Find whether the given mathematical expression has balanced parentheses, braces and square brackets.

**Input Format**

The input has an expression with parantheses, braces and square brackets.

**Constraints**

1 <= length of the expression <= 100

**Output Format**

For each expression, print "**YES**" if it's balanced, else print "**NO**". (The double quotes is just for clarity)

**Sample Input 0**

(2+2)

**Sample Output 0**

YES

There is an array of random integers. This array is passed to a class to which you have no access to. The class has a method **ConstructBST** which takes an array as input and constructs a BST and stores it inside the class object.

**Remember, you don't have access to the constructed tree as well. All you have is the array which was initially passed to the class.**

Using this array, you have to print the **inorder traversal** of the constructed binary search tree.

**Input Format**

The first line has an integer **N**, the number of elements in the array.

The next line has N integers.

**Constraints**

1 <= N <= 10000

1 <= arr[i] <= 1000;

**Output Format**

Print the required inorder traversal in the same line each number separated by a space.

From the given binary rectangular matrix, find the shortest path's length in the matrix from the given source index to the destination index. The path can only be constructed with value 1 and at any moment we can only make one step in one of the following directions.

Go Up: (x, y) -> (x-1, y)

Go Left : (x, y) -> (x, y-1)

Go Down : (x, y) -> (x+1, y)

Go Right : (x, y) -> (x, y+1)

Note: Here x denotes Row Index and y denotes Column Index.

For example consider the following (9 \* 7) binary matrix. If the source = (0, 0) and destination = (6, 4), the shortest path length from source to destination is 11.

1 0 1 0 0 0 0

1 1 1 1 0 1 1

1 1 0 1 1 1 0

0 1 1 0 0 1 1

0 0 1 1 0 0 1

1 1 0 1 0 1 1

0 1 0 1 1 1 0

1 0 1 0 0 1 1

0 0 0 0 1 0 1

For the following example multiple paths can be created for given source and destination.

Path 1 Path 2 Path 3

1 . . . . . . 1 . . . . . . 1 . . . . . .

1 1 . . . . . 1 . . . . . . 1 1 1 1 . . .

. 1 . . . . . 1 1 . . . . . . . . 1 1 1 .

. 1 1 . . . . . 1 1 . . . . . . . . . 1 1

. . 1 1 . . . . . 1 1 . . . . . . . . . 1

. . . 1 . . . . . . 1 . . . . . . . . 1 1

. . . 1 1 . . . . . 1 1 . . . . . . 1 1 .

. . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . .

Length 10 Length 10 Length 14

Path 3's length is greater than Path 1 and Path 2. And Path 1 and Path 2 have the same length. So the shortest length from the given source to the destination is 11. If a path cannot be constructed for the given source to destination then return -1.

**Input Format**

The first line contains two space-separated integers n and m, denoting the number of rows and the number of columns in the matrix. The second line contains two space-separated integers sourceRow and sourceCol, denoting the source index pair in the matrix. The third line contains two space-separated integers destRow and destCol, denoting the destination index pair in the matrix. Each of the next n lines contains m space-separated binary value 0's and 1's.

**Constraints**

0 < n,m < 20 0 <= sourceRow, destRow <= n 0 <= sourceCol, destCol <= m

**Output Format**

Returns the shortest path's length for the given source and destination. If a path can't be constructed then return -1.

**Sample Input 0**

7 4

5 0

1 2

0 1 1 0

0 0 1 0

0 1 1 0

0 1 0 0

1 1 0 0

1 0 0 0

0 0 0 0

**Sample Output 0**

6

**Sample Input 1**

5 3

0 0

4 2

1 0 0

0 1 0

0 1 1

0 1 1

1 1 1

**Sample Output 1**

-1

Given a string ***S***, determine whether any permutation of it is a palindrome.

**Input Format**

A single line which contains the string ***S***

**Constraints**

1 <= |S| <= 10^5

**Output Format**

Print "**Yes**" if any permutation is a palindrome, if not "**No**"

**Sample Input 0**

aaabbbb

**Sample Output 0**

Yes

**Sample Input 1**

zoho

**Sample Output 1**

No