EL9343 Homework 07

Spring 2022

Name: Sagar Patel
NETID: SP5894

Question 01

Suppose that we wish to implement a dynamic, open-address hash table. Why might we consider the table to be full when its load factor reaches some value α that is strictly less than 1? Describe briefly how to make insertion into a dynamic, open-address hash table run in such a way that the expected value of the amortized cost per insertion is $O(1)$. Why is the expected value of the actual cost per insertion not necessarily $O(1)$ for all insertions? (Exercise 17.4-1 in CLRS Textbook, page 471)

Solution

A. For a static open-address hash table, we need to note that by Theorem 11.6, if the load factor α is euqal to 1, the number of probes in an unsuccessful search may reach infinity. Therefore, we must assure that α is strictly less than 1.
Theorem 11.6 states that − "Given an open-address hash table with a load factor $\alpha = \frac{n}{m} < 1$, the expected number of probes in an unsuccessful search is at most $\frac{1}{1-\alpha}$, assuming uniform hashing"

B. For a static open-address hash table, by Theorem 11.7, as long as we can ensure that table expansion does not change the amortized worst case behavior of insertion, we are guaranteed $O(1)$ insertion.
Theorem 11.7 states that − "Inserting an element into an open-address hash table with a load factor α requires at most $\frac{1}{1-\alpha}$ probes on average, assuming uniform hashing"
To keep insertion time reasonable, insertion into a dynamic open-address hash table can be made to run in $O(1)$ time by expanding when $\alpha \geq 0.75$ and contracting when $\alpha \leq 0.25$

If the table is at least half full − $\phi_i = \frac{8}{3} num_i - size_i$
If the table is less than half full − $\phi_i = \frac{1}{2} size_i - num_i$

The expansion factor and contraction factor of table size is 2 and $\frac{1}{2}$ respectively. When the load factor is less than $\frac{1}{4}$, the proof of deletion operation is the same. Our objective is to prove that the amortized cost when we expand the hash table is still $O(1)$.
$num_i = num_{i-1} + 1$
$size_i = 2.size_{i-1}$
$num_{i-1} = \frac{3}{4} size_{i-1}$

Therefore, $\alpha_i = c_i + \phi_i - \phi_{i-1}$
$\Rightarrow \alpha_i = (num_{i-1} + 1) + (\frac{1}{2} size_i - num_i) - (\frac{8}{3} num_{i-1} - size_{i-1})$
$\Rightarrow \alpha_i = (num_{i-1} + 1) + 2.size_{i-1} - \frac{11}{3} num_{i-1} - 1$
$\Rightarrow \alpha_i = (num_{i-1} + 1) + \frac{8}{3} num_{i-1} - \frac{11}{3} num_{i-1} - 1$
$\Rightarrow \alpha_i = 0$
$\Rightarrow \alpha_i = O(1)$

C. The expected value of the actual cost per insertion is not necessarily $O(1)$ for all insertions because the cost of inserting the m-th element into a "full" table costs $O(m)$, since all $m - 1$ items must be copied into the new table before the new element can be inserted.

Question 02

Write pseudocode for RIGHT-ROTATE. (Exercise 13.2-1 in CLRS Textbook, page 313)

Solution

The pseudocode is as follows −

```
RIGHT-ROTATE(T, Y)
    X = Y.left
    Y.left = X.right
    if (X.right != T.nil):
        X.right.p = Y
    X.p = Y.p
    if (Y.p == T.nil):
        T.root = X
    elif (Y == Y.p.right):
        Y.p.right = X
    else:
        Y.p.left = X
    X.right = Y
    Y.p = X
```

Question 03

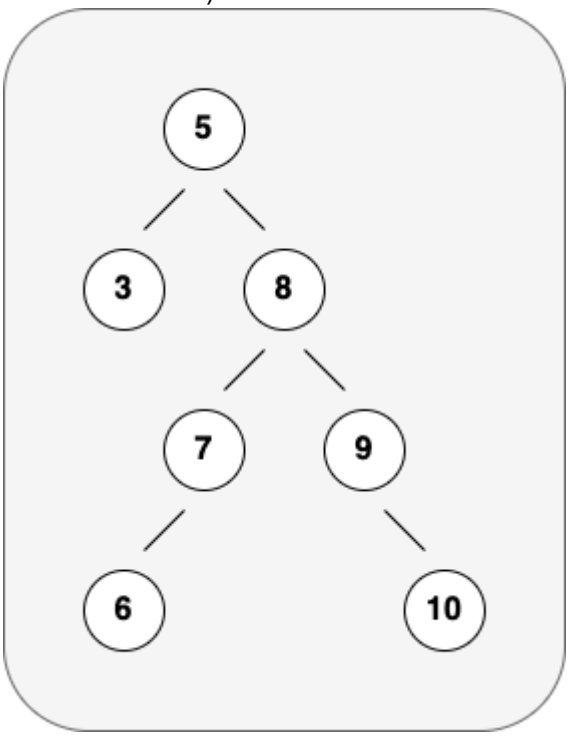
Demonstrate what happens when we insert the keys $[5, 28, 19, 15, 20, 33, 12, 17, 10]$ into a hash table with collisions resolved by chaining. Let the table have 9 slots and let the hash function be $h(k) = k. mod. 9.$

Solution

h(k)	keys
0	
1	10 -> 19 -> 28
2	20
3	12
4	
5	5
6	33 -> 15
7	
8	17

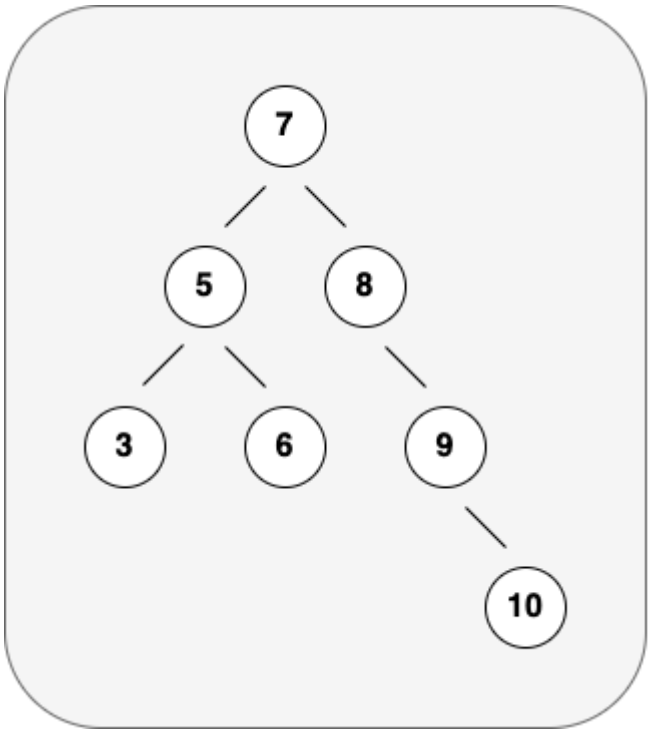
Question 04

Building an AVL Tree out of the Binary Search Tree according to the rotation operations in the lecture. (You can simply give the final result.)

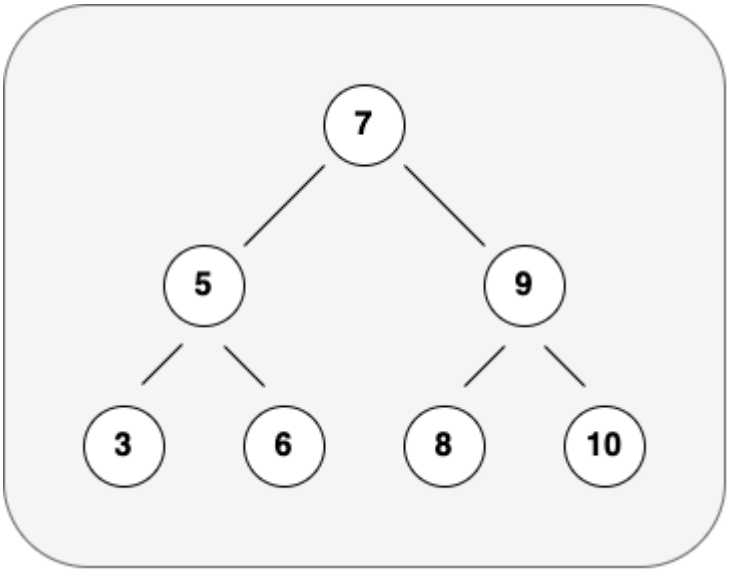


Solution

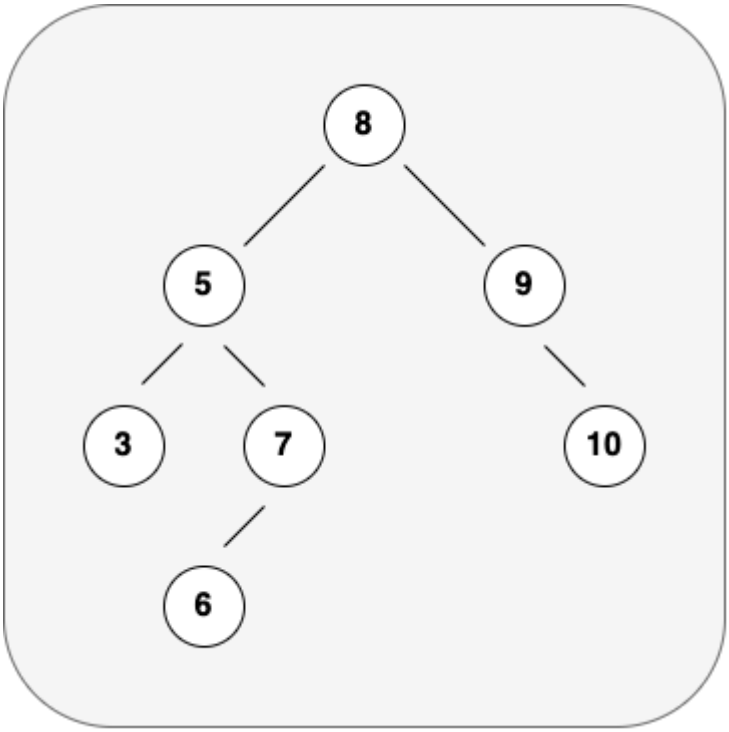
-- A double rotation + A single rotation
After double rotation —



Then a single rotation —



-- A single rotation —


8
5
9
3
7
10
6