0-1 Sequences

Counting inversions is the topic of the problem [1] in this note.

[1] Tung Kam Chuen. 0-1 sequences. 2016. URL https://open.kattis.com/problems/sequences

Problem

You are given a sequence, in the form of a string with characters '0', '1', and '?' only. Suppose there are k '?'s. Then there are 2^k ways to replace each '?' by a '0' or a '1', giving 2^k different 0-1 sequences (0-1 sequences are sequences with only zeroes and ones).
For each 0-1 sequence, define its number of inversions as the minimum number of adjacent swaps required to sort the sequence in non-decreasing order. In this problem, the sequence is sorted in non-decreasing order precisely when all the zeroes occur before all the ones. For example, the sequence 11010 has 5 inversions. We can sort it by the following moves: $11010 \to 11001 \to 10101 \to 01101 \to 01011 \to 00111$.
Find the sum of the number of inversions of the 2^k sequences, modulo $10^9 + 7$.

There are two ways to count the necessary inversions to sort the 2^k 0-1 sequences: we could count for each '0' how many '1' to its left are marching by in the right direction on their way to being sorted. Or we could count for each '1' how many zeros to its right are marching by in the left direction on their way to being sorted.

We arbitrary choose the first way of counting the inversions.

In the sequence $\mathbf{b} = (b_0, b_1, \ldots, bn - 1)$ with characters '0', '1', and '?' we will look at each position i where $\mathbf{b}[i] =$ '0' and each position i where $\mathbf{b}[i] =$ '?'.

We define $q(i)$ to be the number of question marks to the left of i and $o(i)$ to be the number of ones to the left of i:

$$q(i) = |\{j : 0 \leq j < i : \mathbf{b}[j] = \text{'?'}\}|$$
$$o(i) = |\{j : 0 \leq j < i : \mathbf{b}[j] = \text{'1'}\}|$$

Let $s(i)$ be the number of inversions coming from $\mathbf{b}[i]$. When $\mathbf{b}[i] =$ '1' we set $s(i) = 0$ so as to not overcount[2].

[2] We chose to count ones marching right, passing zeros.

When $\mathbf{b}[i] =$ '0' we know that all 2^k 0-1 sequences will have $o(i)$ ones to the left of i. These definitely will count in $s(i)$. We also need to consider all ones coming from setting '?' into '1' to the left of i. There are $q(i)$ possibilities here. For each $j : 1 \leq j \leq q(i)$ we can turn j question marks into ones. We have to choose the subset of size j of positions from the set of $q(i)$ positions with question marks[3]. It follows that:

[3] As a convenience we label the $q(i)$ positions with question marks as position $1, 2, \ldots q(i)$.

$$s(i) = 2^k o(i) + 2^{k-q(i)} (\sum_{j=1}^{q(i)} \binom{q(i)}{j} j)$$

There is a neat way to simplify the sum with the binomial above using a combinatorial proof: Given a set of people of size N, count in how many ways you can choose a team and from that team choose a leader. There are two ways to count here. In the first way count the number of ways to choose a leader: N ways. Then count the number of ways to choose the rest of the team, which is the number of subsets from the set of people without the leader, so 2^{N-1}. In the second way for each possible team size, count the number of possible teams and then count the number of possible leader in that team. Because both ways count the same things, we have:

$$N2^{N-1} = \sum_{j=1}^{N} \binom{N}{j} j$$

Applied to our $s(i)$ we get:

$$\begin{aligned} s(i) &= 2^k o(i) + 2^{k-q(i)} q(i) 2^{q(i)-1} \\ &= 2^k o(i) + 2^{k-1} q(i) \end{aligned}$$

For $\mathbf{b}[i] =$ '?' we do a similar calculation[4], with the only difference being the number of question marks to the right of i: $2^{k-q(i)-1}$ (one less than in the previous calculation, since position i is a question mark). We get:

[4] We instantiate this question mark as a zero. The case where this position gets instantiated as a one is covered by other zero positions.

$$s(i) = 2^k o(i) + 2^{k-2} q(i)$$

The two cases cover all the counts and we can write the following loop (in Go, leaving out the modulus optimizations):

```
seen_ones := 0
seen_qmarks := 0
num_inversions := 0

for i:=0; i < n; i++ {
   switch {
   case b[i] == '0':
     num_inversions += 2^k * seen_ones + 2^(k-1) * seen_qmarks
   case b[i] == '1': seen_ones++
   case b[i] == '?':
     num_inversions += 2^k * seen_ones + 2^(k-2) * seen_qmarks
     seen_qmarks++
   }
}
```

For a more complete implementation in C++, see
https://github.com/uwedeportivo/kattis/tree/main/sequences.

Bibliography

Tung Kam Chuen. 0-1 sequences. 2016. URL https://open.kattis.com/problems/sequences.