# CS3343 Analysis of Algorithms Fall 2017

## Homework 5

Due 10/22/17 before 11:59pm (Central Time)

### 1. Hash Table Probabilities (3 points)

(1) (1 point) Suppose 2 keys are inserted into an empty hash table with $m$ slots. Assuming simple uniform hashing, what is the probability of:

(a) exactly 0 collisions occurring

1 chance of no collision on first insertion and $(m-1)/m$ chance on second insertion, so probability of 0 collisions is $\frac{m-1}{m}$

(b) exactly 1 collisions occurring

1 chance of no collision on first insertion and $1/m$ chance of only collision on second insertion, so probability of 1 collision is $\frac{1}{m}$

(2) (2 points) Suppose 3 keys are inserted into an empty hash table with $m$ slots. Assuming simple uniform hashing, what is the probability of:

(a) exactly 0 collisions occurring

1 chance of no collision on first insertion, $(m-1)/m$ chance on second insertion, and $(m-2)/m$ on third insertion, so probability is $1 \cdot \frac{m-1}{m} \cdot \frac{m-2}{m} = \frac{m^2-3m+2}{m^2}$

(b) exactly 1 collisions occurring

1 chance of no collision on first insertion, now two cases of only collision occurring, either on second or third insertion. Chance of collision on second and not on third is $1 \cdot \frac{1}{m} \cdot \frac{m-2}{m} = \frac{m-2}{m^2}$. Chance of collision on third but not on second is $1 \cdot \frac{m-1}{m} \cdot \frac{2}{m} = \frac{2m-2}{m^2}$. Total chance of 1 collision is $\frac{m-2}{m^2} + \frac{2m-2}{m^2} = \frac{3m-4}{m^2}$

(c) exactly 2 collisions occurring

1 chance of no collision on first insertion, $1/m$ chance of first collision on second insertion, and $2/m$ chance of second collision on third insertion so probability of 2 collisions is $1 \cdot \frac{1}{m} \cdot \frac{2}{m} = \frac{2}{m^2}$

### 2. Red-Black Trees (2 points)

(1) Company X has created a new variant on red-black trees which also uses blue as a color for the nodes. They call these "red-black-blue trees". Below are the new rules for these trees:

- Every node is red, blue, or black.
- The root is black.
- Every leaf (NIL) is black.
- If a node is red, then both its children are black.
- If a node is blue, then both its children are red or black.
- For each node, all simple paths from the node to descendant leaves contain the same number of black nodes.

(a) (2 points) In class we found that the height, $h$, of a red-black tree is $\leq 2\log_2(n+1)$ (where $n$ is the number of keys). Find and prove that a similar bound on height of the red-black-blue trees.
(Hint: You can use the same approach as we did to show $h \leq 2\log_2(n+1)$).

The number of nodes that are black on any simple path from the root to a leaf is at least $h/3$ implied by property 4 and 5. It can be shown that a subtree rooted at any node $x$ contains at least $2^{bh(x)} - 1$ internal nodes. Using the root of the tree as $x$ we know that the number of nodes $n \geq 2^{h/3} - 1 \rightarrow n + 1 \geq 2^{h/3} \rightarrow \log_2(n+1) \geq h/3 \rightarrow 3\log_2(n+1) \geq h$. Thus $h \leq 3\log_2(n+1)$

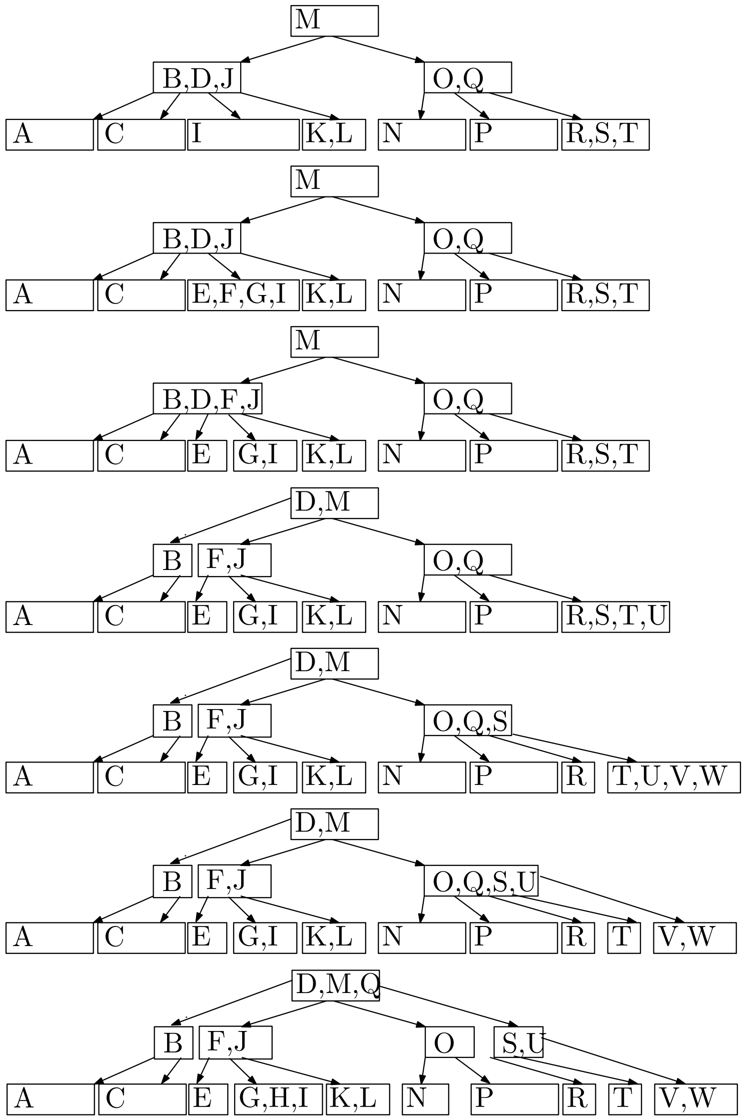
(b) (0 points - just for fun) Adding an additional color didn't seem to improve our bound on $h$ (i.e., 3 colors allows the tree to become more unbalanced than with 2 colors). What benefit might we get from the extra color?
Insertion would be easier and run in $O(1)$ time since it would only require a change in color to complete the insertion rather than $O(log(n))$

**3. B-trees (4 points)**

(1) (2 points) Show the results of inserting the keys

$E, F, G, U, V, W, H$

in order into the B-tree shown below. Assume this B-tree has minimum degree $k = 2$. Draw only the configurations of the tree just before some node(s) must split, and also draw the final configuration.




(2) (2 points) Suppose you have a B-tree of height $h$ and minimum degree $k$. What is the largest number of keys that can be stored in such a B-tree? Prove that your answer is correct.
(Hint: Your answer should depend on $k$ and $h$. This is similar to theorem we proved in the B-tree notes).

| level | num nodes | num keys |
|---|---|---|
| 0 | 1 | $2k-1$ |
| 1 | $2k$ | $2k(2k-1)$ |
| 2 | $(2k)^2$ | $(2k)^2(2k-1)$ |
| $\vdots$ | $\vdots$ | $\vdots$ |
| h | $(2k)^h$ | $(2k)^h(2k-1)$ |

This table shows the maximum number of keys for a B-tree of degree $k$ and height $h$ at each given level. Summing the keys gives:

$$\begin{aligned}
\sum_{i=0}^{h}(2k)^i(2k-1) &= (2k-1)\sum_{i=0}^{h}(2k)^i \\
&= (2k-1)\frac{(2k)^{h+1}-1}{2k-1} \\
&= (2k)^{h+1}-1
\end{aligned}$$

So the maximum number of keys in the given B-tree is $(2k)^{h+1}-1$.

## 4. Choose Function (4 points)

Given $n$ and $k$ with $n \geq k \geq 0$, we want to compute the choose function $\binom{n}{k}$ using the following recurrence:

Base Cases: $\binom{n}{0} = 1$ and $\binom{n}{n} = 1$, for $n \geq 0$
Recursive Case: $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$, for $n > k > 0$

(1) (1 point) Compute $\binom{5}{3}$ using the above recurrence.

$$\begin{aligned}
\binom{5}{3} &= \binom{4}{2} + \binom{4}{3} \\
&= \binom{3}{1} + \binom{3}{2} + \binom{3}{2} + \binom{3}{3} \\
&= \binom{2}{0} + \binom{2}{1} + \binom{2}{1} + \binom{2}{2} + \binom{2}{1} + \binom{2}{2} + 1 \\
&= 1 + \binom{1}{0} + \binom{1}{1} + \binom{1}{0} + \binom{1}{1} + 1 + \binom{1}{0} + \binom{1}{1} + 1 + 1 \\
&= 1+1+1+1+1+1+1+1+1+1 \\
&= 10
\end{aligned}$$

(2) (2 points) Give pseudo-code for a **bottom-up** dynamic programming algorithm to compute $\binom{n}{k}$ using the above recurrence.

**Algorithm 1** int choose(int $n$, int $k$)

```
C[n+1][k+1];
for i = 0 to n do
    j = 0;
    while j ≤ i and j ≤ k do
        if j == 0 or j == i then
            C[i][j] = 1;
        else
            C[i][j] = C[i-1][j-1] + C[i-1][j];
        end if
        j++;
    end while
end for
return C[n][k];
```

(3) (1 point) Show the dynamic programming table your algorithm creates for $\binom{5}{3}$.

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 1 | | | |
| 1 | 1 | 1 | | |
| 2 | 1 | 2 | 1 | |
| 3 | 1 | 3 | 3 | 1 |
| 4 | 1 | 4 | 6 | 4 |
| 5 | 1 | 5 | 10 | 10 |