JooHyun – Lee (comkiwer)

TS삼국지게임

Hancom Education Co. Ltd.

Problem

Problem

당신은 어린 시절 삼국지 게임 매니아였다.
어린 시절 추억을 살려 삼국지 게임을 만들어보기로 한다.

구현하려는 기능은 구체적으로 아래와 같다.

- 어떤 군주와 다른 군주와의 동맹 (ally)
- 어떤 군주와 그 동맹 군주들이 다른 군주의 영토를 공격 (attack)
- 병사 모집 (recruit)

전체 영토는 N x N 영토로 구성되어 있다.

각 영토에는 영토를 다스리는 군주가 있으며, 군주의 이름에 중복은 없으며,
모든 군주는 단 하나의 영토를 통치한다.

초기에는 군주들은 동맹이나 적대관계가 없다.

각 영토에는 병사들이 있다.

Problem

[Fig. 1] 은 N이 4인 경우, 군주의 이름과 각 영토의 병사 수의 예이다.

chengpu 12	sunce 5	huanggai 12	caozhi 9
liuzhang 9	xiahouyuan 12	lingtong 14	caopi 11
yuanshu 9	machao 6	guanyu 15	gaoren 10
weiyan 5	huangzhong 7	lukang 11	xuhuang 10

[Fig. 1]

군주들은 서로 동맹을 한다.

동맹을 하면 전투 시 함께 공격하고,
함께 방어를 할 수 있다.

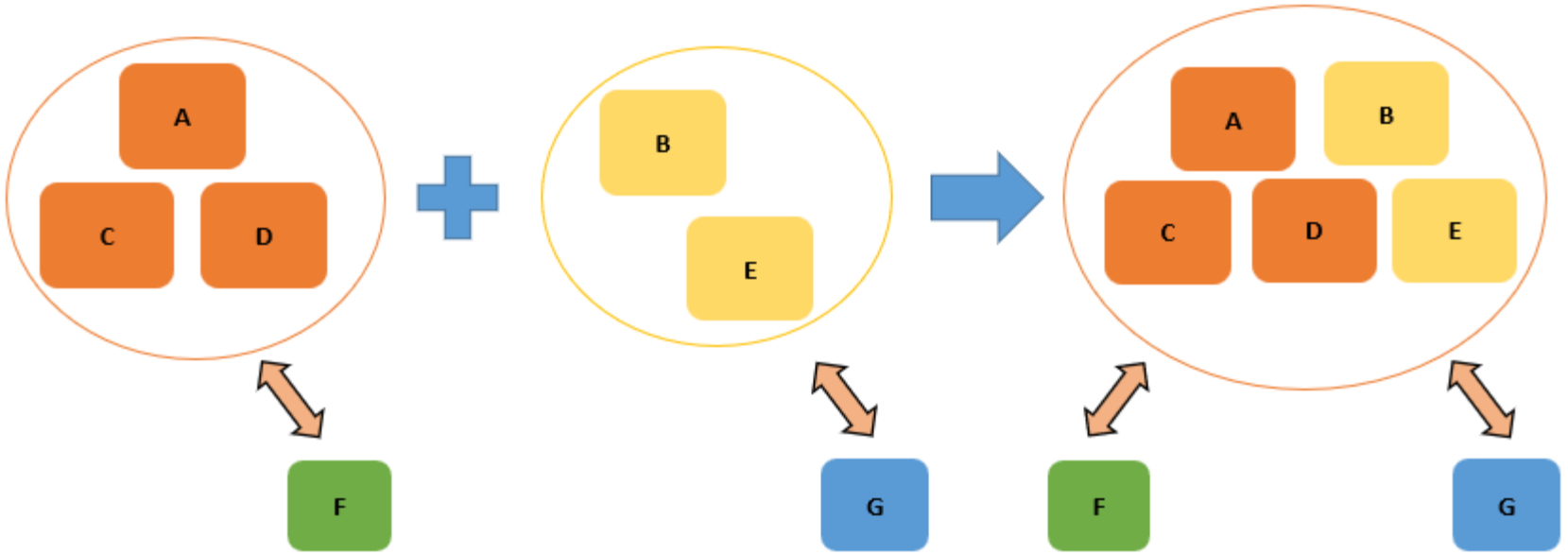
두 군주들이 동맹을 하면
서로의 모든 동맹까지 함께 동맹을 맺는다.

동맹을 맺을 때, 동맹을 맺는 두 군주를 포함한
서로의 모든 동맹 군주 간 적대관계가 있는 경우 동맹을 맺지 않는다.

동맹을 맺을 때 두 군주와 그 동맹들의 적대관계는 새로운 동맹에 그대로 유지가 된다.

Problem

[Fig. 2] 와 같이 A, C, D가 동맹이고 F가 적대 관계이고 B, E가 동맹이고 G와 적대 관계일 때, A와 B가 동맹을 맺으면 A, B, C, D, E 가 동맹이 되고, F, G 와는 적대 관계가 된다.



[Fig. 2]

Problem

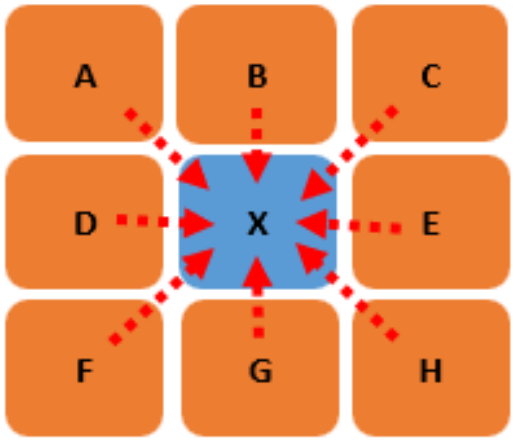
삼국지 게임의 핵심은 전투이다. 전투는 동맹과 동맹 간의 전투다.

공격을 받는 군주가 공격하는 군주의 동맹에 속해 있는 경우에 전투는 발생하지 않는다.

전투는 공격하는 군주의 또는 그의 동맹 (이하 군주의 동맹) 영토가 피 공격 영토에 인접해 있을 경우에 만 발생한다.

[Fig. 3] 에서 A ~ H 를 X 에 인접한 군주라 한다.

군주 X 의 영토를 공격할 때 전투가 발생하기 위해서는 공격하는 군주의 동맹이 A ~ H 에 하나라도 있어야 한다.



[Fig. 3]

Problem

전투가 발생하는 경우 공격하는 동맹의 각 군주들은
방어하는 각 군주들과 서로 적대관계가 된다.
실제 전투가 일어나지 않는 경우는 적대관계가 되지 않는다.

전투는 공격하는 군주의 동맹들이 함께 공격을 하고,
공격을 받는 군주의 동맹도 함께 방어를 한다.
공격하는 인접 동맹들은 자신이 가진 병사의 절반을 공격 대상 영토에 보내 함께 공격을 하고,
방어를 하는 군주의 인접 동맹에서도 병사의 절반씩을 보내어 함께 방어를 한다.

Problem

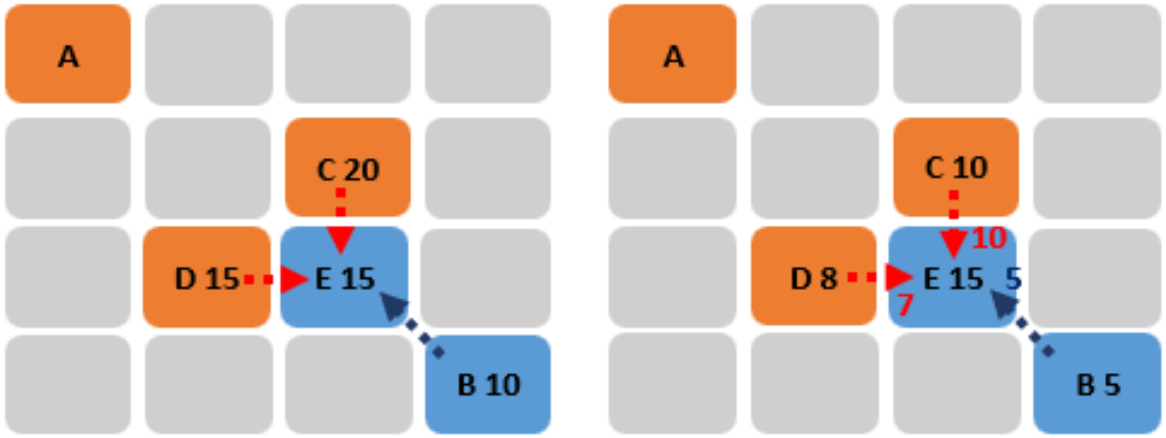
[Fig. 4] 는 군주 A, C, D 가 동맹이고, 군주 B, E가 동맹일 때,

군주 A가 군주 E를 공격하는 경우에 대한 예시다.

군주 E 의 영토가 A의 동맹 C, D 와 인접하므로 전투가 발생한다.

이 때 공격하는 병사의 수는 군주 E 에 인접한 군주 C의 병사 절반 10명과
군주 D의 병사 절반 7명을 합한 17명이 된다.

방어하는 병사의 수는 군주 E의 영토에 있는 병사 15명과
군주 B로부터 병사 절반인 5명을 지원받아 총 20명이 된다.



[Fig. 4]

Problem

병사들은 상대 병사와 1:1로 싸워 함께 전사한다.

전투의 승리 여부는 남은 병사의 수로 판별한다.

만약 공격한 병사의 수가 방어한 병사의 수보다 많다면 공격은 성공한다.

영토가 함락 되었으므로 패배한 군주는 처형되고,
공격을 지휘한 장수가 해당 영토의 새로운 군주가 된다.

새로운 군주의 동맹과 적대관계는 공격한 군주의 동맹 및 적대 관계를 그대로 승계한다.

새로운 군주의 병사의 수는 공격하고 남은 병사의 수가 된다.

방어하는 쪽의 병사의 수가 공격하는 쪽의 병사의 수보다 많거나 같으면 공격은 실패한다.

방어하는 군주의 병사의 수는 방어하고 남은 병사의 수가 된다.

아래 API 설명을 참조하여 각 함수를 구현하라.

Problem

TS삼국지게임

void init(**int** N, **int** soldier[][], **char** monarch[][][]) :

각 테스트 케이스의 처음에 호출된다.

전체 영토는 N x N의 격자 모양으로 이루어져 있다.

soldier 은 각 영토의 병사의 수이다.

monarch 는 각 영토의 군주 이름이다.

군주 이름은 알파벳 소문자로 이루어져 있으며, 길이는 4 이상 10 이하의 문자열이다.

초기에는 모든 군주들은 동맹관계도 적대관계도 없다.

Parameters

N : 전체 영토의 크기 (4 ≤ N ≤ 25, 16 ≤ N x N ≤ 625)

soldier : 각 영토의 병사의 수 (4 ≤ soldier[][] ≤ 100)

monarch : 각 영토의 군주의 이름 (4 ≤ monarch[][]의 길이 ≤ 10)

void destroy()

각 테스트 케이스의 마지막에 호출된다.

빈 함수로 두어도 채점에는 영향을 주지 않는다.

Problem

TS삼국지게임

```
int ally(char monarchA[], char monarchB[])
```

군주 monarchA 의 동맹들이 군주 monarchB 의 동맹들과 동맹을 맺는다.

군주 monarchA 와 군주 monarchB 가 동일 하거나 이미 동맹관계이면 -1을 반환한다.

군주 monarchA 의 동맹과 군주 monarchB 의 동맹 간에 적대관계가 있으면 -2를 반환한다.

위의 두 경우가 아닌 경우 동맹관계가 맺어지고, 1을 반환한다.

각 군주 이름은 알파벳 소문자로 이루어져 있으며, 길이는 4 이상 10 이하의 문자열이다.

monarchA 와 monarchB 는 현재 군주임이 보장된다.

Parameters

monarchA : 군주의 이름 (4 ≤ 길이 ≤ 10)

monarchB : 군주의 이름 (4 ≤ 길이 ≤ 10)

Returns

동맹의 결과 (이미 동맹관계이면 -1, 적대관계가 있으면 -2, 성공하면 1)

Problem

TS삼국지게임

```
int attack(char monarchA[], char monarchB[], char general[])
```

군주 monarchA 와 동맹들이 군주 monarchB 의 영토를 공격한다.

공격을 지휘하는 장수는 general 이다.

군주 monarchA 와 군주 monarchB 가 동맹관계 이면 -1을 반환하고, 전투는 일어나지 않는다.

군주 monarchA 의 영토 또는 동맹 영토가 군주 monarchB 의 영토와 인접하지 않다면 -2을 반환하고, 전투는 일어나지 않는다.

전투가 발생하면 군주 monarchA 의 동맹과 군주 monarchB 의 동맹은 서로 적대관계가 된다.

Problem

int attack(char monarchA[], char monarchB[], char general[]) : (계속)

전투가 발생하면 군주 monarchB 의 영토에 인접한 군주 monarchA 를 포함한 모든 동맹들은 보유한 병사의 절반을 보내 함께 공격한다.

군주 monarchB 의 영토에 인접한 군주 monarchB 의 모든 동맹들도 보유한 병사의 절반을 monarchB 의 영토로 보내 방어를 돕는다.

보내는 병사 계산시 소수점은 버린다.

공격하는 병사의 수가 0명이라도 전투가 발생한 것이다.

전투 시 병사들은 상대 병사와 1:1로 싸워 함께 전사한다.

전투의 결과는 남은 병사로 결정한다.

공격하는 쪽의 병사가 남았다면, 공격 성공으로 1을 반환하고,

방어하는 쪽의 병사가 남았거나, 모든 병사가 사망한 경우 0을 반환한다.

공격을 지휘한 장수는 병사 수에 포함하지 않는다.

Problem

TS삼국지게임

int attack(char monarchA[], char monarchB[], char general[]) : (계속)

공격이 성공하면 군주 monarchB 는 처형되고,

monarchB 가 다스렸던 영토는 멸망하여 동맹관계도 적대관계도 없는 새로운 영토가 된다.

새로운 영토의 군주는 general 이 되고, monarchA의 동맹에 편입되며,
적대 관계는 monarchA 의 적대 관계와 동일하다.

각 군주 이름은 알파벳 소문자로 이루어져 있으며, 길이는 4 이상 10 이하의 문자열이다.

monarchA 와 monarchB 는 현재 군주임이 보장된다. general 는 군주가 아님이 보장된다.

Parameters

monarchA : 공격하는 군주의 이름 (4 ≤ 길이 ≤ 10)

monarchB : 공격을 받는 영토의 군주의 이름 (4 ≤ 길이 ≤ 10)

general : 공격을 지휘하는 장수의 이름 (4 ≤ 길이 ≤ 10)

Returns

공격의 결과 (공격이 승리하면 1, 방어에 성공하면 0, 이미 동맹관계이면 -1,
공격 영토 주변에 공격 하는 동맹이 없는 경우 -2)

Problem

TS삼국지게임

```
int recruit(char monarch[], int num, int sign)
```

병사를 모집한다.

sign 이 0 일 때,

- 군주 monarch 의 영토에 num 명의 병사를 모집한다.
- 병사 모집 이후에 군주 monarch 영토의 병사의 수를 반환한다.

sign 이 1 일 때,

- 군주 monarch 를 포함한 모든 동맹의 영토에 각각 num 명의 병사를 모집한다.
- 병사 모집 이후에 군주 monarch 동맹의 모든 병사의 수 합산하여 반환한다.

군주 이름은 알파벳 소문자로 이루어져 있으며, 길이는 4 이상 10 이하의 문자열이다.

monarch 는 현재 군주임이 보장된다

Parameters

monarch : 군주의 이름 (4 ≤ 길이 ≤ 10)

num : 병사의 수 (1 ≤ num ≤ 200)

Returns

병사의 수

Problem

[제약사항]

1. 각 테스트 케이스 시작 시 init() 함수가, 종료 시 destroy() 함수가 호출된다.
2. 각 테스트 케이스에서 ally() 함수는 최대 8,000회 호출된다.
3. 각 테스트 케이스에서 attack() 함수는 최대 8,000회 호출된다.
4. 각 테스트 케이스에서 recruit() 함수는 최대 13,000회 호출된다.
5. 각 API에 전달되는 모든 문자열은 소문자이며, '\0' (NULL 문자) 로 끝난다.

Problem analysis

Problem analysis : 예제

TS삼국지게임

[#1]

mA 12	mB 5	mC 12	mD 9
mE 9	mF 12	mG 14	mH 11
mJ 9	mK 6	mP 6	mQ 10
mR 5	mT 7	mU 11	mX 14

#	Function	Description	return
1	init(4,	전체 영토는 4 x 4이다.	
	{{12, 5, 12, 9}, ...},	각 영토의 병사의 수와 군주의 이름이 입력으로 주어진다.	
	{{mA,mB,.....})	각 군주의 동맹관계나 적대관계는 없다.	

[#3]

mA 12	mB 5	mC 12	mD 9
mE 9	mF 12	mG 14	mH 11
mJ 9	mK 6	mP 15	mQ 10
mR 5	mT 7	mU 11	mX 10

#	Function	Description	return
2	ally(mA, mG)	군주 mA 와 군주 mG 이 동맹을 맺는다.	1
3	ally(mK, mG)	군주 mK 와 군주 mG 이 동맹을 맺는다.	1

Problem analysis : 예제

TS삼국지게임

[#3]

mA 12	mB 5	mC 12	mD 9
mE 9	mF 12	mG 14	mH 11
mJ 9	mK 6	mP 15	mQ 10
mR 5	mT 7	mU 11	mX 10

#	Function	Description	return
2	ally(mA, mG)	군주 mA 와 군주 mG 이 동맹을 맺는다.	1
3	ally(mK, mG)	군주 mK 와 군주 mG 이 동맹을 맺는다.	1

[#4]

mA 12	mB 5	mC 12	mD 9
mE 9	mF 12	mG 14	mH 11
mJ 9	mK 9	mP 15	mQ 10
mR 5	mT 7	mU 11	mX 10

#	Function	Description	return
4	recruit(mK, 3, 0)	군주 mK 의 영토에 병사 3명을 모집한다.	9

Problem analysis : 예제

TS삼국지게임

[#4]

mA 12	mB 5	mC 12	mD 9
mE 9	mF 12	mG 14	mH 11
mJ 9	mK 9	mP 15	mQ 10
mR 5	mT 7	mU 11	mX 10

#	Function	Description	return
4	recruit(mK, 3, 0)	군주 mK 의 영토에 병사 3명을 모집한다.	9

[#5]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 20	mH 11
mJ 9	mK 15	mP 15	mQ 10
mR 5	mT 7	mU 11	mX 10

#	Function	Description	return
5	recruit(mA, 6, 1)	군주 mA 의 모든 동맹(mA, mG, mK)의 영토에 병사 6명씩 모집한다. mA 18명, mG 20명, mK 15명, 총 53명이다.	53

Problem analysis : 예제

TS삼국지게임

[#5]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 20	mH 11
mJ 9	mK 15	mP 15	mQ 10
mR 5	mT 7	mU 11	mX 10

#	Function	Description	return
5	recruit(mA, 6, 1)	군주 mA 의 모든 동맹(mA, mG, mK)의 영토에 병사 6명씩 모집한다.	53
		mA 18명, mG 20명, mK 15명, 총 53명이다.	

[#6]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 20	mH 11
mJ 9	mK 15	mP 15	mQ 10
mR 5	mT 7	mU 11	mX 10

#	Function	Description	return
6	ally(mP, mX)	군주 mP 와 군주 mX 이 동맹을 맺는다.	1
7	attack(mA, mK, ganada)	군주 mA 가 군주 mK 를 공격한다.	-1
		두 군주는 동맹관계이므로 실제 전투를 일어나지 않는다.	
8	attack(mA, mX, gaboja)	군주 mA 가 군주 mX 를 공격한다. 군주 mX의 주변에 군주 mA 의 동맹이 없으므로, 실제 전투는 일어나지 않는다.	-2

Problem analysis : 예제

TS삼국지게임

[#6]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 20	mH 11
mJ 9	mK 15	mP 15	mQ 10
mR 5	mT 7	mU 11	mX 10

#	Function	Description	return
6	ally(mP, mX)	군주 mP 와 군주 mX 이 동맹을 맺는다.	1
7	attack(mA, mK, ganada)	군주 mA 가 군주 mK 를 공격한다. 두 군주는 동맹관계이므로 실제 전투를 일어나지 않는다.	-1
8	attack(mA, mX, gaboja)	군주 mA 가 군주 mX 를 공격한다. 군주 mX의 주변에 군주 mA 의 동맹이 없으므로, 실제 전투는 일어나지 않는다.	-2

[#9]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 10	mH 11
mJ 9	mK 8	mP 3	mQ 10
mR 5	mT 7	mU 11	mX 5

{mK mA mG} <-> {mP mX}

#	Function	Description	return
9	attack(mA, mP, chacha)	군주 mA 가 군주 mP 를 공격한다. 군주 mP 의 주변에 군주 mA 의 동맹 mK 와 mG 이 있으므로, 전투가 일어난다. 전투가 발생하므로, mA 의 동맹과 mP 동맹은 서로 적대관계가 된다. 공격의 병사수는 mK, mG 에서 각각 7명, 10명을 보내어 총 17명이다. 방어의 병사수는 mP 의 15명에 mX 이 5명을 보내어 총 20명이다. 방어의 병사수가 많으므로 공격은 실패이다.	0

Problem analysis : 예제

TS삼국지게임

[#9]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 10	mH 11
mJ 9	mK 8	mP 3	mQ 10
mR 5	mT 7	mU 11	mX 5

{mK mA mG} <-> {mP mX}

#	Function	Description	return
9	attack(mA, mP, chacha)	군주 mA 가 군주 mP 를 공격한다. 군주 mP 의 주변에 군주 mA 의 동맹 mK 와 mG 이 있으므로, 전투가 일어난다. 전투가 발생하므로, mA 의 동맹과 mP 동맹은 서로 적대관계가 된다. 공격의 병사수는 mK, mG 에서 각각 7명, 10명을 보내어 총 17명이다. 방어의 병사수는 mP 의 15명에 mX 이 5명을 보내어 총 20명이다. 방어의 병사수가 많으므로 공격은 실패이다.	0

[#10]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 10	mH 11
mJ 9	mK 8	mP 13	mQ 10
mR 5	mT 7	mU 11	mX 15

#	Function	Description	return
10	recruit(mX, 10, 1)	군주 mX 를 포함한 모든 동맹의 영토에 각각 10명의 병사를 모집한다.	28

Problem analysis : 예제

TS삼국지게임

[#10]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 10	mH 11
mJ 9	mK 8	mP 13	mQ 10
mR 5	mT 7	mU 11	mX 15

#	Function	Description	return
10	recruit(mX, 10, 1)	군주 mX 를 포함한 모든 동맹의 영토에 각각 10명의 병사를 모집한다.	28

[#11]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 10	mH 11
mJ 9	mK 8	mP 7	mQ 10
mR 5	mT 7	igija 2	mX 8

#	Function	Description	return
11	attack(mP, mU, igija)	군주 mP 가 군주 mU 을 공격한다.	1
		공격의 병사수가 많으므로, 공격 선봉 장수인 igija 가 새로운 군주가 된다.	

Problem analysis : 예제

TS삼국지게임

[#11]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 10	mH 11
mJ 9	mK 8	mP 7	mQ 10
mR 5	mT 7	igija 2	mX 8

#	Function	Description	return
11	attack(mP, mU, igija)	군주 mP 가 군주 mU 을 공격한다. 공격의 병사수가 많으므로, 공격 선봉 장수인 igija 가 새로운 군주가 된다.	1

[#14]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 10	mH 11
mJ 9	mK 8	mP 7	mQ 10
mR 5	mT 7	igija 2	mX 8

#	Function	Description	return
12	ally(igija, mA)	군주 igija 와 군주 mA 는 적대 관계이다.	-2
13	ally(mR, mT)	군주 mR 과 군주 mT 가 동맹을 맺는다.	1
14	ally(mE, mF)	군주 mE 와 군주 mF 가 동맹을 맺는다.	1

Problem analysis : 예제

TS삼국지게임

[#14]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 10	mH 11
mJ 9	mK 8	mP 7	mQ 10
mR 5	mT 7	igija 2	mX 8

#	Function	Description	return
12	ally(igija, mA)	군주 igija 와 군주 mA 는 적대 관계이다.	-2
13	ally(mR, mT)	군주 mR 과 군주 mT 가 동맹을 맺는다.	1
14	ally(mE, mF)	군주 mE 와 군주 mF 가 동맹을 맺는다.	1

[#16]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 10	mH 11
mJ 9	mK 8	mP 4	mQ 10
mR 5	mT 4	igija 16	mX 4

{mP mX igija} <-> {mK mA mG} {mT mR}

#	Function	Description	return
15	recruit(igija, 10, 0)	군주 igija 의 영토에 병사 10명을 모집한다.	12
16	attack(mT, igija, jjukkumi)	군주 mT가 군주 igija 를 공격하지만 실패한다. {mP mX igija} 동맹과 {mT mR}동맹은 적대관계가 된다.	0

16(igija) = 2 + 10(recruit) + 3(mP) + 4(mX) - 3(mT)

Problem analysis : 예제

TS삼국지게임

[#16]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 10	mH 11
mJ 9	mK 8	mP 4	mQ 10
mR 5	mT 4	igija 16	mX 4

{mP mX igija} <-> {mK mA mG} {mT mR}

#	Function	Description	return
15	recruit(igija, 10, 0)	군주 igija 의 영토에 병사 10명을 모집한다.	12
16	attack(mT, igija, jjukkumi)	군주 mT가 군주 igija 를 공격하지만 실패한다. {mP mX igija} 동맹과 {mT mR}동맹은 적대관계가 된다.	0

16(igija) = 2 + 10(recruit) + 3(mP) + 4(mX) - 3(mT)

[#19]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 10	mH 11
mJ 9	mK 8	mP 4	mQ 10
mR 5	mT 4	igija 16	mX 4

{mP mX igija mB} <-> {mK mA mG} {mT mR}

#	Function	Description	return
17	ally(mD, mH)	군주 mD 과 군주 mH 가 동맹을 맺는다.	1
18	ally(mQ, mD)	군주 mQ 과 군주 mD 가 동맹을 맺는다.	1
19	ally(mB, mX)	군주 mB 과 군주 mX 가 동맹을 맺는다.	1

Problem analysis : 예제

TS삼국지게임

[#19]

mA 18	mB 5	mC 12	mD 9
mE 9	mF 12	mG 10	mH 11
mJ 9	mK 8	mP 4	mQ 10
mR 5	mT 4	igija 16	mX 4

{mP mX igija mB} <-> {mK mA mG} {mT mR}

#	Function	Description	return
17	ally(mD, mH)	군주 mD 과 군주 mH 가 동맹을 맺는다.	1
18	ally(mQ, mD)	군주 mQ 과 군주 mD 가 동맹을 맺는다.	1
19	ally(mB, mX)	군주 mB 과 군주 mX 가 동맹을 맺는다.	1

[#20]

mA 18	mB 5	mC 3	mD 5
mE 9	mF 12	mG 10	mH 6
mJ 9	mK 8	mP 4	mQ 10
mR 5	mT 4	igija 16	mX 4

{mP mX igija mB} <-> {mK mA mG} {mT mR}
{mC} <-> {mQ mD mH}

#	Function	Description	return
20	attack(mD, mC, holro)	군주 mD가 군주 mC를 공격한다. 4(mD) + 5(mH) = 9로 공격하지만 12(mC)로 방어한다. mC와 {mQ mD mH}동맹은 적대관계가 된다.	0

Problem analysis : 예제

TS삼국지게임

[#20]

mA 18	mB 5	mC 3	mD 5
mE 9	mF 12	mG 10	mH 6
mJ 9	mK 8	mP 4	mQ 10
mR 5	mT 4	igija 16	mX 4

{mP mX igija mB} <-> {mK mA mG} {mT mR}
{mC} <-> {mQ mD mH}

#	Function	Description	return
20	attack(mD, mC, holro)	군주 mD가 군주 mC를 공격한다. 4(mD) + 5(mH) = 9로 공격하지만 12(mC)로 방어한다. mC와 {mQ mD mH}동맹은 적대관계가 된다.	0

[#22]

mA 18	mB 5	mC 3	mD 5
mE 9	mF 12	mG 10	mH 6
mJ 9	mK 8	mP 4	mQ 10
mR 5	mT 4	igija 16	mX 4

{mP mX igija mB} <-> {mK mA mG} {mT mR}
{mC mJ} <-> {mQ mD mH}

#	Function	Description	return
21	ally(mJ, mC)	mJ와 mC가 동맹을 맺는다.	1
22	ally(mH, mJ)	mH와 mJ는 적대관계이므로 동맹을 맺을수 없다.	-2

Problem analysis : 예제

TS삼국지게임

[#22]

mA 18	mB 5	mC 3	mD 5
mE 9	mF 12	mG 10	mH 6
mJ 9	mK 8	mP 4	mQ 10
mR 5	mT 4	igija 16	mX 4

{mP mX igija mB} <-> {mK mA mG} {mT mR}
{mC mJ} <-> {mQ mD mH}

#	Function	Description	return
21	ally(mJ, mC)	mJ와 mC가 동맹을 맺는다.	1
22	ally(mH, mJ)	mH와 mJ는 적대관계이므로 동맹을 맺을수 없다.	-2

[#23]

mA 18	mB 5	mC 3	mD 5
mE 9	mF 12	mG 10	mH 6
mJ 9	mK 8	mP 4	mQ 10
mR 5	mT 4	igija 16	mX 4

{mP mX igija mB} <-> {mK mA mG} {mT mR}
{mE mF mJ mC} <-> {mQ mD mH}

#	Function	Description	return
23	ally(mF, mJ)	{mE mF}동맹이 {mJ mC}동맹과 동맹을 맺는다.	1

Problem analysis : 예제

TS삼국지게임

[#23]

mA 18	mB 5	mC 3	mD 5
mE 9	mF 12	mG 10	mH 6
mJ 9	mK 8	mP 4	mQ 10
mR 5	mT 4	igija 16	mX 4

{mP mX igija mB} <-> {mK mA mG} {mT mR}
{mE mF mJ mC} <-> {mQ mD mH}

#	Function	Description	return
23	ally(mF, mJ)	{mE mF}동맹이 {mJ mC}동맹과 동맹을 맺는다.	1

[#26]

mA 18	mB 15	mC 10	mD 5
mE 16	mF 19	mG 10	mH 6
mJ 16	mK 8	mP 14	mQ 10
mR 25	mT 4	igija 26	mX 14

#	Function	Description	return
24	recruit(mP, 10, 1)	{mP mX igija mB} 동맹이 각 영토에 병사 10명씩을 모집한다.	69
25	recruit(mR, 20, 0)		25
26	recruit(mF, 7, 1)		61

Problem analysis : 예제

TS삼국지게임

[#26]

mA 18	mB 15	mC 10	mD 5
mE 16	mF 19	mG 10	mH 6
mJ 16	mK 8	mP 14	mQ 10
mR 25	mT 4	igija 26	mX 14

#	Function	Description	return
24	recruit(mP, 10, 1)	{mP mX igija mB} 동맹이 각 영토에 병사 10명씩을 모집한다.	69
25	recruit(mR, 20, 0)		25
26	recruit(mF, 7, 1)		61

[#28]

mA 18	mB 15	mC 10	mD 5
mE 8	mF 10	mG 5	mH 6
mJ 8	biribiri 12	mP 14	mQ 10
mR 25	mT 4	igija 26	mX 14

{mP mX igija mB} <-> {mA mG}
{mP mX igija mB} <-> {mT mR}
{mE mF mJ mC biribiri} <-> {mQ mD mH}
{mE mF mJ mC biribiri} <-> {mA mG}

#	Function	Description	return
27	ally(mE, mC)	군주 mE 와 mC 는 이미 동맹 관계이다.	-1
28	attack(mC, mK, biribiri)	{mE mF mJ mC}동맹이 mK군주를 공격하여 성공하였다. biribiri 장군이 mK군주를 대신하여 새로운 군주가 된다. {mE mF mJ mC}동맹은 {mE mF mJ mC biribiri}로 {mK mA mG}동맹은 { mA mG}로 바뀐다. 두 동맹은 적대 관계가 된다.	1

Problem analysis : 예제

TS삼국지게임

[#28]

mA 18	mB 15	mC 10	mD 5
mE 8	mF 10	mG 5	mH 6
mJ 8	biribiri 12	mP 14	mQ 10
mR 25	mT 4	igija 26	mX 14

{mP mX igija mB} <-> {mA mG}
{mP mX igija mB} <-> {mT mR}
{mE mF mJ mC biribiri} <-> {mQ mD mH}
{mE mF mJ mC biribiri} <-> {mA mG}

#	Function	Description	return
27	ally(mE, mC)	군주 mE 와 mC 는 이미 동맹 관계이다.	-1
28	attack(mC, mK, biribiri)	{mE mF mJ mC}동맹이 mK군주를 공격하여 성공하였다. biribiri 장군이 mK군주를 대신하여 새로운 군주가 된다. {mE mF mJ mC}동맹은 {mE mF mJ mC biribiri}로 {mK mA mG}동맹은 { mA mG}로 바뀐다. 두 동맹은 적대 관계가 된다.	1

[#30]

mA 18	mB 15	mC 10	mD 5
mE 8	mF 10	mG 5	mH 6
mJ 8	biribiri 12	mP 14	mQ 10
mR 25	mT 4	igija 26	mX 14

{mP mX igija mB} <-> {mA mG mQ mD mH}
{mP mX igija mB} <-> {mT mR}
{mE mF mJ mC biribiri} <-> {mQ mD mH mA mG}

#	Function	Description	return
29	ally(biribiri, mA)	두 군주는 적대 관계이다.	-2
30	ally(mD, mA)	두 동맹{mQ mD mH}와 {mA mG}가 동맹을 맺는다.	1

Problem analysis : 예제

TS삼국지게임

[#30]

mA 18	mB 15	mC 10	mD 5
mE 8	mF 10	mG 5	mH 6
mJ 8	biribiri 12	mP 14	mQ 10
mR 25	mT 4	igija 26	mX 14

{mP mX igija mB} <-> {mA mG mQ mD mH}
{mP mX igija mB} <-> {mT mR}
{mE mF mJ mC biribiri} <-> {mQ mD mH mA mG}

#	Function	Description	return
29	ally(biribiri, mA)	두 군주는 적대 관계이다.	-2
30	ally(mD, mA)	두 동맹{mQ mD mH}와 {mA mG}가 동맹을 맺는다.	1

[#33]

mA 38	mB 15	mC 10	mD 25
mE 8	mF 34	mG 25	mH 26
mJ 8	biribiri 34	mP 14	mQ 30
mR 25	mT 4	igija 26	mX 14

#	Function	Description	return
31	recruit(mG, 20, 1)	{mA mG mQ mD mH}동맹이 각 영토에 병사 20명씩을 모집한다	144
32	recruit(biribiri, 22, 0)	biribiri 군주가 자신의 영토에 명사 22명을 모집한다.	34
33	recruit(mF, 24, 0)	mF 군주가 자신의 영토에 명사 24명을 모집한다.	34

Problem analysis : 예제

TS삼국지게임

[#33]

mA 38	mB 15	mC 10	mD 25
mE 8	mF 34	mG 25	mH 26
mJ 8	biribiri 34	mP 14	mQ 30
mR 25	mT 4	igija 26	mX 14

#	Function	Description	return
31	recruit(mG, 20, 1)	{mA mG mQ mD mH}동맹이 각 영토에 병사 20명씩을 모집한다	144
32	recruit(biribiri, 22, 0)	biribiri 군주가 자신의 영토에 명사 22명을 모집한다.	34
33	recruit(mF, 24, 0)	mF 군주가 자신의 영토에 명사 24명을 모집한다.	34

[#34]

mA 38	mB 15	mC 10	mD 25
mE 8	mF 17	mG 25	mH 26
mJ 8	biribiri 17	mP 0	mQ 30
mR 25	mT 4	igija 13	mX 7

{mP mX igija mB} <-> {mA mG mQ mD mH}
{mP mX igija mB} <-> {mT mR}
{mE mF mJ mC biribiri} <-> {mQ mD mH mA mG}
{mE mF mJ mC biribiri} <-> {mP mX igija mB}

#	Function	Description	return
34	attack(mJ, mP, nononono)	군주 mJ 가 군주 mP 를 공격한다.	0
		공격의 병사수와 방어의 병사수가 같아 남은 병사가 0이므로, 공격에 실패한다. {mE mF mJ mC biribiri} <-> {mP mX igija mB} 는 적대관계가 된다.	

Problem analysis : 예제

TS삼국지게임

[#34]

mA 38	mB 15	mC 10	mD 25
mE 8	mF 17	mG 25	mH 26
mJ 8	biribiri 17	mP 0	mQ 30
mR 25	mT 4	igija 13	mX 7

{mP mX igija mB} <-> {mA mG mQ mD mH}
{mP mX igija mB} <-> {mT mR}
{mE mF mJ mC biribiri} <-> {mQ mD mH mA mG}
{mE mF mJ mC biribiri} <-> {mP mX igija mB}

#	Function	Description	return
34	attack(mJ, mP, nononono)	군주 mJ 가 군주 mP 를 공격한다. 공격의 병사수와 방어의 병사수가 같아 남은 병사가 0이므로, 공격에 실패한다. {mE mF mJ mC biribiri} <-> {mP mX igija mB} 는 적대관계가 된다.	0

[#35]

mA 38	mB 15	mC 10	mD 13
mE 8	mF 17	mG 13	mH 65
mJ 8	biribiri 17	mP 0	mQ 15
mR 25	mT 4	igija 13	mX 7

{mP mX igija mB} <-> {mA mG mQ mD mH}
{mP mX igija mB} <-> {mT mR}
{mE mF mJ mC biribiri} <-> {mQ mD mH mA mG}
{mE mF mJ mC biribiri} <-> {mP mX igija mB}

#	Function	Description	return
35	attack(mB, mH, noway)	군주 mB 가 군주 mH 를 공격한다. 공격하는 병사가 0명이지만, 전투는 발생하고, 공격에 실패한다.	0

Problem analysis : 제약조건 및 API

- 군주의 이름과 장수의 이름이 문자열로 주어진다.
  renumbering int id로 바꾸어 사용하는 것이 필요할 것으로 보인다.

- 영토의 최대 크기가 25이므로 초기 군주의 수는 25*25 = 625가 최대 이다.
  이때 동맹의 수는 625개로 생각 할 수 있다.

- 이후 공격의 결과로 새로운 군주가 최대 8000번 나타날 수 있다.
  그렇다 하더라도 영토의 크기가 변하지 않으므로 동맹의 개수는 625개를 넘지 않는다.

- ally() 명령으로 한 무리의 동맹 번호가 변경될 수 있다.
  A동맹의 군주수가 X이고 B동맹의 군주수가 Y이라고 할 때,
  두 동맹을 어떻게 합칠 것인가?

Problem analysis

- attack() 함수가 8000회 호출된다.
  1. 전투가 발생하는지 확인하는 데는 주변 8곳을 확인하면 되므로
     O(상수) 시간에 처리할 수 있다.
  2. 전투가 발생하는 경우를 처리하는 것도 주변 8곳을 확인하면 되므로
     O(상수) 시간에 처리할 수 있다.
  3. 전투 결과는 방어한 경우이거나 점령당하는 것인데 이 또한
     O(상수) 시간에 처리할 수 있다.

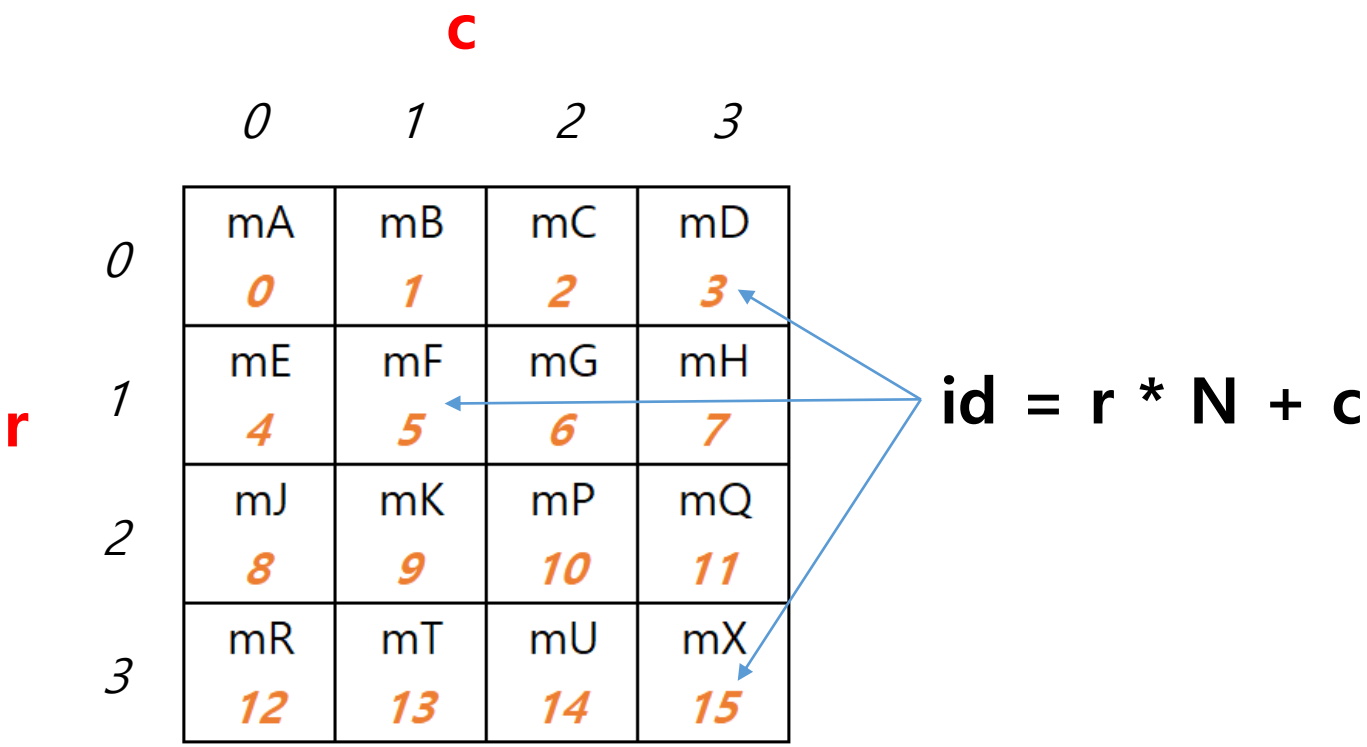
- recruit() 함수가 13000번 호출된다.
  1. sign==0인 경우 하나의 군주만 업데이트 하면 된다.
  2. sign==1인 경우 동맹에 포함된 한 무리의 군주 모두를 업데이트 해야 한다.

  어떻게 처리할 것인가?

Solution sketch

Solution sketch

- 문자열은 정수 id를 부여하여 사용하는 것으로 한다.
  N = 4인 경우 다음과 같이 numbering 할 수 있다.



Solution sketch

TS삼국지게임

- 두 동맹을 합치는 일은 union-find 알고리즘을 사용하는 것으로 한다.

- 두 그룹을 합치는데 경로압축을 사용하여 찾는 Find함수를 사용하는 경우
  시간 복잡도는 O(log*N) 이므로 거의 상수시간으로 볼 수 있다.

```
int Find(int r) {
    if (root[r] == r) return r;
    return root[r] = Find(root[r]);    // path compressing
}
```

- 한 동맹은 다른 여러 적대적 동맹들이 있을 수 있다.
  이들을 관리하기 위하여 각 동맹별로 unordered_set 을 사용할 수 있다.

```
unordered_set<int> enemyList[LM * 2]; // LM = 625
```

Solution sketch

- NN = N * N이라고 할 때,
  전체 영토의 각 군주 별 영토 번호는 0 ~ NN-1로 하고
  동맹 번호는 NN ~ 2*NN - 1로 하기로 한다.

```
int root[LM * 2]; // LM = 625
```

- 군주 별 영토번호로 동맹의 대표번호를 하는 경우
  동맹의 대표번호를 가진 군주가 점령당하는 상황에서 문제가 발생할 수 있기 때문이다.

- 군주 별 병사수를 관리하기 위하여 배열을 사용할 수 있다.

```
int soldierCnt[LM]; // LM = 625
```

- 이제 각 API함수 별 할 일을 정리해 보자

Solution sketch

TS삼국지게임

void init(int N, int soldier[][], char monarch[][][])

- :: N = N, NN = N * N
- renumbering hash table 초기화 `htab.clear();`
- 적대적 관계 목록 초기화 `enemyList[0 ~ NN * 2].clear();`
- renumbering id 등록
- 초기 군주 별 병사 수 등록
- 군주 별 초기 동맹 번호 지정 `root[mid] = mid + NN;`
- 초기 최고 조상 동맹 번호 지정 `root[mid + NN] = mid + NN;`
  `(mid = i * N + j)`

Solution sketch

TS삼국지게임

int ally(char monarchA[], char monarchB[])

- 군주A와 군주B의 renumbering id 를 구한다.

```
int aid = htab[monarchA], bid = htab[monarchB];
```

- 최고 조상 동맹 id 를 구한다.

```
int aroot = Find(aid), broot = Find(bid);
```

- 같은 동맹이라면 -1반환과 함께 함수 종료
- 적대적인 상태라면 -2반환과 함께 함수 종료
- 동맹이 가능한 경우 aroot로 합치기(broot로 합치는 것도 가능하다.)
- broot동맹의 적들 목록을 순회하며 aroot와도 적대적인 관계로 업데이트 한다.

```
for (int id : enemyList[broot]) {
    int froot = Find(id);              // 최고 조상 찾기
    enemyList[aroot].insert(froot); // aroot동맹의 적으로 froot 등록
    enemyList[froot].insert(aroot); // froot동맹의 적으로 aroot 등록 *****
}
```

- 동맹을 맺은 경우 1 반환과 함께 함수 종료

Solution sketch

TS삼국지게임

int attack(char monarchA[], char monarchB[], char general[])

- 군주A와 군주B의 renumbering id 를 구한다.

```
int aid = htab[monarchA], bid = htab[monarchB];
```

- 최고 조상 동맹 id 를 구한다.

```
int aroot = Find(aid), broot = Find(bid);
```

- 같은 동맹이라면 -1반환과 함께 함수 종료
- 공격받는 영토 주변에 공격하는 동맹이 없는 경우 -2반환과 함께 함수 종료

```
int isAroot(int r, int c, int aroot) { // (r, c)주변에 동맹번호 aroot가 있는지 검사
    for (int i = r - 1; i <= r + 1; ++i) {
        for (int j = c - 1; j <= c + 1; ++j) {
            if (i < 0 || i >= N || j < 0 || j >= N) continue;
            int id = i * N + j;
            if (Find(id) == aroot) return 1;
        }
    }
    return 0;
}
```

Solution sketch

TS삼국지게임

int attack(char monarchA[], char monarchB[], char general[]) : (계속)

- 전투가 발생한다면 적대적 관계로 등록한다.

```
enemyList[aroot].insert(broot);                    // 적대적 관계 등록
enemyList[broot].insert(aroot)
```

- 전투지역으로 모이는 aroot의 병사수와 broot의 병사수를 구한다.

```
int asum = getSum(br, bc, aroot);          // 전투 지역으로 모인 aroot동맹의 병사수
soldierCnt[bid] += getSum(br, bc, broot); // 전투 지역으로 모인 broot동맹의 병사수
```

- if (soldierCnt[bid] < asum) 인 경우 aroot동맹으로 흡수된다.
  1. 이전 군주 이름을 htab에서 삭제한다.
  2. 새로운 군주 general 을 등록한다.
  3. 새로운 군주의 초기 병사수를 등록한다.
  4. 동맹번호를 수정한다.
  5. 1을 반환하고 함수 종료한다.
- 지켜낸 경우 soldierCnt[bid] -= asum으로 업데이트 후 0을 반환하고 함수 종료한다.

Solution sketch

int attack(char monarchA[], char monarchB[], char general[]) : (계속)

```
int getSum(int r, int c, int gnum) {        // (r, c)와 그 주변에 동맹 번호가 gnum인 경우
    int sum = 0;                            // 가진 병사의 절반을 구하여 그 합을 반환
    for (int i = r - 1; i <= r + 1; ++i) {
        for (int j = c - 1; j <= c + 1; ++j) {
            if (i < 0 || i >= N || j < 0 || j >= N) // 범위를 벗어난 경우
                continue;
            int id = i * N + j;
            if (Find(id) == gnum) {
                int cnt = soldierCnt[id] / 2;
                soldierCnt[id] -= cnt;
                sum += cnt;
            }
        }
    }
    return sum;
}
```

Solution sketch

TS삼국지게임

int recruit(char monarch[], int num, int sign)

- 군주의 renumbering id 를 구한다.

```
int mid = htab[monarch];
```

- sign == 0 인 경우
  soldierCnt[mid] 의 병사수만 num만큼 증가시킨다.
- sign == 1인 경우
  mroot = Find(mid) 를 구하고 동맹 번호가 mroot인 모든 영역에 병사를 증가시킨다.
  이때 모든 지역을 탐색하므로 시간 복잡도는 O(NN) 될 수 있다.
  전체 시간 복잡도는 625(NN)*13000(함수호출수)*50(TC) = 406,250,000 이므로
  3초 안에 실행하는데 문제가 없다.

Solution sketch

[Summay]

- 문자열을 정수로 치환하여 배열의 인덱스로 사용할 수 있다.
  hash , trie, unordered_map<string, int> 등을 사용할 수 있다.

- 상호독립집합(Disjoint-Set)을 하나로 합치는 문제에서
  Union-Find 알고리즘을 사용할 수 있다.

- 함수 정의를 위해 고안한 알고리즘의 시간 복잡도를 계산하여
  문제 해결에 적용할 수 있다.

Code example

Code example

TS삼국지게임

```
#include <cmath>
#include <cstring>
#include <string>
#include <algorithm>
#include <unordered_set>
#include <unordered_map>
using namespace std;

const int LM = 25 * 25;
int N, NN, root[LM * 2], soldierCnt[LM];            // i*N+j
unordered_map<string, int> htab;
unordered_set<int> enemyList[LM * 2];

int Find(int r) {                                   // path compressing
    if (root[r] == r) return r;
    return root[r] = Find(root[r]);
}
```

Code example

TS삼국지게임

```
void init(int N, int soldier[25][25], char monarch[25][25][11]) {
    ::N = N, NN = N * N;
    htab.clear();
    for (int i = 0; i < NN * 2; ++i) {
        enemyList[i].clear();                       // 적대적 관계 목록 초기화
    }

    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            int mid = i * N + j;                    // renumbering id
            htab[monarch[i][j]] = mid;              // renumbering id 등록

            soldierCnt[mid] = soldier[i][j];        // 초기 병사수 등록

            enemyList[mid].clear();
            root[mid] = mid + NN;                   // 초기 동맹 번호 지정
            root[mid + NN] = mid + NN;              // 초기 최고 조상 동맹 번호
        }
    }
}

void destroy() {}
```

Code example

TS삼국지게임

```
int ally(char monarchA[11], char monarchB[11]) {     // O(NN)
    int aid = htab[monarchA], bid = htab[monarchB]; // renumbering int id 구하기
    int aroot = Find(aid), broot = Find(bid);        // 동맹 id 구하기
    if (aroot == broot) return -1;                   // 같은 동맹인가?
    if (enemyList[aroot].count(broot)) return -2;    // 적인 경우

    root[broot] = aroot;                             // aroot 동맹으로 합치기
    for (int id : enemyList[broot]) {
        int froot = Find(id);
        enemyList[aroot].insert(froot);              // aroot동맹의 적으로 froot 등록
        enemyList[froot].insert(aroot);              // froot동맹의 적으로 aroot 등록 *****
    }
    return 1;
}

int isAroot(int r, int c, int aroot) {              // (r, c)주변에 동맹번호 aroot가 있는지 검사
    for (int i = r - 1; i <= r + 1; ++i) {
        for (int j = c - 1; j <= c + 1; ++j) {
            if (i < 0 || i >= N || j < 0 || j >= N) continue;
            int id = i * N + j;
            if (Find(id) == aroot) return 1;
        }
    }
    return 0;
}
```

Code example

TS삼국지게임

```
int getSum(int r, int c, int gnum) {                    // (r, c)와 그 주변에 동맹 번호가 gnum인 경우
    int sum = 0;                                        // 가진 병사의 절반을 구하여 그 합을 반환
    for (int i = r - 1; i <= r + 1; ++i) {
        for (int j = c - 1; j <= c + 1; ++j) {
            if (i < 0 || i >= N || j < 0 || j >= N) // 범위를 벗어난 경우
                continue;
            int id = i * N + j;
            if (Find(id) == gnum) {
                int cnt = soldierCnt[id] / 2;
                soldierCnt[id] -= cnt;
                sum += cnt;
            }
        }
    }
    return sum;
}
```

Code example

TS삼국지게임

```
int attack(char monarchA[11], char monarchB[11], char general[11]) { // O(상수)
    int aid = htab[monarchA], bid = htab[monarchB];
    int aroot = Find(aid), broot = Find(bid);
    if (aroot==broot) return -1;                    // 이미 동맹관계
    int br = bid / N, bc = bid % N;
    if (isAroot(br, bc, aroot)==0)                  // 공격받는 영토 주변에 공격하는 동맹이 없는 경우
        return -2;

    enemyList[aroot].insert(broot);                 // 적대적 관계 등록
    enemyList[broot].insert(aroot);

    int asum = getSum(br, bc, aroot);               // 전투 지역으로 모인 aroot동맹의 병사수
    soldierCnt[bid] += getSum(br, bc, broot);       // 전투 지역으로 모인 broot동맹의 병사수

    int res = 0;                                    // 공격실패를 가정하여 초기값으로
    if (soldierCnt[bid] < asum) {                   // aroot동맹으로 흡수됨
        htab.erase(monarchB);
        res = 1;                                    // 공격이 성공함
        htab[general] = bid;                        // 새로운 군주 등록
        soldierCnt[bid] = asum - soldierCnt[bid];   // 새로운 군주 초기 병사수
        root[bid] = aroot;                          // 동맹번호 변경
    }
    else soldierCnt[bid] -= asum;                   // 지켜낸 경우

    return res;
}
```

Code example

TS삼국지게임

```
int recruit(char monarch[11], int num, int sign) { // O(NN)
    int mid = htab[monarch];
    if (sign == 0) {                                // mid의 병사만 증가
        soldierCnt[mid] += num;
        return soldierCnt[mid];
    }
    int res = 0;
    int mroot = Find(mid);                          // 동맹번호가 mroot인 모든 지역에 병사 증가
    for (int i = 0; i < NN;++i) {                   // O(NN)
        if (Find(i) == mroot) {
            soldierCnt[i] += num;
            res += soldierCnt[i];
        }
    }
    return res;
}
```

Thank you.