Brackets

task: `brackets`	input file: `stdin`	output file: `stdout`
points: `100`	time limit: `200 ms`	memory limit: `1 GB`

Task

A bracket symbol is one of the following: `()[]{}<>`. A correct *bracket expression* is any string consisting of bracket symbols, such that:

- Every left bracket has a matching right bracket of the same kind, and every right bracket is matched;
- No two pairs of matching brackets cross – for every two such pairs, they are either disjoint or one is contained inside the other.

For example, `([])<>` is a correct bracket expression, whereas `<{>}` is not, as the curly brackets and angle brackets cross each other.

You are given a graph of n vertices in which every (directed) edge is labeled with one of the bracket symbols. A path in this graph is *valid*, if its edges form a correct bracket expression. For some two vertices s and t, determine the length of a shortest valid path between s and t. We allow the path to pass multiple times through any vertex.

Input

On the first line of input there are four integers n, m, s, t ($1 \leq n \leq 200$, $0 \leq m \leq 2000$, $1 \leq s, t \leq n$) – the number of vertices, edges, starting and ending vertex, respectively. Each of the following m lines contains two integers x, y and a bracket symbol b ($1 \leq x, y \leq n$), which describe one graph edge. Note that there may be loops and multiple edges.

Output

Output a single line containing a single integer – the length of the shortest valid path between s and t. If there is no such path, output -1. You may assume that if a path exists, its length does not exceed 10^{18}.

Subtasks

Subtask	Points	Description
1	16	$n \leq 10, m \leq 50$
2	16	$n \leq 20, m \leq 100$
3	16	$n \leq 50$
4	16	$n \leq 100$
5	10	$s = 1$, $t = n$, $a < b$ for every edge (a, b)
6	26	no additional constraints

Samples

input

```
4 4 1 4
1 2 (
2 2 [
2 3 ]
3 4 )
```

output

```
4
```

input

```
5 4 1 5
1 2 <
2 3 {
3 4 >
4 5 }
```

output

```
-1
```