Up Down sequence

- Problem Summary: Given a permutation of length N & a string of length N-1 consisting of 'U' & 'D'. Find a the maximal length subsequence s.t. $a_i < a_{i+1}$ if $S_i = U$ & $a_i > a_{i+1}$ if $S_i = D$

- The problem is very similair to a classical problem called Longest increasing subsequence if $S = "UUUUU \ldots U"$ then the problem is equivalent to finding the LIS

- we can form a dp recurrence

  DP[i][j] = if there is a valid subsequence of length j using the ith value

  $DP[i][j] = \max_{k<i} DP[k][j-1]$ & $[P_k > P_i$ if $S_j = D$ or $P_k < P_i$ if $S_j = U]$

  which is $O(n^3)$

- if we change the dp to DP[i] = longest valid subsequence using P_i

  $DP[i] = \max_{j<i} DP[j] = L$; $P_i < P_j$ if $S_L = D$; $P_i > P_j$ if $S_L = U$

- we still have a $O(n^2)$ solution, but this matches the standard LIS solution. In the classical problem we can use a greedy solution by storing the lowest value to make an LIS of length K and use binary search to update values $O(n \cdot \log n)$

- In this case that isn't obvious to do. But what we can do is use a fast data structure a segment tree

- Lets say we have the longest valid subsequence using element i and that value is L then we can check S_L to see if the next value needs to be greater or smaller

- we keep two segment trees where each index corresponds to p_i

  - segment tree 1: stores all DP[j] s.t. $S_{DP[j]} = D$
  - segtree 2: stores all DP[j] s.t. $S_{DP[j]} = U$

- Now we can compute $DP[i] = \max(\text{segtree 1}_{A_i < x \le n}, \text{segtree 2}_{0 \le x < A_i})$

- we then update 1 segmentree depending on the value of DP[i]

- total complexity is $O(n \cdot \log n)$