Problem Statement (MagicBoard)

Magic Board is a lock that has the shape of a rectangular board, divided into a grid of n × m unit squares. Some of the unit squares contain a Magic Diamond. You are given the contents of the Magic Board as a String[] **board**. The number of elements in **board** is n, and the length of each element is m. Empty squares are denoted by periods ('.'), squares with a Magic Diamond are denoted by '#'.

Let S = (S[0], S[1], ...) be a sequence of cells on the board. We will consider the cells in S one after another, picking up a Magic Diamond from each of them. Doing this will unlock the Magic Board if and only if S has the following properties:

- S contains each cell with a Magic Diamond exactly once.
- S does not contain any other cells.
- For each even i, S[i] and S[i+1] are in the same row of **board**.
- For each odd i, S[i] and S[i+1] are in the same column of **board**.

You are given the String[] **board**. Determine whether this Magic Board can be unlocked. Return "YES" (quotes for clarity) if it can be unlocked and "NO" otherwise.

Definition

Class:	MagicBoard
Method:	ableToUnlock
Parameters:	String[]
Returns:	String
Method signature:	String ableToUnlock(String[] board)

(be sure your method is public)

Constraints

-**board** will contain between 1 and 50 elements, inclusive.

-**board**[0] will contain between 1 and 50 characters, inclusive.

-All elements of **board** will contain the same number of characters.

-Each character in each element of **board** will be either '.' or '#'.

-**board** will contain at least one '#'.

Examples

0)

```
{"##",
 ".#"}
```

```
Returns: "YES"
```

The only solution: Start in the upper left corner, then move to the right, and finally move down.

1)
```
{"#.",
 ".#"}
```

```
Returns: "NO"
```
In this case there is no solution. Regardless of which Magic Diamond we start with, we will not be allowed to take the other one, as it is not in the same row.

2)
```
{"##",
 "##",
 "##"}
```

```
Returns: "YES"
```

3)
```
{"###",
 "###",
 "###"}
```

```
Returns: "NO"
```

4)
```
{"###",
 "..#",
 "###",
 "#..",
 "###"}
```

```
Returns: "NO"
```

5)
```
{".................",
 ".######..######.",
 ".######..######.",
 ".##......##..##.",
 ".##......##..##.",
 ".######..######.",
 ".######..######.",
 ".....##..##..##.",
 ".....##..##..##.",
 ".######..######.",
 ".######..######.",
 "................."}
```

```
Returns: "YES"
```

6)
```
{"#"}
```

```
Returns: "YES"
```