Insane DFS

Imagine you have a rooted tree consisting of n vertices. Consider the following function:

```
int order[n]; // initially consists of -1
int pointer = 0;
void dfs(int vertex, int depth) {
  order[pointer] = depth;
  pointer++;
  for each child of vertex
    dfs(child, depth + 1);
}
```

In the end this function produces an array $order[]$. We will call an array *suitable* if and only if it can be produced as a result of running $dfs(root, 0)$ on a rooted tree with n vertices.

You are given an array a, whose elements are either question signs or integer numbers. How many suitable arrays can you get, if you are allowed to replace any question sign with non-negative integer number? Print this number modulo $10^9 + 7$.

Input Format

The first line contains a single integer n, that is the size of array a. The next line contains n elements of the array: $a[0], a[1], \ldots, a[n-1]$. Each element is either a question sign, or a non-negative integer number which doesn't exceed 200.

Constraints

$1 \leq n \leq 10^5$
$0 \leq a[i] \leq 200$

Output Format

Print a single integer — the number of suitable arrays you can get modulo $10^9 + 7$.

Sample Input #0

```
3
? ? ?
```

Sample Output #0

```
2
```

Sample Input #1

```
2
1 ?
```

Sample Output #1

```
0
```

Sample Input #2

```
4
0 ? 1 ?
```

Sample Output #2

```
2
```

Explanation

In **sample#0** there are two possible arrays: [0, 1, 2] and [0, 1, 1];

In **sample#1** there cannot be any suitable arrays, because each of them starts with 0.