**Scenario-Based Coding Problem: Forest Fire Simulation**

**Problem Statement:**

In a dense forest, a fire has started at a specific tree and is spreading in all four directions (up, down, left, right). The forest is represented as an m x n grid where each cell can either be a tree (T), an empty space (.), or a rock (#). The fire cannot spread through rocks but will spread through trees and empty spaces. The fire takes 1 minute to spread from one tree to an adjacent tree or empty space. Given the grid and the initial fire starting positions, determine the minimum number of minutes required for the fire to completely spread throughout the forest.

**Input Format:**

* The first line contains two integers m and n, the number of rows and columns in the grid.
* The next m lines contain n characters each representing the grid.
* The last line contains two integers r and c, the row and column indices of the initial fire starting position.

**Output Format:**

* Print the minimum number of minutes required for the fire to spread completely throughout the forest, followed by the word "minutes".
* If the fire cannot spread to certain parts of the forest, print -1.

**Constraints:**

* 1 <= m, n <= 1000
* The grid contains only characters T, ., and #.
* The starting position r, c will always be a tree (T) or an empty space (.).

**Sample Input 1:**

5 5

TTTTT

T###T

TT.TT

TTTTT

T###T

1 1

**Sample Output 1:**

8 minutes

**Explanation:**

* In the first sample, the fire spreads to all the trees in 8 minutes:
  + At minute 0, fire starts at (1, 1).
  + At minute 1, fire spreads to (1, 2), (2, 1).
  + At minute 2, fire spreads to adjacent cells of the newly burnt trees, and so on.

**Sample Input 2:**

3 3

T#T

###

T#T

1 1

**Sample Output 2:**

-1

**Explanation:**

* In this sample input, trees are isolated by rocks, making it impossible for the fire to spread to them, hence the output is -1.

**Sample Input 3:**

4 4

TT#T

T#TT

TT#T

#TTT

4 4

**Sample Output 3:**

7 minutes

**Solution:**

python

from collections import deque

def min\_fire\_spread\_time(m, n, grid, start\_row, start\_col):

# Directions for moving in the grid (right, down, left, up)

directions = [(0, 1), (1, 0), (0, -1), (-1, 0)]

# Queue for BFS and a set for visited positions

queue = deque([(start\_row, start\_col, 0)]) # (row, col, time)

visited = set([(start\_row, start\_col)])

# BFS to spread the fire

max\_time = 0

while queue:

x, y, time = queue.popleft()

max\_time = max(max\_time, time)

for dx, dy in directions:

nx, ny = x + dx, y + dy

# Check if the new position is within bounds and is not a rock

if 0 <= nx < m and 0 <= ny < n and (nx, ny) not in visited and grid[nx][ny] != '#':

visited.add((nx, ny))

queue.append((nx, ny, time + 1))

# Check if there are any trees or empty spaces that the fire didn't reach

for i in range(m):

for j in range(n):

if grid[i][j] in ('T', '.') and (i, j) not in visited:

return -1

return max\_time

# Reading input

m, n = map(int, input().split())

grid = [input().strip() for \_ in range(m)]

start\_row, start\_col = map(int, input().split())

# Solving the problem

result = min\_fire\_spread\_time(m, n, grid, start\_row, start\_col)

if result == -1:

print(result)

else:

print(f"{result} minutes")

**Test Cases:**

**Test Case 1:**

Input:

5 5

TTTTT

T###T

TT.TT

TTTTT

T###T

3 3

Output:

6 minutes

**Test Case 2:**

Input:

4 4

TT#T

T#TT

TT#T

#TTT

1 1

Output:

9 minutes

**Test Case 3:**

Input:

3 3

T#T

###

T#T

1 1

Output:

-1

**Test Case 4:**

Input:

3 3

TTT

T#T

TTT

1 1

Output:

4 minutes

**Test Case 5:**

Input:

3 3

TTT

TTT

TTT

1 1

Output:

4 minutes