Hacker Cup 2018 Qualification Round Solutions

FACEBOOK HACKER CUP · TUESDAY, 10 JULY 2018

Here are the solutions to the Hacker Cup 2018 Qualification Round problems. If you had a rejected solution and want to find out where you went wrong, read on and download the official input and output!

Input / Output / Solutions:
https://www.dropbox.com/sh/h68xcs9kniokeiu/AAALoB9pZq0FoLSs-nKBClzKa?dl=0

The problems in this round were written by Jacob Plachta. Test data was prepared by Wesley May.

Tourist

Overall, Alex will want to first see attractions `1` to `N` in order, then see them all a second time in the same order, then all a third time in the same order, and so on. If we write out an infinite sequence of attractions `1, 2, ..., N-1, N, 1, 2, ..., N-1, N, 1, ...`, then Alex will see the first `K` attractions in this list on his first visit, the next `K` on his second visit, and so on. This means that, on his `V` th visit, Alex will see the `(K*(V-1)+i)` th attraction in the list for `i=1..K`. We can convert each of these values into its corresponding attraction index between `1` and `N` using modulus, sort these indices in increasing order, and then output their corresponding attraction names in that order.

Interception

Writing out how the polynomial will be evaluated once the order of operations is reversed, we end up with a series of `N+2` terms all exponentiated together (right-associatively): `(P_N * x) ^ ((N * P_{N-1}) * x) ^ ... ^ ((2 + P_1) * x) ^ ((1 + P_0) * x) ^ (0)`. The only way for such an expression to potentially evaluate to `0` is if the first term is equal to `0`, as `a^b ≠ 0` when `a` is a non-zero real number (unless `b = -infinity`, which isn't possible here). Since `P_N` is guaranteed to be non-zero, this means that the polynomial can only possibly evaluate to `0` when `x = 0`.

Now, when `x = 0`, it's clear that each of the `N+2` terms above is also equal to `0`. So, we're interested in evaluating the expression `0^0^...^0^0`, with `N+2` `0`'s. We can observe an alternating pattern based on the number of `0`'s: `0^0 = 1`, `0^0^0 = 0^1 = 0`, `0^0^0^0 = 1`, `0^0^0^0^0 = 0`, and so on. So, this expression evaluates to `0` when the number of `0`'s is odd, and to `1` when the number of `0`'s is even. It follows that the polynomial has a single x-intercept at `x = 0` when `N` is odd, and no x-intercepts when `N` is even, independent of its coefficients.

Ethan Searches for a String

If an occurrence of `A` exists within `B` starting at some position `B_j`, then Ethan's algorithm will fail to find that occurrence if and only if it reaches Step 2 with `i > 1` and `A_i = B_j = A_1`. This, in turn, will occur if and only if a length-`k` prefix of `A` exists within `B` ending at position `B_j`, such that `k > 1`. Therefore, if there's an index `k` such that `A_k = A_1` and `k > 1`, then we can construct a string `B` of length `|A|+k-1` by taking `A`'s length-`(k-1)` prefix followed by a full copy of `A`, and Ethan's algorithm will fail to find the occurrence of `A` which starts at position `B_k`. For example, if `A = “ABACUS”`, then we can choose `k = 3` to yield the string `B = “ABABACUS”`. If there's no such index `k`, then it's impossible for Ethan's algorithm to incorrectly return `false`.

However, choosing any such index `k` is insufficient to guarantee that Ethan's algorithm will return `false`, as if there's another occurrence of `A` within `B` (in particular, starting at position `B_1`), then his algorithm will still find that one and correctly return `true`. For example, if `A = “FBFBF”`, then choosing `k = 3` would yield the string `B = “FBFBFBF”`, which is no good. One possibility is to try each valid index `k`, and choose one which results in the resulting `B` string not containing `A` as a prefix, which can be done in `O(N^2)` time. The time complexity may also be improved to `O(N)` by observing that, if the earliest valid index `k` doesn't work out, then no later ones will either, due to `A` necessarily being “periodic” — made up entirely of two or more copies of its length-`(k-1)` prefix (with the last copy possibly being incomplete). In the above example, `“FBFBF”` is made up entirely of copies of `“FB”`, so neither `k = 3` nor `k = 5` will work out.