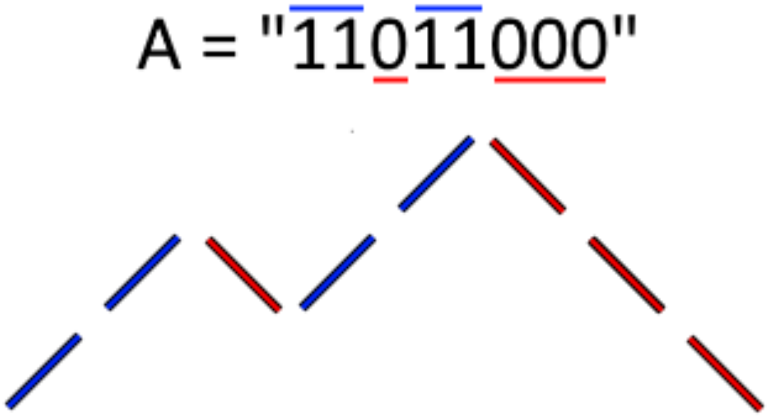
761. Special Binary String

Approach #1: Recursion [Accepted]

Intuition

We can represent binary strings as "up and down" drawings, as follows:



In such a drawing, `"1"` is a line up one unit, and `"0"` is a line down one unit, where all lines are 45 degrees from horizontal.

Then, a binary string is *special* if and only if its up and down drawing does not cross below the starting horizontal line.

Now, say a special binary string is a *mountain* if it has no special proper prefix. When viewed through the lens of up and down drawings, a special binary string is a mountain if it touches the starting horizontal line only at the very beginning and the very end of the drawing. Notice that every special string can be written as consecutive mountains.

Without loss of generality, we can assume we only swap mountains. Because if we swap special adjacent substrings A and B, and A has mountain decomposition $A = M_1 M_2 \dots M_k$, then we could instead swap B and M_k, then B and M_{k-1}, and so on.

Also, if S has mountain decomposition $S = M_1 M_2 \dots M_k$, and we choose A to start not at the start of some M_i, then A has global height $h > 0$, and so A cannot be special if it includes parts of another mountain M_{i+1} as the end of mountain M_i will cause it to dip to global height $0 < h$.

Algorithm

Say `F(String S)` is the desired `makeLargestSpecial` function. If `S` has mountain decomposition $S = M_1 M_2 \dots M_k$, the answer is $\text{reverse\_sorted}(F(M_1), F(M_2), \dots, F(M_k))$, as swaps `A, B` involving multiple M_i cannot have `A` or `B` start differently from where these M_i start.

It remains to determine how to handle the case when $S = S_0, S_1, \dots, S_{N-1}$ is a mountain. In that case, it must start with `"1"` and end with `"0"`, so the answer is `"1" + F([S[1], S[2], ..., S[N-2]]) + "0"`.

Java Python Copy

Complexity Analysis

- Time Complexity: $O(N^2)$, where N is the length of S.
- Space Complexity: $O(N)$.

Analysis written by: @awice (https://leetcode.com/awice).

Join the conversation

Signed in as **williamfu4leetcode**.

Post a Reply

(https://discuss.leetcode.com/user/chrislzm)

chrislzm commented 3 months ago

@invalid (https://discuss.leetcode.com/uid/67997) Worst case, we will touch n-2 elements in each level of recursion, so the total amount of work would be:

n + (n-2) + (n-4) + ... 4 + 2 = sum of even numbers through n ~= O(n^2)

I

(https://discuss.leetcode.com/user/invalid)

invalid commented 4 months ago

@awice (https://discuss.leetcode.com/uid/71269) how can I analyze time complexity of this problem to get to O(n^2)?