from collections import deque

ROW = int(input())

COL = int(input())

class Point:

def \_\_init\_\_(self,x: int, y: int):

self.x = x

self.y = y

class queueNode:

def \_\_init\_\_(self,pt: Point, dist: int):

self.pt = pt

self.dist = dist

def isValid(row: int, col: int):

return (row >= 0) and (row < ROW) and (col >= 0) and (col < COL)

rowNum = [-1, 0, 0, 1]

colNum = [0, -1, 1, 0]

def BFS(mat, src: Point, dest: Point):

if ((97<=mat[src.x][src.y]<=122 or 65<=mat[src.x][src.y]<=90)==False or (97<=mat[dest.x][dest.y]<=122 or 65<=mat[dest.x][dest.y]<=90)==False):

return -1

visited = [[False for i in range(COL)]

for j in range(ROW)]

visited[src.x][src.y] = True

q = deque()

s = queueNode(src,0)

q.append(s)

while q:

curr = q.popleft()

pt = curr.pt

if pt.x == dest.x and pt.y == dest.y:

return curr.dist

for i in range(4):

row = pt.x + rowNum[i]

col = pt.y + colNum[i]

if (isValid(row,col) and chr(mat[row][col]).isalpha() == True and not visited[row][col]):

visited[row][col] = True

Adjcell = queueNode(Point(row,col),curr.dist+1)

q.append(Adjcell)

return -1

'''

3

10

0 0

2 0

67 0 2 2 2 1 0 1 1 1

67 0 2 2 2 1 0 1 1 1

67 0 2 2 2 1 0 1 1 1

'''

def main():

l1=list(map(int, input().split()))

l2=list(map(int, input().split()))

source = Point(l1[0],l1[1])

dest = Point(l2[0],l2[1])

mat = []

for i in range(0,ROW):

addl=list([int(item) for item in input().split()])

mat.append(addl)

dist = BFS(mat,source,dest)

if dist!=-1:

print("Shortest Path is",dist)

else:

print("Shortest Path doesn't exist")

main()