Tree Permutations

This problem appeared as a lemma in the work I did on dynamic clustering, and uses many different fun tools from different branches.

Definitions and Problem

Complete binary trees

As seen before, a tree is a connected graph without cycles. A binary tree is a special kind of tree which has the following properties :

- It has a special node called a root.
- All nodes have either two or zero children (hence degree 1 – a leaf – or 3).

We also want the tree to be complete, which means it also satisfies :

- All the leaves are on the same level (hence at the same distance from the root.

This means that a complete binary has 2^i nodes on the i-th level, hence a total of $\sum_{i=0}^{height} 2^i = 2^{height+1} - 1$.
From now on, we will write h for the height of the tree, and $n = 2^{h+1} - 1$ as the total number of nodes

Permutations on trees

As seen before, a permutation is a bijective function from [0;n] to [0;n], hence a reordering of the elements. We shall consider the permutations on the tree, and see it as the following :

- Each node gets an initial unique label between 1 and n.
- We apply a uniform random permutation on those labels.

The word uniform means that every possible permutation has the same probability, which implies that every node has an equal chance of getting any number between 1 and n.

Remark. In the previous line, the implication only goes one way. To prove this consider the n permutations of the kind$(1, 2...n)$; $(2, 3...n, 1)$; ... ; $(n, 1, 2, ..., n-1)$. If we take any of these with probability $1/n$, each label has an equal probability of getting any number, but the probabilities are not independent, and many possible permutations have probability 0.

However, we will actually only be interested by whether a label has a higher or lower value than another, and there's a very nice property we can use for that :
Suppose we take a uniform random permutation of labels on a set of n nodes. If we look at a subset of k nodes and look at the order there (hence who has the biggest, second biggest and so on), the probability of a node being higher than another is exactly one half. Moreover, the order on those k nodes will exactly correspond to a uniform random permutation of k labels (hence if we relabel the labels with 1 being smallest and k being biggest, we get a uniform random permutation on k labels). This is even true when we set the value of a node outside this subset. The order on the k labelsents only depends on the k labels, and nothing that could happen on the other $n-k$ labels affects it.

The Problem

We ask the following question : given a complete binary tree of height h, and a uniform permutations on its labels, what is the probability $p(h)$ of finding at least one branch (path from root to leaf) where all the labels are smaller than the root's label (we will call this the P property)?

Remark. By symmetry, this is equivalent to the question of finding a branch with all labels bigger than the root's.

Bounds

Lower Bounds

We can easily compute some lower bounds, because of a nice property :

Proposition. *$p(h)$ is a decreasing function.*

Proof. Consider a complete binary tree of height $h+1$. If we look at the subset of nodes consisting of everything but the leaves, we get a tree of height h. For every permutation that satisfies P in the big tree, we get a permutation that satisfies P in the smaller tree (because the branch from which we remove the last node is a branch in the smaller tree). However the converse is not true. This means that the probability of getting a permutation satisfying P is at most as high in the big tree as in the small (and in practice smaller because for some of the permutations we get a branch in the small tree that does not reach the leaves in the big tree. □

This means that if we can get an upper bound on the tree of height h, we know that for all bigger trees the probability will be lower than that bound. If the height is 0, the probability is 1. If the height is 1, the probability becomes 2/3. However, if we looked naively at height 2, we'd have to look at $7! = 5040$ permutations. However, we can simplify this by doing a case disjunction (splitting the problem in smaller problems). Let's consider the possible value for the label of the root.

- If it's 7, all labels are smaller and we have a branch.
- If it's 6, we can still always find a branch.
- If it's 1 or 2 we can't find a branch.

This means that with probability 2/7 we fail, and 2/7 we win (by finding a branch). Now for the harder cases :

- If it's 5, then we win, unless the root's two children are 6 and 7. Either 6 is the left or the right child, and in any case there are $4! = 24$ possibilities for the four remaining nodes. As there are $6! = 720$ possibilities, we can find a branch with probability $\frac{6!-2\times4!}{6!} = \frac{14}{15}$. However we have to count the probability of having the root's label equal to 5 – which is $\frac{1}{7}$ – hence the total is that we have a probability $\frac{2}{15}$ of winning.
- If it's 3, then the labels on the branch have to be either 1 then 2 or 2 then 1, and there are four possible branches, hence 8 possible ways of finding a branch, multiplied by the ways to arrange the remaining nodes. In total we get $\frac{8\times4!}{6!} = \frac{4}{15}$, multiplied by the probability of having 3 as the root, hence $\frac{4}{105}$.
- If it's 4, we have a complex case, and we need to refine it further.
  - If all smaller nodes are in one subtree, we win. There are $2 \times 3! \times 3!$ ways of having this (choosing which subtree has the small labels, and arranging the labels in each subtree).
  - If two of them are in one subtree, we win if and only if they are not both leaves. There are 2 ways to choose which subtree has 2 small labels, 3 ways of choosing which small labels, 2 ways to choose the order on them (which is on top), 2 ways of choosing which leaf, 3 ways of choosing where to put the last small leaf, and finally 3! ways to arrange the remaining nodes. Hence a total of $2 \times 3 \times 3 \times 2 \times 2 \times 3!$ total winning possibilities.
  - This means that the total number of winning possibilities is $(2 + 2 \times 3!) \times 3! \times 3! = 504$.

To conclude this means that the total probability of winning is

$$\frac{2}{7} + \frac{2}{15} + \frac{4}{105} + \frac{504}{5040} = \frac{39}{70} \approx 0.557$$

As we can see, this technique should not be used by hand to compute the value when the height is 3. A computer can compute that value, but no single computer can naively compute the value for $h = 4$, this is because the number of possible permutations is $(2^{h+1} - 1)!$, and for $h = 4$ a naive algorithm would need about 2×10^{35}operations (good computers can do about 10^{10} operations per second. If we took all computers on earth and made them naively compute the probability for $h = 4$, we would need 10^{15} seconds, so about 30 million years). Thankfully, we can estimate the probability easily, by just taking random permutations and checking if they have the property. Check the report at the end for experimental results.

Lower Bounds

We have found constant upper bounds, and shown that it is hard to get better exact upper bounds, so the task is now to bound as precisely as possible from beneath. We've seen that the probability decreases with h, but the question remains whether it goes towards 0, and how fast.

Trivial bound

We can first see a trivial lower bound of finding a good branch : $\frac{1}{2^{h+1}-1}$: the probability that the root has the biggest label. In fact, this can even be improved to $\frac{2}{2^{h+1}-1}$, as having one of the two biggest labels is enough.

Simple bound

Let's now consider one branch (without knowing whether the labels are in correct order). The branch has $h+1$ nodes, and as we've seen at the beginning, if we look only at the order on those nodes, we get a permutation on $h+1$ labels. This means that with probability $\frac{1}{h+1}$ the branch is a good branch (and this, no matter what branch we choose initially). The problem is that this of course doesn't work with multiple branches at the same time, as we don't have independence. It is still possible to get a slightly better result : consider two branches which only have the root in common. Then one of them is good if one of the following things happens :

- The root has the biggest label, or second biggest label (probability $\frac{2}{2h+1}$)
- The root has the $k-th$ biggest label, and all k bigger labels are on the same branch (which happens with probability that gets arbitrarily close to $\frac{1}{2h+1} \times 2^{-k}$, for fixed k and increasing h).

Hence we can get arbitrarily close to $\frac{3}{2h+1}$ (but not reach it by using this method).

$\sqrt{h}$ bound

Using the previous method and considering a set of h branches forming a sort of rake and handling the independence issues, it is possible to get a lower bound of $\frac{c}{\sqrt{h}}$ for a constant c. This is left as exercise to the reader.

Constant bound

We shall see a schematic version of a proof that the $p(h) > \frac{1}{16}$. First, let's suppose that the root has a label in the top quarter of labels. Then each time we look at other nodes, we have a probability around 3/4 of finding nodes which have smaller labels. This is very called to something called a Galton-Watson process (used to study evolution of populations and names). It turns out that if the average number of "good" children (hence with smaller label) from a node is greater than 1 (here it's around 3/2), then we have a constant probability of having arbitrarily long branches. This could work, but the problem is that we don't have independence, so each time we find a good child we diminish the future probability of finding good children, and that this is very hard to handle generally. To limit the impact of this, we only look at the tree cut at height $h/2$. This has $2^{h/2+1}-1$ nodes, so a proportion about $2^{-h/2}$ of the total number of nodes. Hence the probability will stay higher than $3/4 - 2^{-h/2}$, and everything behaves nicely. Once we have a branch going to height $h/2$, we can look at a subtree having rooted at the end of that partial branch. This means that with constant probability we have a good branch from $h/2$ to h, hence a good branch from root to leaf.

This can be improved quite substantially to show that $p(h) \to 1 - \ln(2)$.

For more information see my report or the paper on dynamic sum-radii clustering on my website www.koliaza.com.