# Robot

## Maze

AIC 2005 (Australia)

Mê cung được thể hiện qua một lưới đơn vị kích thước n× m gồm các ô trắng và đen. Các dòng của mê cung được đánh số 1..n từ trên xuống, các cột: 1..m từ trái qua phải. Tại ô trắng trên dòng d, cột c có một quân cờ X. Mỗi nước đi, bạn có thể di chuyển quân cờ dọc hoặc ngang theo số lượng tuỳ ý các ô trắng liên tiếp nhau nhưng không được đi vào hoặc vượt qua ô đen. Ví dụ, từ ô X trong hình bạn có thể thực hiện 4 nước đi khác nhau như sau:

1. Đi xuống 1 ô,

2. Đi xuống 2 ô,

3. Đi xuống 3 ô,

4. Qua phải 1 ô.

Bạn cần di chuyển quân cờ X đến một ô trắng bất kì trong mê cung nhưng không được sử dụng quá k nước đi. Hãy cho biết giá trị nhỏ nhất của k.

Giới hạn max n = max m = 100.

Table

Description automatically generated

*Input text file: MAZE.INP*

*Dòng đầu tiên: hai giá trị n và m, 1 ≤ n, m ≤ 100.*

*Dòng thứ hai: d và c cho biết vị trí của quân cờ X.*

*n dòng tiếp theo, mỗi dòng một xâu m kí tự trong đó 0 thể hiện ô trắng, 1 thể hiện ô đen.*

*Output: giá trị k trên màn hình.*

### Thuật toán

Ta phỏng theo thuật toán sửa đỉnh của Dijkstra để giải bài này. Gọi a là ma trận 2 chiều chứa giá trị xuất phát của mỗi ô (i, j). Trước hết ta quy định các chỉ số được tính từ 0, vì quy định này không làm thay đổi kết quả của bài toán.

Đọc dữ liệu: Trong pha này ta chuyển đổi các ô màu đen có giá trị 1 là những ô cấm không được vượt qua thành BLACK = -1, các ô trắng có giá trị 0 thành các ô nhận giá trị vô cùng inf = nm+10, trong đó n là số dòng, m là số cột, nm là diện tích của ma trận a. Riêng ô xuất phát X là ô (d,c) ta gán trị 0.

Sửa trị: Xuất phát từ ô a[d][c] = 0, ta sửa trị của các ô trên làn kề trái, kề phải, kề trên và kề dưới với ô này thành trị 1. Một làn kề với ô (i,j) là dãy các ô trắng kề liên tiếp với ô (i,j).

Ví dụ ô (2,2) chứa X có

Làn kê trái có 0 ô

Làn kề phải có 1 ô là (2,3)

Làn kề trên có 0 ô

Làn kề dưới có 3 ô là (3,2), (4,2) và (5,2)

Tiếp đến ta thực hiện vòng lặp với các trị (màu) v = 1..k:

Tìm toàn bộ các ô (i,j) có trị v để sửa trị các ô trên các làn kề với ô (i,j) thành trị v+1.

Thuật toán sẽ kết thúc khi mọi ô trắng đều được sửa trị.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *Tô ô X bằng màu 0*  *Tô các ô theo làn kề trai, phải,*  *trên, dưới từ màu v thành màu v+1* |
| *0* | 3 | 4 | 4 |  | 4 | 5 | 5 | 5 |
| *1* | 3 | 4 |  | 2 | 3 |  | 6 |  |
| *2* | 3 |  | 0 | 1 |  | 5 | 6 | 6 |
| *3* | 2 | 2 | 1 |  | 3 | 4 |  | 7 |
| *4* | 3 |  | 1 | 2 | 2 |  | 8 | 7 |
| *5* | 2 | 2 | 1 |  | 3 |  | 8 | 7 |
| *6* |  | 3 |  | 5 |  | **9** |  | 7 |
| *7* | 4 | 3 | 4 | 4 |  | 8 | 8 | 7 |

### Chương trình

# Maze

FN = "MAZE.INP"

BLACK = -1

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def ReadInput():

global n, m, inf, a, d, c

with open(FN) as f:

n, m = map(int,f.readline().split())

inf = n\*m + 10 # gía trị vô cùng

print(n, 'dòng', m, 'cột')

print(' vô cùng =', inf)

d, c = map(int,f.readline().split())

d -= 1; c -= 1

print(' Ô xuất phát X (chỉ số tính từ 0):', d, c)

a = [] # sư đồ mê cung

for i in range(n):

a.append(list(map(int,f.readline().split())))

print(' Maze map:')

for i in range(n):

for j in range(n):

a[i][j] = BLACK if a[i][j] == 1 else inf

a[d][c] = 0

# Sửa trị V2 cho các ô kề ngang, dọc với ô (d,c)

def Update(d, c, v2):

global a, dt # dt: diện tích mê cung

v1 = a[d][c]+1

# qua trái

for j in range(c-1, -1, -1):

if a[d][j] == BLACK: break

if v1 < a[d][j]:

a[d][j] = v2

dt -= 1

# qua phải

for j in range(c+1, m):

if a[d][j] == BLACK: break

if v1 < a[d][j]:

a[d][j] = v2

dt -= 1

# lên

for i in range(d-1, -1, -1):

if a[i][c] == BLACK: break

if v1 < a[i][c]:

a[i][c] = v2

dt -= 1

# xuống

for i in range(d+1, n):

if a[i][c] == BLACK: break

if v1 < a[i][c]:

a[i][c] = v2

dt -= 1

def Scan(v):

for i in range(n):

for j in range(m):

if a[i][j] == v:

Update(i, j, v+1)

def Maze():

global dt

ReadInput()

for i in range(n): print(a[i])

dt = sum([a[i].count(inf) for i in range(n)])

print(' Tổng số ô cần sửa trị', dt)

Update(d,c,1)

for v in range(1, inf):

if dt == 0: break

Scan(v) # v -> v+1

print(' Final:')

for i in range(n): print(a[i])

print(' Min step:', v)

# APPLICATION

Maze()

print(" T h e E n d")

### Output

8 dòng 8 cột

vô cùng = 74

Ô xuất phát X (chỉ số tính từ 0): 2 2

Maze map:

[74, 74, 74, -1, 74, 74, 74, 74]

[74, 74, -1, 74, 74, -1, 74, -1]

[74, -1, 0, 74, -1, 74, 74, 74]

[74, 74, 74, -1, 74, 74, -1, 74]

[74, -1, 74, 74, 74, -1, 74, 74]

[74, 74, 74, -1, 74, -1, 74, 74]

[-1, 74, -1, 74, -1, 74, -1, 74]

[74, 74, 74, 74, -1, 74, 74, 74]

Tổng số ô cần sửa trị 46

Final:

[3, 4, 4, -1, 4, 5, 5, 5]

[3, 4, -1, 2, 3, -1, 6, -1]

[3, -1, 0, 1, -1, 5, 6, 6]

[2, 2, 1, -1, 3, 4, -1, 7]

[3, -1, 1, 2, 2, -1, 8, 7]

[2, 2, 1, -1, 3, -1, 8, 7]

[-1, 3, -1, 5, -1, 9, -1, 7]

[4, 3, 4, 4, -1, 8, 8, 7]

Min step: 9

T h e E n d

### Độ phức tạp

Gọi L là chiều dài dữ liệu vào, L = nm = số phòng (ô) của mê cung. Thuật toán lặp tối đa L lần, giả sử mỗi lần chỉ sửa được một ô thì độ phức tạp tối đa là .

## Where

Người ta lập trình cho một Robot xuất phát từ điểm A(ax,ay) trên mặt phẳng tọa độ Descartes, mặt quay về hướng Bắc, tức là nhìn theo trục tung Oy và cần đi theo một bản chỉ đường S để đến điểm đích B. Bản chỉ đường là một dãy kí tự gồm các chữ cái F, R và L viết liền nhau thể hiện các lệnh với ý nghĩa như sau:

Box and whisker chart

Description automatically generated *F – đi thẳng theo 1 đơn vị dài, chẳng hạn 1 mét,*

*R – quay phải 90 độ,*

*L – quay trái 90 độ.*

*Biết điểm A(ax,ay) và bản chỉ đường s.*

*Hãy xác định điểm B(bx, by)?*

*Dữ liệu input:*

*s – bản chỉ đường dưới dạng string.*

*(ax, ay) – tọa độ điểm xuất phát.*

*Dữ liệu output*

*(bx, by) – tọa độ điểm đến.*

### Thuật toán

Bài này khá dễ giải. Robot cần xuất phát từ điểm A(ax, ay) đi theo bản chỉ dẫn s để tìm đến điểm B. Hàm Where hướng dẫn cách đi.

Biến face cho biết hướng nhìn của robot. Tại điểm xuất phát, robot nhìn về hướng Bắc, ta có:

face = 0

Gặp chữ cái R robot cần đổi hướng 90 độ:

R: face = (face + 1) % 4

Bắc (0) 🡪 Đông (1) 🡪 Nam (2) 🡪 Tây (3) 🡪 Bắc (0)

Gặp chữ cái L robot cần đổi hướng ngược lại:

L: face = (face + 3) % 4

Bắc (0) 🡨 Đông (1) 🡨 Nam (2) 🡨 Tây (3) 🡨 Bắc (0)

Các giá trị trong mảng dx và dy được tính sẵn và biểu thị sự thay đổi tọa độ theo trục Ox và Oy tùy theo robot đang hướng về phía nào: Bắc (0), Đông (1), Nam (2) hay Tây(3).

# Bac Dong Nam Tay

dx[4] = { 0, 1, 0, -1 };

dy[4] = { 1, 0, -1, 0 };

Giả sử robot đang đứng ở điểm C(cx, cy). Khi gặp chữ cái F robot đi thêm 1 đơn vị dài theo hường nhìn. Ta có,

* Theo hướng Bắc: tọa độ y được tăng thêm 1, tọa độ x giữ nguyên:

cy += 1

* Theo hướng Đông: tọa độ x được tăng thêm 1, tọa độ y giữ nguyên:

cx += 1

* Theo hướng Nam: tọa độ y được giảm bớt 1, tọa độ x giữ nguyên:

cy -= 1

* Theo hướng Tây: tọa độ x được giảm bớt 1, tọa độ y giữ nguyên:

cx -= 1

* Tổng quát ta có

cx += dx[face]; cy += dy[face]

A picture containing text, light

Description automatically generated

Ví dụ

Nếu robot xuất phát từ điểm A(-3, 2) mắt nhìn về hướng Bắc và đi theo bản chỉ dẫn RFRFFLF thì sẽ đến điểm B(-1,0).

Độ phức tạp tính toán cỡ n, với n là số kí tự trong bản chỉ đường.

### Chương trình

# Robot

# Bac Dong Nam Tay

dx = [ 0, 1, 0, -1 ]

dy = [ 1, 0, -1, 0 ]

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def Where(ch):

global face, x, y

if ch == 'R':

face = (face + 1) % 4

return

if ch == 'L':

face = (face + 3) % 4

return

if ch == 'F':

x += dx[face]

y += dy[face]

def Robot(ax, ay, s):

global x, y, face

x, y = ax, ay

face = 0

print(' Begin point', ax, ay)

print(' Path:',s)

for ch in s: Where(ch)

print(' End point', x, y)

# APPLICATION

Robot(-3, 2, "RFRFFLF")

print(" T h e E n d")

### Output

Begin point -3 2

Path: RFRFFLF

End point -1 0

T h e E n d

# Luyện tập Robot

## Lost

*AIC 2007* *(Australia)*

*Box and whisker chart

Description automatically generatedRobot xuất phát từ điểm gốc tọa độ Descartes A(0,0), mặt quay về hướng Bắc, tức là theo trục tung Oy và cần đi theo một bản chỉ đường để đến tọa độ B(bx,by) cho trước. Bản chỉ đường là một dãy kí tự s gồm các chữ cái F, R và L viết liền nhau với ý nghĩa như sau:*

*F – đi thẳng theo 1 đơn vị dài, chẳng hạn 1 mét,*

*R – quay phải 90 độ,*

*L – quay trái 90 độ.*

*Điều không may là trong bản chỉ đường s thiếu mất một chữ cái R.*

*Bạn hãy cho biết chữ cái bị thiếu đứng thứ mấy trong dãy chữ cái của bản chỉ đường s. Trật tự các chữ cái trong bản chỉ đường được tính từ 0.*

*Dữ liệu input:*

*s: bản chỉ đường dưới dạng xâu kí tự và thiếu 1 chữ cái R,*

*ax ay: tọa độ của điểm xuất phát A*

*bx by: tọa độ của điểm đích B*

Ví dụ

ax = ay = 0 # Điểm xuất phát

face = 0

bx = 2; by = 1 # Đích cần đến

s = "RFLFFFLLF" # Bản chỉ đường

# Đáp số: Lost = 4

# Bản chỉ đường thiếu chữ cái R tại vị trí thứ 4

# s = "RFLF**R**FFLLF"

## Smart Robot

Chart, box and whisker chart

Description automatically generatedNgười ta lập trình cho một Robot xuất phát từ điểm A(ax,ay) trên mặt phẳng tọa độ Descartes, mặt quay về hướng Bắc, tức là theo trục tung Oy và cần đi theo một bản chỉ đường để đến điểm đích B. Bản chỉ đường là một dãy kí tự gồm các chữ cái F, R và L viết liền nhau thể hiện các lệnh với ý nghĩa như sau:

F – đi thẳng theo 1 đơn vị dài, chẳng hạn 1 mét,

R – quay phải 90 độ,

L – quay trái 90 độ.

Bạn hãy cung cấp một bản chỉ đường khác với ít chữ cái nhất mà vẫn đảm bảo cho Robot đến được đích B.

Input:

s – bản chỉ đường dưới dạng xâu kí tự.

(ax, ay) – tọa độ điểm xuất phát.

Output:

u – bản chỉ đường ngắn nhất do bạn đề xuất.

Ví dụ

Xuất phát: 1, 2

Chỉ đường: RFRFFLFLFR

Đích: 3, 1

Đường ngắn nhất: RFFRF

## Box and whisker chart Description automatically generatedHành tinh Alpha

Người ta lập trình cho hai Robot thám hiểm trên hành tinh Alpha. Robot thư nhất xuất phát từ điểm A(ax,ay) trên mặt phẳng tọa độ Descartes, mặt quay về hướng Bắc, tức là theo trục tung Oy và cần đi theo một bản chỉ đường để đánh dấu các địa điểm cần khoan thăm dò. Bản chỉ đường là một dãy kí tự gồm các chữ cái F, R, L và K viết liền nhau thể hiện các lệnh với ý nghĩa như sau:

F – đi thẳng theo 1 đơn vị dài, chẳng hạn 1 mét,

R – quay phải 90 độ,

L – quay trái 90 độ,

K – đánh dấu vị trí cần khoan thăm dò.

Robot thư hai cũng xuất phát từ điểm A(ax,ay) trên cùng mặt phẳng tọa độ Descartes, mặt quay về hướng Bắc có nhiệm vụ tìm đến các địa điểm cần khoan theo đúng trật tự do robot thứ nhất đã chỉ định để tiến hành khoan. Bạn hãy cung cấp một bản chỉ đường với ít chữ cái nhất để robot thứ hai hoàn thành được nhiệm vụ.

Input:

s – bản chỉ đường dưới dạng xâu kí tự.

(ax, ay) – tọa độ điểm xuất phát.

Output:

u – bản chỉ đường ngắn nhất do bạn đề xuất.

Cả hai bản chỉ đường s và u đều phải chứa các chữ K thể hiện địa điểm cần khoan.

Ví dụ

Xuất phát: 2, 0

s = FFRFRFKLFFLFK, len = 13

u = FRFKFFLFK, len = 9

# Bài giải Robot

## Lost

### Algorithm

#### Phương án 1. Vét cạn.

Độ phức tạp: với *n* là chiều dài của bản chỉ đường.

Lặp i = 0..n lần, mỗi lần thực hiện như sau:

* Thêm kí tự R vào vị trí i của bản chỉ đường s để thu được bản chỉ đường w.
* Xuất phát từ A theo bản chỉ đường w tìm đến điểm C.
* Nếu điểm C = B thì w sẽ là bản chỉ đường hoàn chỉnh.

#### Phương án 2. Vận dụng phép quay.

Độ phức tạp n.

Thuật toán gồm hai pha:

Pha 1. Xuất phát từ A(0,0) theo bản chỉ dẫn s khuyết kí tự R tìm đến điểm E. Pha này sẽ lưu lại đường đi a bao gồm dãy điểm liên tiếp từ A đến E.

Như vậy điểm E có thể lệch với điểm đích B. Do bản chỉ dẫn s khuyết một kí tự R nên cần quay điểm E một góc 90 độ theo chiều kim đồng hồ. Thời điểm quay diễn ra vào lúc nào? Dễ thấy thời điểm đó sẽ trùng với thời điểm phải xuất hiện kí tự R trong bản chỉ dẫn khuyết. Do đó ta cần thực hiện pha 2 kèm thêm phép quay mặt phẳng.

Pha 2. Mỗi bước i = 0..len(s), ta thử quay điểm E quanh điểm a[i] qua phải 90 độ. Nếu phép quay này chuyển E đến được đích B thì i chính là một ứng viên cho vị trí thiếu lệnh R.

Chart, line chart

Description automatically generatedPhép quay điểm A(ax,ay) quanh gốc tọa độ (0,0) một góc 90 độ theo chiều kim đồng hồ để đến điểm M(mx, my) rất dễ thực hiện:

mx = ay

my = -ax

Ví dụ

Quay điểm A(5,2) một góc 90 quanh gốc toạ độ O(0,0)

sẽ thu được điểm (2, -5)

Nếu ta phải quay điểm C(cx,cy) quanh tâm A(ax,ay) một góc 90 độ theo chiều kim đồng hồ để đến điểm M(mx, my) thì ta thực hiện theo ba bước sau đây:

* Bước 1. Chuyển trục tọa độ về điểm A (tức là A sẽ trở thành gốc tọa độ). Khi đó điểm C sẽ có tọa độ mới là:

x1 = cx – ax

y1 = cy - ay

* Bước 2. Quay điểm (x1,y1) quanh gốc tọa độ A một góc 90 độ theo chiều kim đồng hồ để nhận được điểm (x2,y2):

x2 = y1

y2 = -x1

* Bước 3. Chuyển lại trục tọa độ về gốc (0,0) cũ:

mx = x2 + ax

my = y2 + ay

Bạn có thể gộp ba bước trên như sau:

mx = ax – ay + cy

my = ax + ay - cx

Ví dụ

Quay điểm C(3,3) một góc 90 quanh điểm A(5,2)

sẽ thu được điểm (ax-ay+cy, ax+ay-cx) = (5-2+3, 5+2-3) = (6,4)

Chương trình dưới đây demo các phép quay phải 90o

# Rotate demo

X, Y = 0, 1

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

# Quay điểm c 90 độ quanh gốc tọa độ (0,0)

def Rotate(c): return c[Y], -c[X]

# Quay điểm c 90 độ quanh điểm A

def RotateA(a, c): return a[X]-a[Y]+c[Y], a[X]+a[Y]-c[X]

def Demo():

a = (5,2)

c = (3,3)

o = (0,0)

d = Rotate(a)

print(' \* Quay phải 90 độ điểm', a)

print(' quanh tâm O',o, ' -> ',d)

d = RotateA(a,c)

print(' \* Quay phải 90 độ điểm', c)

print(' quanh tâm',a, ' -> ',d)

# APPLICATION

Demo()

print(" T h e E n d")

### Output

\* Quay phải 90 độ điểm (5, 2)

quanh tâm O (0, 0) -> (2, -5)

\* Quay phải 90 độ điểm (3, 3)

quanh tâm (5, 2) -> (6, 4)

T h e E n d

### Chương trình

# Lost. Phương án 2: vận dụng phép quay

X, Y = 0, 1 # trục tọa độ

# Bắc Đông Nam Tây

dx = [ 0, 1, 0, -1 ]

dy = [ 1, 0, -1, 0 ]

huong = ['Bấc', 'Đông', 'Nam', 'Tây']

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

# Quay phải c 90 độ quanh gốc tọa độ (0,0)

def Rotate(c): return c[Y], -c[X]

# Quay phải điểm c 90 độ quanh điểm A

def RotateA(a, c): return a[X]-a[Y]+c[Y], a[X]+a[Y]-c[X]

# Xuát phát từ a, mắt nhìn theo hướng face,

# đi theo đường dẫn w sẽ đến điểm nào ?

def LastPoint(a, w, face = 0):

x, y = a

for ch in w:

if ch == 'R': face = (face + 1) % 4

elif ch == 'L': face = (face + 3) % 4

elif ch == 'F': x += dx[face]; y += dy[face]

return (x,y)

# Robot đi theo chỉ dẫn s khuyết R

# đến điểm e: Ghi nhận các tọa độ

def Phase1(a, s, face = 0):

x, y = a # xuất phát từ gốc tọa độ (0,0)

p = [(x,y)] # Ghi nhận điểm xuất phát

for ch in s:

if ch == 'R': face = (face + 1) % 4

elif ch == 'L': face = (face + 3) % 4

elif ch == 'F': x += dx[face]; y += dy[face]

p.append((x,y)) # Ghi nhận điểm trung gian

return p

# Quay e 90 độ tại các điểm trung gian trong a

# sao cho trùng với b

def Phase2(p, s, b):

e = p[-1] # điểm cuối của dãy a

print('Điểm đến theo dường khuyết R:',e)

for i in range(len(p)):

# quay e quanh a[i]

# mong nhận được b ?

if RotateA(p[i],e) == b:

# Kiểm tra

if LastPoint(p[0],s[:i]+'R'+s[i:]) == b:

return i

# Xuất phát từ gốc tọa độ (0,0) theo lost path s

# cần đến (bx,by). Tìm kí tự lost R

def Lost(a, b, s, face = 0):

print('Xuất phát:', a)

print(' Đích cần đến:', b)

print(' Bản chỉ đường khuyết R:',s)

print(' Hướng nhìn:', huong[face])

p = Phase1(a, s, face)

i = Phase2(p, s, b)

print(' Result:', i)

print(' Correct path:', s[:i]+'R'+s[i:])

# APPLICATION

Lost((0,0), (2,1), "RFLFFFLLF")

print(" T h e E n d")

### Output

Xuất phát: (0, 0)

Đích cần đến: (2, 1)

Bản chỉ đường khuyết R: RFLFFFLLF

Hướng nhìn: Bấc

Điểm đến theo dường khuyết R: (1, 2)

Result: 4

Correct path: RFLFRFFLLF

T h e E n d

### Gợi ý

Bạn cần test tại mỗi điểm quay hợp lý để đảm bảo tính chắc chắn của kết quả thu được.

## Rotate

Chúng ta xét hai phép quay thường gặp.

* Quay phải điểm A(ax, ay) quanh gốc toạ độ O(0,0) một góc 90 sẽ nhận được điểm C(cx, cy):

cx = ay; cy = -ax

* Quay trái điểm A(ax, ay) quanh gốc toạ độ O(0,0) một góc 90 sẽ nhận được điểm D(dx, dy):

dx = -ay; dy = ax

* Chuyển trục toạ độ
* Điểm C có toạ độ (cx,cy) trên mặt phẳng nhận điẻm O(0,0) làm gốc toạ độ. Nay chọn điểm A(ax, ay) làm gốc toạ độ thì toạ độ mới của điểm C sẽ là (cx’, cy’):

cx’ = cx - ax; cy’ = cy – ay

* Chuyển ngược lại: Điểm C có toạ độ (cx’, cy’) trên mặt phẳng nhận điểm A(ax, ay) làm gốc toạ độ nay chọn lại điểm O(0,0) làm gốc toạ độ thì toạ độ cũ (cx, cy) của điểm C sẽ là::

cx = cx’ + ax; cy = cy’ + ay

* Quay điểm B(bx, by) quanh điểm A(ax,ay) một góc 90 độ sẽ nhận được điểm C(cx, cy) theo các bước sau:
* Chuyển trục toạ độ O → A:

bx -= ax; by -= ay

* Quay B quanh gốc toạ độ A:

c = Rotate(b);

* Chuyển về trục toạ độ O:

cx += ax; cy -= ay;

Tuỳ theo dấu của góc quay mà ta có chiều quay xuôi (quay phải) hay ngược (quay trái) kim đồng hồ.

Ta viết các hàm Rotate với tham biến chỉ hướng quay, right = 1 là quay phải, nếu right = -1 thì quay trái.

### Chương trình

# Rotate

X, Y = 0, 1 # truc toa do X Y

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

# Quay điểm c 90 độ quanh gốc tọa độ (0,0)

# right = 1: quay phải, right = -1: quay trái

def Rotate(c, right = 1): # ngầm định la quay phải

return right\*c[Y], -right\*c[X]

# Quay c 90 độ quanh a

# right = 1: quay phải, right = -1: quay trái

def RotateA(a, c, right = 1): # ngam dinh la quay phai

return right\*(c[Y]-a[Y])+a[X], -right\*(c[X]-a[X])+a[Y]

def Demo():

o, a, c = (0,0), (2,5), (4,2)

print(' Quay phải',c,'quanh gốc tọa độ', o)

for i in range(4):

c = Rotate(c)

print(' . ', c)

print(' Quay trái',c,'quanh gốc tọa độ', o)

for i in range(4):

c = Rotate(c,-1)

print(' . ', c)

print(' Quay phải',c,'quanh', a)

for i in range(4):

c = RotateA(a, c)

print(' . ', c)

print(' Quay trái',c,'quanh', a)

for i in range(4):

c = RotateA(a, c,-1)

print(' . ', c)

# APPLICATION

Demo()

print(" T h e E n d")

### Output

Quay phải (4, 2) quanh gốc tọa độ (0, 0)

. (2, -4)

. (-4, -2)

. (-2, 4)

. (4, 2)

Quay trái (4, 2) quanh gốc tọa độ (0, 0)

. (-2, 4)

. (-4, -2)

. (2, -4)

. (4, 2)

Quay phải (4, 2) quanh (2, 5)

. (-1, 3)

. (0, 8)

. (5, 7)

. (4, 2)

Quay trái (4, 2) quanh (2, 5)

. (5, 7)

. (0, 8)

. (-1, 3)

. (4, 2)

T h e E n d

### Nâng cao

Bạn thử giải bài Lost với điều kiện sau đây: Bản chỉ đường khuyết một ký tự quay tại một vị trí nào đó, nhiệm vụ của bạn là chỉ ra ký tự khuyết (R hoặc L) và vị trí khuyết.

Ví dụ

Lost(2, 1, "RFLFFFLLF") # R, 4: RFLFRFFLLF

Lost(2, 1, "RFLFRFFLF") # L, 7: RFLFRFFLLF

## Smart Robot

### Algorithm

Mọi đường ngắn nhất từ A đến B theo các cạnh của ô đơn vị trong lưới đều có chiều dài bằng nửa chu vi của hình chữ nhật nhận A, B làm hai đỉnh đối diện: dài + rộng = |ax-bx|+|ay-by|.

|  |  |  |  |
| --- | --- | --- | --- |
| *A* |  |  | *Bản chỉ dẫn "RFFFFRFF" cho Robot*  *đi từ A đến B theo đường ngắn nhất*  *theo các cạnh của hình chữ nhật* [*A,B*].  *Tổng chiều dài sẽ là nửa chu vi của hình*  *chữ nhật* [*A,B*] = dài + rộng = 6*.* |
|  | A black and white grid  Description automatically generated |  |
|  | *B* |  |
|  |  |  |

Muốn đạt tối thiểu số lần rẽ (L hoặc R) thì đi theo các cạnh của hình chữ nhật nhận A và B làm hai đỉnh đối diện.

Gọi điểm xuất phát là A. Trước hết ta tìm điểm đích B theo bản chỉ dẫn s, sau đó ta xây dựng đường ngắn nhất theo nguyên tắc trên. Chú ý các trường hợp ax = bx và ay = by, khi AB là một điểm hoặc AB là đường thẳng song song với một trong hai trục toạ độ. Ngoài ra bạn cần lưu ý rằng, tại điểm xuất phát A robot nhìn theo hướng bắc.

Ta tạm kí hiệu F(d) là chỉ thị cho Robot đi thẳng theo trục x hoặc y một khoảng d, ví dụ, F(4) = “FFFF”.

Có tất cả 9 tình huống thể hiện sự tương quan giữa hai điểm: điểm xuất phát A(ax,ay) và điểm đích B(bx,by) như sau.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *A* • | *B* |  |  |  |  | *(1) AB là một điểm* |
|  |  |  |  |  |  | *Không làm gì* |
|  |  |  |  |  |  | *s = ""* |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | *A* |  | |  | |  | |  | | *B* | | *(2) AB là đường thẳng ngang* | | |  |  |  | |  | |  | |  | | *ax < bx; ay = by* | | |  |  |  | |  | |  | |  | | *s = "RF(bx-ax)"* | | |  |  |  | |  | |  | |  | |  | | | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | *B* |  |  |  |  | *A* | *(3) AB là đường thẳng ngang* | |  |  |  |  |  |  | *bx < ax; ay = by* | |  |  |  |  |  |  | *s = "LF(ax-bx)"* | |  |  |  |  |  |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | *A* |  |  |  |  |  | *(4) AB là đường thẳng dọc* | |  |  |  |  |  |  | *ax = bx; ay > by* | |  |  |  |  |  |  | *s = "RRF(ay-by)"* | | *B* |  |  |  |  |  |  | | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | *B* |  |  |  |  |  | *(5) AB là đường thẳng dọc* | |  |  |  |  |  |  | *ax = bx; ay < by* | |  |  |  |  |  |  | *s = "F(by-ay)"* | | *A* |  |  |  |  |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | *A* |  |  |  |  |  | *(6)* | |  |  |  |  |  |  | *ax < bx; ay > by* | |  |  |  |  |  |  | *s = "RF(bx-ax)RF(ay-by)"* | |  |  |  |  |  | *B* |  | | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | *A* | *(7)* | |  |  |  |  |  |  | *ax > bx; ay > by* | |  |  |  |  |  |  | *s = "LF(ax-bx)LF(ay-by)"* | | *B* |  |  |  |  |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | *B* | *(8)* | |  |  |  |  |  |  | *ax < bx; ay < by* | |  |  |  |  |  |  | *s = "F(by-ay)RF(bx-ax)"* | | *A* |  |  |  |  |  |  | | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | *B* |  |  |  |  |  | *(9)* | |  |  |  |  |  |  | *ax > bx; ay < by* | |  |  |  |  |  |  | *s = "F(by-ay)LF(ax-bx)"* | |  |  |  |  |  | *A* |  | |

### Chương trình

# Smart Robot

from random import randint

X, Y = 0, 1 # trục rọa độ X Y

# Bắc Đông Nam Tây

dx = [ 0, 1, 0, -1 ]

dy = [ 1, 0, -1, 0 ]

huong = ['Bác','Đông','Nam','Tây']

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

# đi thẳng n bước

def F(n): return 'F'\*n

# đường tối ưu: a->(s)->b

def OptimalPath(a, s):

global x, y, face

x, y = a[X], a[Y]

face = 0

for ch in s: Where(ch)

b = (x,y)

print(a, ' -> (', s, ') -> ', b)

if a == b: return ""

doc, ngang = F(abs(b[Y]-a[Y])), F(abs(b[X]-a[X]))

if a[X] == b[X]: # a thang cot voi b

return doc if a[Y] < b[Y] else "RR"+doc

if a[X] < b[X]: # a nam ben trai b: di ngang

if a[Y] == b[Y]: return "R"+ngang

return doc+"R"+ngang if a[Y] < b[Y] else "R"+ngang+"R"+doc

# a[X] > b[X]: a nam ben phai b

if a[Y] == b[Y]: return "L"+ngang

return doc+"L"+ngang if a[Y] < b[Y] else "L"+ngang+"L"+doc

def Where(ch):

global face, x, y

if ch == 'R': face = (face + 1) % 4; return

if ch == 'L': face = (face + 3) % 4; return

if ch == 'F': x += dx[face]; y += dy[face]

def Test1():

a = (-1,8)

s = "FRLR"

w = OptimalPath(a,s)

print(' Test 1: Optimal Path:',w) # (-1,9) F

a = (15,-1)

s = "RRRFFRFFFRLRFL"

w = OptimalPath(a,s)

print(' Test 1: Optimal Path:',w) # (14, 2) FFFLF

# sinh ngẫu nhiên string chứa n kí tự "LFR"

def Gen():

w = "LFR"

n = randint(3,20)

return "".join(w[randint(0,2)] for i in range(n))

# sinh du lieu ngau nhien

def Test2():

while True:

a = (randint(-10,10), randint(-10,10)) # diem a

s = Gen() # ban chi duong

w = OptimalPath(a,s)

print(' Test 2: Optimal Path:',w)

Go()

# APPLICATION

Test1()

Test2()

print(" T h e E n d")

### Output

(-1, 8) -> FRLR -> (-1, 9)

Test 1: Optimal Path: F

(15, -1) -> RRRFFRFFFRLRFL -> (14, 2)

Test 1: Optimal Path: FFFLF

(7, 6) -> FRFRLLRLLLR -> (8, 7)

Test 2: Optimal Path: FRF

(-3, 1) -> FFLLLFFL -> (-1, 3)

Test 2: Optimal Path: FFRFF

(-10, -10) -> LLLFLFRFRLLFFFFF -> (-8, -4)

Test 2: Optimal Path: FFFFFFRFF

T h e E n d

Độ phức tạp cỡ len(s).

## Box and whisker chart Description automatically generatedHành tinh Alpha

### Algorithm

Tương tự như bài Smart Robot.

Cho robot thứ nhất đi từ A theo s và ghi nhận các tọa độ cần khoan.

Khi gặp điểm cần khoan (bx,by) ta gọi robot thứ hai đi từ địa điểm khoan trước (kx,ky) theo đường ngắn nhất để đến điểm (bx,by), tức là theo các cạnh của hình chữ nhật nhận (kx,ky) và (bx,by) là hai đỉnh đối diện để thực hiện mũi khoan. Bạn cần cần cập nhật riêng rẽ hành vi (cấu hình) của hai robot. Cấu hình cho robot thứ nhất gồm:

Điểm xuất phát: ax, ay

Hướng nhìn: face

Điểm trung gian: bx, by

Cấu hình cho robot thứ hai gồm:

Điểm xuất phát: ax, ay

Hướng nhìn: kface

Điểm trung gian: kx, ky

### Chương trình

# Smart Robot

from random import randint

X, Y = 0, 1

# Bắc Đông Nam Tây

dx = [ 0, 1, 0, -1 ]

dy = [ 1, 0, -1, 0 ]

huong = ['Bắc','Đông','Nam','Tây']

Bac, Dong, Nam, Tay = 0, 1, 2, 3

def Go(msg=' ? '):

if input(msg) == '.': exit(0)

# đi thẳng n bước

def F(n): return 'F'\*n

# đường đi tối ưu a -> b theo hướng fc

# output: str

def OptimalPath(a,b,fc = 0): # ngầm định fc = 0 (hướng Bắc)

print(a,'(',huong[fc], ') ->',b)

doc = F(abs(a[Y]-b[Y])) # theo trục Y

ngang = F(abs(a[X]-b[X])) # theo trục X

if a == b: return '' # một điểm

if fc == Bac or fc == Nam:

if a[X] == b[X]: # ab theo chieu doc

if a[Y] < b[Y]: # a duoi b

return doc if fc == Bac else "RR"+doc

# a trên b

return "RR"+doc if fc == Bac else doc

if a[X] < b[X]: # a ben trai b

if a[Y] == b[Y]:

return "R"+ngang if fc == Bac else "L"+ngang

if a[Y] < b[Y]:

return doc+"R"+ngang if fc == Bac else "L"+ngang+"L"+doc

return "R"+ngang+"R"+doc if fc == Bac else doc+"L"+ngang

# a[X] > b[X] a bên phải b

if a[Y] == b[Y]:

return "L"+ngang if fc == Bac else "R"+ngang

if a[Y] < b[Y]:

return doc+"L"+ngang if fc == Bac else "R"+ngang+"R"+doc

# a[Y] > b[Y]

return "L"+ngang+"L"+doc if fc == Bac else doc+"L"+ngang

# fc = Đông or Tây

if a[X] == b[X]: # ab thang theo chieu doc

if a[Y] < b[Y]:

return "L"+doc if fc == Dong else "R"+doc

return "R"+doc if fc == Dong else "L"+doc

if a[X] < b[X]: # a trai b

if a[Y] == b[Y]:

return ngang if fc == Dong else "RR"+ngang

if a[Y] < b[Y]:

return ngang+"L"+doc if fc == Dong else "R"+doc+"R"+ngang

return ngang+"R"+doc if fc == Dong else "L"+doc+"L"+ngang

# ax > bx

if a[Y] == b[Y]:

return "RR"+ngang if fc == Dong else ngang

if a[Y] < b[Y]:

return "L"+doc+"L"+ngang if fc == Dong else ngang+"R"+doc

return "R"+doc+"R"+ngang if fc == Dong else ngang+"L"+doc

def Where(ch):

global face, x, y

if ch == 'R': face = (face + 1) % 4; return

if ch == 'L': face = (face + 3) % 4; return

if ch == 'F': x += dx[face]; y += dy[face]

# robot 1 xuat phat tu diem a, mat nhin fc

# di theo duong s1: chi dinh cac diem can khoan

def Robot1(a, fc, s1):

global x, y, face

x, y, face = a[X], a[Y], fc # xuat phat

# print(' Robot 1:')

khoan = [] # cac diem can khoan

for ch in s1:

if ch == 'K':

khoan.append((x,y))

continue

Where(ch)

return khoan

# Xuat phat : mat nhin theo huong fc

# di theo duong ti uu s

# ket thuc thi mat nhin theo huong nao

def GetFace(s, fc):

for ch in s:

if ch == "R": fc = (fc+1) % 4

else:

if ch == "L": fc = (fc+3) % 4

return fc

# robot 2 xuat phat tu a, mat quay ve huong fc

# di theo duong toi uu den cac diem khoan

def Robot2(a,fc,khoan):

# print(' Robot 2:')

print(' Cac diem can khoan:', khoan)

v = OptimalPath(a, khoan[0], fc)

fc = GetFace(v, fc)

s2 = v + "K"

for i in range(1, len(khoan)):

v = OptimalPath(khoan[i-1], khoan[i], fc)

fc = GetFace(v, fc)

s2 += v + "K"

return s2

def Alpha():

# diem xuat phat cua 2 robot

s1 = "FFRFRFKLFFLFK"

#s1 = "KFFRKFRFKLFFLFKK"

a = (2,0)

khoan = Robot1(a, 0, s1)

s2 = Robot2(a, 0, khoan)

print(' Result:', s2)

# APPLICATION

Alpha()

print(" T h e E n d")

### Output

Cac diem can khoan: [(3, 1), (5, 2)]

(2, 0) ( Bac ) -> (3, 1)

(3, 1) ( Dong ) -> (5, 2)

Result: FRFKFFLFK

T h e E n d

# Tài liệu tham khảo

1. Bùi Việt Hà, *Nhập môn thuật toán,* Nhà xuất bản Đại học Quốc gia Hà Nội, 2021.
2. Nguyễn Xuân Huy, *Sáng tạo trong thuật toán và lập trình, T1 – T7,* Nhà xuất bản Thông tin và Truyền thông*,* 2010 - 2022.
3. Nguyễn Xuân Huy, *Sáng tạo trong thuật toán và lập trình với Python* (tuyển các bài toán tin nâng cao cho học sinh và sinh viên giỏi), Nhà xuất bản Đại học Quốc gia Hà Nội, 2021.
4. [Crandall Richard](https://en.wikipedia.org/wiki/Richard_Crandall), [Pomerance Carl](https://en.wikipedia.org/wiki/Carl_Pomerance), [*Prime Numbers: A Computational Perspective*](https://books.google.com/books?id=RbEz-_D7sAUC&pg=PA121) (2nd ed.), Springer, p. 121, 2005, [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [9780387252827](https://en.wikipedia.org/wiki/Special:BookSources/9780387252827).
5. Ellis Horowitz, Sartal Sahni, *Fundamentals of Computer Algorithms*, PITMAN, 1978.
6. Donald E. Knuth, *The Art of ComputerPprogramming*, Addison Wesley PC, T1,2,3, 1974.
7. Kenneth H. Rosen, *Elementary Number Theory And Its Applications* (3rd ed.), Addison Wesley PC, 1992.
8. <https://stats.ioinformatics.org/>
9. <https://icpc.global/regionals/abouticpc>

17/04/2024