Problem Set 12

Justin Ely

615.202.81.FA15 Data Structures

24 November, 2015

---

1)

Assuming this question means: assuming a tree is constructed at random, show that balanced trees are more probable to be assembled than straight-line trees, then the likely hood of a given tree being constructed comes from the number of possible states that tree could take. For straight-line trees, there is only a single physical configuration that satisfies the straight-line requirement. For a balanced tree, there are numerous ways it could be constructed - making it more probable to occur.

2)

```
def delete(key1, key2):
    for node in inOrderTraverse(Tree):
        while key1 <= node.value <= key2:
            tmp = node
            del node
            node = tmp.inOrderSuccessor()
```

3)

```
def delete(record, node):
    if node.isleaf():
        del record
    else:
        del record
        rebalanceTree()
```

4)

Approximating the uniform spread of the hash with a random process, each key has a 50% chance to have collided with any of the filled spaces. Representing the filled space as the load factor, a key has a $.5(lf)$ chance of having collided. Since there will be n keys, but the first could not have collided, the total number of average collisions is: $(n-1)\frac{lf}{2}$.

5)

The maximum number of comparisons performed will be 1 for each possible space in the table, plus 1 for that last check that determines there is no more room available: $(tablesize + 1)$. Dividing this value by the number of open spots in the table then gives the value of the likely number of comparisons performed, plus 1 as a single check is always performed. With the substitution $n_{open} = tablesize - n$, the desired equation is found

$$n_{comp} = \frac{(tablesize + 1)}{(tablesize - n + 1)} \tag{1}$$

Linear probing doesn't satisfy this condition because it breaks from the randomness of the probability, and instead simply steps linearly through the table until a new position is found.