Code Jam 2013 - World Finals

Analysis: Can't Stop

Divide and Conquer

There is a nice divide-and-conquer solution. Split the input down the middle; the best interval is either entirely in the left half, entirely in the right half, or crosses the middle. We handle the right and left cases recursively; if we can handle the "crosses" case in linear time, we will have an `O(N lg N)` algorithm, which is good enough.

If the best interval crosses the middle, we know it must use the middle element, so we must pick one of the **D** numbers in that roll to use. Now go out as far as we can to the left and right with that number. Once we stop, we can extend either to the left or the right, so there are `2D` different numbers to try. Expand again, and get `2D` more numbers to try for a final expansion. In total, we only tried `D*2D*2D` different choices, so the whole thing runs in `256N` time, and we get our `O(N lg N)` time algorithm.

Note: surprisingly for a divide and conquer algorithm, we don't use any information from the left and right halves to solve the middle case; the fact that it crosses is enough.

Slow and Simple

There's another set of approaches to this problem. Try every starting position, expanding out to the right and only making choices when necessary (just as in the divide-and-conquer). This is $O(D^K * N)$ for each starting position, or $O(D^K * N^2)$ overall, which is too slow. However, there are several different improvements (some small, some large) one could make in order to make this linear in N.

Linear Time

One improvement works in this way: Every time you pick a number, check if the roll before your starting position contains that number. If so, we can safely ignore that choice (because we would have done better starting one roll earlier and choosing the same set of numbers).

It turns out that this runs in linear time, but the reason why isn't obvious. We want to prove that a particular index `i` is only visited from `O(1)` starting positions. Imagine we have reached `i` from `start`. Now imagine starting at `i` and going leftward (as usual, we expand left as far as possible before picking each new number). If we only choose numbers from the set that got us from `start` to `i`, we must get back to `start` (and no further, since the optimization above guaranteed that `start - 1` doesn't contain any of our numbers). But there are only $1 + D + D^2 + D^3$ different choices of numbers starting from `i`, so there are only that many possible positions for `start`, so `i` will only be visited $O(D^3)$ times.