J212 - Paint the Floor

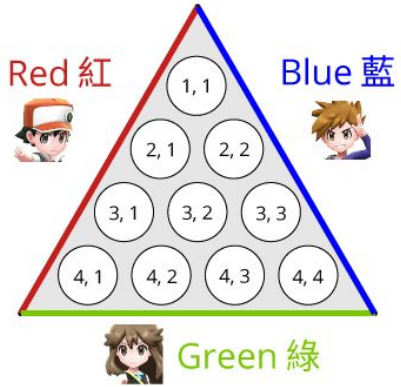
Percy Wong {percywtc}

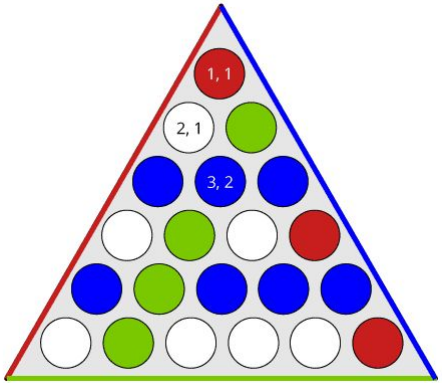
Background

Author: percywtc

Setters: s13215, christycty

Pictures: microtony

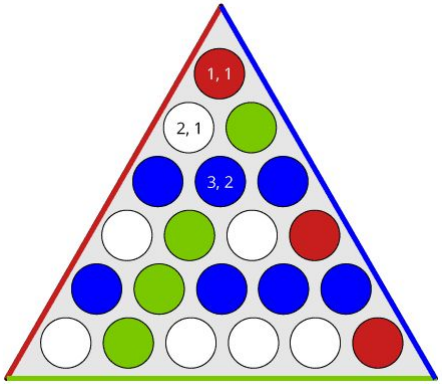


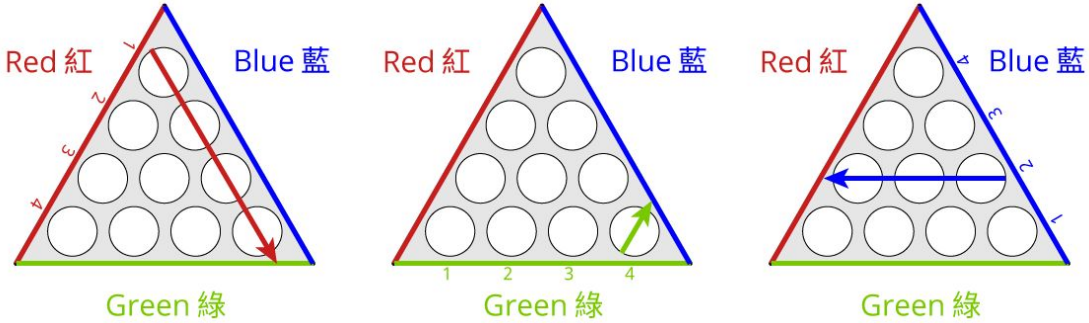


The Problem

Painting a triangular grid with 3 colors, in 3 directions

First perform some painting operations,
Then query the color of some cells





Subtasks

N = Height of the grid

P = Number of painting operations

Q = Number of queries

For all cases: $1 \leq N, P, Q \leq 200000$

	Points	Constraints
1	12	$1 \leq N, P, Q \leq 10$ Only blue paint is used
2	12	Only blue paint is used
3	15	$1 \leq N, P \leq 500$ Only green and blue paints are used
4	15	Only green and blue paints are used
5	23	$1 \leq N, P \leq 500$
6	23	No additional constraints

Statistics

12 points	6+2+0+0=8
24 points	2+3+0+0=5
27 points	0+4+0+0=4
39 points	0+ 1+0+0=1
50 points	2+5+4+0=11
62 points	0+2+2+0=4
77 points	0+ 1+ 1+0=2
100 points	0+ 1+9+8=18

First **Accepted** by **dbsboscowang** at **14m44s**

For all cases: $1 \leq N, P, Q \leq 200000$

	Points	Constraints
1	12	$1 \leq N, P, Q \leq 10$ Only blue paint is used
2	12	Only blue paint is used
3	15	$1 \leq N, P \leq 500$ Only green and blue paints are used
4	15	Only green and blue paints are used
5	23	$1 \leq N, P \leq 500$
6	23	No additional constraints

Subtasks Analysis

We can solve the task in two dimensions:

Subtasks	Blue	Blue, Green	Blue, Green, Red
N, P small	***1***	***1+3***	***1+3+5***
N, P large	***1+2***	***1+2+3+4***	***1+2+3+4+5+6***

Points	Blue	Blue, Green	Blue, Green, Red
N, P small	***12***	***27***	***50***
N, P large	***24***	***54***	***100***

For all cases: $1 \leq N, P, Q \leq 200000$

	Points	Constraints
1	12	$1 \leq N, P, Q \leq 10$ Only blue paint is used
2	12	Only blue paint is used
3	15	$1 \leq N, P \leq 500$ Only green and blue paints are used
4	15	Only green and blue paints are used
5	23	$1 \leq N, P \leq 500$
6	23	No additional constraints

Solution for N, P small

We can simply perform simulation of the painting operations

Store the color in a 2-dimensional array of characters


```
char[1..500][1..500] grid := {'W'...}      // set all as white

for each paint operation OP:
  for each cell (x,y) to be painted in OP:
    grid[x][y] := OP.color

for each query (x,y):
  print grid[x][y]
```

Solution for N, P small

We can simply perform simulation of the painting operations

Store the color in a 2-dimensional array of characters


```
char[1..500][1..500] grid := {'W'...}      // set all as white

for each paint operation OP:
  for each cell (x,y) to be painted in OP:
    grid[x][y] := OP.color

for each query (x,y):
  print grid[x][y]
```

How?

Solution for N, P small

How to find which cells to be painted?

Solution for N, P small - Blue

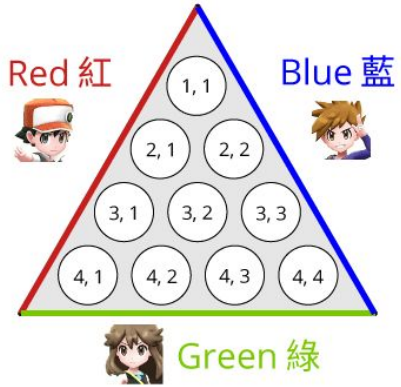
Blue operations paint cells of the same row

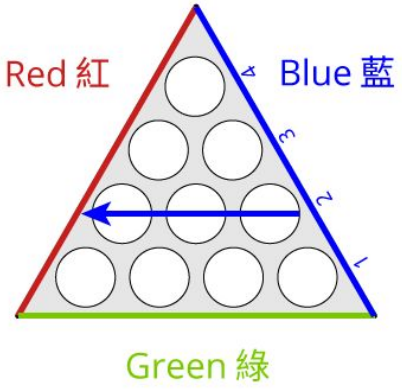
Which row?

Row (**N** + 1 - **L**)

L = 1	Row 4
L = 2	Row 3
L = 3	Row 2
L = 4	Row 1

```
row := N + 1 - L
for col from 1 to row:
  grid[row][col] := 'B'
```





Solution for N, P small - Green

Green operations paint cells of the same column

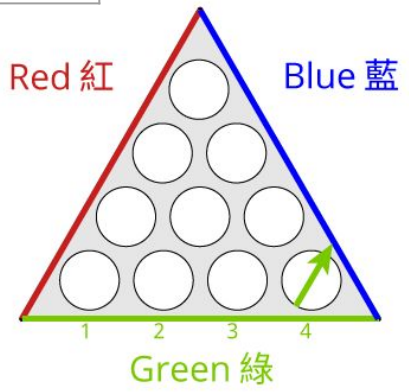
Column **L**

Which rows do column **L** has?

Row **L** to **N**

```
col := L
for row from L to N:
  grid[row][col] := 'G'
```

Column 1	Row 1 to 4
Column 2	Row 2 to 4
Column 3	Row 3 to 4
Column 4	Row 4 to 4



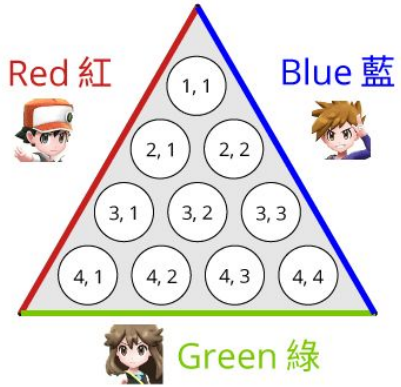
Solution for N, P small - Red

Red operations paint cells of the same column

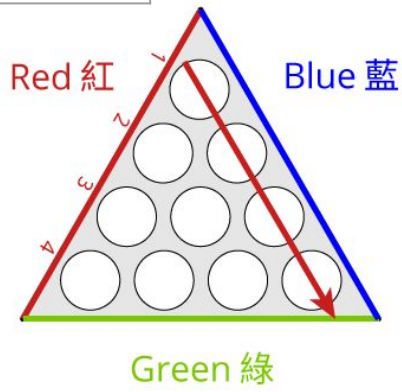
Starts at row **L**,

Column increases by 1 per row

L = 1	(1,1) (2,2) (3,3) (4,4)
L = 2	(2,1) (3,2) (4,3)
L = 3	(3,1) (4,2)
L = 4	(4,1)




```
col := 1
for row from L to N:
  grid[row][col] := 'R'
  col := col + 1
```



Solution for N, P large

For larger N and P, e.g. 200000

Worst case: need to update $200000 \times 200000 = 4 \times 10^{10}$ times, Time Limit Exceeded

How to speed up? Let's try to solve from subtasks!

Solution for N, P large - Blue

Blue operations paint cells of the same row

Painting Line **L** is Row (**N** + 1 - **L**),
in other words, row **X** is painted iff there exists some blue **L** = **N** + 1 - **X**

We can just memorize which lines are painted, which lines are not :)

```
bool[1..200000] bluePainted = {false...}  // set all as false

for each paint operation OP:
  bluePainted[OP.L] := true                  // mark which “L”s are painted

for each query (x,y):
  print ‘B’ if bluePainted[N+1-x] else ‘W’
```

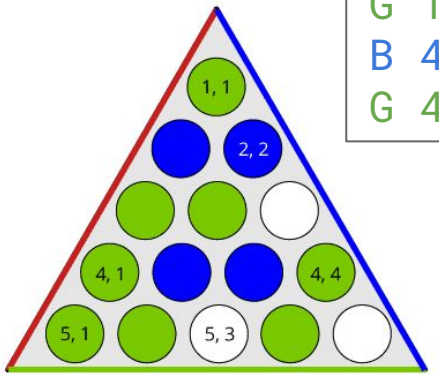
Solution for N, P large - Blue + Green

Can we apply similar idea as the previous one?

Column **Y** is painted iff there exists some green **L** = **Y**

What to do if `bluePainted[N+1-x]` and `greenPainted[y]` are both true?

G	2
B	2
B	5
G	1
B	4
G	4



Solution for N, P large - Blue + Green

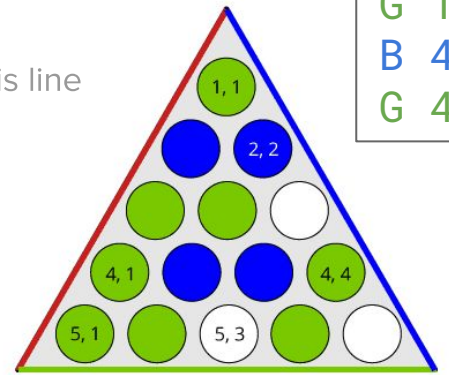
What to do if `bluePainted[N+1-x]` and `greenPainted[y]` are both true?

It actually depends on which color comes later

Instead of memorizing lines painted or not (boolean),
We can maintain the latest time painting the lines (integer)

```
for the i-th paint operation:
  if C[i]='B': blueLast[L[i]] := i    // update the latest time painting this line
  if C[i]='G': greenLast[L[i]] := i
```

G	2
B	2
B	5
G	1
B	4
G	4



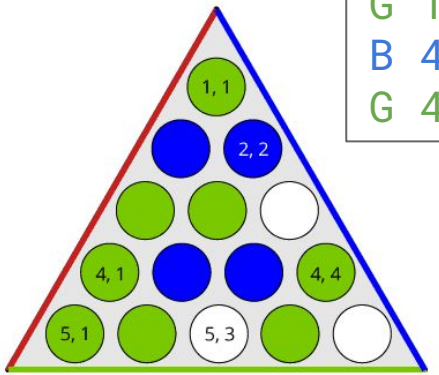
Solution for N, P large - Blue + Green

```
for the i-th paint operation:
  if C[i]='B': blueLast[L[i]] := i    // update the latest time painting this line
  if C[i]='G': greenLast[L[i]] := i
```

	1	2	3	4	5
blueLast		2		5	3
greenLast	4	1		6	

```
for each query (x,y):
  print 'B' if blueLast[N+1-x] > greenLast[y] else 'G'
// How do handle white (not painted)?
```

G	2
B	2
B	5
G	1
B	4
G	4

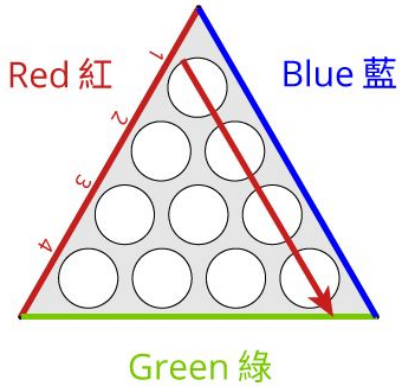


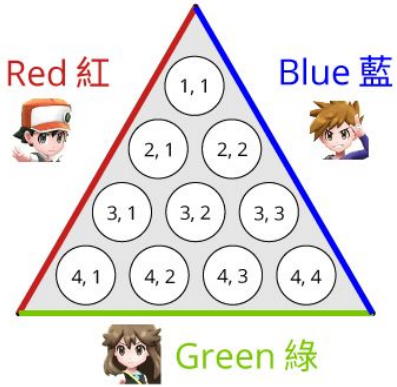
Solution for N, P large - Blue + Green + Red

We can do similar stuff with Red operations!

What kind of cells get affected when we paint Red on Line **L**?

L = 1	(1,1) (2,2) (3,3) (4,4)
L = 2	(2,1) (3,2) (4,3)
L = 3	(3,1) (4,2)
L = 4	(4,1)





Solution for N, P large - Blue + Green + Red

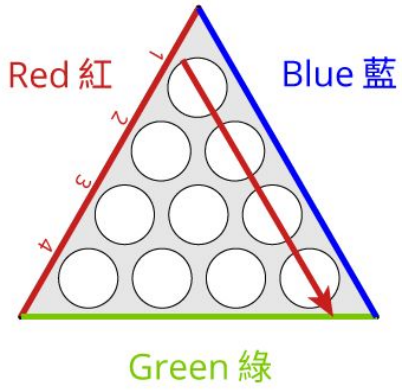
We can do similar stuff with Red operations!

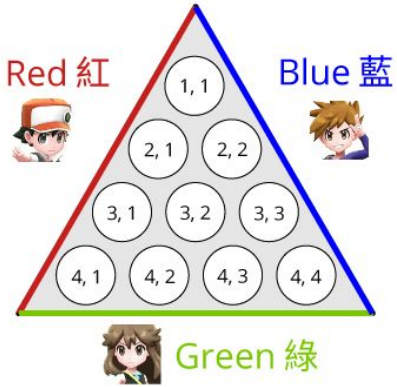
What kind of cells get affected when we paint Red on Line **L**?

It's when **x - y** + 1 = **L**

L = 1	(1,1) (2,2) (3,3) (4,4)
L = 2	(2,1) (3,2) (4,3)
L = 3	(3,1) (4,2)
L = 4	(4,1)

We can mark `redLast[1..200000]` similarly





Accepted