Code Jam 2015 - World Finals

Analysis: Costly Binary Search

Let us call the length of the input sequence N. There are N+1 possible "slots" in which the new element can end up — we number them from 0 to N, where 0 is the slot before the first element of **a**, and N is the slot after the last element of **a**.

At any point in our search, we will have narrowed down the possible location of the correct slot to an interval of possible slots. We then want to find out which slot in that interval is the correct one; we call this "solving" the interval.

For a given cost C and position X, we want to compute the largest possible interval of slots, beginning at X, which we can solve in cost C or less. Call the rightmost slot of that interval f(C,X). We will compute f using dynamic programming.

Clearly if C=0, then f(C,X)=X and the interval is empty. (Since each comparison costs at least 1, we cannot make any comparisons.)

Otherwise, assume we have an interval [X,Y], where X < Y, that we can solve at a cost no greater than C. The solution must involve an initial comparison at an index i, where i is in [X,Y-1], which costs a_i. Then we will have narrowed down the correct slot to either the interval of slots to the left of index i, which is [X,i]; or the interval of slots to the right of index i, which is [i+1,Y].

Each of these intervals must be solvable with a cost no greater than C-a_i, i.e., f(C-a_i, X) ≥ i, and f(C-a_i, i+1) ≥ Y.

We want to find the largest Y for which the above statements hold. To limit the number of possible choices for i and Y, we always use Y=f(C-a_i, i+1), since by definition this is the largest possible Y for a given choice of X and i. Also, instead of trying every value of i in [X, f(C-a_i, X)], we iterate through each possible value of a_i from 1 to 9, and try the largest index i in [X, f(C-a_i, X)] with that value. (Smaller indices i with the same value of a_i could not produce a larger Y). If there are no usable values of i (because C is too small, or X=N) then f(C,X)=X.

We compute f(C,X) in this way for increasing values of C, until we find the minimal C such that f(C,0)=N. This C is the answer.

The maximum value for the resulting C occurs when every array value is 9, in which case we can't do better than a standard binary search, which takes O(log N) comparisons of cost 9 each. That means computing f(C,X) over a domain of size 9 log N × N. Since the computation at each position takes constant time (we try 9 things), the overall algorithm takes O(N log N) time, with a somewhat large constant because of the two factors of 9 mentioned previously.