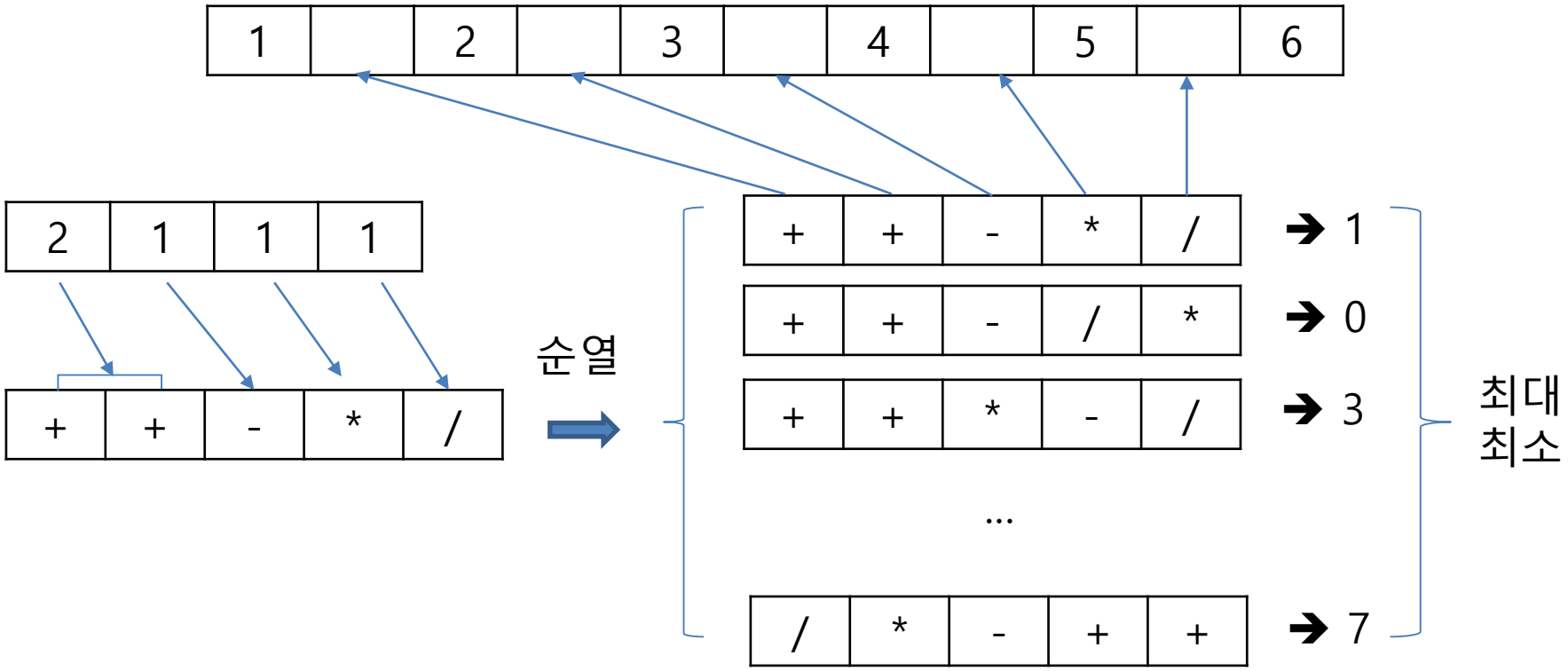
완전검색 : 백트래킹

AD 보충수업 3일차

연사자끼워넣기

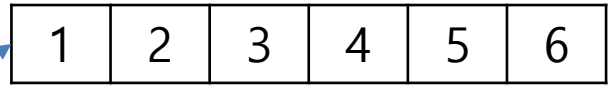
https://www.acmicpc.net/problem/14888

연사자끼워넣기

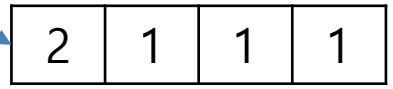


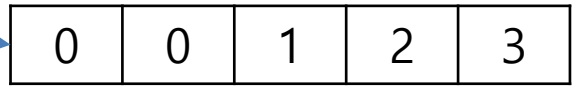
연사자끼워넣기

```
N = int(input())
nums = list(map(int, input().split()))
opc = list(map(int, input().split()))

ops = []
for i in range(4):
    ops += [i] * opc[i]

maxans, minans = -1e10, 1e10
solve(0)
print("%d₩n%d" % (maxans, minans))
```



(+, -, *, / 의 개수)



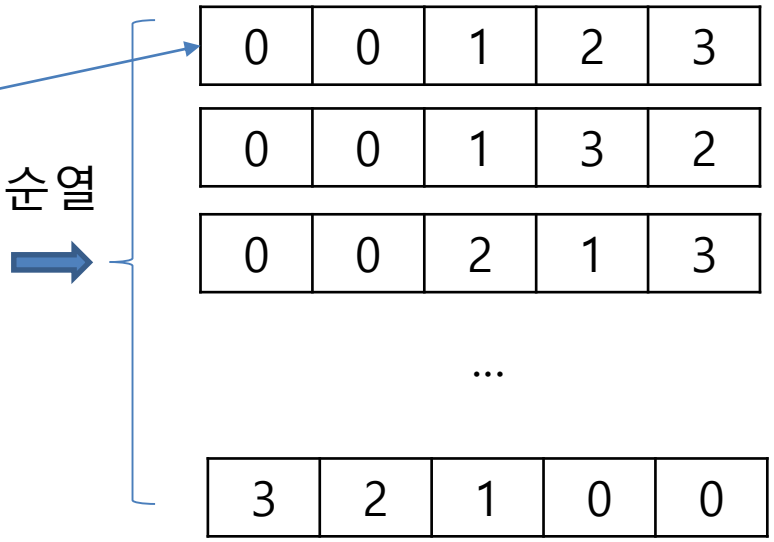


(+,+,-,*,/)

연사자끼워넣기

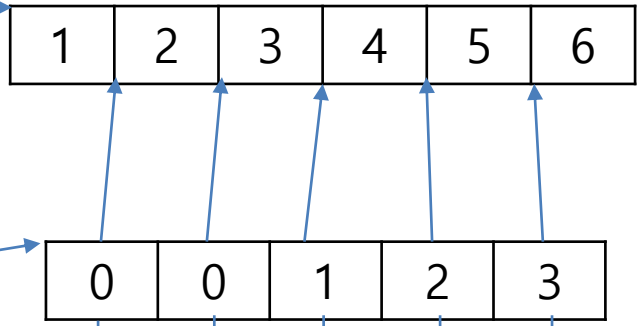
```
def solve(k):
    global maxans, minans
    if k == N - 1:
        ...
    else:
        for i in range(k, N - 1):
            ops[k], ops[i] = ops[i], ops[k]
            solve(k + 1)
            ops[k], ops[i] = ops[i], ops[k]
```

순열



연사자끼워넣기

```
def solve(k):
    global maxans, minans
    if k == N - 1:
        val = nums[0]
        for i in range(N - 1):
            if ops[i] == 0:
                val += nums[i + 1]
            elif ops[i] == 1:
                val -= nums[i + 1]
            elif ops[i] == 2:
                val *= nums[i + 1]
            else:
                val = int(val / nums[i + 1])
        maxans = max(maxans, val)
        minans = min(minans, val)
    else:
        ...
```

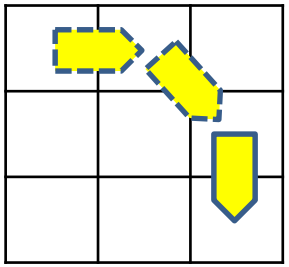


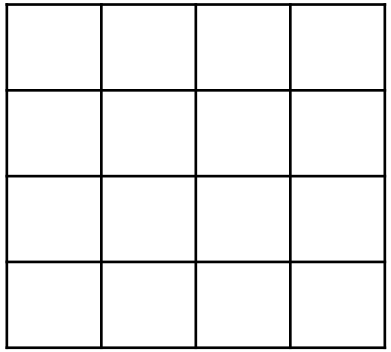
= 1

파이프 옮기기1

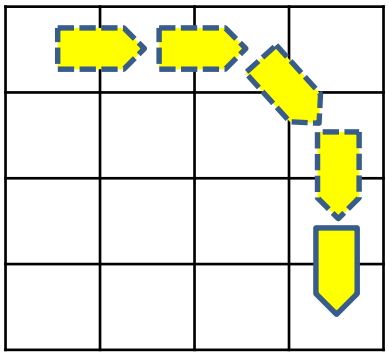
https://www.acmicpc.net/problem/17070

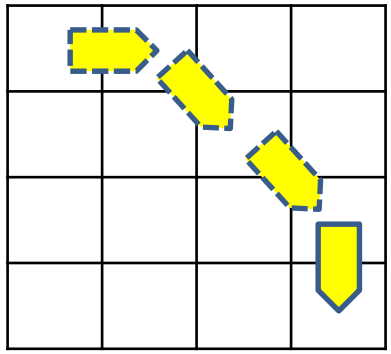
파이프 옮기기1

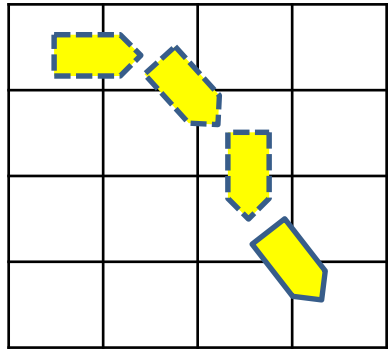




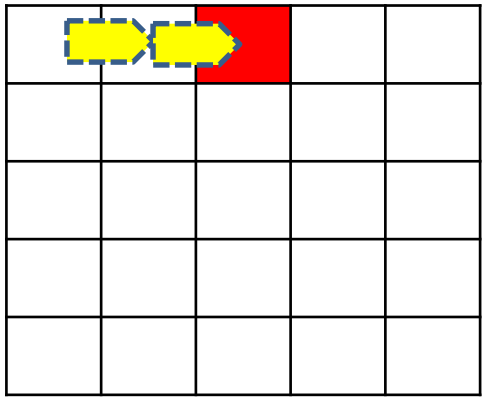
파이프 옮기기1

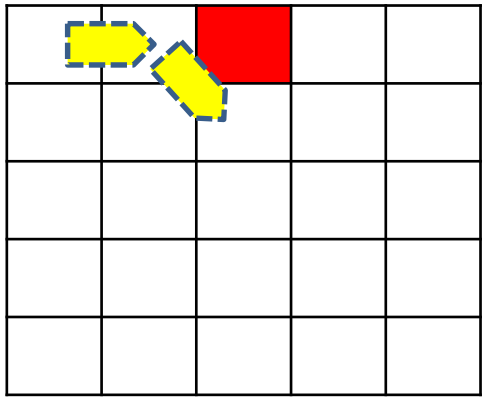




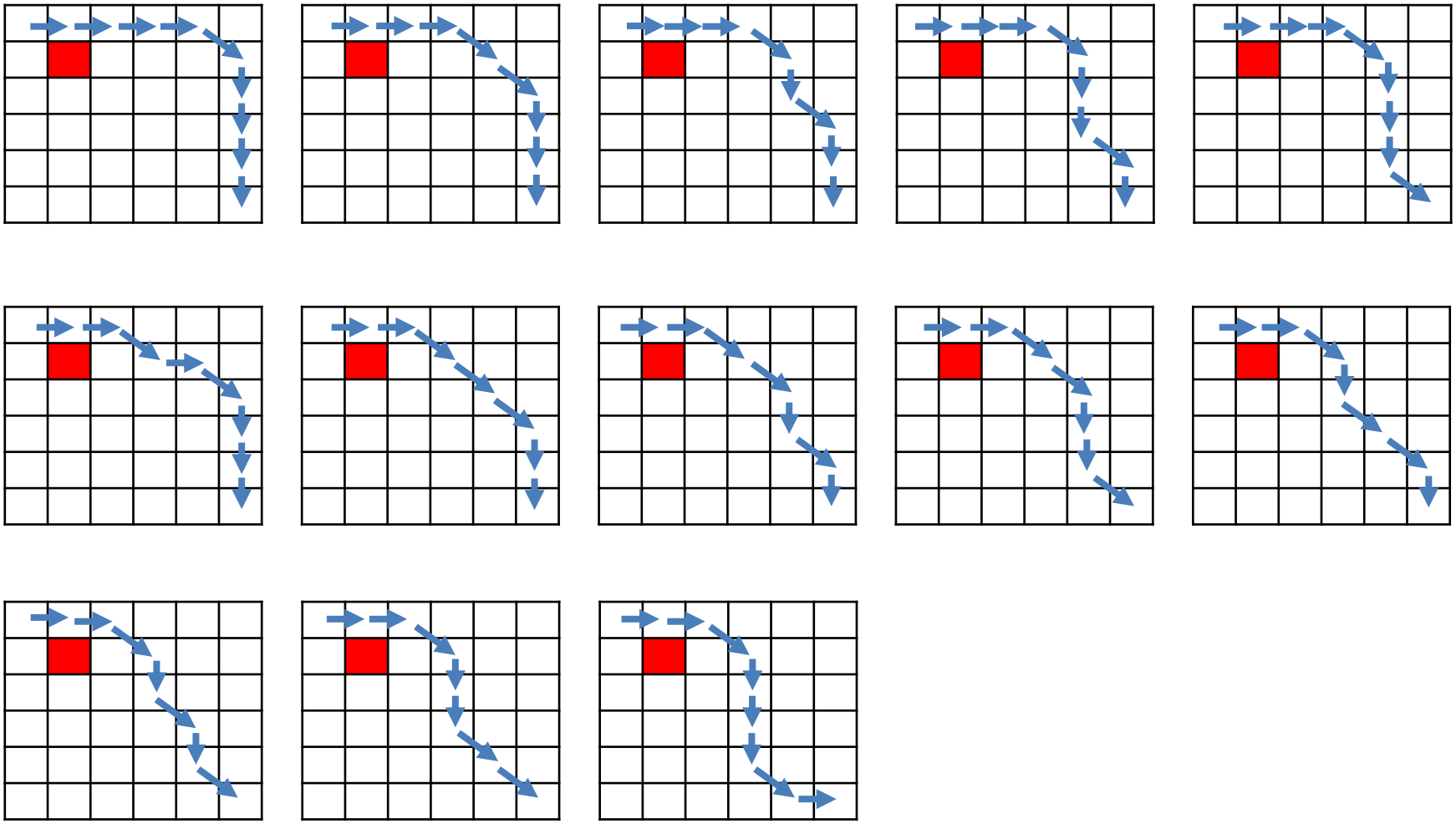


파이프 옮기기1



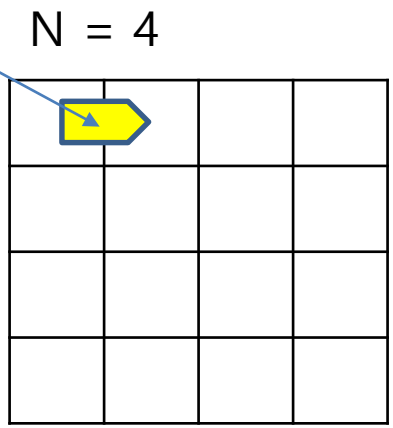


파이프 옮기기1



파이프 옮기기1

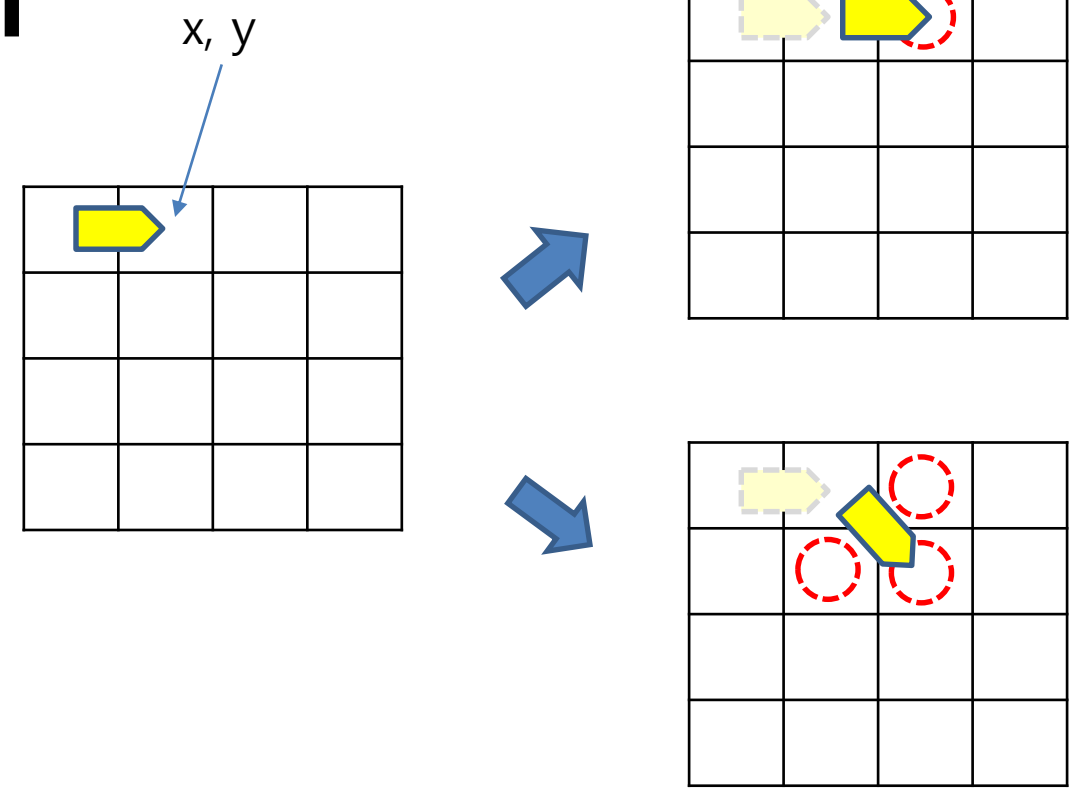
```
N = int(input())
mat = [[*map(int, input().split())] for _ in range(N)]
ans = 0
solve(0, 1, 0)  #(x,y,d) d: →0, ↓1, ↘2
print(ans)
```



파이프 옮기기1

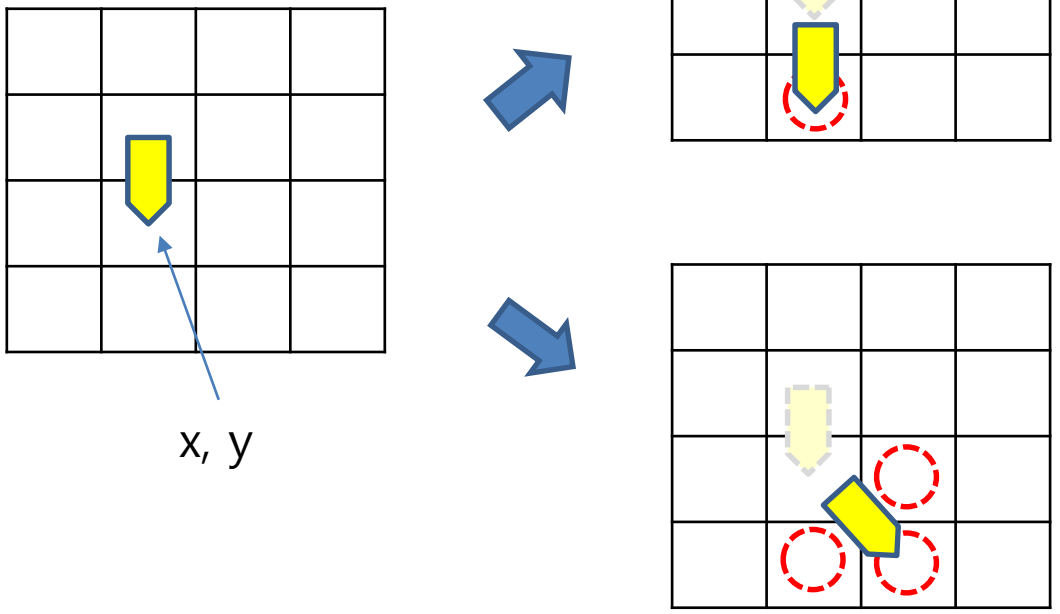
```
def solve(x, y, d):  # d: → 0, ↓ 1, ↘ 2
    global ans
    if x == N - 1 and y == N - 1:
        ans += 1
```

파이프 옮기기1



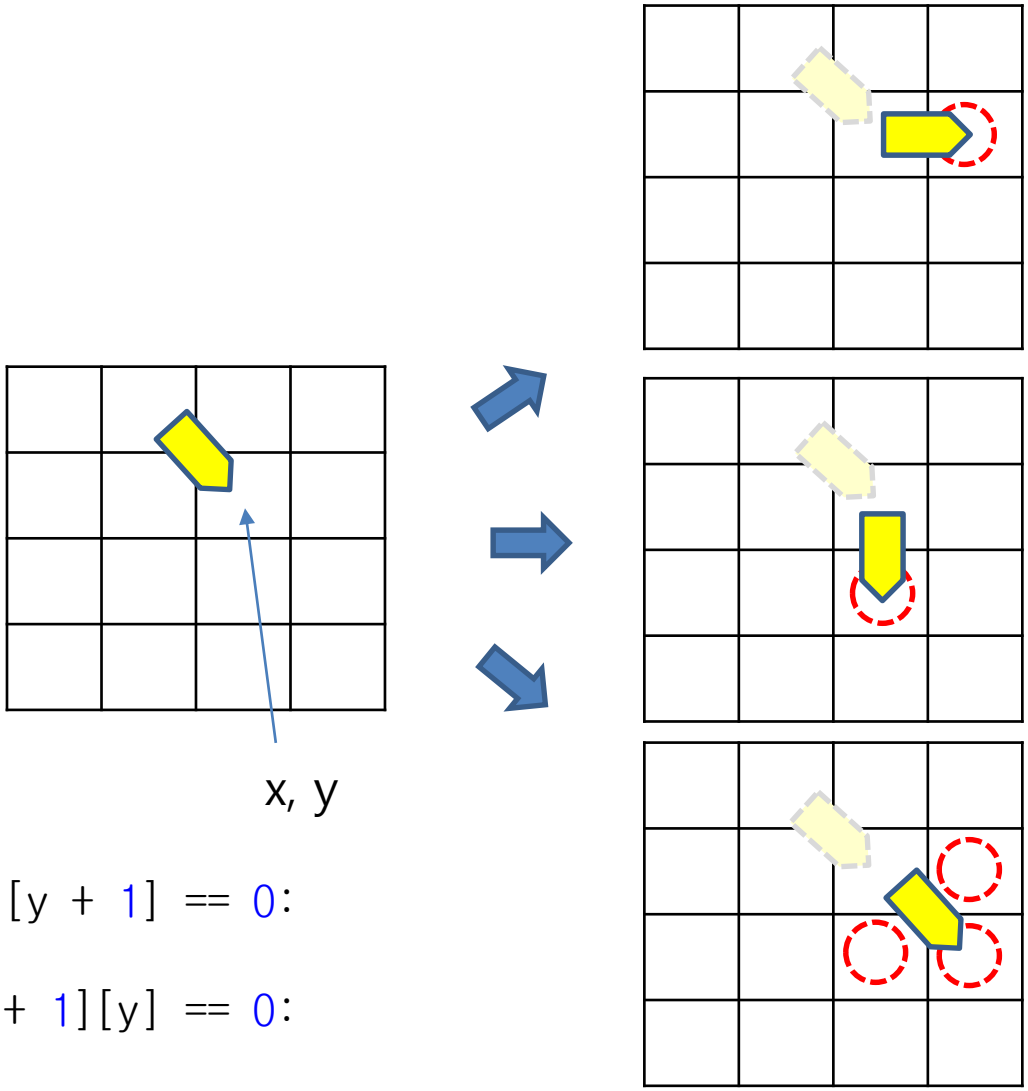

```
# d: → 0, ↓ 1, ↘ 2
def solve(x, y, d):
    global ans
    …

    if d == 0:
        if y + 1 < N and mat[x][y + 1] == 0:
            solve(x, y + 1, 0)
        if x + 1 < N and y + 1 < N and ₩
                mat[x + 1][y] == mat[x][y + 1] == mat[x + 1][y + 1] == 0:
            solve(x + 1, y + 1, 2)

    …
```

파이프 옮기기1




```
# d: → 0, ↓ 1, ↘ 2
def solve(x, y, d):
    global ans

    ...

    if d == 1:
        if x + 1 < N and mat[x + 1][y] == 0:
            solve(x + 1, y, 1)
        if x + 1 < N and y + 1 < N and ₩
                mat[x + 1][y] == mat[x][y + 1] == mat[x + 1][y + 1] == 0:
            solve(x + 1, y + 1, 2)

    ...
```

파이프 옮기기1




```
# d: → 0, ↓ 1, ↘ 2
def solve(x, y, d):
    global ans
    ...

    if d == 2:
        if y + 1 < N and mat[x][y + 1] == 0:
            solve(x, y + 1, 0)
        if x + 1 < N and mat[x + 1][y] == 0:
            solve(x + 1, y, 1)
        if x + 1 < N and y + 1 < N and ₩
                mat[x + 1][y] == mat[x][y + 1] == mat[x + 1][y + 1] == 0:
            solve(x + 1, y + 1, 2)
```

캐슬디펜스

https://www.acmicpc.net/problem/17135

캐슬디펜스

1번예제

5, 5
길이 d = 1

5곳에서 3군데 고르는
경우의 수 ${}_5C_3$ = 10

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
︽	︽	︽		

3

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
︽	︽		︽	

3

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
︽	︽			︽

3

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
︽		︽	︽	

3

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
︽		︽		︽

3

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
︽			︽	︽

3

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
	︽	︽	︽	

3

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
	︽	︽		︽

3

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
	︽		︽	︽

3

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
		︽	︽	︽

3

최대값 3

캐슬디펜스

2번예제

5, 5
길이 d = 1

5곳에서 3군데 고르는 경우의 수 $_5C_3 = 10$

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
^	^	^		

길이가 1 아래로 내려올때 까지 기다려야 한다.

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
^	^	^		

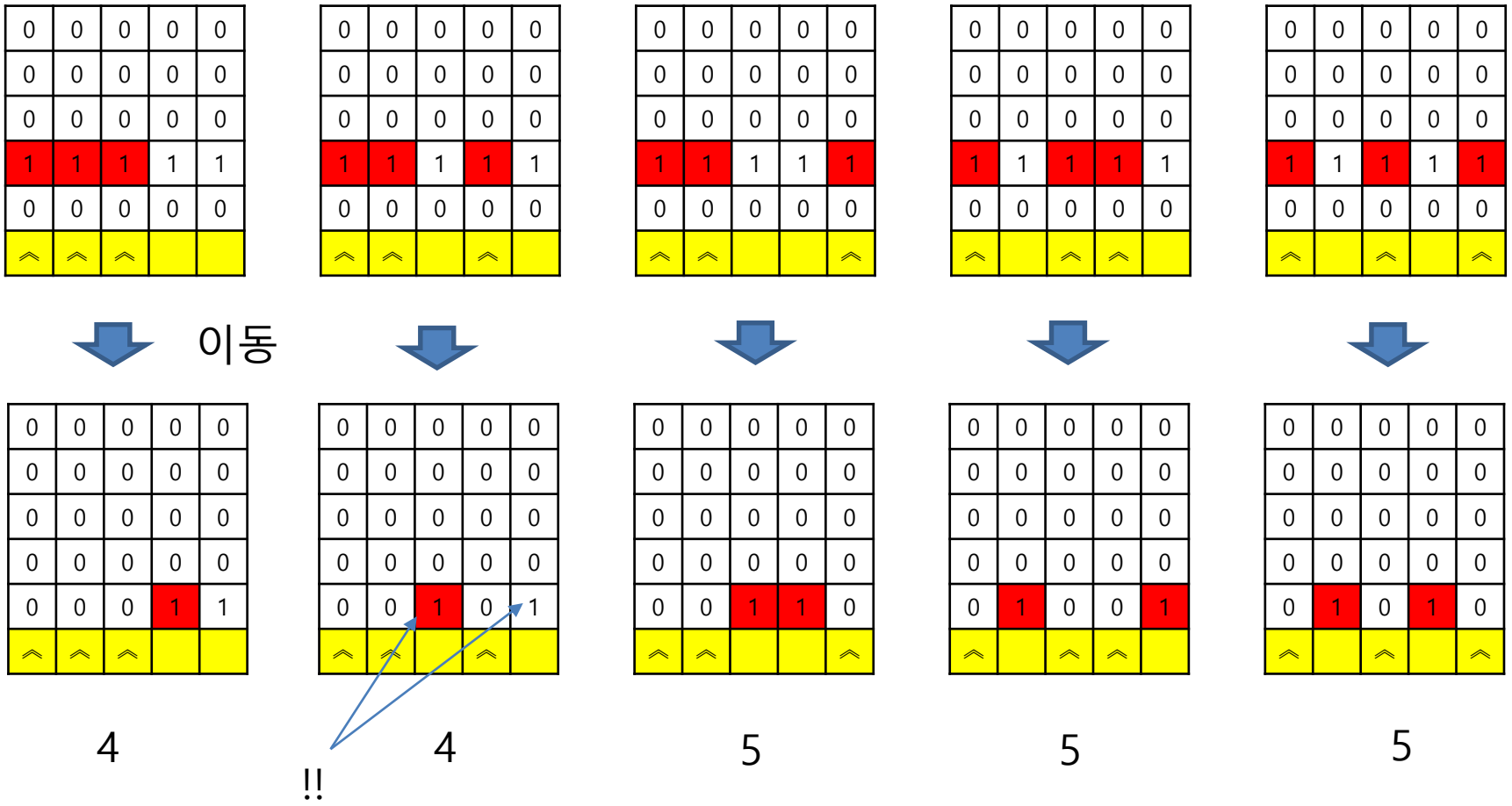
첫 번째 문제와 같은 상황

캐슬디펜스

3번예제

5, 5
길이 d = 2

5곳에서 3군데 고르는 경우의 수 $_5C_3 = 10$

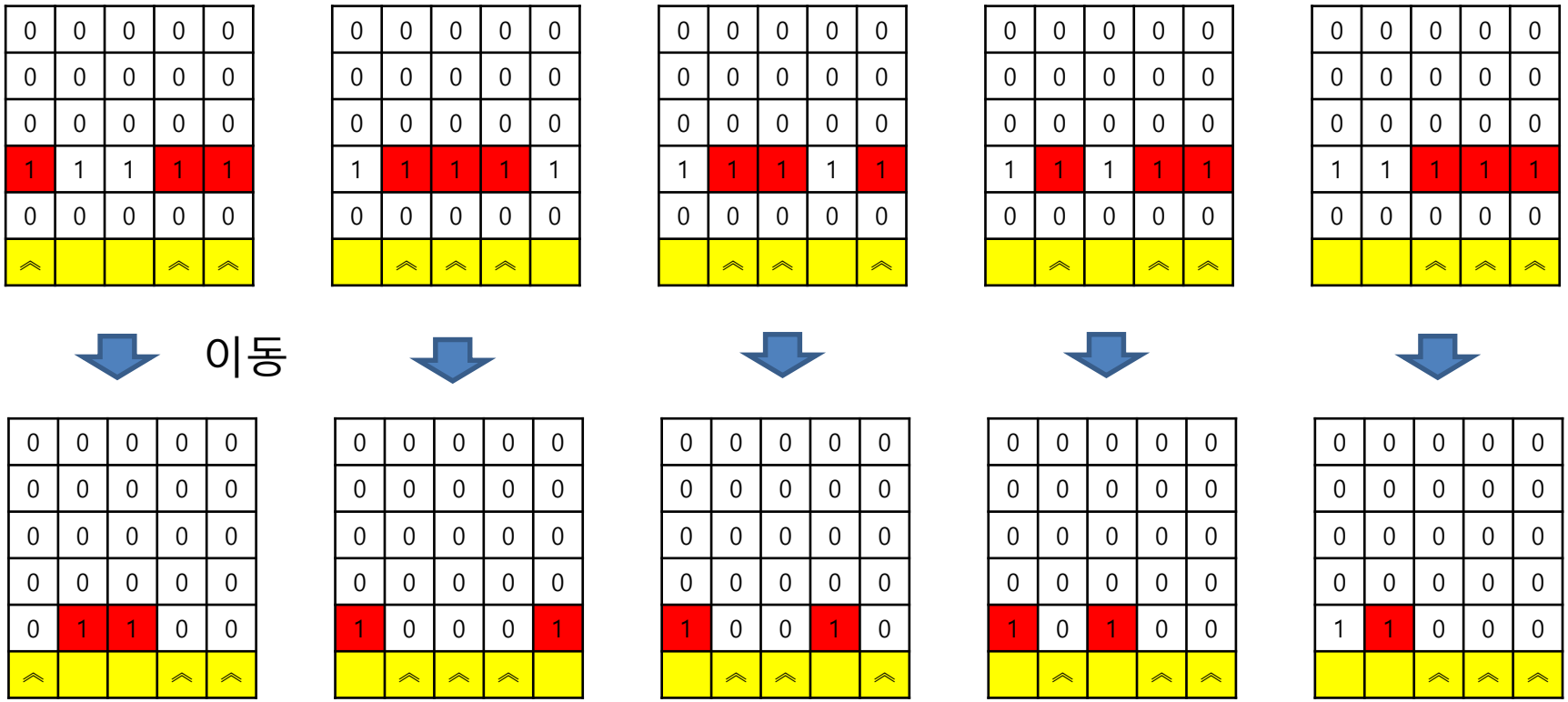


캐슬디펜스

3번예제

5, 5
길이 d = 2

5곳에서 3군데 고르는 경우의 수 $_5C_3 = 10$



최대값 5

캐슬디펜스

5번예제

6, 5
길이 d = 1

5곳에서 3군데 고르는
경우의 수 ${}_5C_3$ = 10

1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
0	0	0	1	1
1	1	0	1	1
0	0	1	0	0
︽	︽	︽	0	0

0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
0	0	0	1	1
1	1	0	1	1
︽	︽	︽	0	0

0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
0	0	0	1	1
︽	︽	︽	0	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
︽	︽	︽	0	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
︽	︽	︽	0	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
︽	︽	︽	0	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
︽	︽	︽	0	0

1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
0	0	0	1	1
1	1	0	1	1
0	0	1	0	0
︽	︽	0	︽	0

0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
0	0	0	1	1
1	1	0	1	1
︽	︽	0	︽	0

0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
0	0	0	1	1
︽	︽	0	︽	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
︽	︽	0	︽	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
︽	︽	0	︽	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
︽	︽	0	︽	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
︽	︽	0	︽	0

...

최개 9

캐슬디펜스

6번예제

6, 5
길이 d = 2

5곳에서 3군데 고르는 경우의 수 ${}_5C_3$ = 10

1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
0	0	0	1	1
1	1	0	1	1
0	0	1	0	0
︽	︽	︽	0	0

0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
0	0	0	1	1
0	0	0	1	1
︽	︽	︽	0	0

0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
0	0	0	1	1
︽	︽	︽	0	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
0	0	0	0	0
︽	︽	︽	0	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
0	0	0	1	0
︽	︽	︽	0	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	1
︽	︽	︽	0	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
︽	︽	︽	0	0

…

1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
0	0	0	1	1
1	1	0	1	1
0	0	1	0	0
︽	0	︽	0	︽

0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
0	0	0	1	1
0	1	0	1	0
︽	0	︽	0	︽

0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
0	0	0	1	1
︽	0	︽	0	︽

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
1	1	0	0	0
︽	0	︽	0	︽

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
0	1	0	1	0
︽	0	︽	0	︽

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	1	0	1
︽	0	︽	0	︽

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
︽	0	︽	0	︽

…

최개 14

캐슬디펜스

```
N, M, D = map(int, input().split())
mat = [list(map(int, input().split())) for _ in range(N)]

archer = [0] * 3
ans = 0
for i in range(M - 2):
    for j in range(i + 1, M - 1):
        for k in range(j + 1, M):
            killed = [[0] * M for _ in range(N)]
            archer[0], archer[1], archer[2] = i, j, k
            solve(N)
            ans = max(ans, sum(sum(killed, [])))
print(ans)
```

거리

$_MC_3$ 조합생성

죽은 적군 계산하고 매번 새로 생성

N 행 이동

죽은 적군 계산, 최대값 갱신

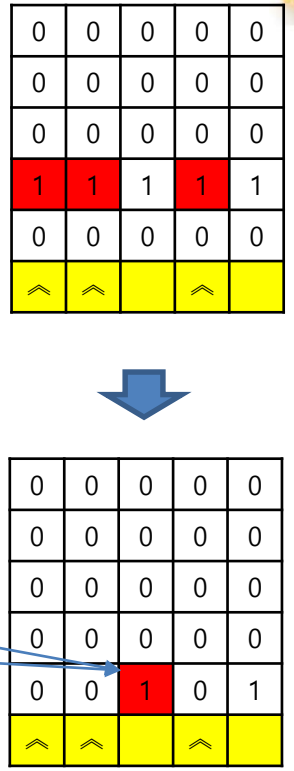
캐슬디펜스

모든 행을 처리

```
def solve(k):
    if k == 0:
        return
    else:
        t = []
        t.append(kill(k, archer[0]))
        t.append(kill(k, archer[1]))
        t.append(kill(k, archer[2]))
        for found, x, y in t:
            if found:
                killed[x][y] = 1
        solve(k - 1)
```

겹칠수 있으므로 각 궁수가 저격 가능한 적의 위치를 모은다.

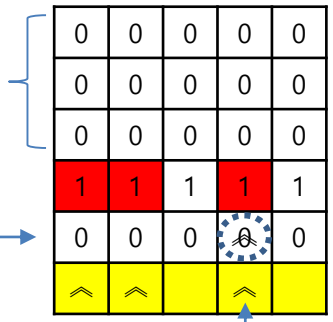
겹친다

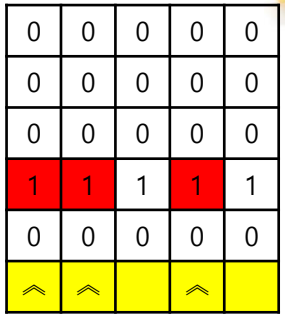


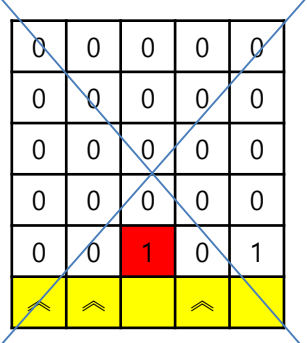
한 행씩 처리

캐슬디펜스

실제 적을 움직이는 것이 아니라 궁수가 이동한다고 가정








```
def kill(k, y):
    xx, yy, min_d = - 1, - 1, 100
    for i in range(k - 1, -1, -1):
        for j in range(M):
            if mat[i][j] and not killed[i][j]:
                td = abs(i - k) + abs(j - y)
                if td < min_d:
                    xx, yy, min_d = i, j, td
                elif td == min_d and j < yy:
                    xx, yy = i, j

    return (min_d <= D, xx, yy)
```

거리가 가장 가까운 적의 위치를 검색

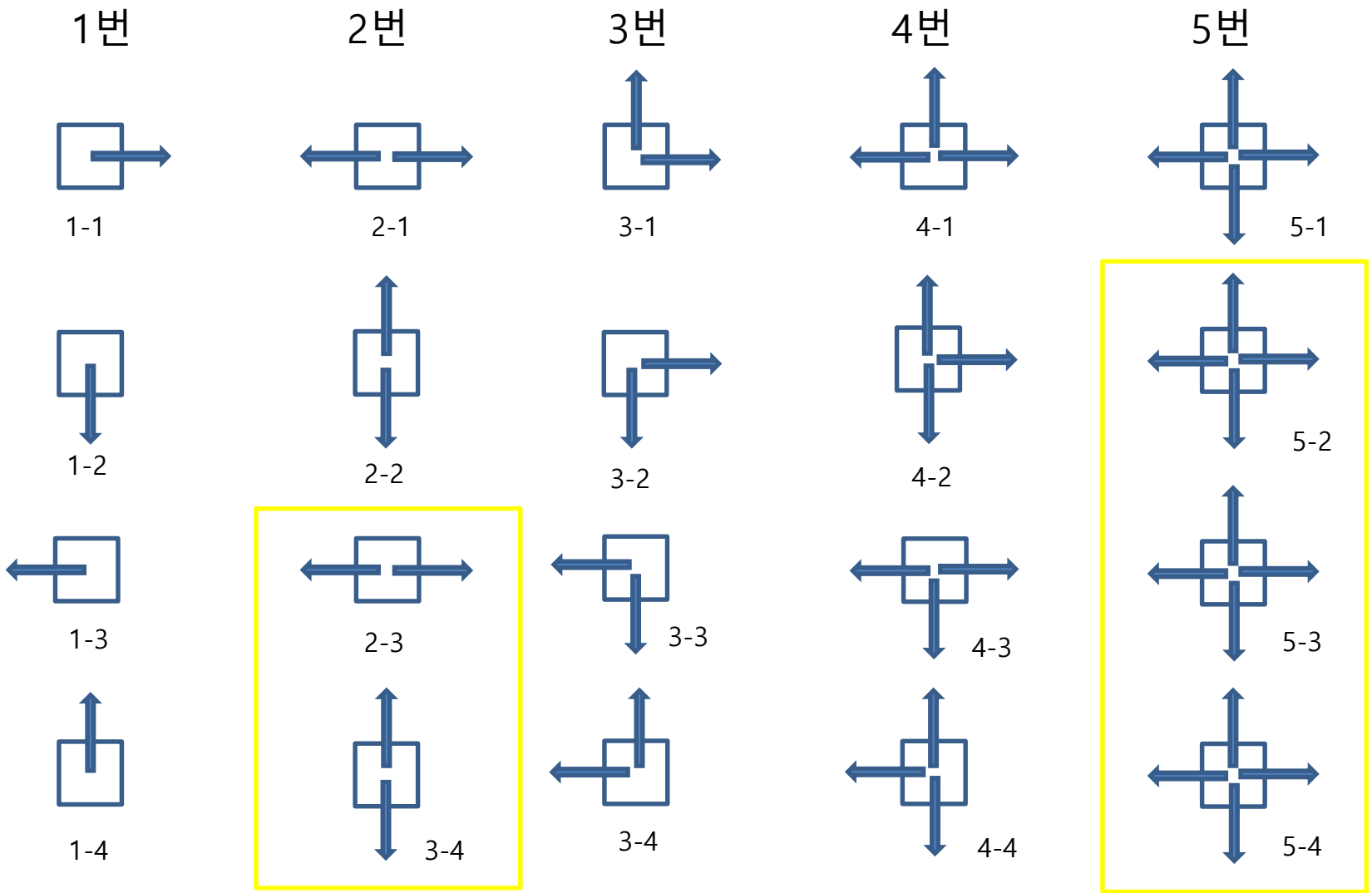
같은 거리면 왼쪽을 선택

가장 가까운 거리라도 사정 거리 이내여야 의미가 있음

감시

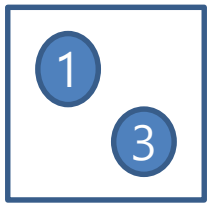
https://www.acmicpc.net/problem/15683

감시



각 종류 CCTV 90도 씩 회전한 모습

감시



사무실 안에 1,3 두 종류의
CCTV가 있다면

1-1 3-1	1-2 3-1	1-3 3-1	1-4 3-1
1-1 3-2	1-2 3-2	1-3 3-2	1-4 3-2
1-1 3-3	1-2 3-3	1-3 3-3	1-4 3-3
1-1 3-4	1-2 3-4	1-3 3-4	1-4 3-4

각 CCTV를 회전 했을 때

서로 다른 16가지 경우가
생긴다.

중복순열이다.

감시

```
N, M = map(int, input().split())
mat = [list(map(int, input().split())) for _ in range(N)]
observed = [[0] * M for i in range(N)]
cctvXYC = []

...

ans = 0
direction = [0] * len(cctvXYC)
solve(0)
print(N*M - ans)
```

CCTV로 감시되는 칸의 정보를 저장

CCTV의 종류와 위치를 저장할 배열

중복순열을 저장할 배열

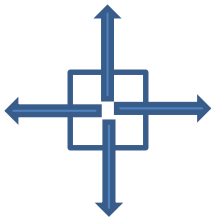
사각지대 = 전체 영역 - 최대감시영역

감시

```
...
for x in range(N):
    for y in range(M):
        if mat[x][y] == 0 : continue
        elif mat[x][y] == 6:
            observed[x][y] = 1
        elif mat[x][y] == 5:
            observed[x][y] = 1
            fill_right(x, y, observed)
            fill_left(x, y, observed)
            fill_up(x, y, observed)
            fill_down(x, y, observed)
        else:
            cctvXYC.append((mat[x][y], x, y))
...
```

벽 감시영역으로 처리

5번



5번 CCTV는 회전의 의미가 없음. 미리 감시영역처리

1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	5	0	0
0	0	5	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

CCTV 종류, 위치

감시

```
def fill_right(x, y, arr):
    yy = y + 1
    while yy < M and mat[x][yy] != 6:
        arr[x][yy] = 1
        yy += 1
```

1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	5	0	0
0	0	5	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

mat

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	1	1	1
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

arr

```
def fill_left(x, y, arr):
    yy = y - 1
    while yy > -1 and mat[x][yy] != 6:
        arr[x][yy] = 1
        yy -= 1
```

1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	5	0	0
0	0	5	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

mat

0	0	0	0	0	0
0	0	0	0	0	0
1	1	1	1	1	1
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

arr

감시

```
def fill_up(x, y, arr):
    xx = x + 1
    while xx < N and mat[xx][y] != 6:
        arr[xx][y] = 1
        xx += 1
```

1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	5	0	0
0	0	5	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

mat

0	0	0	1	0	0
0	0	0	1	0	0
1	1	1	1	1	1
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

arr

```
def fill_down(x, y, arr):
    xx = x - 1
    while xx > -1 and mat[xx][y] != 6:
        arr[xx][y] = 1
        xx -= 1
```

1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	5	0	0
0	0	5	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

mat

0	0	0	1	0	0
0	0	0	1	0	0
1	1	1	1	1	1
0	0	0	1	0	0
0	0	0	1	0	0
0	0	0	1	0	0

arr

감시

```
def solve(k):
    global direction
    if k == len(cctvXYC):
        observe()
    else:
        if cctvXYC[k][0] == 2:
            for i in range(2):
                direction[k] = i
                solve(k + 1)
        else:
            for i in range(4):
                direction[k] = i
                solve(k + 1)
```

4방향의 중복순열을 저장할 배열, 0,1,2,3
➔ 우,하,좌,상

5번을 제외하고 CCTV개수 만큼 중복순열을 생성했으면

2번 CCTV는 2번만 회전

2번

4방향에 대한 중복 순열 생성

감시

mat

1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	5	0	0
0	0	5	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

```
def observe():
    global ans
    tobserved = [[0] * M for i in range(N)]
    for i in range(N):
        for j in range(M):
            tobserved[i][j] = observed[i][j]

    for i in range(len(cctvXYC)):
        cctvC, x, y = cctvXYC[i]
        dir = direction[i]
        tobserved[x][y] = 1
        if cctvC == 1:
            ...
        elif cctvC == 2:
            ...
        elif cctvC == 3:
            ...
        elif cctvC == 4:
            ...

    ans = max(ans, sum(sum(tobserved, [])))
```

5번 CCTV를 미리 처리
한 정보를 받아옴

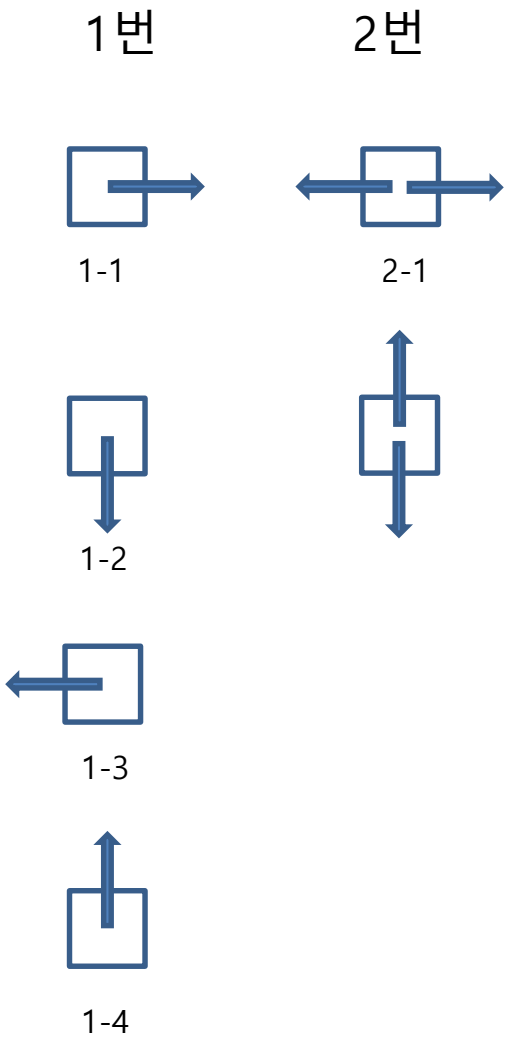
(1,0,0)	(1,1,1)	(1,2,2)	(1,3,3)	(1,4,4)	(1,5,5)

1	1	1	1	1	1

1	1	1	1	1	1
0	1	1	1	1	1
0	0	1	1	1	1
1	1	1	1	1	1
0	0	1	0	1	1
0	0	1	0	0	1

감시

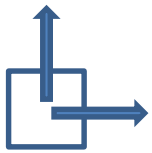
```
if cctvC == 1:
    if dir == 0:   fill_right(x, y, tobserved)
    elif dir == 1: fill_down(x, y, tobserved)
    elif dir == 2: fill_left(x, y, tobserved)
    elif dir == 3: fill_up(x, y, tobserved)
elif cctvC == 2:
    if dir == 0:
        fill_right(x, y, tobserved)
        fill_left(x, y, tobserved)
    elif dir == 1:
        fill_up(x, y, tobserved)
        fill_down(x, y, tobserved)
```



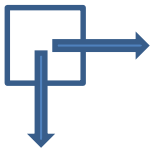
감시

```
elif cctvC == 3:
    if dir == 0:
        fill_up(x, y, tobserved)
        fill_right(x, y, tobserved)
    elif dir == 1:
        fill_right(x, y, tobserved)
        fill_down(x, y, tobserved)
    elif dir == 2:
        fill_down(x, y, tobserved)
        fill_left(x, y, tobserved)
    elif dir == 3:
        fill_left(x, y, tobserved)
        fill_up(x, y, tobserved)
```

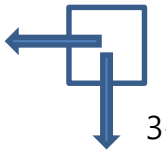
3번



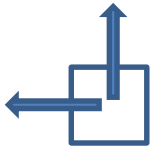
3-1



3-2



3-3

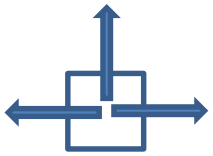


3-4

감시

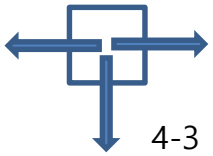
```
elif cctvC == 4:
    if dir == 0:
        fill_right(x, y, tobserved)
        fill_left(x, y, tobserved)
        fill_up(x, y, tobserved)
    elif dir == 1:
        fill_right(x, y, tobserved)
        fill_down(x, y, tobserved)
        fill_up(x, y, tobserved)
    elif dir == 2:
        fill_right(x, y, tobserved)
        fill_down(x, y, tobserved)
        fill_left(x, y, tobserved)
    elif dir == 3:
        fill_down(x, y, tobserved)
        fill_left(x, y, tobserved)
        fill_up(x, y, tobserved)
```

4번



4-1

4-2



4-3

게리맨더링

https://www.acmicpc.net/problem/17471

게리맨더링

구역이 [1,2,3,4], 4개 있다면 하나의 선거구가 구성할 수 있는 방법은?

[]
[1]
[2]
[3]
[4]
[1, 2]
[1, 3]
[1, 4]
[2, 3]
[2, 4]
[3, 4]
[1, 2, 3]
[1, 2, 4]
[1, 3, 4]
[2, 3, 4]
[1, 2, 3, 4]

게리맨더링

다른 선거구가 가지는 방법이 있어야 하므로 [], [1, 2, 3, 4]을 제외하면 14가지 경우가 생긴다.

[1]
[2]
[3]
[4]
[1, 2]
[1, 3]
[1, 4]
[2, 3]
[2, 4]
[3, 4]
[1, 2, 3]
[1, 2, 4]
[1, 3, 4]
[2, 3, 4]

1선거구

[2, 3, 4]
[1, 3, 4]
[1, 2, 4]
[1, 2, 3]
[3, 4]
[2, 4]
[2, 3]
[1, 4]
[1, 3]
[1, 2]
[4]
[3]
[2]
[1]

2선거구

게리맨더링

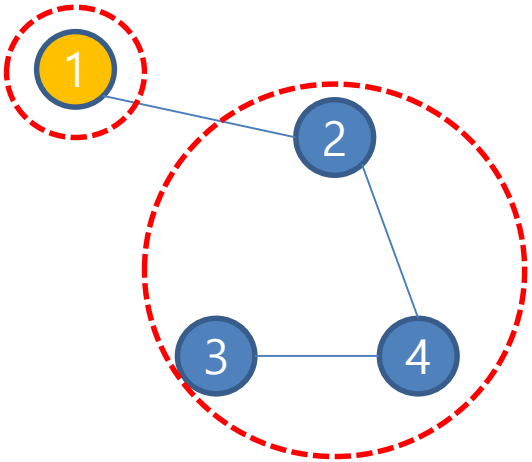
즉 부분 집합과 연관되어 있다. 1선거구를 부분 집합으로 구하고 전체 선거구에서 1선거구를 제거 하여 2선거구를 만들 수 있다.

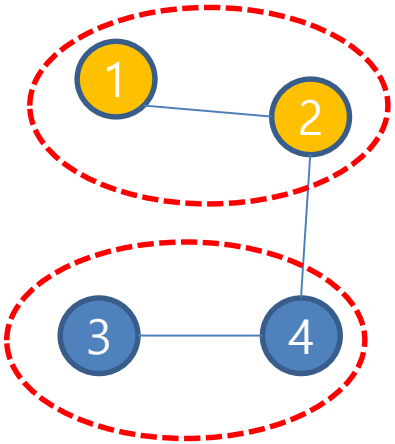
[1, 2, 3, 4]	-	[1]	=	[2, 3, 4]
		[2]		[1, 3, 4]
		[3]		[1, 2, 4]
		[4]		[1, 2, 3]
		[1, 2]		[3, 4]
		[1, 3]		[2, 4]
		[1, 4]		[2, 3]
		[2, 3]		[1, 4]
		[2, 4]		[1, 3]
		[3, 4]		[1, 2]
		[1, 2, 3]		[4]
		[1, 2, 4]		[3]
		[1, 3, 4]		[2]
		[2, 3, 4]		[1]
		1선거구		2선거구

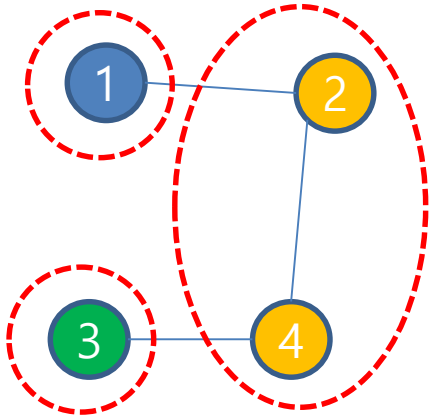
게리맨더링

나누어진 2개 선거구를 이용하여 선거구의 연결 상태를 확인한다.

1선거구		2선거구
[1]		[2, 3, 4]
[2]		[1, 3, 4]
[3]	…	[1, 2, 4]
[4]		[1, 2, 3]
[1, 2]		[3, 4]
[1, 3]		[2, 4]
[1, 4]	…	[2, 3]
[2, 3]		[1, 4]
[2, 4]		[1, 3]
[3, 4]		[1, 2]
[1, 2, 3]		[4]
[1, 2, 4]	…	[3]
[1, 3, 4]		[2]
[2, 3, 4]		[1]





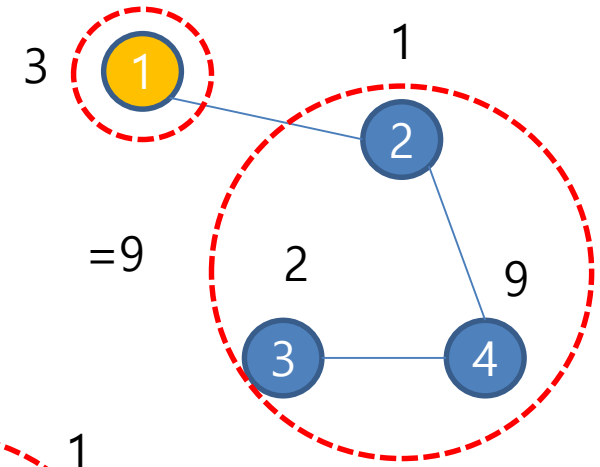


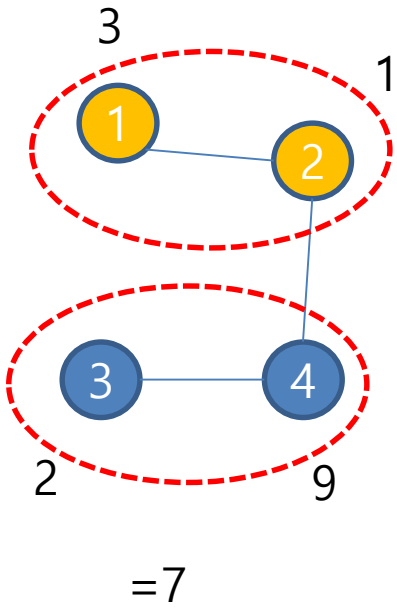
2개의 선거구로 분리되지 않는 경우도 있다.

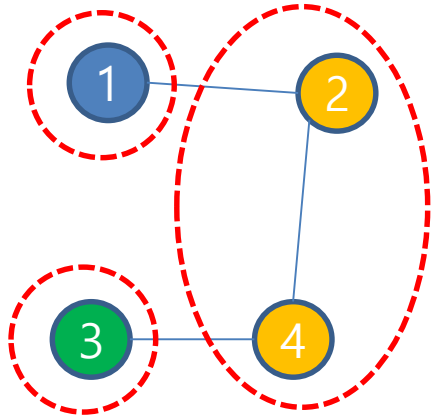
게리맨더링

두 개 선거구의 인구수를 구하고 그 차가 최소인 것을 찾는다.

1선거구		2선거구
[1]		[2, 3, 4]
[2]		[1, 3, 4]
[3]	…	[1, 2, 4]
[4]		[1, 2, 3]
[1, 2]		[3, 4]
[1, 3]		[2, 4]
[1, 4]	…	[2, 3]
[2, 3]		[1, 4]
[2, 4]		[1, 3]
[3, 4]		[1, 2]
[1, 2, 3]		[4]
[1, 2, 4]	…	[3]
[1, 3, 4]		[2]
[2, 3, 4]		[1]







2개의 선거구로 분리되지 않는 경우도 있다.

게리맨더링

5	2	3	4	1	2

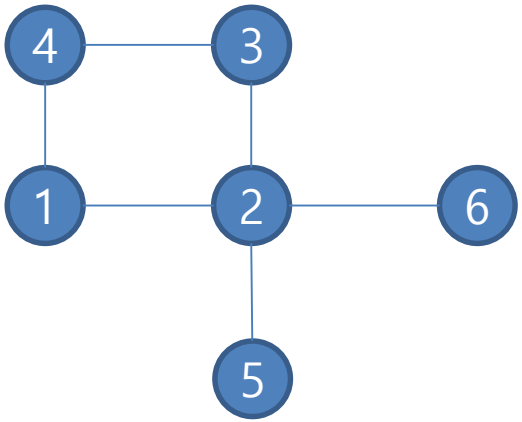
```
6
5 2 3 4 1 2
2 2 4
4 1 3 6 5
2 4 2
2 1 3
1 2
1 2
```

구역인구수

```
N = int(input())
people = list(map(int, input().split()))
G = []

for i in range(N):
    tlist = list(map(int, input().split()))
    G.append(tlist[1:])

ans = 1e9
subset = [0] * N
solve(0)

if ans == 1e9:
    print(-1)
else:
    print(ans)
```

[[2,4], [1,3,6,5], [4,2], [1,3], [2], [2]]

두 구역으로 나눌 수 있는 방법이 없으면



게리맨더링

subset

0	0	0	0	0	0

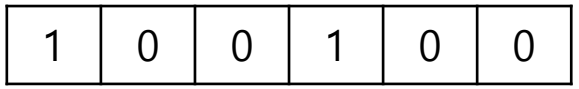
1	1	1	1	1	1

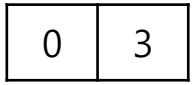
```
def solve(k):
    global ans
    if k == N:
        if sum(subset) == 0 or sum(subset) == N: return

        ...
    else:
        subset[k] = 1; solve(k + 1)
        subset[k] = 0; solve(k + 1)
```

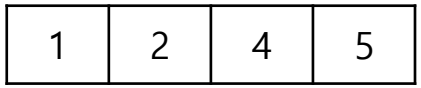
구역 개수의
모든 부분집합
을 생성

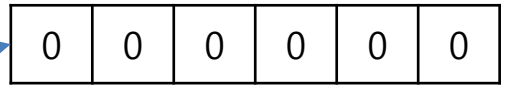
게리맨더링

```
def solve(k):
    global ans
    if k == N:
        ...
        area1, area2 = [], []
        for i in range(N):
            if subset[i]:
                area1.append(i)
            else:
                area2.append(i)
        visited = [0] * N
        v1 = dfs(area1[0], area1, visited)
        v2 = dfs(area2[0], area2, visited)
        if sum(visited) == N:
            ans = min(ans, abs(v1 - v2))
    else:
        ...
```





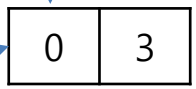


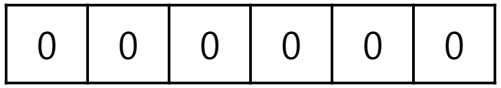


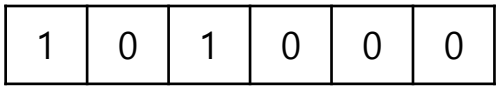
visited를 매 경우 마다 새로 만든다.

두 지역구를 조사 한 후 모든 구역이 선택 됐으면...
두 구역의 인구수의 최소 차를 구한다.

게리맨더링

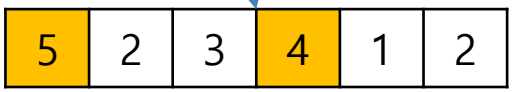


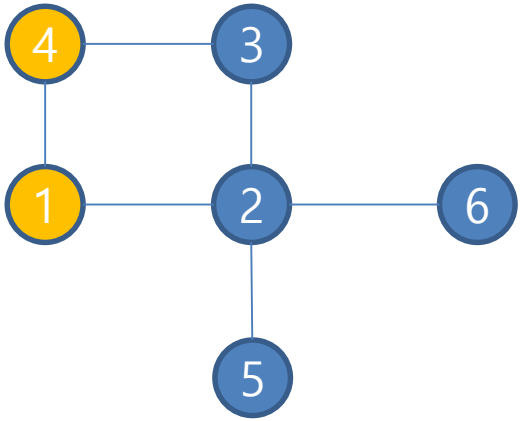




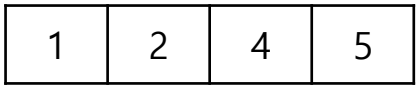

```
def dfs(v, area, visited):
    ret = people[v]
    visited[v] = 1
    for u in G[v]:
        if not visited[u - 1] and u - 1 in area:
            ret += dfs(u - 1, area, visited)
    return ret
```

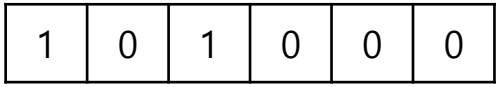
최종 9 반환

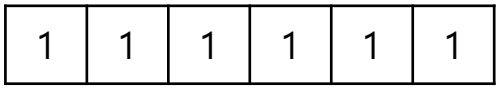




게리맨더링








```
def dfs(v, area, visited):
    ret = people[v]
    visited[v] = 1
    for u in G[v]:
        if not visited[u - 1] and u - 1 in area:
            ret += dfs(u - 1, area, visited)
    return ret
```

최종 8 반환