Illinois Institute of Technology
Department of Computer Science

Solutions to Homework 6

CS 430 Introduction To Algorithms
Spring Semester, 2013

1. (a) Let $T(h)$ be the minimum number of nodes in a height balanced tree of height h. We proceed by induction. For the base cases note that $T(1) \geq T(0) \geq 1$, thus $T(1) \geq F_1$ and $T(0) \geq F_0$. Now assume that $T(h') \geq F_{h'}$ for all $h' < h$. The root node in an AVL-tree of height will have two children: one with height $h-1$, and the other with height at least $h-2$. The minimum number of nodes in an AVL-tree of height can therefore be bounded in terms of $T(h-1)$ and $T(h-2)$, $T(h) \geq T(h-1) + T(h-2)$. By induction hypothesis, this implies $T(h) \geq F_{h-1} + F_{h-2} = F_h$. And we have the fact that $F_h \geq 1.6^h$, so $n \geq 1.6^h$, $h = O(\lg n)$.

   (b) The procedure BALANCE(x) is as follows.

Algorithm 1: BALANCE(x)

```
if height(x.left)− height(x.right) ≤ 1 then
|   return x ;
else if height(x.left) < height(x.right) then
|   y ← x.right ;
|   if y.left < y.right then
|   |   return left-rotate(x) ;
|   else
|   |   right-rotate(y);
|   |   return left-rotate(x);
else
|   y ← x.left ;
|   if y.right < y.left then
|   |   return right-rotate(x) ;
|   else
|   |   left-rotate(y) ;
|   |   return right-rotate(x) ;
```

   (c) The procedure AVL-INSERT(x, z) is as follows.

   (d) Since only pointers are exchanged in rotation so it takes only $O(1)$ time on rotation. The height of AVL-tree with n nodes is $O(\lg n)$. AVL-insert takes $O(\lg n)$ time to insert and balancing can take almost $O(\lg n)$ time. So AVL insert takes O(lg n) time.

2. As the expected number of nodes for each red-black tree is n/m for n keys and m hash values, the expected height of each tree is $O(\lg(n/m))$. Since the it takes time $O(1)$ to compute a hash value, the expected time for successful and unsuccessful search on the red-black tree is $O(1 + \lg(n/m))$.

Algorithm 2: AVL-INSERT(x, z)

```
if x = nil then
    height[z] ← 0
    return z
if key[z] ≤ key[x] then
    y ← AVL-INSERT(left[x], z)
    left[x] ← y
else
    y ← AVL-INSERT(right[x], z)
    right[x] ← y
parent[y] ← x
height[x] ← height[y] + 1
x ← BALANCE(x)
return x
```

3. Suppose that

$$h_1(k) + jh_2(k) = h_1(k) + ih_2(k) (\text{mod } m)$$
$$(j - i)h_2(k) = 0 (\text{mod } m)$$
$$(j - i)h_2(k) = c \cdot m \text{ where } c \text{ is some integer}$$

If $h_2(k)$ and m are not relatively prime, then the above equation holds. Thus the hash function repeats at the i-th and the j-th step, and hence the hashing function will not generate m different locations.