Problem Title Description. Problem Name: Wormtongue's Mind Author's Name: Min-Max Expression Problem Code: MINMAX Alphabet: H

Problem: Given an expression consisting of min and max operations over 'N' independent U[0, 1] random variables $x_1, x_2, \ldots, x_N$, find its expected value.

Solution: It turns out that this problem, though it looks hard, can be solved by figuring out the probability distribution of the expression.

Lets try to find the CDF (Cumulative Distribution Function) of expressions recursively. Note that we consider $x \in [0, 1]$ only.

For an expression of the form "x" (i.e. just a uniform random variable X), $F_X(x) := Prob\{X <= x\} = x$

For an expression of the form "max(expr1, expr2)" (i.e. something that looks like $X = max(X_1, X_2)$ where X_1 and X_2 are expressions),
$F_X(x) := Prob\{X <= x\}$
$= Prob\{max(X_1, X_2) <= x\}$
$= Prob\{X_1 <= x \text{ and } X_2 <= x\}$.
Now, since X_1 and X_2 consist of independent random variables, we get that
$Prob\{X_1 <= x \text{ and } X_2 <= x\}$
$= Prob\{X_1 <= x\} * Prob\{X_2 <= x\}$
$= F_{X_1}(x) * F_{X_2}(x)$

Similarly for an expression of the form "min(expr1, expr2)". Let $X = min(X_1, X_2)$. Now, $F_X(x) = Prob\{min(X_1, X_2) <= x\} = Prob\{X_1 <= x$ OR $X_2 <= x\}$. In terms of sets, this becomes $Prob(\{X_1 <= x\} \bigcup \{X_2 <= x\})$. Finally, using $Prob(A \bigcup B) = Prob(A) + Prob(B) - Prob(A \bigcap B)$, along with (as in the case of max) the fact that the random variables are independent, we get $F_X(x) = F_{X_1}(x) + F_{X_2}(x) - F_{X_1}(x) * F_{X_2}(x)$.

Thus, we notice that in all cases, the distribution turns out to be some *polynomial* in x. From here, finding the expected value can be got by integration.

Recall that $\int x^n = \frac{1}{n+1} x^{n+1}$ (ignoring constants of integration etc).
Also, $E[X] = \int_0^1 x f_X(x) dx$, where $f_X(x)$ is the probability density function of X, and is dF_X/dx. Thus, if $F_X = \sum_{i=0}^{k} c_i x^i$, then, $x f_X = \sum_{i=1}^{k} i c_i x^i$, and hence the integral (with limits from 0 to 1) would be $\sum_{i=1}^{k} \frac{i}{i+1} c_i x^i$.

Alternately, once you have the CDFs, then you can also use $E[X] = \int_0^\infty Prob(X >= x) dx$, which holds whenever X is a non-negative random variable. In this case, this is
$E[X] = \int_0^1 (1 - F_X(x)) dx$
$= 1 - \int_0^1 F_X dx$

Note on Implementation: You are given the input in the form of a preorder traversal of the expression tree. It would be good to actually build a tree out of this, and store "cdfs" related to each node (which corresponds to an "expression")

Also, there is heavy use of Polynomials. Hence, it is also advised to use a Polynomial class imbued with operations of "+", "-" and "*". Finally, with the given constraints ($N <= |S|/2$ where S is the input string length), we get that degree of polynomial is linear in number of random variables, and hence O(N^2) per polynomial multiplication is good enough. Polynomials can be stored using an array of 64-bit integers that store the coefficients.

Finally, due to precision requirements, using a double (even for the final calculation) is not good enough (atleast by this approach of calculating polynomials etc.). Hence it was specified to use long double and long long datatypes. In Java, BigDecimal solution passes.