Subproblems

Define $F(v,i)$ for $v \in T$ and $i \in [0,k]$ such that $F(v,i)$ is the maximum number of leaves that can be chosen from the subtree T_v, given the constraint specified in the problem that every leaves are at least k distance apart and the additional constraint that every leaves are at least i from the v. The final answer is $F(r,0)$ where r is the root.

Recursive Formula

For base case, if $v \in T$ is a leaf, then $F(v,0) = 1$ (choosing itself) and $F(v,i) = 0$ for $i > 0$ (only possible choice is itself, yet the distance to itself is 0 so no possible leaves).

For recurrence, if $v \in T$ is a internal node, then we have its left child v_L and right child v_R. $F(v,i)$ is the sum of $F(v_L, j)$ and $F(v_R, \ell)$, for some j and l. We need to make sure that the constraint for $T(v,i)$ is still satisfied, therefore $j + \ell \geq k - 2$ so that two leaves in T_{v_L} and T_{v_R} still are at least k apart, and that $\min(j,\ell) \geq i - 1$ so that every leave is still i apart from v. Therefore (**the naive version**)

$$F(v,i) = \max_{\substack{j,\ell \in [0,k] \\ j+\ell \geq k-2 \\ \min(j,\ell) \geq i-1}} F(v_L, j) + F(v_R, \ell)$$

But notice that we should notice that $F(v,i)$ is monotonically non-increasing with i. (Intuitively, the cases when given constraint that at least i' away from root always include the cases when given constraint that at least away i from root given that $i' < i$) Therefore, we want to make $j + \ell = k - 2$ so that the sub cases could have as small j and k possible therefore bigger $F(v_L, j)$ and $F(v_R, \ell)$. So, a **better** recurrence formula is:

$$F(v,i) = \max_{\substack{j,\ell \in [0,k] \\ j+\ell = k-2 \\ \min(j,\ell) \geq i-1}} F(v_L, j) + F(v_R, \ell)$$

Evaluation Order

Since each $F(v,i)$ only depends on its children, we could traverse tree in **post-order** (bottom-up), in each node, we need to evaluate for all $i \in [0,k]$. (Order doesn't matter for i, but let's do it in increasing order).

Complexity

Since we need to traverse every node, so there are $O(n)$ node. In each node we need to consider j, ℓ for $O(k)$ cases (as their sum goes to $k-2$). In each case, the summation only takes $O(1)$ time. In total, it takes $O(nk)$. (If it's the naive approach, it will take $O(nk^2)$ instead). Since we save $F(v,i)$ for all node $v \in T$ and $i \in [0,k]$, it will take $O(nk)$. (Technically, we only need to save the root-to-leaf path which only have $O(h)$ node so the space complexity is $O(hk)$ but in worst case $O(h) = O(n)$, so it doesn't have effect on final space complexity)