You are given an m x n matrix board, representing the**current**state of a crossword puzzle. The crossword contains lowercase English letters (from solved words), ' ' to represent any **empty**cells, and '#' to represent any **blocked** cells.

A word can be placed**horizontally** (left to right **or** right to left) or **vertically** (top to bottom **or** bottom to top) in the board if:

* It does not occupy a cell containing the character '#'.
* The cell each letter is placed in must either be ' ' (empty) or **match** the letter already on the board.
* There must not be any empty cells ' ' or other lowercase letters **directly left or right**of the word if the word was placed **horizontally**.
* There must not be any empty cells ' ' or other lowercase letters **directly above or below** the word if the word was placed **vertically**.

Given a string word, return true*if*word*can be placed in*board*, or*false***otherwise***.

**Example 1:**

A picture containing square

Description automatically generated

**Input:** board = [["#", " ", "#"], [" ", " ", "#"], ["#", "c", " "]], word = "abc"

**Output:** true

**Explanation:** The word "abc" can be placed as shown above (top to bottom).

**Example 2:**

Chart

Description automatically generated

**Input:** board = [[" ", "#", "a"], [" ", "#", "c"], [" ", "#", "a"]], word = "ac"

**Output:** false

**Explanation:** It is impossible to place the word because there will always be a space/letter above or below it.

**Example 3:**

A picture containing square

Description automatically generated

**Input:** board = [["#", " ", "#"], [" ", " ", "#"], ["#", " ", "c"]], word = "ca"

**Output:** true

**Explanation:** The word "ca" can be placed as shown above (right to left).

**Constraints:**

* m == board.length
* n == board[i].length
* 1 <= m \* n <= 2 \* 105
* board[i][j] will be ' ', '#', or a lowercase English letter.
* 1 <= word.length <= max(m, n)
* word will contain only lowercase English letters.