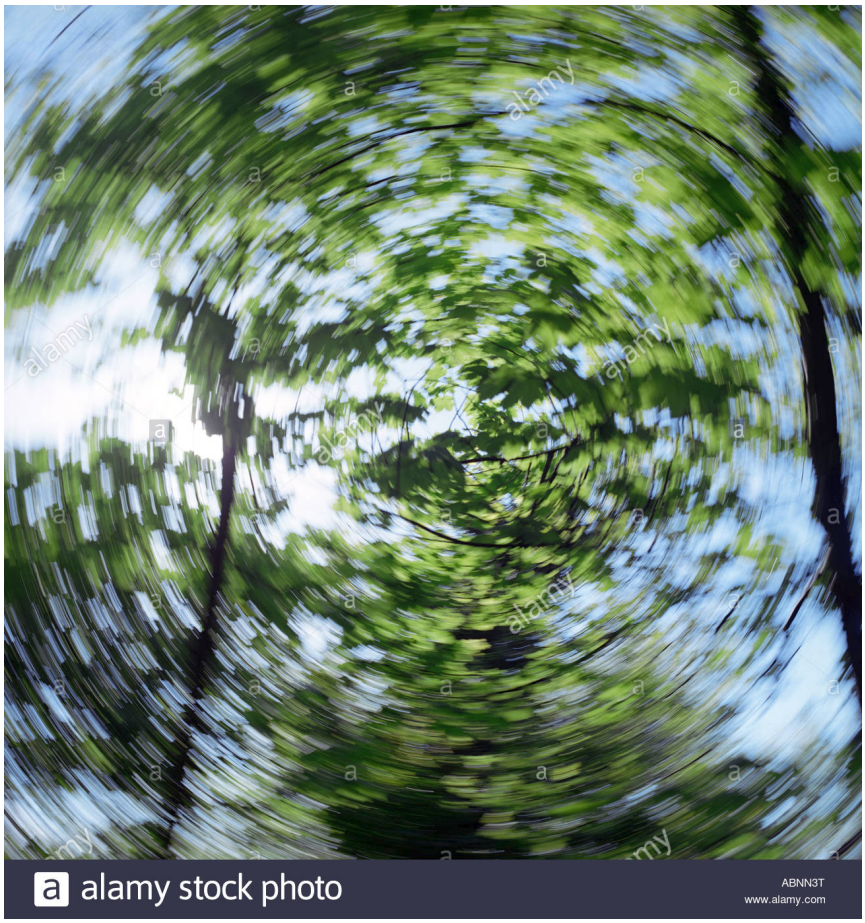
> Spanning Tree). You don't need Mr. K in another problem in coding practice.

After learning a bunch of tree data structures, you become dizzy and the trees look like they are spinning in your view.



You see an expression tree. The tree in your view can spin in two directions:

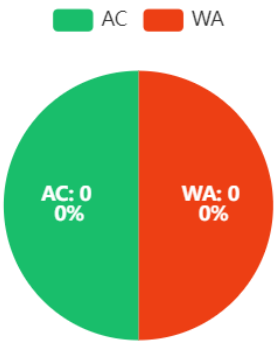
- `Left Spin` : Given a tree rooted at P, where Q is the right child of P, and B is the left child of Q (see the lower right figure). After a left spin, the new root becomes Q. The left child of Q becomes P, and the right child of P becomes B. If P does not have a right child, the left spin cannot be performed.
- `Right Spin` : Given a tree rooted at Q, where P is the left child of Q, and B is the right child of P (see the lower left figure). After a right spin, the new root becomes P. The right child of P becomes Q, and the left child of Q becomes B. If Q does not have a left child, the right spin cannot be performed.

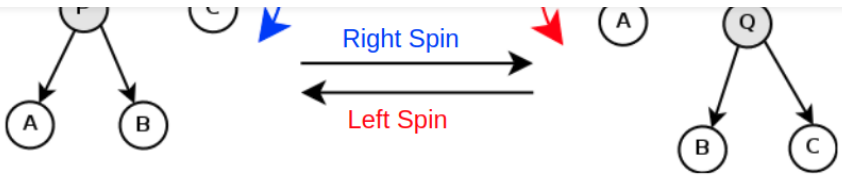
Submissions
Rankings
View Contest

Information

ID	4
Time Limit	1500MS
Memory Limit	256MB
IO Mode	Standard IO
Created By	ta_redleaf
Level	Hidden
Score	100
Tags	Show

Statistic Details





If the expression tree after several (or zero) times of spinning can still denote a valid expression, we say that the tree is a good spinning tree; otherwise we say that the tree is a bad spinning tree.

The evaluation of the tree changes after the tree spins. Given an evaluation tree, you want to know that what is the minimum evaluation among all good spinning trees.

Input

The first line of the input is an integer N, being the length of the prefix expression of the expression tree.

The next line is the expression tree (in prefix expression). The expression contains only `+`, `-`, `*` and digits from 1 to 9.

Constraints

- $3 \leq N \leq 3 \cdot 10^5$
- For all test cases, it is guaranteed intermediate values during calculation is between -10^9 and 10^9. That is, using 32-bit signed integer will not overflow (if your code does not have bugs).

Output

Output the minimum evaluation among all good spinning trees.

Sample Input 1

```
11
+-1+23*4-56
```

Sample Output 1

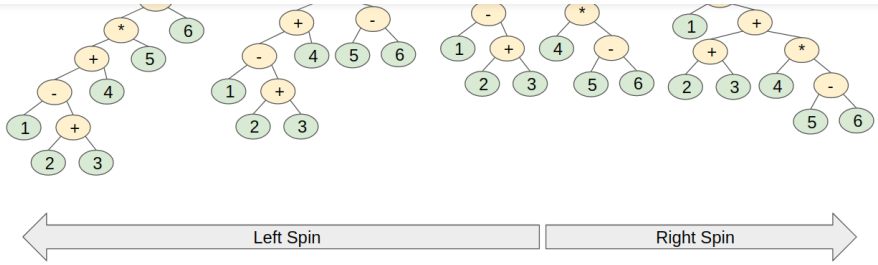
```
-8
```

Hint

Explanation of Sample IO

For the given expression tree, it has four good spinning trees which are shown in the below figure.

- Initially its evaluation is -8
- After one left spin, its evaluation becomes 0
- After two left spins, its evaluation becomes -6
- After one right spin, its evaluation becomes 0



Detailed Constraints

For test ID 1 (5% of total points):

- same as sample IO

For test ID 2 (5% of total points):

- $N \leq 50$
- It is guaranteed that there are only three good spinning trees : the giving tree, the tree that left spins once, and the tree that right spin once.
- i.e. evaluate the three good spinning trees and output the minimum value, and then you get AC in this test case

For test ID 3 - 6 (50% of total points):

- $N \leq 2500$

For test ID 7 - 10 (40% of total points):

- No additional constraints

Hints

Hint 1 (For test ID 3 - 10):

- Which nodes may become root of spinning trees?
- How many spinning trees are there?

Hint 2 (For test ID 7 - 10):

- Augment information in each node may help.
  - Store additional useful information in each node
  - These information can be updated efficiently
- Recall BST (binary search tree) lecture : We introduced BSTs that supports rank and select operation. We augment a "size" information that stores subtree size for each node to speed up rank and select operations.
- Recall AVL tree lecture: AVL tree augments a "balance factor" that helps itself determine how rotations should be done to keep the tree balanced.
- What information can be augmented in this problem so that evaluation of the expression tree can be speed up?

Language: C Theme: Solarized Light

Submit for Sample Test Submit

Sample Test Input

```
11
+-1+23*4-56
```

Sample Test Output