# Yet another 01 problem

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 5 seconds |
| Memory limit: | 1024 megabytes |

Yana, Mino, White, and Huzz are best friends.

Mino has been feeling confused lately. Obsessed with his past failures in OI, he has been struggling to plan his future while managing his busy school schedule. His three friends suggested he evaluate the importance of his tasks and prioritize them.

Assume there are $n$ different tasks numbered from 1 to $n$. Each time, Huzz selects two adjacent tasks, White compares them, and Yana merges them into a single task. Mino is surprised to find that this process accidentally forms a segment tree, which does not necessarily split in the middle. More precisely, it forms a tree with $2n - 1$ nodes, where every subtree corresponds to a contiguous segment. Note that all $n$ tasks are leaves, while the other $n - 1$ nodes each have two children. These nodes, which result from comparisons, define the values of the edges to their children: 0 for the smaller one and 1 for the larger one.

Mino is very fond of calculations. He defines the weight of each task as the XOR sum of the edges along the path from the task to the root. He wonders: if the weights are given, how many ways are there to construct such trees and compare the children?

Once again, Mino is not skilled in OI, which is why he has turned to you for help. To simplify the problem, you only need to find the answer modulo $998,244,353$.

## Input

The first line of the input contains a positive integer $n$ ($1 \leq n \leq 250000$), the number of tasks.

The second line contains a binary string $S$ of length $n$, where $S_i$ represents the Bitwise-XOR of all values on the edges from task $i$ to the root.

## Output

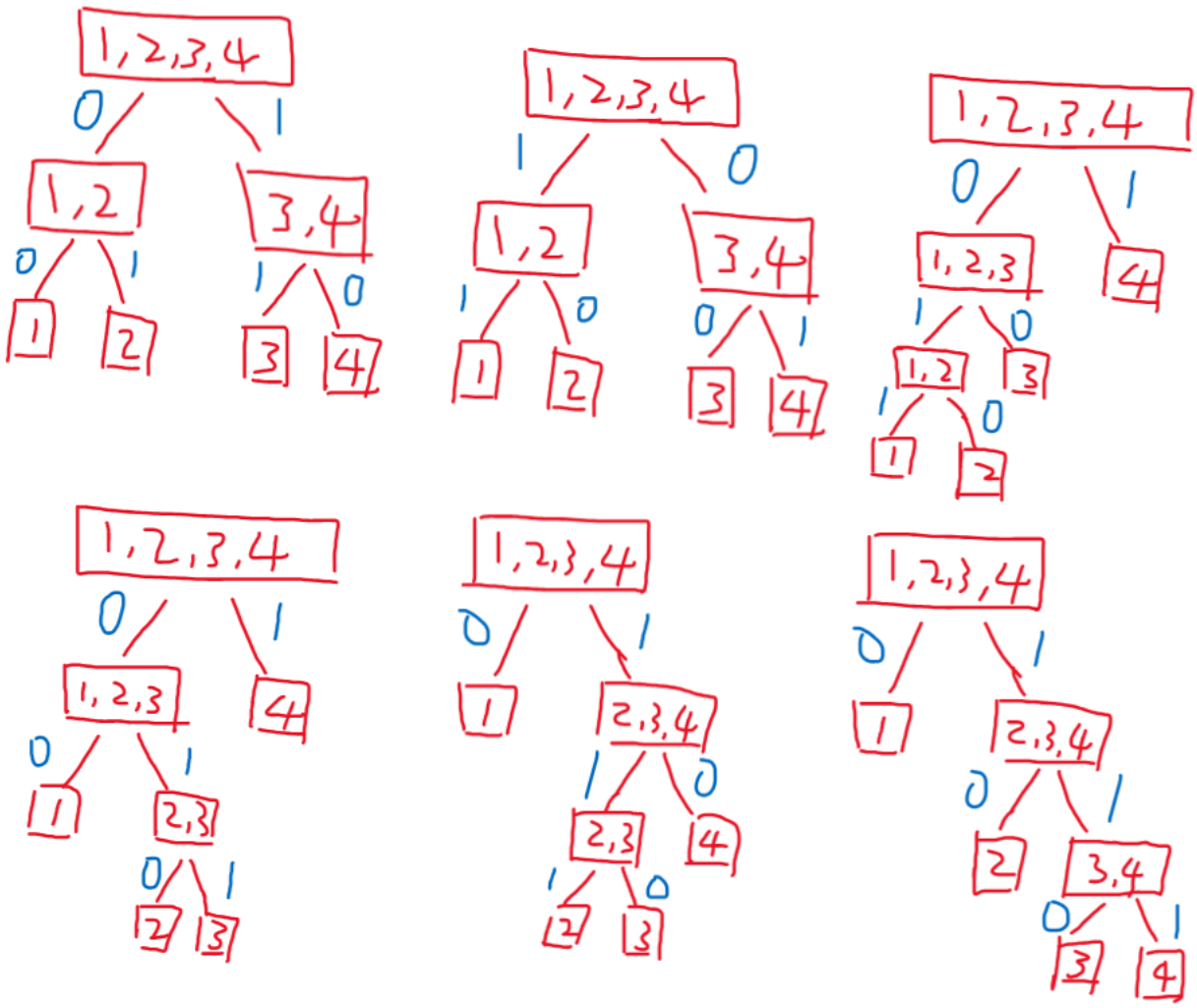
Print an integer, the answer modulo $998,244,353$.

## Example

| standard input | standard output |
|---|---|
| 4<br>0101 | 6 |

## Note

There are 6 ways to construct the tree and compare the children:


1,2,3,4
0
1
1,2
3,4
0
1
1
0
1
2
3
4
1,2,3,4
1
0
1,2
3,4
1
0
0
1
1
2
3
4
1,2,3,4
0
1
1,2,3
4
1
0
1,2
3
1
0
1
2
1,2,3,4
0
1
1,2,3
4
0
1
1
2,3
0
1
2
3
1,2,3,4
0
1
1
2,3,4
1
0
2,3
4
1
0
2
3
1,2,3,4
0
1
1
2,3,4
0
1
2
3,4
0
1
3
4