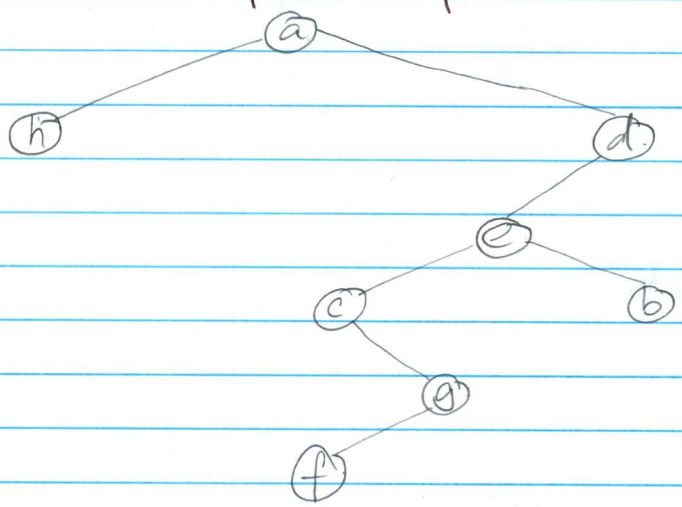
YUQIAN ZHANG # N19965556 Recitation: Ojas

Fundamental Algorithms Problem Set 6

Q1. BST (continuation of Problem from Last Assignment)



a) The successor of c is f.

According to the program SUCCESSOR, x is c

if (x.right ≠ NIL) is true, where x.right is g

then return TREE-MINIMUM (g), which is f

b). The minimal element is h.

According to the program MIN, Root x is the given node a

① while x.left ≠ NIL (a.left is h)

x = x.left here x is h

② x.left = NIL (h.left is NIL)

Return x Thus, h is the minimal element

c) Delete [e]

According to the TREE Delete program (T, e)

Because e.left ≠ NIL and e.right ≠ NIL, y = TREE-MINIMUM(e.right)

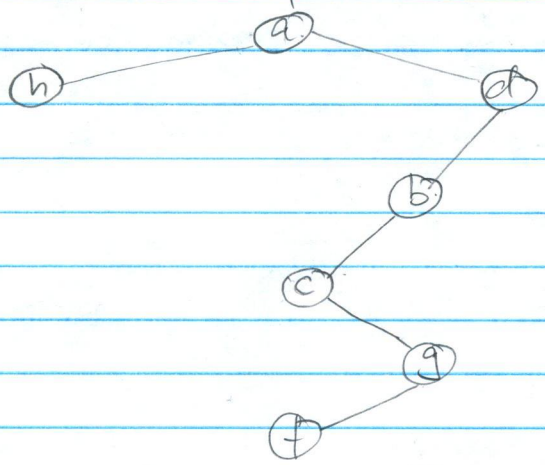
y = b, y.p = e

Transplant (T, e, b) b.left = e.left

b.left.p = b sets new parent to c

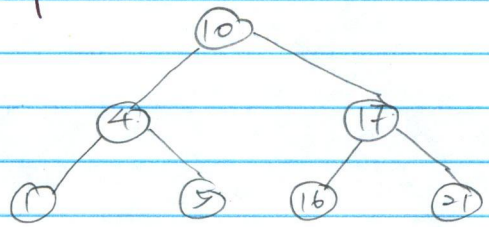
(continue on the next page)

Thus, after the deletion of Delete [e], the BST will be.

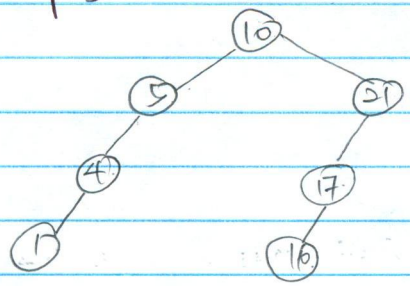


Q2. Draw BST of height 2, 3, 4, 5, 6 on the set of keys {1, 4, 5, 10, 16, 17, 21}

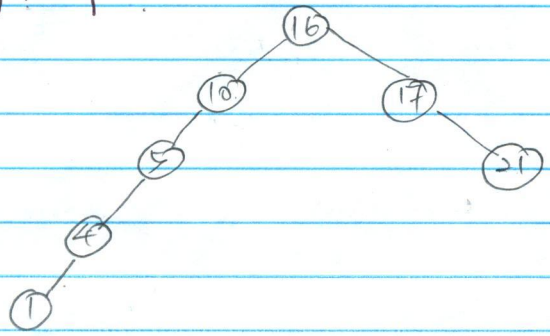
height of 2.



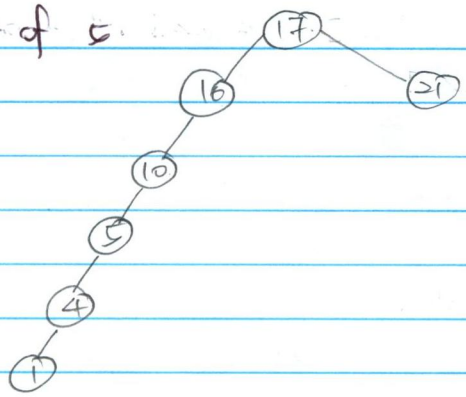
height of 3



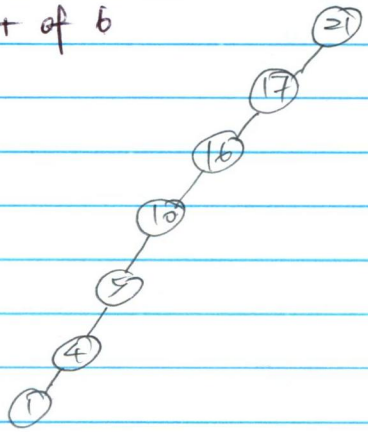
height of 4



height of 5



height of 6



Q3. What's the difference between the BST and heap property?

Both BST and heap are semi sorted algorithm. The BST can print out the keys of an n-node tree in sorted order in $O(n)$ time, but heap cannot. Because any node x, any node y in left tree has key(y) ≤ key(x), any node z in right tree has key(z) ≥ key(x)

eg In Max heap, it is promised that the parent node is greater than its children. However, it's not promised that left child is certainly less than right child. Thus, using heap-sort to print out the keys of an n-node in sorted order takes at least $O(n \lg n)$ run time

In addition, heap-sort is limited to print inorder only. The BST can print preorder and postorder as well

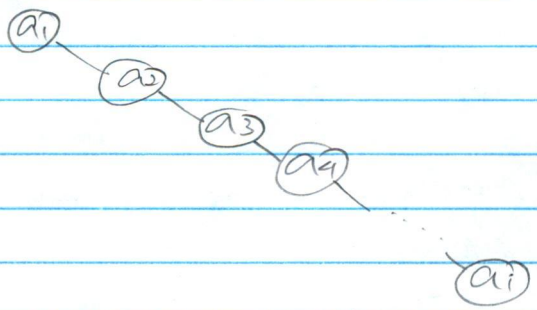
Q4. Given an array A[1...n], a hash function h: Ω → {1,... n} and a Table T[1...n] of linked lists, initially all empty.

Algorithm: For i from 1 to n Of array A {
h(i, T);
if (there is collision of h[key(i)]) {
Search collision chaining linked list;
if (there is a duplicate)
return BAD;
}
}
return GOOD

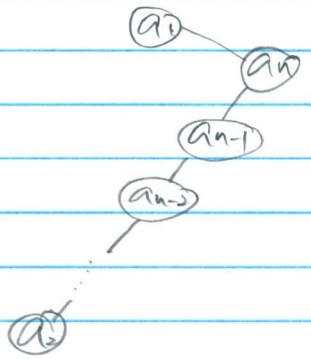
[Assumption①] Assumption is the hash function is uniform distributed

Under assumption that calculating the hash function takes unit time. The time of the Algorithm is $O(n)$.

Q5. BST root: a_1 $key[a_i] = 1$ $key[a_i] = i$ for $2 \le i \le n$.
Insert $[a_2]$... , Insert $[a_n]$



Insert $[a_n]$... , Insert $[a_2]$



Q4 continued: [Assumption ②]

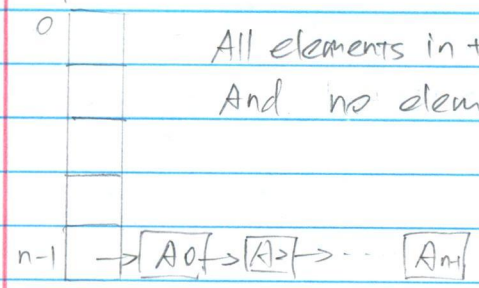
Assumption is the hash function is not uniform distributed

```
For i from 1 to n of array A {
        h(i, T):
        if there is a collision of h[key(i)] {
            Search collision chaining linked list;
            if (there is a duplicate)
                return BAD;
        }
}
return GOOD
```

The worst case analyze:

hash Table T

Array A



For the hash function, it is $\Theta(1)$ each time

Thus, the time complexity is,

$1+2+3+\cdots n-1 = \frac{n(n-1)}{2} \sim O(n^2)$