



# 인구 이동 (beakjoon 16234)

categories: 구현, 그래프 이론, 그래프 탐색, 너비 우선 탐색, 시뮬레이션

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	512 MB	51272	20176	11711	36.302%

## 문제

$N \times N$  크기의 땅이 있고, 땅은  $1 \times 1$  개의 칸으로 나누어져 있다. 각각의 땅에는 나라가 하나씩 존재하며,  $r$  행  $c$  열에 있는 나라에는  $A[r][c]$  명이 살고 있다. 인접한 나라 사이에는 국경선이 존재한다. 모든 나라는  $1 \times 1$  크기이기 때문에, 모든 국경선은 정사각형 형태이다.

오늘부터 인구 이동이 시작되는 날이다.

인구 이동은 하루 동안 다음과 같이 진행되고, 더 이상 아래 방법에 의해 인구 이동이 없을 때까지 지속된다.

- 국경선을 공유하는 두 나라의 인구 차이가  $L$  명 이상,  $R$  명 이하라면, 두 나라가 공유하는 국경선을 오늘 하루 동안 연다.
- 위의 조건에 의해 열어야 하는 국경선이 모두 열렸다면, 인구 이동을 시작한다.
- 국경선이 열려있어 인접한 칸만을 이용해 이동할 수 있으면, 그 나라를 오늘 하루 동안은 연합이라고 한다.
- 연합을 이루고 있는 각 칸의 인구수는 (연합의 인구수) / (연합을 이루고 있는 칸의 개수)가 된다. 편의상 소수점은 버린다.
- 연합을 해체하고, 모든 국경선을 닫는다.

각 나라의 인구수가 주어졌을 때, 인구 이동이 며칠 동안 발생하는지 구하는 프로그램을 작성하시오.

## 입력

첫째 줄에  $N, L, R$ 이 주어진다. ( $1 \leq N \leq 50, 1 \leq L \leq R \leq 100$ )

둘째 줄부터  $N$ 개의 줄에 각 나라의 인구수가 주어진다.

$r$  행  $c$  열에 주어지는 정수는  $A[r][c]$ 의 값이다. ( $0 \leq A[r][c] \leq 100$ )

인구 이동이 발생하는 일수가 2,000번 보다 작거나 같은 입력만 주어진다.

## 출력

인구 이동이 며칠 동안 발생하는지 첫째 줄에 출력한다.



# 인구 이동

(beakjoon 16234)

categories: 구현, 그래프 이론, 그래프 탐색, 너비 우선 탐색, 시뮬레이션

## 입력1

```
2 20 50
50 30
20 40
```

## 출력1

```
1
```

초기 상태는 아래와 같다.

50	30
20	40

$L = 20$ ,  $R = 50$  이기 때문에, 모든 나라 사이의 국경선이 열린다. (열린 국경선은 점선으로 표시)

50	30
20	40

연합은 하나 존재하고, 연합의 인구는  $(50 + 30 + 20 + 40)$  이다. 연합의 크기가 4이기 때문에, 각 칸의 인구수는  $140/4 = 35$ 명이 되어야 한다.

35	35
35	35



# 인구 이동

(beakjoon 16234)

categories: 구현, 그래프 이론, 그래프 탐색, 너비 우선 탐색, 시뮬레이션

## 입력2

2 40 50  
50 30  
20 40

## 출력2

1

초기 상태는 아래와 같다.

50	30
30	40

L = 20, R = 50이기 때문에, 아래와 같이 국경선이 열린다.

50	30
30	40

합쳐져 있는 연합의 인구수는  $(50+30+30) / 3 = 36$  (소수점 버림)이 되어야 한다.

36	36
36	40



# 인구 이동

(beakjoon 16234)

categories: 구현, 그래프 이론, 그래프 탐색, 너비 우선 탐색, 시뮬레이션

## 입력3

```
2 40 50
50 30
20 40
```

## 출력3

```
0
```

경계를 공유하는 나라의 인구 차이가 모두 L보다 작아서 인구 이동이 발생하지 않는다.

## 입력4

```
3 5 10
10 15 20
20 30 25
40 22 10
```

## 출력4

```
2
```

## 입력5

```
4 10 50
10 100 20 90
80 100 60 70
70 20 30 40
50 20 100 10
```

## 출력5

```
3
```



Woody K

Contact: [woody35545@gmail.com](mailto:woody35545@gmail.com)

Github: <https://github.com/woody35545>

CNU, Computer Science and Engineering



# 인구 이동 (beakjoon 16234)

categories: 구현, 그래프 이론, 그래프 탐색, 너비 우선 탐색, 시뮬레이션

## Answer(1/2)

```

1  from collections import deque
2
3  graph = []
4
5  N,L,R = map(int,input().split(" "))
6  for _ in range(N):
7      graph.append(list(map(int, input().split(" "))))
8
9
10 ROW_MAX, COL_MAX = N,N
11 visited = [[0]*COL_MAX for _ in range(ROW_MAX)]
12
13 DIRECTIONS = [(1,0),(-1,0),(0,1),(0,-1)]
14
15 is_valid = lambda x,y : 0<=x<ROW_MAX and 0<=y<COL_MAX
16 can_open = lambda x1,y1,x2,y2 : L <= abs(graph[x1][y1]-graph[x2][y2]) <= R
17
18 stage_count = 0
19 def bfs(start_vertex:tuple):
20     global graph, visited
21     block_elements = deque([]) # 현재 블록 원소들의 좌표를 저장하기 위한 덱
22     queue = deque([start_vertex])
23     open_wall_occur = False
24     while queue:
25         cx,cy = queue.popleft()
26
27         if not visited[cx][cy]:
28             visited[cx][cy] = 1
29             block_elements.append((cx,cy))
30
31             for k in range(len(DIRECTIONS)):
32                 nx,ny = cx + DIRECTIONS[k][0], cy+ DIRECTIONS[k][1]
33
34                 if is_valid(nx,ny) and can_open(cx,cy,nx,ny) and visited[nx][ny] == 0:
35                     open_wall_occur = True
36                     queue.append((nx,ny))
37     if open_wall_occur:
38         # initialize
39         current_block_total = 0
40         current_block_total_length = len(block_elements)
41
42         for i in range(len(block_elements)):
43             current_block_total += graph[block_elements[i][0]][block_elements[i][1]]
44
45         new_value = int(current_block_total / current_block_total_length)
46         for i in range(len(block_elements)):
47             cur_x, cur_y = block_elements[i]
48             graph[cur_x][cur_y] = new_value
49
50     return open_wall_occur

```





# 인구 이동

(beakjoon 16234)

categories: 구현, 그래프 이론, 그래프 탐색, 너비 우선 탐색, 시뮬레이션

## Answer(2/2)

```
72 def solve():
73     global stage_count
74     stage_count = 0
75     while True:
76         open_wall_occur = next_stage()
77
78         if not open_wall_occur:
79             break
80
81     print(stage_count)
82
83
84 solve()
```



Woody K

Contact: [woody35545@gmail.com](mailto:woody35545@gmail.com)

Github: <https://github.com/woody35545>

CNU, Computer Science and Engineering