Problem Title Description. Problem Name: The Loyalty of the Orcs Author's Name: File System Sync Problem Code: FSSYNC Alphabet: K

Problem: You are given a tree of N vertices, some of which correspond to "dead" orcs. Fix a permutation of the vertices: the order in which you perform a roll-call. You "save" a roll-call for a particular orc, if it occurs *after* the roll-call of any of its dead ancestors. What is the expected number of roll-calls that you save?

Solution: This problem asks for the expectation of some quantity. There are various approaches used to solve such problems. One might think of expectation as $E[X] = \sum_x x * P[x]$. Here, counting the number of saved roll-calls ("x") and finding the probability of this number is not so easy.

Instead, we solve this using an approach termed **Linearity of Expectation**.

$$E[\sum_i X_i] = \sum_i E[X_i]$$

The next useful concept we would like to use, is that of **Indicator Variables**. An indicator variable is associated with an event A, and takes value 1 if the event A occurs, else it takes value 0.

A standard example of where we use these two is to find the expected number of heads when tossing a coin n times, which has the probability of turning up heads with probability p.
Let indicator variable X_i be associated with the event that the i'th coin toss is a head. Let X be the random variable that counts the number of heads. $X = \sum_{i=1}^{n} X_i$.
$E[X] = E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i]$. Note that, $E[X_i] = 1 * (p) + 0 * (1 - p)$ (from the definition of expectation). Thus, $E[X] = np$ (a well-know result: The expectation of a binomial distribution B(n, p)).

In this problem, let X_v denote the event that the vertex v's roll-call is saved. Let $X = \sum_v X_v$. We are interested in finding $E[X]$. Using linearity of expectation, $E[X] = \sum_v E[X_v]$.

We now ask ourselves, given a vertex v, with what probability will the event X_v occur? (i.e. with what probability does it occur *after* all its dead ancestors?).

Let $d(v)$ = the number of dead ancestors of node v. Now,
$Prob[v$ occurs after all its dead ancestors]
$= 1 - Prob[v$ occurs before all its dead ancestors]
$= 1 - Prob[v$ occurs before $d(v)$ particular nodes]
$= 1 - \frac{d(v)!}{(d(v)+1)!} = 1 - \frac{1}{d(v)+1} = \frac{d(v)}{d(v)+1}$

There are various ways to convince yourself of the above probability being $\frac{d(v)!}{(d(v)+1)!}$. One way is given as follows: count the number of such permutations

(and then divide by $N!$). The number of such permutations = First fix $d(v)+1$ positions in which to place these $d(v)$ dead nodes as well as the node v. Now, since v comes before all the others, its position is fixed, while the others can be varied anyhow: in $d(v)!$ ways. Also, among the remaining positions, you can permute the remaining numbers as you wish, hence giving you another $(N-(d(v)+1))!$ options. So, the total number of such good permutations are: $\binom{N}{d(v)+1}\ d(v)!(N-d(v)-1)!$

This simplifies to just $N!\frac{d(v)!}{(d(v)+1)!}$, which gives the required probability when dividing by $N!$.

Finally, note that $d(v) = d(v.\text{parent})+ 1$ or 0 depending on whether v.parent is dead or not. This means that the d values can be computed in O(N) time. Hence, overall time complexity = O(N).