CSCE 500 Homework Assignment #2

Assigned: October 18, 2021
Due immediately after the exam on October 25, 2021

Work on the following exercise problems:
(1) **11.4-5** (pp. 277)

11.4-5 ★
Consider an open-address hash table with a load factor α. Find the nonzero value α for which the expected number of probes in an unsuccessful search equals twice the expected number of probes in a successful search. Use the upper bounds given by Theorems 11.6 and 11.8 for these expected numbers of probes.

Solution: $\frac{1}{1-\alpha} = 2\left(\frac{1}{\alpha}\ln\frac{1}{1-\alpha}\right)$, *solving numerically*, $\alpha \approx 0.7153319$

Theorem 11.6
Given an open-address hash table with load factor $\alpha = n/m < 1$, the expected number of probes in an unsuccessful search is at most $1/(1-\alpha)$, assuming uniform hashing.

Theorem 11.8
Given an open-address hash table with load factor $\alpha < 1$, the expected number of probes in a successful search is at most

$$\frac{1}{\alpha}\ln\frac{1}{1-\alpha},$$

assuming uniform hashing and assuming that each key in the table is equally likely to be searched for.

Proof A search for a key k reproduces the same probe sequence as when the element with key k was inserted. By Corollary 11.7, if k was the $(i+1)$st key inserted into the hash table, the expected number of probes made in a search for k is at most $1/(1-i/m) = m/(m-i)$. Averaging over all n keys in the hash table gives us the expected number of probes in a successful search:

$$\begin{aligned}
\frac{1}{n}\sum_{i=0}^{n-1}\frac{m}{m-i} &= \frac{m}{n}\sum_{i=0}^{n-1}\frac{1}{m-i} \\
&= \frac{1}{\alpha}\sum_{k=m-n+1}^{m}\frac{1}{k} \\
&\le \frac{1}{\alpha}\int_{m-n}^{m}(1/x)\,dx \quad \text{(by inequality (A.12))} \\
&= \frac{1}{\alpha}\ln\frac{m}{m-n} \\
&= \frac{1}{\alpha}\ln\frac{1}{1-\alpha}. \qquad \blacksquare
\end{aligned}$$

If the hash table is half full, the expected number of probes in a successful search is less than 1.387. If the hash table is 90 percent full, the expected number of probes is less than 2.559.

(2) **12.2-5** (pp. 293)

12.2-5

Show that if a node in a binary search tree has two children, then its successor has no left child and its predecessor has no right child.

Solution:

(A) If a node z has two children, then z.left ≠ NIL and z.right ≠ NIL.

(B) Since node z has two children, z's predecessor and z's successor must be descendants of z.

(C) Since z's predecessor must be descendants of z, z's predecessor will not have a right child because if it did, then real predecessor of z would be the right child of the assumed predecessor of z.

(D) Likewise, since z's successor must be descendants of z, z's successor will not have a left child because if it did, then real successor of z would be the left child of the assumed successor of z.

(3) **12.3-6** (pp. 299)

12.3-6

When node z in TREE-DELETE has two children, we could choose node y as its predecessor rather than its successor. What other changes to TREE-DELETE would be necessary if we did so? Some have argued that a fair strategy, giving equal priority to predecessor and successor, yields better empirical performance. How might TREE-DELETE be changed to implement such a fair strategy?

Solution:

TREE-DELETE (T, z) //using predecessor

```
if z.left == NIL
    TRANSPLANT (T, z, z.right)
elseif z.right == NIL
    TRANSPLANT (T, z, z.left)
else y = TREE-MAXIMUM(z.left) // Get z's predecessor
    if y.p ≠ z
        TRANSPLANT (T, y, y.left)
        y.left = z.left
        y.left.p = y
TRANSPLANT (T, z, y)
y.right = z.right
y.right.p = y
```

(4) **18.1-4** (pp. 491)

18.1-4

As a function of the minimum degree t, what is the maximum number of keys that can be stored in a B-tree of height h?

Solution:

$$h \leq \log_t \frac{n+1}{2}$$

$$t^h \leq \frac{n+1}{2}$$

$$2t^h \leq n+1$$

$$2t^h - 1 \leq n$$

For the minimum degree t, $t = 2$.

$$2 \cdot 2^h - 1 \leq n$$

$$2^{h+1} - 1 \leq n$$

Therefore, the maximum number of keys, n, that can be stored is $2^{h+1} - 1$.

(5) **18.2-1** (pp. 497)

18.2-1

Show the results of inserting the keys

$F, S, Q, K, C, L, H, T, V, W, M, R, N, P, A, B, X, Y, D, Z, E$

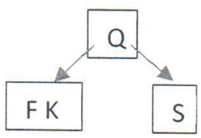
in order into an empty B-tree with minimum degree 2. Draw only the configurations of the tree just before some node must split, and also draw the final configuration.

Adding F

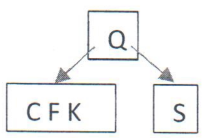
Adding S

Adding Q

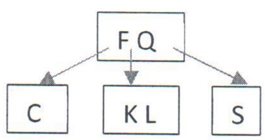
Adding K



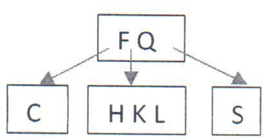
Adding C



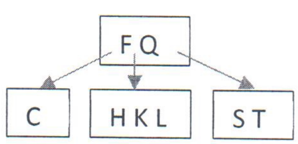
Adding L



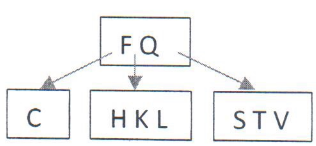
Adding H



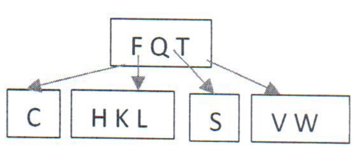
Adding T



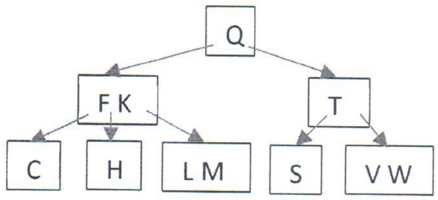
Adding V



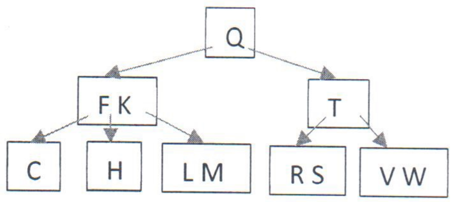
Adding W



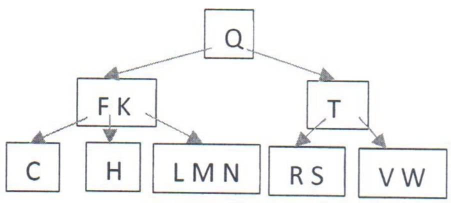
Adding M



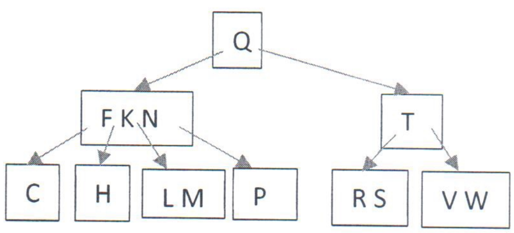
Adding R



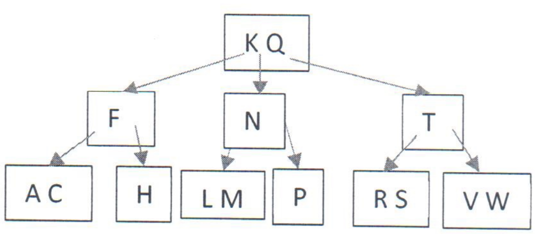
Adding N



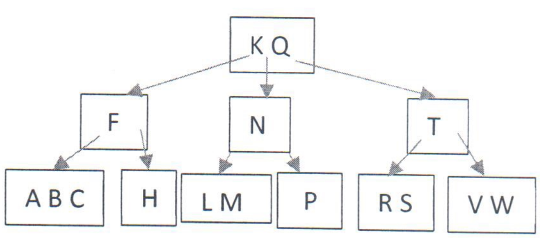
Adding P



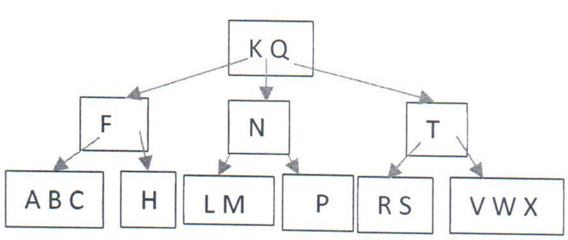
Adding A



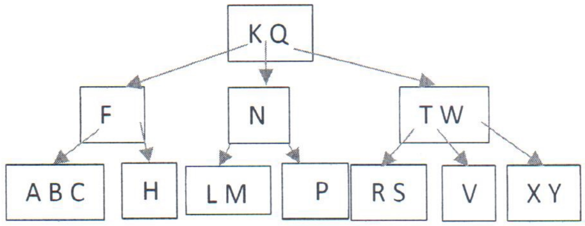
Adding B



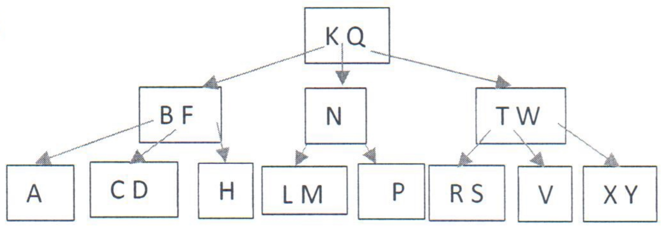
Adding X



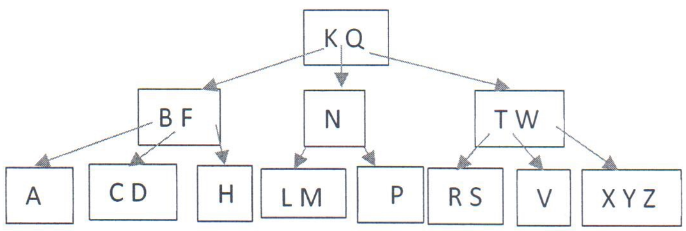
Adding Y



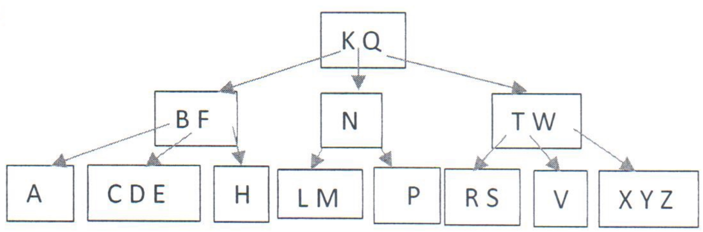
Adding D



Adding Z



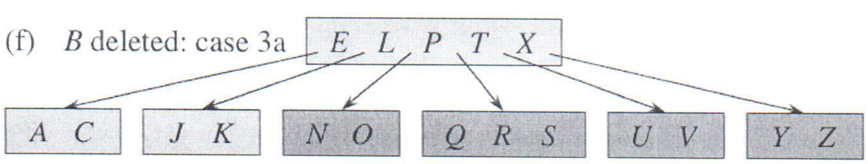
Adding E



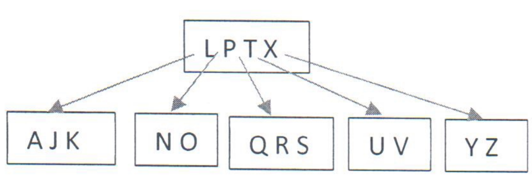
18.3-1

Show the results of deleting C, P, and V, in order, from the tree of Figure 18.8(f).

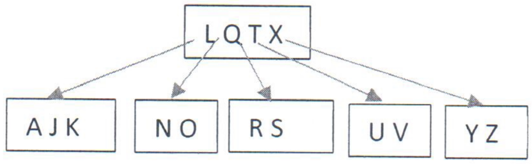
(f) B deleted: case 3a



Deleting C



Deleting P



Deleting V