#!/usr/bin/env python3

from \_\_future\_\_ import print\_function

from builtins import input

import numpy as np

import sys

import itertools

import math

from py3\_map import Map

def make\_cost\_map(m, miny):

cost\_m = Map(m.w, m.h, 1000, 1000)

x = -1

maxy = min(m.maxy, m.h)

while x <= m.w:

x += 1

if m.getXY(x - 1, maxy) > 0 and m.getXY(x, maxy) == 0:

x1 = x + 1

while m.getXY(x1, maxy) == 0:

x1 += 1

for x2 in range(x, x1, 1):

cost\_m.setXY(x2, maxy, m.w - min(x2 - x, x1 - x2 - 1))

for y in range(maxy - 1, max(0, miny) - 1, -1):

for x in range(m.w):

if m.getXY(x, y) == 0:

cost\_m.setXY(x, y, cost\_m.getXY(x, y + 1) + 15)

for x in range(m.w):

if m.getXY(x, y) == 0:

v = min(cost\_m.getXY(x, y), cost\_m.getXY(x - 1, y) + 12)

cost\_m.setXY(x, y, v)

for x in range(m.w - 1, -1, -1):

if m.getXY(x, y) == 0:

v = min(cost\_m.getXY(x, y), cost\_m.getXY(x + 1, y) + 12)

cost\_m.setXY(x, y, v)

for x in range(m.w):

if m.getXY(x, y) == 0:

if m.getXY(x, y + 1) == 0:

cost\_m.setXY(x, y, cost\_m.getXY(x, y) - 2)

if m.getXY(x + 1, y) == 0 and m.getXY(x, y + 1) == 0:

cost\_m.setXY(x, y, cost\_m.getXY(x, y) - 2)

if m.getXY(x - 1, y) == 0 and m.getXY(x, y + 1) == 0:

cost\_m.setXY(x, y, cost\_m.getXY(x, y) - 2)

for y in range(maxy + 1, maxy + 4):

for x in range(m.w):

cost\_m.setXY(x, y, cost\_m.getXY(x, y - 1))

return cost\_m

def readline():

x = input()

print(str(x), file=sys.stderr)

return x

def next\_state(p, ac, m):

next\_v = p[1] + ac

next\_p = p[0] + next\_v

if m.has\_collision(p[0], next\_p):

return (p[0], next\_v)

return (next\_p, next\_v)

def evaluate(ps, m, cost\_m):

cost = -cost\_m.getP(ps[0])

return cost - 10 \* np.sum(np.absolute(ps[1]))

def search(pos, m, cost\_m, depth):

if depth == 0:

return evaluate(pos, m, cost\_m)

maxy = min(m.maxy, m.h)

if pos[0][1] + pos[1][1] >= maxy + 4:

return evaluate(pos, m, cost\_m) + depth \* 100

maxv = float('-inf')

for ax, ay in itertools.product(range(-1, 2), range(-1, 2)):

next\_pos = next\_state(pos, np.array([ax, ay]), m)

v = search(next\_pos, m, cost\_m, depth - 1)

maxv = max(maxv, v)

return maxv

def main():

total\_time = int(readline())

max\_step = int(readline())

width, height = [int(x) for x in readline().split()]

view = int(readline())

print(0)

m = Map(width, height)

if total\_time < 5000:

depth = 2

else:

depth = 3

while True:

step = int(readline())

time = int(readline())

ps = []

for j in range(2):

xs = [int(x) for x in readline().split()]

p = [np.array([xs[0], xs[1]]), np.array([xs[2], xs[3]])]

ps.append(tuple(p))

for y in range(ps[0][0][1] - view, ps[0][0][1] + view + 1, 1):

ls = [int(v) for v in readline().split()]

if y > 0:

m.setline(y, ls)

acs, maxv = None, float('-inf')

cost\_m = make\_cost\_map(m, ps[0][0][1] - view)

for ax, ay in itertools.product(range(-1, 2), range(-1, 2)):

next\_pos = next\_state(ps[0], np.array([ax, ay]), m)

if (next\_pos[0] == ps[0][0]).all():

continue

v = search(next\_pos, m, cost\_m, depth)

if v > maxv:

maxv = v

acs = (ax, ay)

if maxv < -1000000.0:

vv = max(np.absolute(ps[0][1]))

for ax, ay in itertools.product(range(-vv, vv + 1), range(-vv, vv + 1)):

next\_pos = next\_state(ps[0], np.array([ax, ay]), m)

if (next\_pos[0] == ps[0][0]).all():

continue

v = max(abs(ax), abs(ay)) \* -100 + evaluate(next\_pos, m, cost\_m)

if v > maxv:

maxv = v

acs = (np.sign(ax), np.sign(ay))

if acs is None:

acs = (-np.sign(ps[0][1][0]), -np.sign(ps[0][1][1]))

cmd = ' '.join(str(x) for x in acs)

print(cmd)

if \_\_name\_\_ == '\_\_main\_\_':

main()