

题目描述：
2XXX 年，人类通过对火星的大气进行宜居改造分析，使得火星已在理论上具备人类宜居的条件；由于技术原因，无法一次性将火星大气全部改造，只能通过局部处理形式；假设将火星待改造的区域为 $row * column$ 的网格，每个网格有 3 个值，宜居区、可改造区、死亡区，使用 YES、NO、NA 代替，YES 表示该网格已经完成大气改造，NO 表示该网格未进行改造，后期可进行改造，NA 表示死亡区，不作为判断是否改造完成的宜居，无法穿过；
初始化下，该区域可能存在多个宜居区，并且每个宜居区能同时在每个太阳日单位向上下左右四个方向的相邻格子进行扩散，自动将 4 个方向相邻的真空区改造成宜居区；请计算这个待改造区域的网格中，可改造区是否能全部变成宜居区，如果可以，则返回改造的太阳日天数，不可以则返回 -1。
输入描述：
输入 $row * column$ 个网格数据，每个网格值枚举值如下：YES，NO，NA；样例：  YES YES NO  NO NO NO  NA NO YES
输出描述：
可改造区是否能全部变成宜居区，如果可以，则返回改造的太阳日天数，不可以则返回 -1。
补充说明：
$grid[i][j]$ 只有 3 种情况，YES、NO、NA  $row == grid.length, column == grid[i].length, 1 \leq row, column \leq 8$
示例 1

输入:

YES YES NO  
NO NO NO  
YES NO NO

输出:

2

说明:  
经过 2 个太阳日，完成宜居改造。

初始化	第一太阳日																		
<table><tr><td>YES</td><td>YES</td><td>NO</td></tr><tr><td>NO</td><td>NO</td><td>NO</td></tr><tr><td>YES</td><td>NO</td><td>NO</td></tr></table>	YES	YES	NO	NO	NO	NO	YES	NO	NO	<table><tr><td>YES</td><td>YES</td><td>YES</td></tr><tr><td>YES</td><td>YES</td><td>NO</td></tr><tr><td>YES</td><td>YES</td><td>NO</td></tr></table>	YES	YES	YES	YES	YES	NO	YES	YES	NO
YES	YES	NO																	
NO	NO	NO																	
YES	NO	NO																	
YES	YES	YES																	
YES	YES	NO																	
YES	YES	NO																	

示例 2

输入:

YES NO NO NO  
NO NO NO NO  
NO NO NO NO  
NO NO NO NO

输出:

6

说明：

经过 6 个太阳日，可完成改造

示例 3

输入：

NO NA

输出：

-1

说明：

无改造初始条件，无法进行改造

示例 4

输入：

YES NO NO YES  
NO NO YES NO  
NO YES NA NA  
YES NO NA NO

输出：

-1

说明：

-1 // 右下角的区域，被周边三个死亡区挡住，无法实现改造

```
import sys
from queue import Queue
```

```
matrix = []
curr = 0
q = [Queue(), Queue()]
not_cleared = 0
```

```
row = 0
m = 0
```

```

n = 0
for line in sys.stdin:
    a = line.split()
    n = len(a)
    m += 1
    row_d = []
    for col in range(n):
        row_d.append(0 if a[col] == 'NO' else 1 if a[col] == 'YES' else -1)
        if a[col] == 'YES':
            q[curr].put((row, col))
        if a[col] == 'NO':
            not_cleared += 1

    matrix.append(row_d)
    row += 1

i = -1
while not q[curr].empty():
    next = 1 - curr
    while not q[curr].empty():
        # print(matrix, list(q[curr].queue), not_cleared)
        (r, c) = q[curr].get()
        # if matrix[r][c] != 0:
        #     continue
        # matrix[r][c] = 1
        # not_cleared -= 1
        for (dr, dc) in [(0,1),(0,-1),(1,0),(-1,0)]:
            rr = r + dr
            cc = c + dc
            if rr >= 0 and rr < m and cc >= 0 and cc < n and matrix[rr][cc] == 0:
                matrix[rr][cc] = 1
                not_cleared -= 1
                q[next].put((rr, cc))

    i += 1
    curr = next

if not_cleared == 0:
    print(i)
else:
    print(-1)

```