# Question 01:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA computer screen shot of a computer screen

Description automatically generated

def findBestPath(n, m, startRow, startColumn, endRow, endColumn, monsterRow, monsterColumn):

def isPathPossible(min\_dist, new\_x, new\_y):

for i in range(len(monsterRow)):

if abs(new\_x - monsterRow[i]) + abs(new\_y - monsterColumn[i]) < min\_dist:

return False

return True

def bfs(min\_dist):

visited = [[False] \* m for \_ in range(n)]

queue = [(startRow, startColumn)]

visited[startRow][startColumn] = True

while queue:

curr\_x, curr\_y = queue.pop(0)

if curr\_x == endRow and curr\_y == endColumn:

return True

for dx, dy in [(0, 1), (1, 0), (0, -1), (-1, 0)]:

new\_x, new\_y = curr\_x + dx, curr\_y + dy

if 0 <= new\_x < n and 0 <= new\_y < m and not visited[new\_x][new\_y] and isPathPossible(min\_dist, new\_x, new\_y):

visited[new\_x][new\_y] = True

queue.append((new\_x, new\_y))

return False

start, end = 0, n \* m

res = -1

while start <= end:

mid = (start + end) // 2

if bfs(mid):

res = mid

start = mid + 1

else:

end = mid - 1

return res

# Question 02:

Question 2 was similar to the question given in this link: <https://leetcode.com/discuss/interview-question/2766517/IMC-or-OA-or-Busy-Intersection>.

Wordings were changed as the example of was Mountain and trails.

from collections import deque

def getResult(arrival, direction):

time = 0

i = 0

prev\_hiker = 1

q1, q0 = deque(), deque()

res = [0] \* len(arrival)

while True:

move = True

while i < len(arrival) and arrival[i] == time:

if direction[i] == 0:

q0.append(i)

else:

q1.append(i)

i += 1

if q0 and not q1:

prev\_hiker = 0

while prev\_hiker == 1 and q1:

res[q1.popleft()] = time

time += 1

move = False

while i < len(arrival) and arrival[i] == time:

if direction[i] == 0:

q0.append(i)

else:

q1.append(i)

i += 1

while prev\_hiker == 0 and q0:

res[q0.popleft()] = time

time += 1

move = False

while i < len(arrival) and arrival[i] == time:

if direction[i] == 0:

q0.append(i)

else:

q1.append(i)

i += 1

if not q0:

prev\_hiker = 1

if move:

time = arrival[i]

continue

if not q0 and not q1 and i == len(arrival):

return res