Coding Competitions Farewell Rounds - Round D

Analysis: Old Gold

Test Set 1

For a given gold placement, we can consider what the markings would look like if they were all known. For example, if two consecutive gold nuggets are 3 kilometers apart, the expected pattern of markings would be $\circ$<>$\circ$. If they are 6 kilometers apart, the expected pattern would be $\circ$<<=>>$\circ$, and if they are 1 kilometer apart, it would be simply $\circ\circ$.

Because each of the expected patterns only depends on the positions of the two gold nuggets to the left and right, our solution can involve repeatedly adding a new gold nugget to the east and checking that all of the markings between the new gold nugget and the previous gold nugget match the expected pattern. We can make this algorithm efficient with dynamic programming by defining $dp(i)$ as the number of gold placements for the first i kilometers that have a gold nugget at the i-th kilometer.

$dp(i)$ is calculated as the sum of $dp(j)$ for all $(j < i)$ where the markings between kilometer j and i are valid, plus 1 if i can be the first gold nugget on the road. We can check if the markings are valid by iterating over all positions between j and i and checking if each one is either (.) or the expected marking of (<), (=), or (>). Also, the positions j and i themselves must have a (.) or ($\circ$).

We also need to specially handle the parts of the road to the west of the first gold nugget and to the east of the last gold nugget. All markings to the west of the first gold nugget must be (.) or (>). Similarly, our final answer will be the sum of $dp(i)$ for all i where all markings to the east are (.) or (<).

In the solution above, we calculate the value of each $dp(i)$ by iterating over all positions for the previous nugget and checking if the markings in between are valid. This is $O(n^2)$, where n is the length of $\mathbf{S}$. Applying this to each element of the dynamic programming table leads to a final time complexity of $O(n^3)$.

Test Set 2

We will solve Test Set 2 by speeding up the calculation of each $dp(i)$. Let's use $\mathbf{S}_x$ to denote the marking at position x, with $\mathbf{S}_0$ being the first marking. Also, let's use p to denote the position of the last known (<, =, >, or $\circ$) marking before i, or -1 if there is no known marking before i. This means that $\mathbf{S}_j$ is (.) for all $(p < j < i)$. We need to find all positions j that are valid and part of our summation for $dp(i)$. We can make the following observations:

1. All positions $(p < j < i)$ are valid since there are only (.)s in this range.
2. If $\mathbf{S}_p$ is ($\circ$), then its position $j = p$ is valid.
3. If $\mathbf{S}_p$ is (<), then it can be part of the left half of the pattern between two gold nuggets. The range of valid positions extends to the left of p as long as the markings are (.) or (<) and the center of the pattern has not passed p. More precisely, let q be the largest position less than p where $\mathbf{S}_q$ is (=), (>), or ($\circ$), or -1 if there is no such position, and let r be the leftmost position for the gold nugget that keeps the center of the pattern to the right of p. In this case, $r = p - (i - (p+1)) = 2p - i + 1$. Then all positions $(max(q, r-1) < j < p)$ where $\mathbf{S}_j$ is (.) are valid. If $\mathbf{S}_q$ is ($\circ$), then $j = q$ can be valid as well as long as $q \geq r$.

4. If $\mathbf{S}_p$ is (=), then it can be the center of the pattern between two gold nuggets, as long as the markings that will end up on the left side of the pattern are (.) or (<). More precisely, let q again be the largest position less than p where $\mathbf{S}_q$ is (=), (>), or (○), or -1 if there is no such position. Let r be the position of the gold nugget that will result in the center of the pattern being at p. In this case, $r = p - (i - p) = 2p - i$. Then $j = r$ is valid as long as $r > q$, or if $\mathbf{S}_q$ is (○) and $r = q$.
5. If $\mathbf{S}_p$ is (>), then it can be part of the right half of the pattern between two gold nuggets. As such, the gold nugget *cannot* be anywhere to the left of p that causes the center of the pattern to be to the right of p. More precisely, let r be the rightmost position for the gold nugget with the center of the pattern to the left of p, so $r = p - 1 - (i - p) = 2p - i - 1$. Then $(r < j < p)$ is *not* valid.

Observations 2, 3, and 4 all impose a limit on the lowest possible values for j, but observation 5 does not. This means that we can traverse backwards through $\mathbf{S}$ applying observation 5 for each (>) until we reach a (○), (<), (=), at which point the limits from observations 2, 3, and 4 take effect, or until we reach the start of the road.

We need to make one more observation about the number of contiguous ranges of positions for j that are valid. Observation 5 causes the next range of valid positions to be *double* the distance from the current (>) to i. Let's suppose that the position of the last (>) is l kilometers to the west of i. If we were trying to maximize the number of contiguous ranges of valid positions, we would place the next (>) at a position $2l + 2$ kilometers west of i so that there is a valid range of length 1 at $2l + 1$ kilometers west. Similarly, we would place the following (>) at a position $2(2l + 2) + 2 = 4l + 6$ kilometers west of i. Continuing this logic and ignoring the constant term in the equation, we get that having k ranges of valid positions requires $n > 2^k l$ kilometers. Therefore, $k < \log n - \log l$, which is $O(\log n)$ even if $l = 1$!

Our resulting algorithm will iterate over all ranges of valid positions for j and calculate the sum of $dp(j)$ in that range. We can calculate this sum for a single range in $O(1)$ time complexity using a prefix sum array. This prefix sum array should be updated as we calculate each $dp(i)$. Our target overall time complexity will be $O(n \log n)$.

Implementation

To aid in the implementation of the observations above, we can precalculate the positions of the last marking for each kilometer on the road, for various combinations of marking types. In particular, we can precalculate the largest $j < i$ such that $\mathbf{S}_j$ is (>) for each possible i. This would be needed to calculate observation 5. We can also precalculate similar arrays for the last instance of (○), (<), or (=) to choose between observations 2, 3, and 4, and the last instance of (○), (=), or (>) for the calculation of observation 3 or 4.

Unfortunately, simply traversing through each (>) until we reach observation 2, 3, or 4 could be $O(n)$ if there are many (>)s. However, if p is the position of the current (>), we know that there are no valid positions between $r = 2p - i - 1$ and p, so we can simply skip to the leftmost (>) in that range, as long as we don't skip past any (○), (<), or (=). We can precalculate an array with the next (>) for each position i for this purpose. Even though we have not travelled our expected 2^k kilometers, we will reach that point in the next iteration anyway because there were no more (>)s before the one we chose.

Also note that, when calculating observation 3, we can use the sum of all values for j in ($max(q, r - 1) < j < p$), even if some of the positions have a <, because those positions cannot have a gold nugget and the corresponding dp value will be 0.

Finally, note that the range of invalid positions in observation 5 between $r = 2p - i - 1$ and p can cause an upper bound on the valid positions when applying observations 2, 3, or 4. An example of this is `o.>.i`, where the (`o`) cannot be a valid j when calculating $dp(i)$.

Since none of these implementation details worsens the time complexity, our final algorithm is $O(n \log n)$

Alternative Implementation

TODO: Add explanation of alternative implementation idea if desired.