Task Homework

Prepared by: Krešimir Nežmah and Dominik Fistrić

We can represent a valid expression by a binary tree with n leaves and $n-1$ inner nodes. The leaves correspond to question marks, and each inner node corresponds to one of the functions `max` or `min`. The problem can now be rephrased as writing a permutation of numbers from 1 to n in the leaves, and propagating these values to the root. The solution for every subtask involves parsing the string from the input and converting it to such a binary tree.

The first subtask can be solved in $O(n! \cdot n)$ by trying out all the different permutation of $\{1, 2, \ldots, n\}$, substituting it in the expression and evaluating it.

The second subtask can be solved with a bitmask dp. The state is `dp[node][mask]`, which represents the set of all possible values obtainable in this node if the allowed values are from the mask. For the transition, we try to partition the mask into two submasks (one for each child), in all possible ways. The time complexity is $O(3^n \cdot n^2)$, but in practice it is much faster because all states for which the number of ones in the mask don't match up with the number of nodes in the subtree can be discarded.

There is also a randomized solution which solves the second subtask. We make a guess that the set of obtainable numbers forms an interval (this guess will later turn out to be true). Then we try to evaluate as many different permutations as possible, and store the smallest and largest number we've come across. We declare our final output to be the length of that interval. Challenge for the reader: prove that the probability of success of such an approach is sufficiently high.

The third subtask is solved by looking at the longest sequence of nodes, starting from the root, which are of the same type (all `min` or all `max`). Let's say that there are k of them. The solution is then $n-k$. The proof is left as an excersize to the reader.

Let's look at a node and its subtree. Let S be a set of distinct numbers whose size equals the number of leaves in the subtree. The full solution requires the observation that the set of number obtainable in this node, using the numbers from S as the values for the leaves, forms an interval in S. Additionally, we must figure out a way to combine the intervals of the left and right child to get the interval for the node itself. The proof is inductive/recursive and at the same time describes a way to obtain the mentioned intervals, allowing us to solve the problem with a tree dp.

Suppose that a node has a left child with L leaves, and a right child with R leaves. Also, let the interval of possible values for the left child be $[a, b] \subseteq [1, L]$, and $[c, d] \subseteq [1, R]$ for the right child.

If the node is of type `max`, the lower limit for the interval of the node turns out to be $a + c$. Indeed, let's call the smallest $a + c - 1$ numbers from $[1, L + R]$ *small*, and the rest of them *large*. Let's say that the left tree contains x small numbers, and the right subtree contains y small numbers. Since $x + y = a + c - 1$, at least one of $x < a$ and $y < b$ must hold. With out loss of generality assume that $x < a$. Now the a-th smallest number in the left subtree is large, so the maximum of the left and right subtrees will also be large. This shows that the node value is at least $a + c$. This value is also obtainable: put the numbers $[1 + c, L + c]$ in the left subree, and the rest of them in right subtree. We can make it so that the value of the left subtree turns out to be $a + c$, and the value of the right subtree is c.

If the node is of type `min`, the lower limit will be $\min(a, c)$. This can be shown in a similar way to what is described above. This time, the small numbers will be the ones smaller than $\min(a, c)$, and the rest will be large. The same type of argument shows that the node value is at least $\min(a, c)$. If $a < c$, this can be obtained by putting $[1, L]$ in the left subtree, and $[L + 1, L + R]$ in the right subtree. The case where $a \geq c$ is similar.

We can show a similar thing for the upper limit. If the node is of type `max`, the upper limit is $\max(b + R, d + L)$, and if it is of type `min`, it is $b + d - 1$. To prove that the values in between are obtainable, we just have to slightly alter the way of distributing the numbers in the left and right subtrees, similar to the constructions for achieving the bound. The total time complexity is $O(n)$.