DSAT (Jan - April 2025)

Minor Answers

Instructions:

1. *In questions asking for algorithm you should write pseudocode.*
2. *All questions are of* 5 *marks, but they are not of equal difficulty level.*
3. *Write answers on paper, take a snapshot, and upload the soft-copy in PDF form.*

1. State which of the following statements are true and which are false:

a) Deleting a red node in an RB-tree does not cause any violation of RB-tree properties.
False. This was tricky as z is red but y can be black as discussed in the lectures.

b) In an RB-tree no path's length from the root to the leaf is more than twice the length of any other path from the root to the leaf.
True as discussed in the lectures.

c) Inserting a new node in an RB-tree will always cause a violation of some RB-tree property.
False as discussed in the lectures.

d) Deleting a black node from an RB-tree will always result in violation of the 5th property.
False as discussed in the lectures.

e) In an RB-tree, a red node cannot have exactly one non-NIL child.
True. What colour can you give to the only non-NIL child of such a red node? It cannot be red as it will violate property 5 and it can also not be black as it will violate black height consistency.

Marking scheme: 1 mark for every right answer.

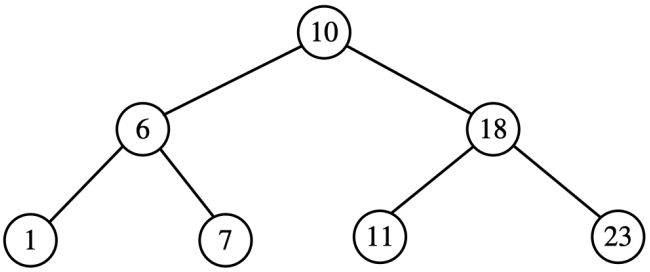
2. Consider inserting the keys 34,21,11,45,8,7,58,99 into a hash table of length $m = 11$ using open addressing. Give the final hash-table after inserting these keys in the given order using double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \,\%\, (m - 1))$.
Solution:

Slot	Element
0	11
1	34
2	NIL
3	58

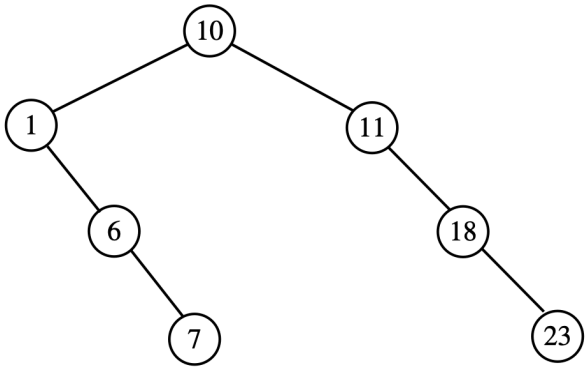
Slot	Element
4	7
5	NIL
6	NIL
7	45
8	8
9	99
10	21

Marking scheme: Partial marks for partial right answers. For instance, if you have inserted first four keys in the right slot, you will get 2.5 marks.

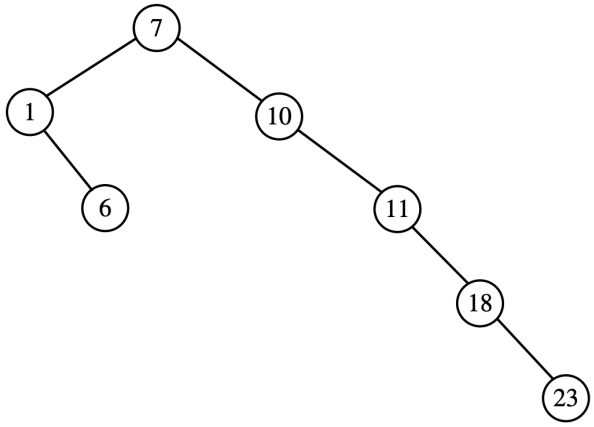
3. For the set {1,6,7,10,11,18,23} of keys, draw BSTs of heights 2, 3, 4, 5, and 6.
Solution:
BST of height 2:



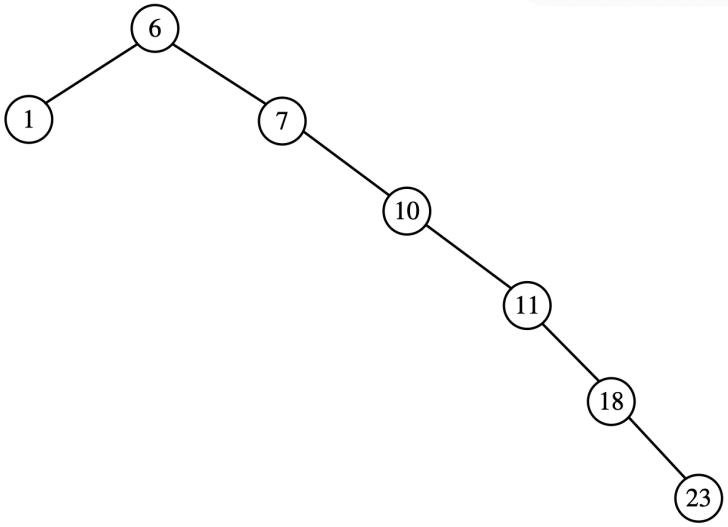
BST of height 3:



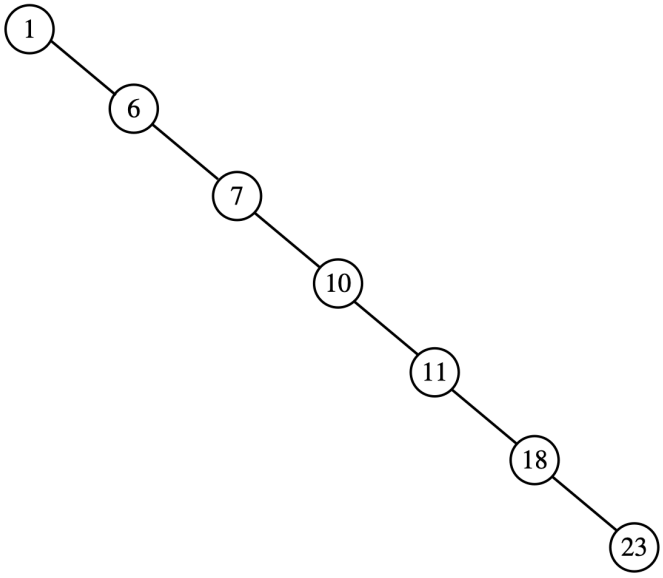
BST of height 4:



BST of height 5:

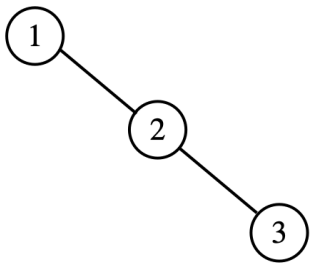


BST of height 6:

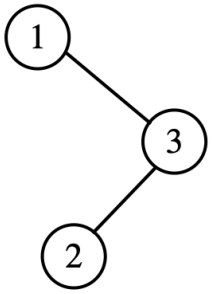


Marking Scheme: 1 mark for every right answer. No partial marking.

4. (a) (2.5 marks) Consider a BST T. Let x and y be two new elements with distinct keys. Is it the case that the BST we get after inserting x and then y is the same as the BST we get after inserting y and then x? Either argue for it or give a counterexample. (Two BSTs are considered different if some node in both the BSTs has different left or right child.)
Solution: The statement is false. Here's a counterexample. Consider a tree with just one node with key 1. Now if you insert 2 and then 3, the BST will look like below:

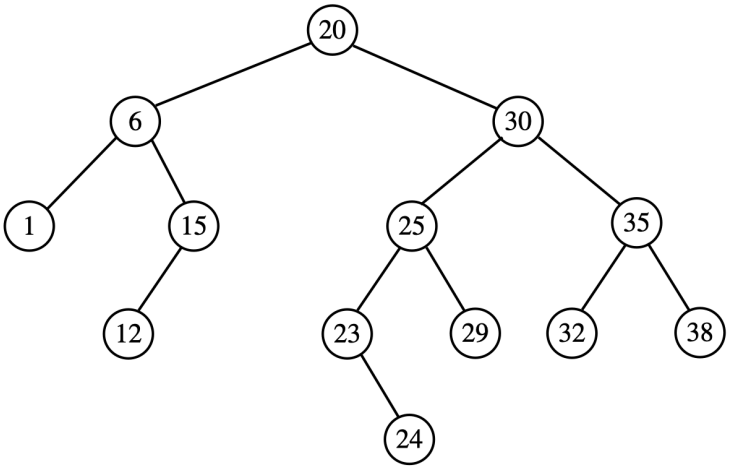


On the other hand, if you insert 3 and then 2, then it will look like below.

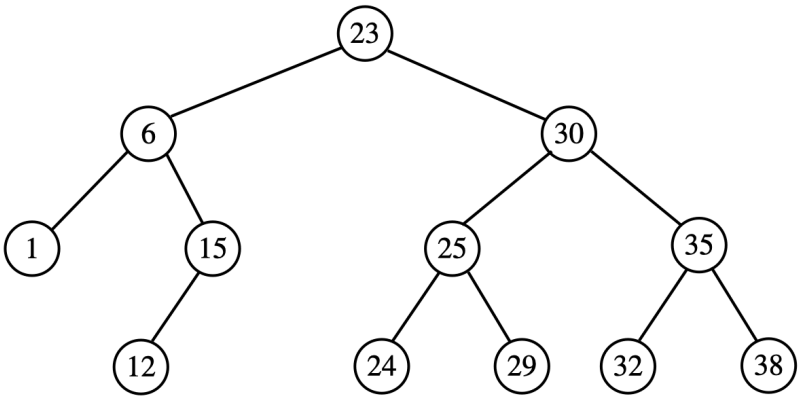


Marking Scheme: 1 mark for saying the statement is false. 1.5 for providing the right counterexample.

(b) (2.5 marks) Show the resulting BST after deleting 20 from the below BST.



Solution:



Marking Scheme: Full marks only when you have drawn the right tree. No marks will be deducted if you considered 21 as the left child of 23 instead of taking 24 as the right child.

5. Describe an algorithm that, given integers in the range 0 to k, preprocesses its input and then answers any query about how many of the integers fall into a range $[a, b]$ in $O(1)$ time. Your algorithm should use $O(n + k)$ preprocessing time and $O(k)$ space. Write complete pseudocode and give informal proof of correctness. (*Hint: Use hashing.*)
Solution: The following function runs in $O(n + k)$ time and does the preprocessing.
Function($A[1 \ldots n]$**)**

create an array $freq[0 \ldots k]$ and initialise every element with 0.
for $i = 1$ **to** n
    $freq[A[i]] = freq[A[i]] + 1$
for $i = 1$ **to** k
    $freq[i] = freq[i-1] + freq[i]$

In the above function, in the first loop we are calculating the frequency of every element in the array A. In the second loop, we are calculating how many integers are smaller than or equal to i in $freq[i]$ by doing cumulative sum. The two loops take $O(n + k)$ time and $O(k)$ space (size of $freq$). After this preprocessing we can compute the number of integers falling in the range of $[a, b]$ by simply outputting $freq[b] - freq[a-1]$. If $a = 0$, we can output $freq[b]$ only.

Marking Scheme: 3 marks for pseudocode, 2 marks for informal proof of correctness or explanation.

6. Given an unsorted array A of distinct elements, you have to reorder it such that $A[0] < A[1] > A[2] < A[3] > A[4] < A[5] > A[6] \ldots\ldots$ For instance, for input

[3,1,0,4,8,6,10] one possible reordering will be [1,3,0,8,4,10,6]. Give an $O(n)$ time algorithm with a casual argument for its correctness.
Solution: The following function runs in $O(n)$ time and gives the desired output.
Function(A)

for $i = 0$ **to** $n - 2$
if (i is even **and** $A[i] >$ A[I+1])
swap $A[i]$ and $A[i + 1]$
else if (i is odd **and** $A[i] < A[i + 1]$)
swap $A[i]$ and $A[i + 1]$

We claim that at the end of the ith iteration subarray $A[0 \ldots (i + 1)]$ is in the desired order. We can prove it using induction. Claim is certainly true for $i = 0$. Now take some $i = k$, assuming that $A[0 \ldots k]$ was set in the right order at the end of the $(k - 1)$th iteration. Assume further without loss of generality that k is even. Since k is even $A[k - 1] > A[k]$. Now in the kth iteration, if $A[k] > A[k + 1]$ then moving $A[k]$ to $(k + 1)$th index and moving $A[k + 1]$ to kth position is perfectly fine as $A[k - 1] > A[k + 1]$. The first "if condition" does just that. If $A[k] < A[k + 1]$, then nothing needs to be done. Hence, at the end of the kth iteration, subarray $A[0 \ldots k + 1]$ in the right order. The argument can be made for odd k.

The basic idea is that doing local swaps to get the right order works because it does not disturb the previous inequalities.

Marking Scheme: 2.5 marks for pseudocode, 2.5 marks for informal proof of correctness or explanation.