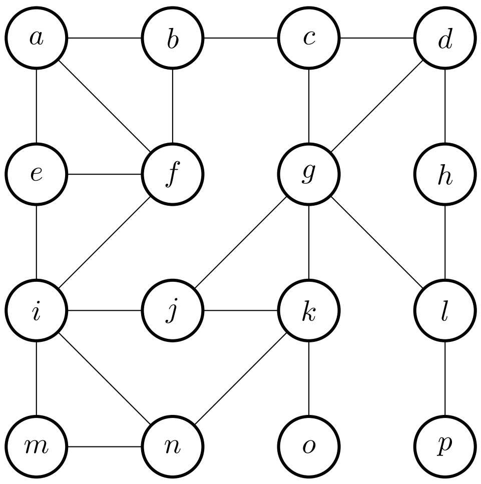
Solution: A red-black tree with maximum black height having minimum number of nodes will have all black nodes and total number of nodes is equal to N for height $= \log N$. So, we need a smallest tree with $N + 3$ nodes and black height $= \log N$. So, smallest tree must have 3 extra nodes alternately distributed as red node if the height has to be increased to maximum. a tree must have at most six nodes (colored alternately as red and black not counting the external leaf) in a path from root to a leaf. So, we should look for tree of black height 3 having minimum number of nodes in which a single path from root to a leaf has alternate red and black node. Such a tree is shown below. All similar examples should be correct.



`+4 for a correct example.`

`+2 for the correct explanation.`

6. (6 points) We performed a DFS of the graph below, starting at node a. When picking up which neighboring node to visit, we opted to go in alphabetical order.



(a) (3 points) Suppose search starts from vertex a, which search DFS or BFS can reach the goal vertex g faster (having to traverse less number of edges)? Give the edge sequence we have to traverse in each case to justify your answer.

(b) (3 points) Suppose we want to reach the goal vertex f from start vertex a which search DFS or BFS will reach us f faster? How many edges we have to traverse for the slower search to the goal vertex. Give the edge sequence we have to traverse in each case to justify your answer.

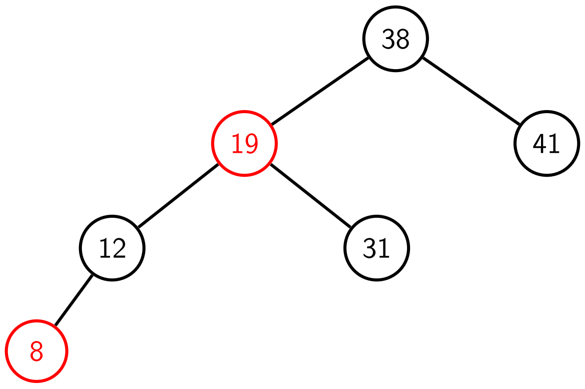
Solution (a): It is DFS. Because in DFS the edge sequence traversed for reaching g is: $a \rightarrow b, b \rightarrow c, c \rightarrow d$, $d \rightarrow g$. But BFS the edge sequence traversed is: $a \rightarrow b, a \rightarrow e, a \rightarrow f, b \rightarrow c, e \rightarrow i, c \rightarrow g$.

`Full mark only if edge sequences are mentioned, otherwise 2`

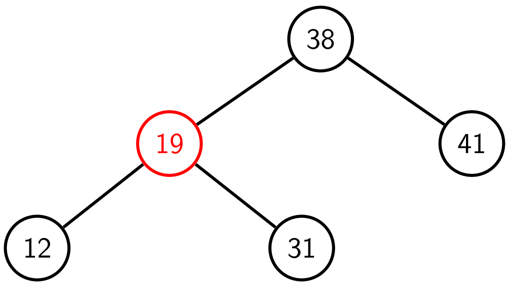
Solutin (b): We can reach f from a faster in BFS than in DFS, because the edge sequence we have traverse for reaching f from a is: $a \rightarrow b, a \rightarrow e$, $a \rightarrow f$. In DFS the edge sequence for reaching f is: $a \rightarrow b$, $b \rightarrow c$, $c \rightarrow d$, $d \rightarrow h$, $h \rightarrow l$, $l \rightarrow g$, $g \rightarrow j$, $j \rightarrow i$, $i \rightarrow c$, $c \rightarrow f$

`Full mark only if edge sequences are mentioned, otherwise 2`

7. (6 points) Show the result of successive deletion of keys 8, 12, 31 from the red-black tree below identifying the cases as discussed in the class.

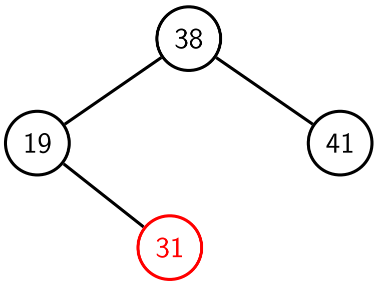


Solution: The deletion of 8 can be done without any problem as it is a red leaf node. It represent **case 0 or base case** as discussed in the class. The tree after deletion of 8 is:

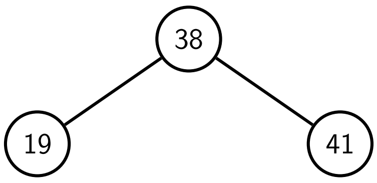


`+1 mark up to here`

Deletion of 12 creates problem as it is a black node. The left external node which replaces 12 will have double black color. Since sibling of 12's child (external leaf) is 31. Color of 31 is black and it has two black children. Therefore, it represents **case 2** as discussed in the class. This case is also simple, we just flip its color to red and move X up to 19. Since, 19 is red we have to flip its color to black. After deletion of 12, the tree is:



The next deletion is 31. As 31 is red, it represents **case 0 or base case**. It deletion does not violate any color invariant for a red black tree. The new tree is:



`Full marks if cases mentioned, otherwise +4 mark (if cases are not mentioned).`

8. (6 points) This question is about closed hashing where elements are directly inserted into the table.
   (a) (2 points) Let H be a hash table where collision is resolved by linear probing. Suppose the size of H is M and we have to perform $n \leq M/2$ distinct insertions. What is the worst case time complexity (in Big-Oh notations) to insert n keys?
   (b) (2 points) Suppose we have to put 1500 items into a hash table, and we are using hashing with chaining. If the table size 375. What is the average number of probes for a successful search?

(c) (1 point) What will be the table size if we want the average number of probes for successful search to be 2?

Solution (a): Since $n \leq M/2$, load factor does not exceed 1/2. But collision may occur with every insertion. So with ith insertion there will be $i-1$ collisions. Therefore total number of collisions for n insertions will be

$$\sum_{0}^{n-1} i = n(n-1)/2 = O(n^2)$$

Hence, the worst case time complexity is O(n^2).

`+1 for correct answer without explanation`

`+2 for correct explanation`

Solution (b): Recall the formula for successful search for hashing with chaining. It is $(1 + \alpha/2)$, where $\alpha = n/m$ is load factor. So if the table size is 375 and number of items is 1500, then $\alpha = 1500/(2*375) = 2$. So the number of average number of probes required for successful search is 1+2 =3.

`No partial credit`

Solution (c) Since we want $\alpha = 1$, the table size must be half the number of keys, i.e, 750.

`No partial credit`

9. (6 points) An undirected graph $G = (V, E)$ is without multiple edges, if no more than one edge exists between any two distinct vertices $v, w \in V$. Similarly, a directed graph $G = (V, E)$ is without parallel edges if no more than one edge exists between any two distinct end points $v, w \in V$. In this question an undirected graph is assumed to be without multiple edges, and a directed graph is assumed to be without parallel edges. Now answer the following questions.

   (a) (2 points) Draw a directed graph with five vertices and seven edges and exactly one self loop. A self loop is an edge which have the same initial and same terminal vertex. If you cannot draw it, give a proof that such a directed graph is not possible.

   (b) (2 points) Draw an undirected graph with four vertices v_1, v_2, v_3, v_4 and five edges such that there are no parallel edges, and there is a path from v_1 to v_3. If you cannot draw it, give a proof that such a graph is not possible.

   (c) (2 points) An undirected graph with five vertices with each vertex having degree three. If you cannot draw it, give a proof that a such graph is not possible.

Solution (a): Many examples are possible.

Solution (b): Many examples are possible.

Solution (c): Such a graph is not possible. Number of odd degree vertices in a graph should be even.

10. (6 points) Answer the following questions in the context of hashing.

(a) (2 points) One of your friends has done the assignment and stored it his area. Suppose you have stolen a copy of the system passwd file. Would that allow you to access the friend's area and steal the assignment file? If yes, explain why? If not, why not?

(b) (2 points) Suppose you know user name of someone and you also know a password of a different person. But the the other password you know has the same hash value as the hash value of the first person's password. Will you be able to login into the account of the first person? If yes, why? If not, explain why?

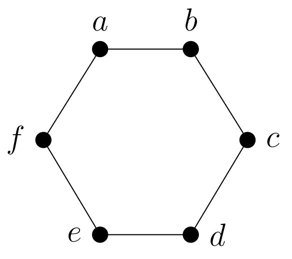
(c) (2 points) Suppse we use hashing with chaining and the following keys are inserted: 5, 28, 19, 15,20, 33, 12, 17, 10, and $m = 9$. For simplicity we use the following hash function $h(x) = x \mod m$. In which slots do the collisions occur?

Solution (a): No, you won't be able to get into his area. You only know hashed password and the user name. However, when you login you have to give password. The password is internally hashed by the login process and matched. Since, you don't know the password you cannot give it. Hashed value of password is of no use for loging into the user's account.

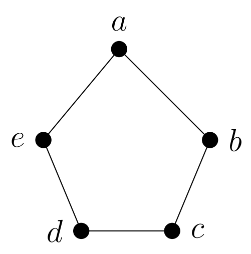
Solution (b): Yes, you have a password which fortunately hashes to same value. So when system internally applies the hash to the password provided by you, it generates the same hash value as the hashed value of the password of the user's account. So it lets you in.

Solution (c): The hash values are 5, 1, 1, 6, 2, 6, 3, 8, 1. So collisions occur with 19, 33, and 10.

11. (6 points) Conider the two graphs (a) and (b) below, where each edge represents one unit of distance.



(a) Graph G_1



(b) Graph G_2

Use BFS with start vertex a to color the vertices of the graphs such that

- all vertices which are at even distance from a are assigned color C_1, and
- all vertices which are at odd distance from a are assigned color C_2.

If no two adjacent vertices have the same color declare the graph as bipartite. Which one of (a) or (b) is bipartite? Give the color assignments to vertices of each graph by BFS.

Solution (a): Bipartite. The color assignment for the graph is

Vertex	a	b	c	d	e	f
Color	$C1$	$C2$	$C1$	$C2$	$C1$	$C2$

So, the vertex colors are alternatively assigned $C1$ and $C2$ around the ring from a clockwise. It means we partitition the graph as (i) a, c, e, and (ii) b, d, f. There no edge between any pair of vertices belonging to a part. That is no edge exists between two vertices in set $\{a, c, e\}$. Similarly, no edge exists between two vertices in set $\{b, d, f\}$,

Solution (b): Not bipartite.

Vertex	a	b	c	d	e
Color	$C1$	$C2$	$C1$	$C1$	$C2$

There is no such partition possible as there is an odd cycle. So graph is not bipartite. The vertex pair c and d have same color and they are adjacent.