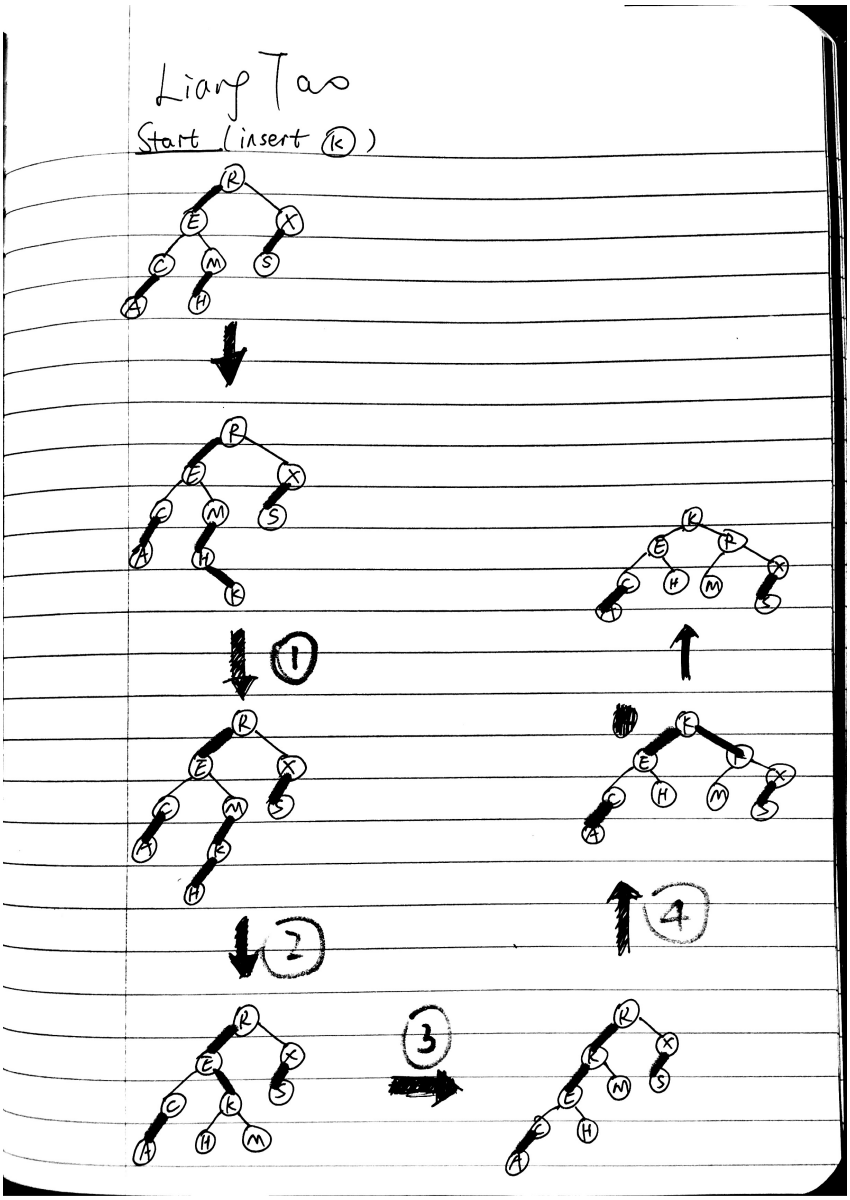
Liang Tao
CS323 hw2
February 15, 2014

Assignment 2

1. Think

Below is one specific case indicated in the question: (scanned from my written script)



2. Read

2a) Here is the description of the method they used:

Unlike path copying method which having a drawback that it uses nonlinear space usage, they take advantage of the fact that old balance information is not essential- and this way helps to avoid copy the entire access path each time an update occurs. They let each node to hold k pointers in addition to its original two (here they adopt the case that k = 1). So when attempting to add a new pointer into a new node, the program will check whether there is still an empty slot for a new pointer. If the answer is no, they copy the node, set the initial left and right pointers of the copy to their latest values, and then they also store a pointer to the copy in the latest parent of the copied one. If the parent also has no free slot, it will be copied too like which described above. The whole copying process will not stop until the root is copied or a node with a free slot is reached.

2b) From the information given by the article, the O(1) additional space per update (insertion or deletion) is bound amortized.

To prove this, the provide a definition that the nodes are partitioned into two categories: like the authors said, live and dead. Also they use the potential paradigm. Potential is defined as the number of live nodes minus 1/k times the number of free slots in live nodes. Then they defined the amortized spec cost of an update operation to be "the actual number of nodes it creates plus the net increase in potential it causes." With such definitions, the authors prove that strong a new pointer in a node has an amortized space cost of 1/k. Then because creation of a new node during an insertion has an amortized space cost of one, and an insertion or deletion requires storing O(l) new pointers not counting node copying, they prove that the amortized space cost of an update is O(1).

2c) The "fat node" means nodes described in the article can store other information including additional 1 pointer (k = 1), time stamp indicating when the change occurred and a bit that indicates whether the new pointer is a left or right pointer. So in this case "fat node" would have 3 pointers in one node.

3. Estimate

3a) average number of empty lists: $100 * e^{(-200/100)} = 13.5335$

average length of the longest list: $\log 200/(\log(\log 200)) = 2.605$

3b) average number of empty lists: $10000 * e^{(-20000/10000)} = 1353.35$

average length of the longest list: $\log 20000/(\log(\log 20000)) = 12.348$

3c) 1. for the average number of empty lists, I used the formula below:

C(number of empty lists) $= k(1 - \frac{1}{k})^n = k(1 - \frac{1}{k})^{-k(-\frac{n}{k})} = ke^{(-\frac{n}{k})}$

where k = size of the total lists

n = size of the total keys

Source: **http://www.cnblogs.com/fengfenggirl/p/hash_prob.html**

2. for the average length of the longest list, I used the formula coming from the book:

Length of the average length of the logest list: L= logN / loglogN;

Source:Algorithm 4th Edition. P466