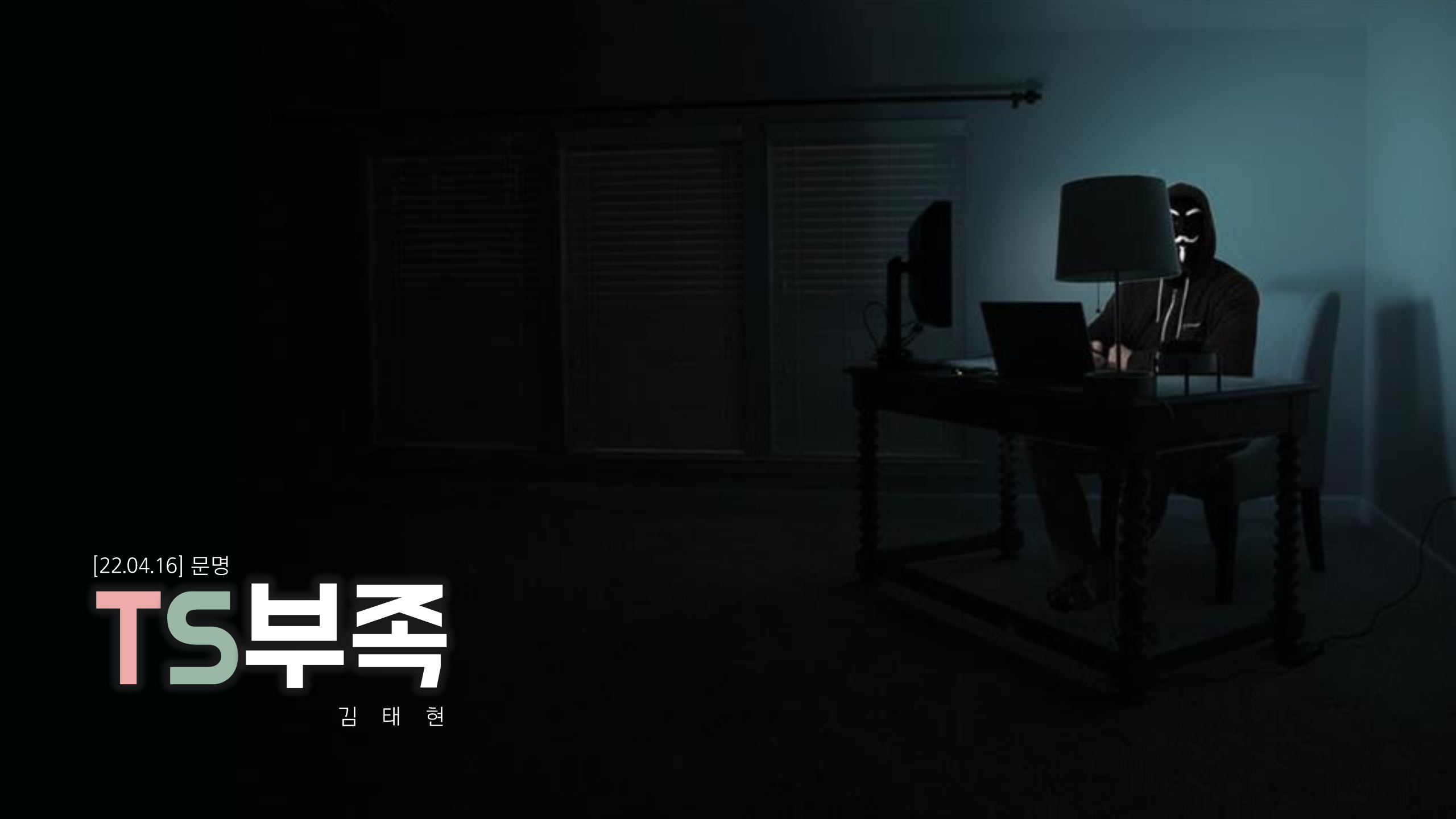


[22.04.16] 문명

TS부족

김 태 현

문 제

(1,1) ~ (N,N) 범위의 땅이 있다.

한 지점에서 고유 번호를 갖는 부족이 발생 / 합병 / 소멸 한다.

```
void init(int N)
```

땅 크기 N <= 1,000

```
int getTribe(int r, int c)
```

(r,c) 땅을 점유하는 부족 id 반환
없을시 0 반환

```
int getTribeArea(int id)
```

id 부족이 점유하는 땅의 개수 반환
부족이 존재하지 않는 경우 0 반환

```
int newTribe(int r, int c, int id)
```

(r,c)에서 id 부족 발생

발생시 규칙
1)발생하는 땅에서 상하좌우 인접한 땅이 없으면 해당 부족 발생
2)인접한 땅이 있으면 가장 많이 인접한 부족, 같다면 id가 가장 작은 부족으로 발생

```
int removeTribe(int id)
```

id 부족이 사라지고 이 부족이 점유하던 땅은 빈 땅으로 변경
점유하던 땅의 개수 반환
부족이 존재하지 않거나 이미 사라진 경우 0 반환

```
int mergeTribe(int id1, int id2)
```

id2 부족이 id1 부족으로 합병
합병 후, id1이 점유하는 땅의 개수 반환
id1, id2 는 최소 1개 땅을 점유하고 있음을 보장

5<=N<=1,000
1<=id<=1,000,000,000
newTribe() <= 60,000
removeTribe() <= 3,000
getTribe(), getTribeArea() <= 각 10,000
mergeTribe() <= 30,000

문제 분석

점령 땅의 수 최대 60,000개
id가 크기 때문에 hash를 이용한다.
(r,c) 가 점유하는 부족을 알아야 한다.
id부족이 점유하는 땅의 수를 알아야 한다.
부족별 점유중인 땅의 좌표를 알아야 한다.
부족간 합병이 효율적이어야 한다.

5<=N<=1,000

1<=id<=1,000,000,000

newTribe() <= 60,000

removeTribe() <= 3,000

getTribe(), getTribeArea() <= 각 10,000

mergeTribe() <= 30,000

※ 눈에 띄는 조건
참고로, 자료 구조 A와 자료 구조 B 중 하나를 변경해야 하는 경우 사이즈가 더 작은 것을 변경하는 것이 유리하다.

자료구성

id->idx : hash table

id	idx
1234	1
234	2
11	3
564	4

idx->id : array

idx	id
1	1234
2	234
3	11
4	564

land status : array

1	1		4	
				4
		2	2	
		3	1	1
				1

부족별 점유 땅 리스트 : linked list

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

부족별 점유 땅 개수 : array

idx	cnt
1	5
2	2
3	1
4	2

※ list의 size()로 대체 가능

`int newTribe(int r, int c, int id)` <= 60,000

1. land[r][c] 의 주위를 보며 가장 많은 부족을 구한다.
   인접한 개수가 같을 때는 idx2id를 참조하여 id가 작은 부족을 선택한다.
2. 만약 없다면 id에 idx를 부여하여 id2idx에 등록한다.
3. tribeList, tribeCnt에 추가하고 개수 증가시킨다.

land[r][c]

1	1		4	
				4
		2	2	
		3	1	1
				1

tribeList

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

tribeCnt

idx	cnt
1	5
2	2
3	1
4	2

id2idx

id	idx
1234	1
234	2
11	3
564	4

idx2id

idx	id
1	1234
2	234
3	11
4	564

`int removeTribe(int id)` <= 3,000

1. id2idx에서 idx를 찾는다.
2. tribeList[idx] 의 모든 좌표(r,c)에 대해 land[r][c] = 0 으로 바꾼다.
3. id2idx hash table에서 삭제 한다.
4. tribeCnt[idx] 를 반환한다.

어차피 많아봐야 60,000개 삭제하므로 시간은 문제 없다.

land[r][c]

1	1		4	
				4
		2	2	
		3	1	1
				1

tribeList

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

tribeCnt

idx	cnt
1	5
2	2
3	1
4	2

id2idx

id	idx
1234	1
234	2
11	3
564	4

idx2id

idx	id
1	1234
2	234
3	11
4	564

`int mergeTribe(int id1, int id2)` <= 10,000

1. id2idx에서 idx1, idx2를 찾는다.
2. tribeList[idx2] 의 모든 좌표 (r,c) 에 대해 land[r][c] = idx1 으로 변경한다.
3. tribeList[idx2] 의 모든 값을 tribeList[idx1] 으로 옮긴다.
4. tribeCnt[idx1] += tribeCnt[idx2]
5. id2idx 에서 id2를 지운다.
6. tribeCnt[idx1] 반환

1회당
O(id2가 점유중인 땅의 수)
worst case O(60,000)

land[r][c]

1	1		4	
				4
		2	2	
		3	1	1
				1

tribeList

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

tribeCnt

idx	cnt
1	5
2	2
3	1
4	2

id2idx

id	idx
1234	1
234	2
11	3
564	4

idx2id

idx	id
1	1234
2	234
3	11
4	564

개 선

```
int mergeTribe(int id1, int id2)
```

참고로, 자료 구조 A와 자료 구조 B 중 하나를 변경해야 하는 경우
사이즈가 더 작은 것을 변경하는 것이 유리하다.

tribeCnt[idx1] < tribeCnt[idx2] 라면

- tribeList[idx1] 의 모든 좌표 (r,c) 에 대해 land[r][c] = idx2 으로 변경한다.
- tribeList[idx1] 의 모든 값을 tribeList[idx2]로 옮긴다.
- tribeCnt[idx2] += tribeCnt[idx1]
- id2idx에서 id1의 값을 idx2로 바꾼다.
- id2idx에서 id2의 값을 지운다.

특정부족의 크기가 커질수록 다른부족의 크기는 작아지므로 비용이 크게 감소한다.

land[r][c]

1	1		4	
				4
		2	2	
		3	1	1
				1

tribeList

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

tribeCnt

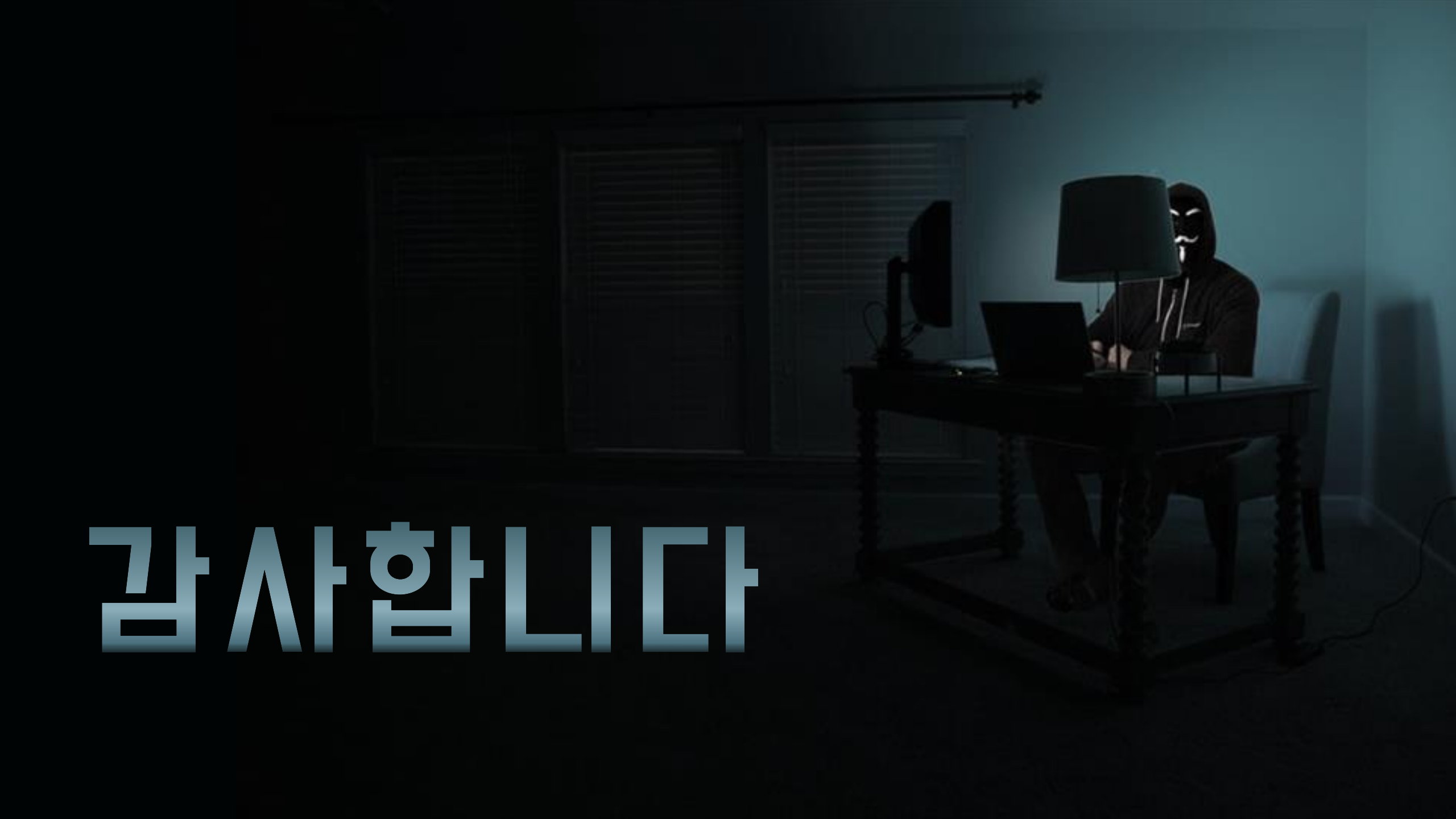
idx	cnt
1	5
2	2
3	1
4	2

id2idx

id	idx
1234	1
234	2
11	3
564	4

idx2id

idx	id
1	1234
2	234
3	11
4	564


감사합니다