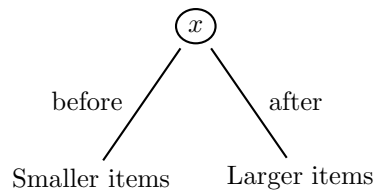


Lecture 11: October 7, 2009

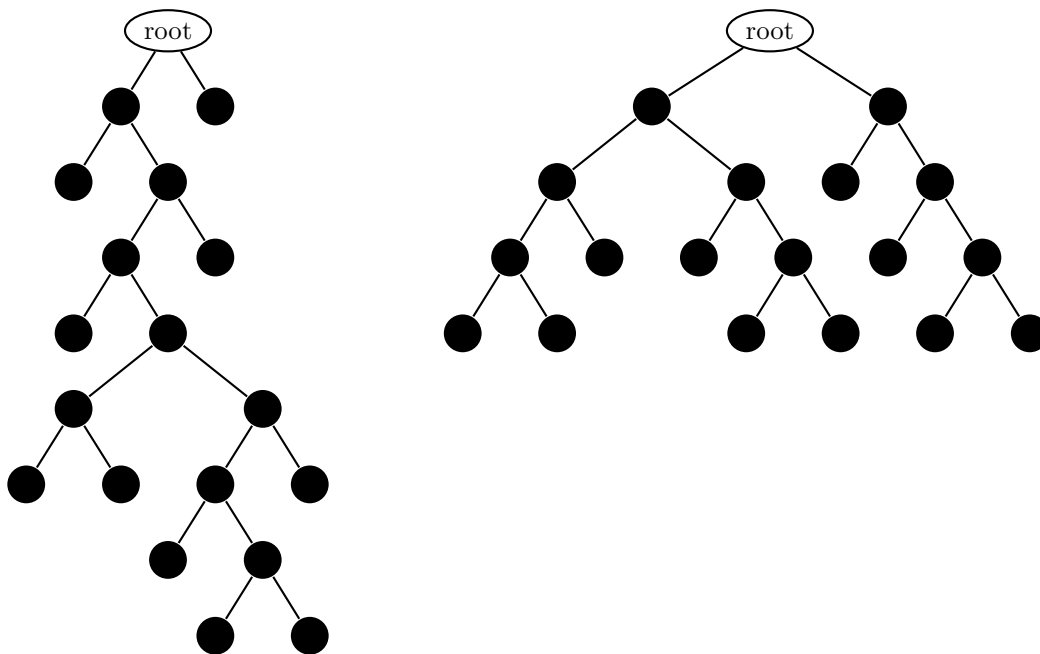
CS 330 Discrete Structures
Fall Semester, 2009

1 Probabilistically Balanced Lexicographical Trees

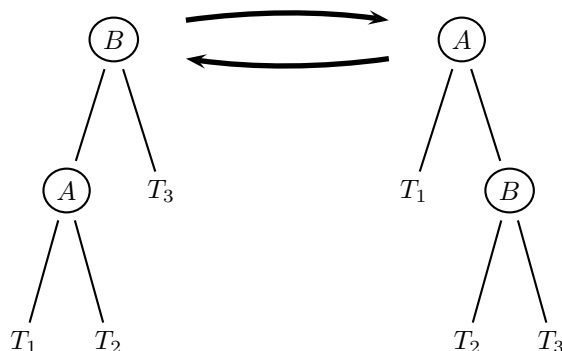
We can see computer application of Conditional Probability in probabilistically balanced lexicographical trees. A lexicographical tree is a data structure that embodies the idea of binary search. When you do a binary search, depending on the relative value of your current position and the value you are looking for, you will descend either left or right. A lexicographical tree does the same:



This rule is then applied recursively to each and every node in the tree. To search through the tree, when you visit a node you compare what you are looking for with the value store in the node. If your value is less, examine the left child, otherwise examine the right child. The maximum number of nodes examined will be the height of the tree. What we want is a tree that has a low height as opposed to a tree with a big height:



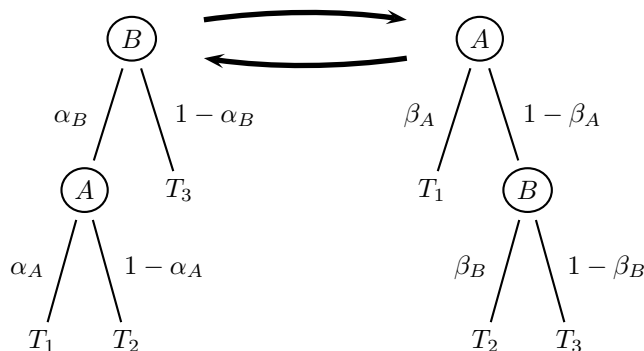
You can see that the tree on the left is a much deeper tree, so it will take longer to search that the tree on the right. In fact, if a tree is really bad, all the nodes will be in a single line. This is why it's very important to balance these trees. The basic step in tree balancing is a **rotation**. In order to rearrange the tree so it's short and fat, we apply the following transformation to certain branches:



You will notice that the lexicographical property is not affected by this transformation; in other words, we may rotate any branch in a lexicographic tree and still have a lexicographic tree.

If we are given the *probability* that, when we visit a node, we will go right, and the *probability* that, when we visit a node, we will go left, we can attempt to equate these probabilities in order to balance our tree. The problem is that after the rotation, the various probabilities will change and mess up our results. We must find a way to keep track of the probabilities of going left and the probabilities of going right, even after we rotate a branch.

We will use α and β , respectively to refer to probabilities in the two trees. We label the various probabilities concerning a rotation as follows:



We want to find the α s in terms of the β s, and vice versa. $\alpha_B = \Pr\{x < B\}$ where x is what we're searching for. We can see that for $x < B$ we need to go to either T_1 or T_2 . To go to T_2 we have to go through A and B . Using the rules of sum and product, we have

$$\alpha_B = \beta_A + (1 - \beta_A)\beta_B.$$

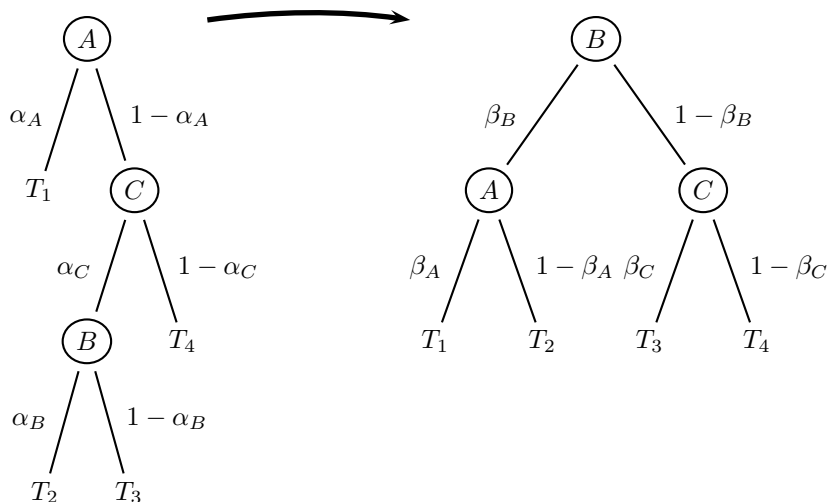
Similarly, we get:

$$\alpha_A = \Pr\{x < A | x < B\} = \frac{\Pr\{x < A \wedge x < B\}}{\Pr\{x < B\}} = \frac{\Pr\{x < A\}\Pr\{x < B | x < A\}}{\Pr\{x < B\}} = \frac{\beta_A \cdot 1}{\beta_A + (1 - \beta_A)(\beta_B)}$$

$$\beta_B = \Pr\{x < B|x > A\} = \frac{\Pr\{x < B\}\Pr\{x > A|x < B\}}{\Pr\{x > A\}} = \frac{\alpha_B(1 - \alpha_A)}{1 - \alpha_B\alpha_A}$$

$$\beta_A = \Pr\{x < A\} = \alpha_B\alpha_A$$

We also need to consider the *double rotation* and the resulting probabilities:



Again, we use Bayes' Theorem and the rules of sum and product:

$$\beta_A = \Pr\{x < A|x < B\} = \frac{\Pr\{x < A\}\Pr\{x < B|x < A\}}{\Pr\{x < B\}} = \frac{\alpha_A}{\alpha_A + (1 - \alpha_A)\alpha_B\alpha_C}$$

$$\beta_B = \Pr\{x < B\} = \alpha_A + (1 - \alpha_A)\alpha_B\alpha_C$$

To compute $\beta_C = \Pr\{x < C|x > B\}$ in terms of the α s, we need only look at the right subtree of the α -tree because $B > A$, thus $x > B$ implies $x > A$ so we take the right branch of A with probability 1:

$$\beta_C = \Pr\{x < C|x > B\} = \frac{\Pr\{x < C\}\Pr\{x > B|x < C\}}{\Pr\{x > B\}} = \frac{\alpha_C(1 - \alpha_B)}{1 - \alpha_B\alpha_C}$$

As an exercise, use the same technique to compute the α s in terms of the β s.