안녕하세요, 23 년 동계 대학생 S/W 알고리즘 역량강화 과정을 수강하고 있는 박희조 교육생입니다. 오늘 진행된 '코드배틀 2. 삼국지 게임' 문제 풀이를 올려드립니다. 프로그래밍 언어는 Java 입니다.

코드배틀 2. 삼국지 게임

- 문제에서 진행하는 시뮬레이션은 크게 3 가지입니다.

1. 어떤 군주와 다른 군주와의 동맹 (ally)
2. 어떤 군주와 그 동맹 군주들이 다른 군주의 영토를 공격 (attack)
3. 병사 모집 (recruit)

- 연결리스트를 활용한 그룹 맺기

문제를 읽으면서 이 문제는 동맹 그룹을 잘 맺는 것이 중요한 포인트라는 생각을 했습니다. 개수 제한이 없고 동맹으로 두 집단이 합쳐질 때의 계산을 편리하게 하기 위해 노드를 활용한 Linked List 를 활용하기로 했습니다.

~~(한 줄로 간단하게 작성됐지만, 이를 떠올리기에는 오랜 시간이 걸렸습니다..)~~

```
class Monarchs{ //군주의 정보
    int groupNo;    //그룹 번호
    int x;  //영토의 x 위치
    int y;  //영토의 y 위치
    Monarchs next;  //다음 군주로의 연결 링크

    Monarchs(int x, int y, int groupNo){
        this.x = x;
        this.y = y;
        this.groupNo = groupNo;
        this.next = null;
    }
}
```

군주를 의미하는 노드 Monarchs 에는 그룹 번호(groupNo), 군주의 좌표(x,y)와 다음 군주로의 연결 링크(next)가 있습니다. 링크(next)를 활용하여 같은 동맹(그룹)인 여러 군주들의 연결 리스트가 구현됩니다.

- 그룹의 기준은 그룹번호

처음에 주어지는 군주들은 총 N*N 명입니다. 이들은 모두 동맹을 맺지 않은 상태기 때문에, 동맹 그룹은 총 N*N 개로 형성이 됩니다.

또한 문제를 진행하면서 그룹이 둘로 쪼개지거나 완전히 새로운 그룹이 등장하는 일은 없습니다. 이를 바탕으로, 그룹에 번호를 매기고 동맹이 되면 하나의 번호로 통합하여 그룹을 관리하도록

했습니다. (통합 방식은 오른쪽 변수에 적힌 군주와 동맹들이 왼쪽 변수에 적힌 군주의 그룹으로 편입됩니다.)

```
class IndexNode{
    int groupNo;    //그룹번호
    List<Integer> enemyGroupNo; //해당 동맹의 적들
    Monarchs allyMonarchs;  //해당 그룹 번호를 가진 동맹국들

    IndexNode(int groupNo){
        this.groupNo = groupNo;
        this.allyMonarchs = new Monarchs(-1, -1, groupNo);
        this.enemyGroupNo = new ArrayList<>();
    }
}
```

이 그룹 번호에 따른 동맹, 적 정보들을 보관하는 노드 IndexNode 입니다. 동맹 목록은 위에서 생성한 노드 Monarchs 로 연결리스트로 이루어져 있습니다. 적 목록은 리스트 형태로 적의 그룹 번호들을 보관합니다.

보다 직관적인 설명을 위해 그림으로 시뮬레이션을 진행해보겠습니다. 문제에서 주어진 예제를 참고해주세요.

예제 풀이

1. init()

[1] init()

soldierMap 병사 수

12	5	12	9
9	12	14	11
9	6	15	10
5	7	11	10

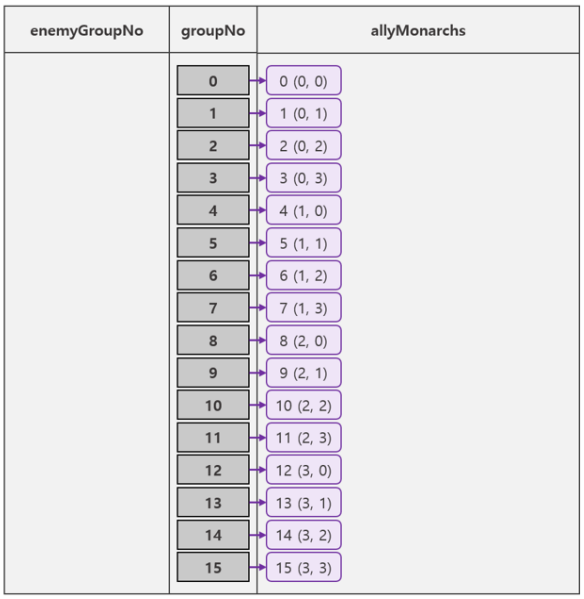
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	lingtong	caopi
yuanshu	machao	guanyu	gaoren
weiyan	huangzhong	lukang	xuhuang

nodeMap 그룹번호 (x좌표, y좌표)

0 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	5 (1, 1)	6 (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	13 (3, 1)	14 (3, 2)	15 (3, 3)

groupList



초기화에는 4 단계가 이루어집니다.

- soldierMap 에 병사 수를 저장
- monarchNameMap 에 군주 이름을 저장
- nodeMap 에 각 군주의 초기 그룹 번호, 좌표를 저장
- groupList 에 그룹 번호별 헤더를 생성하고 군주들을 연결

2. ally(chengpu, lingtong)

[2] ally(chengpu, lingtong)

soldierMap 병사 수

12	5	12	9
9	12	14	11
9	6	15	10
5	7	11	10

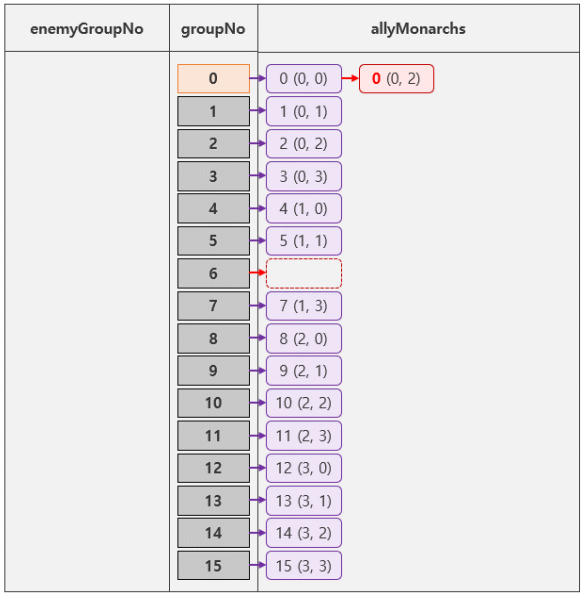
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	**lingtong**	caopi
yuanshu	machao	guanyu	gaoren
weiyan	huangzhong	lukang	xuhuang

nodeMap 그룹번호 (x좌표, y좌표)

0 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	5 (1, 1)	6->**0** (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	13 (3, 1)	14 (3, 2)	15 (3, 3)

groupList



군주 'cheungpu'와 'lingtong'가 서로 동맹을 맺습니다. 동맹은 항상 오른쪽 군주가 왼쪽 군주로

흡수된다고 가정을 했기 때문에, lingtong 은 그룹 번호 6 을 cheungpu 의 그룹번호 0 으로 변경하고, 이에 따라 groupList 의 그룹 번호 6 에서 0 으로 이동합니다.

3. ally(machao, lingtong)

[3] ally(machao, lingtong)

soldierMap 병사 수

12	5	12	9
9	12	14	11
9	6	15	10
5	7	11	10

monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	**lingtong**	caopi
yuanshu	**machao**	guanyu	gaoren
weiyan	huangzhong	lukang	xuhuang

nodeMap 그룹번호 (x좌표, y좌표)

0->**9** (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	5 (1, 1)	0->**9** (1, 2)	7 (1, 3)
8 (2, 0)	**9** (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	13 (3, 1)	14 (3, 2)	15 (3, 3)

groupList

enemyGroupNo	groupNo	allyMonarchs
	0	
	1	1 (0, 1)
	2	2 (0, 2)
	3	3 (0, 3)
	4	4 (1, 0)
	5	5 (1, 1)
	6	
	7	7 (1, 3)
	8	8 (2, 0)
	9	9 (2, 1) → **9** (0, 0) → **9** (1, 2)
	10	10 (2, 2)
	11	11 (2, 3)
	12	12 (3, 0)
	13	13 (3, 1)
	14	14 (3, 2)
	15	15 (3, 3)

군주 'machao'와 'lingtong'이 서로 동맹을 맺습니다. 마찬가지로 동맹은 오른쪽 군주들이 왼쪽 그룹 번호로 이동합니다. lingtong 의 그룹 번호는 0 번인데, 0 번 그룹에는 'chungpu'도 있으니 두 군주가 machao 의 그룹 번호 9 로 이동합니다.

4. recruit(machao, 3, 0)

[4] recruit(machao, 3, 0)

soldierMap 병사 수

12	5	12	9
9	12	14	11
9	6 **+3 =9**	15	10
5	7	11	10

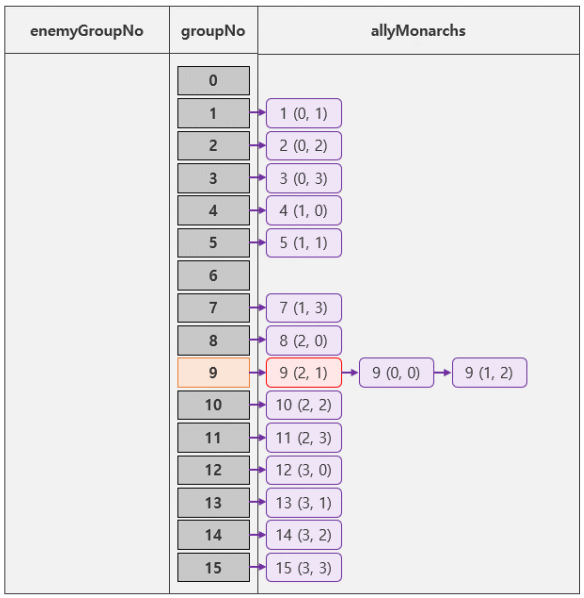
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	lingtong	caopi
yuanshu	**machao**	guanyu	gaoren
weiyan	huangzhong	lukang	xuhuang

nodeMap 그룹번호 (x좌표, y좌표)

9 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	5 (1, 1)	9 (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	13 (3, 1)	14 (3, 2)	15 (3, 3)

groupList



군주 'machao'의 병력을 3 증가시킵니다. soldierMap 의 해당 위치의 값을 3 증가시킵니다.

5. recruit(chengpu, 6, 1)

[5] recruit(chengpu, 6, 1)

soldierMap 병사 수

12 **+6 =18**	5	12	9
9	12	14 **+6 =20**	11
9	9 **+6 =15**	15	10
5	7	11	10

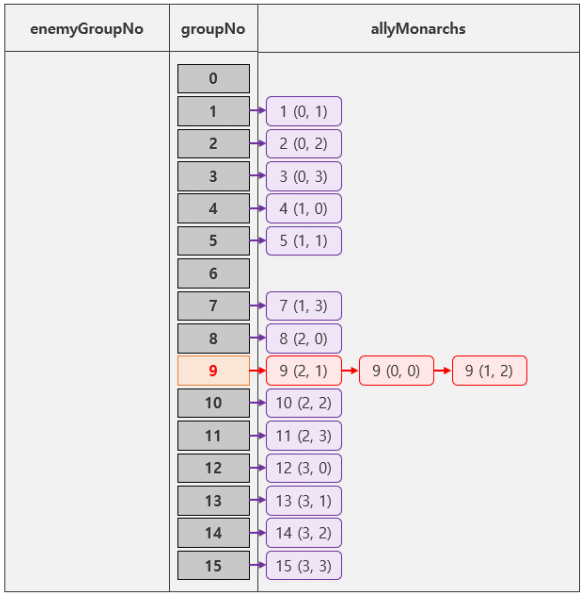
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	**lingtong**	caopi
yuanshu	**machao**	guanyu	gaoren
weiyan	huangzhong	lukang	xuhuang

nodeMap 그룹번호 (x좌표, y좌표)

9 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	5 (1, 1)	9 (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	13 (3, 1)	14 (3, 2)	15 (3, 3)

groupList



군주 'chengpu'와 그 동맹국들의 병력을 6 씩 증가시킵니다. 군주 'cheungpu'의 그룹번호는 9 번이며, 그룹 9 번에는 'machao'와 'lingtong'이 함께 있습니다. 이들의 병력들을 모두 6 증가시킵니다.

6. ally(guanyu, xuhuang)

[6] ally(guanyu, xuhuang)

soldierMap 병사 수

18	5	12	9
9	12	20	11
9	15	15	10
5	7	11	10

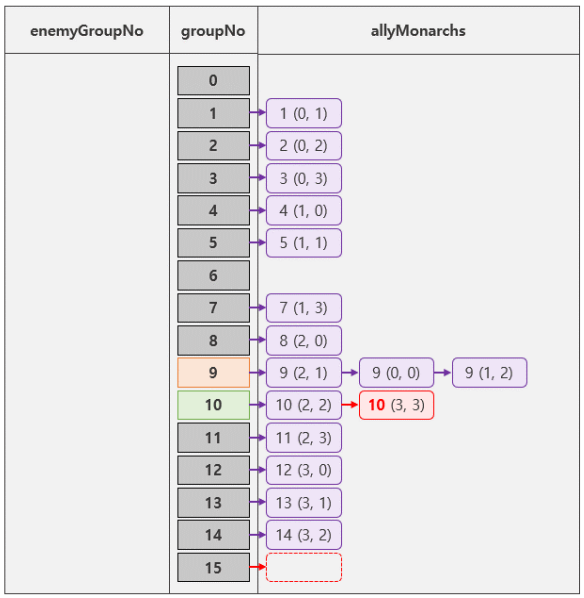
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	lingtong	caopi
yuanshu	machao	**guanyu**	gaoren
weiyan	huangzhong	lukang	**xuhuang**

nodeMap 그룹번호 (x좌표, y좌표)

9 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	5 (1, 1)	9 (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	**10** (2, 2)	11 (2, 3)
12 (3, 0)	13 (3, 1)	14 (3, 2)	~~15~~->**10** (3, 3)

groupList



이번에는 군주 'guanyu'와 'xuhuang'이 동맹을 맺습니다. (역시 오른쪽에서 왼쪽으로 동맹을 맺는다고 가정했기 때문에) 군주 'xuhuang'은 기존 그룹번호 15 에서 군주 'guanyu'의 그룹번호인 10 번으로 변경 및 노드가 이동합니다.

7. attack(chengpu, machao, ganada)

[7] attack(chengpu, machao, ganada)

soldierMap 병사 수

18	5	12	9
9	12	20	11
9	15	15	10
5	7	11	10

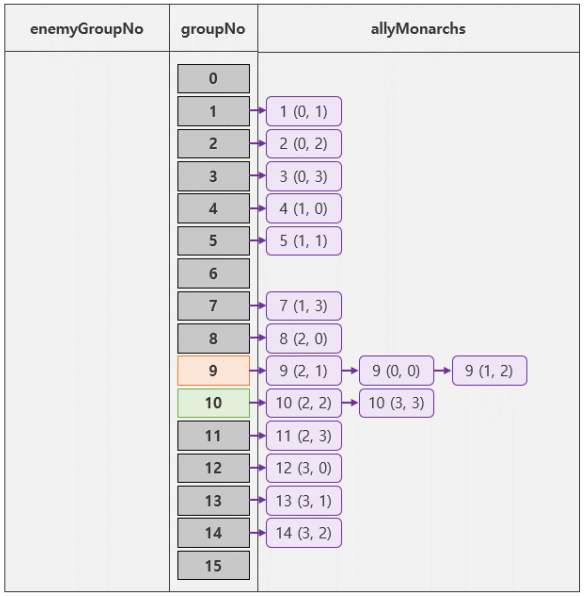
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	lingtong	caopi
yuanshu	**machao**	guanyu	gaoren
weiyan	huangzhong	lukang	xuhuang

nodeMap 그룹번호 (x좌표, y좌표)

9 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	5 (1, 1)	9 (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	13 (3, 1)	14 (3, 2)	10 (3, 3)

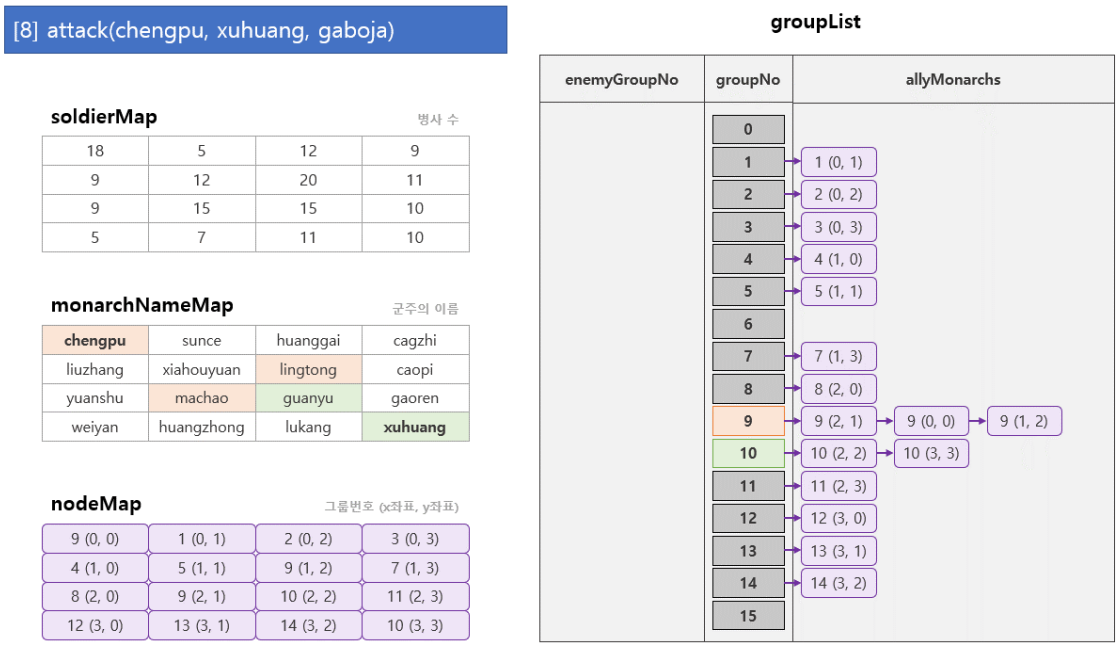
groupList



군주 'cheungpu'가 군주 'machao'를 공격하려 합니다. 그런데 확인해보니 군주 'cheungpu'와 군주 'machao'의 그룹번호가 9 로 동일합니다. 이는 서로가 동맹임을 의미하기 때문에, 공격이

이루어지지 않습니다.

8. attack(chengpu, xuhuang, gaboja)



이번에는 군주 'chengpu'가 군주 'xuhuang'을 공격하려 합니다. 그런데 'xuhuang' 주변 8 개의 구역에는 군주 'chengpu'의 그룹 번호 9 를 가진 군주들이 없습니다. 이렇게 되면 공격이 이루어지지 않습니다.

9. attack(chengpu, guanyu, chacha)

[9] attack(chengpu, guanyu, chacha)

soldierMap 병사 수

18	5	12	9
9	12	20 -> **10**	11
9	15 -> **8**	15 -> **20-17**	10
5	7	11	10 -> **5**

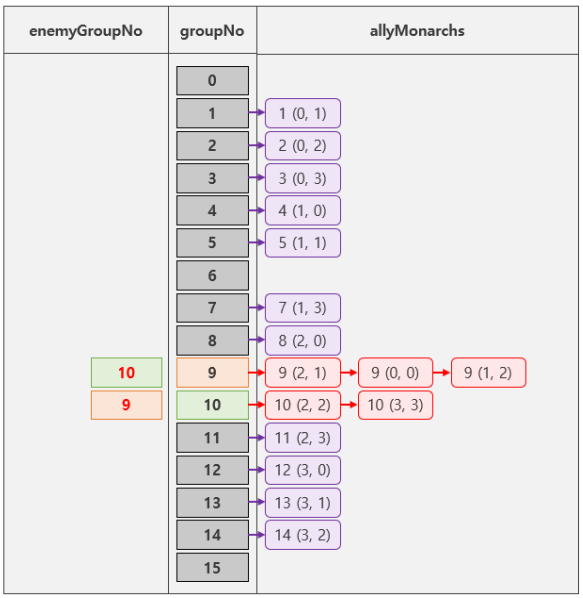
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	lingtong	caopi
yuanshu	machao	**guanyu**	gaoren
weiyan	huangzhong	lukang	xuhuang

nodeMap 그룹번호 (x좌표, y좌표)

9 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	5 (1, 1)	9 (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	13 (3, 1)	14 (3, 2)	10 (3, 3)

groupList



끈질긴 군주 'chengpu'가 이번엔 군주 'guanyu'를 공격하려 합니다. 다행히 군주 'guanyu'와 그룹 번호도 다르고, 주변 구역에 'chengpu'의 동맹 군주들이 있습니다. 문제에서 주어진대로 파견할 병사 수를 계산하여 공격을 진행합니다.
아쉽게도 병력이 부족하여 공격에 실패했습니다. 다만 서로 공격을 진행했기 때문에 그룹 번호 9 와 10 을 가진 두 연합은 서로 적대 관계가 되었습니다. 이를 표시하기 위해 groupNo 에 적의 그룹 번호를 enemyGroupNo 에 추가합니다.

이러한 방식으로 프로그램은 진행됩니다. 이후 부분은 사진으로 대체하겠습니다.

10. recruit(xuhuang, 10, 1)

[10] recruit(xuhuang, 10, 1)

soldierMap 병사 수

18	5	12	9
9	12	10	11
9	8	3 -> **13**	10
5	7	11	5 -> **15**

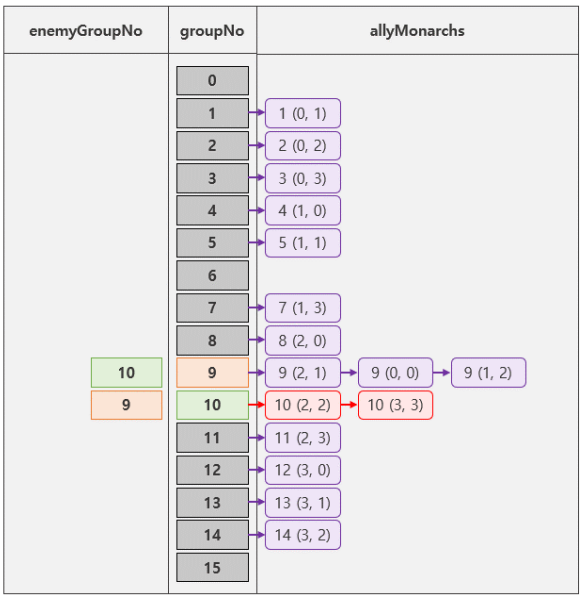
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	lingtong	caopi
yuanshu	machao	**guanyu**	gaoren
weiyan	huangzhong	lukang	**xuhuang**

nodeMap 그룹번호 (x좌표, y좌표)

9 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	5 (1, 1)	9 (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	13 (3, 1)	14 (3, 2)	10 (3, 3)

groupList



11. attack(guanyu, lukang, igija)

[11] attack(guanyu, lukang, igija)

soldierMap 병사 수

18	5	12	9
9	12	10	11
9	8	13 -> **7**	10
5	7	11 -> **2**	15 -> **8**

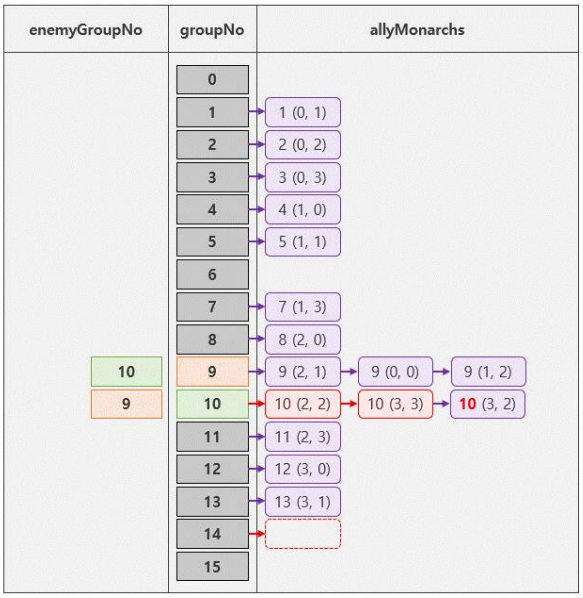
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	lingtong	caopi
yuanshu	machao	**guanyu**	gaoren
weiyan	huangzhong	~~**lukang**~~ **igija**	xuhuang

nodeMap 그룹번호 (x좌표, y좌표)

9 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	5 (1, 1)	9 (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	13 (3, 1)	~~14~~->**10** (3, 2)	10 (3, 3)

groupList



12. ally(igija, chengpu)

[12] ally(igija, chengpu)

soldierMap 병사 수

18	5	12	9
9	12	10	11
9	8	7	10
5	7	2	8

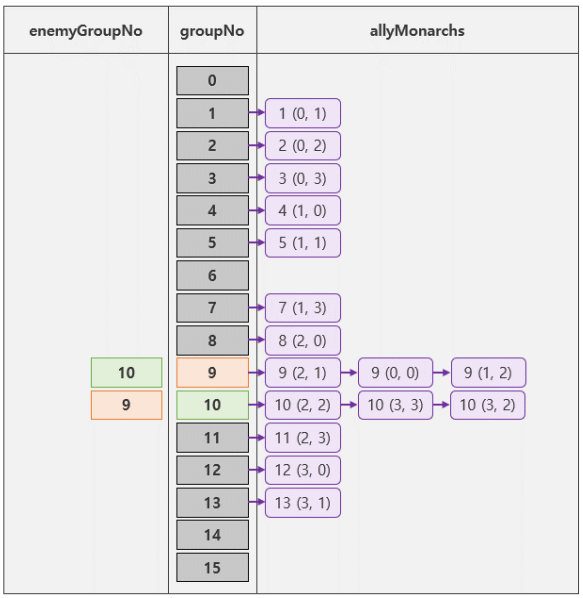
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	lingtong	caopi
yuanshu	machao	guanyu	gaoren
weiyan	huangzhong	**igija**	xuhuang

nodeMap 그룹번호 (x좌표, y좌표)

9 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	5 (1, 1)	9 (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	13 (3, 1)	10 (3, 2)	10 (3, 3)

groupList



13. ally(weiyan, huangzhong)

[13] ally(weiyan, huangzhong)

soldierMap 병사 수

18	5	12	9
9	12	10	11
9	8	7	10
5	7	2	8

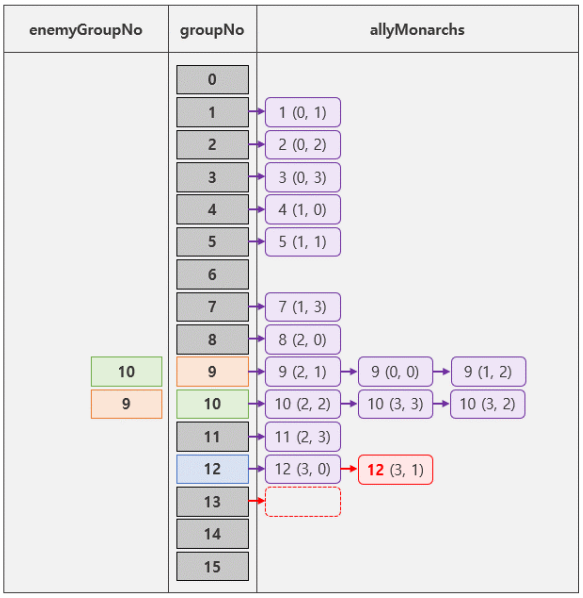
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	lingtong	caopi
yuanshu	machao	guanyu	gaoren
weiyan	**huangzhong**	igija	xuhuang

nodeMap 그룹번호 (x좌표, y좌표)

9 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	5 (1, 1)	9 (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	~~13~~ ->**12** (3, 1)	10 (3, 2)	10 (3, 3)

groupList



14. ally(liuzhang, xiahouyuan)

[14] ally(liuzhang, xiahouyuan)

soldierMap 병사 수

18	5	12	9
9	12	10	11
9	8	7	10
5	7	2	8

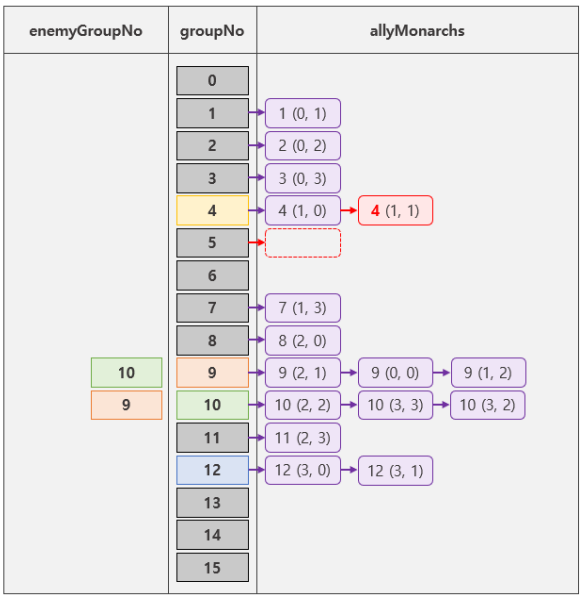
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	**xiahouyuan**	lingtong	caopi
yuanshu	machao	guanyu	gaoren
weiyan	huangzhong	igija	xuhuang

nodeMap 그룹번호 (x좌표, y좌표)

9 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	~~5~~->**4** (1, 1)	9 (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	12 (3, 1)	10 (3, 3)	10 (3, 2)

groupList



15. recruit(igija, 10, 0)

[15] recruit(igija, 10, 0)

soldierMap 병사 수

18	5	12	9
9	12	10	11
9	8	7	10
5	7	2 +10 = **12**	8

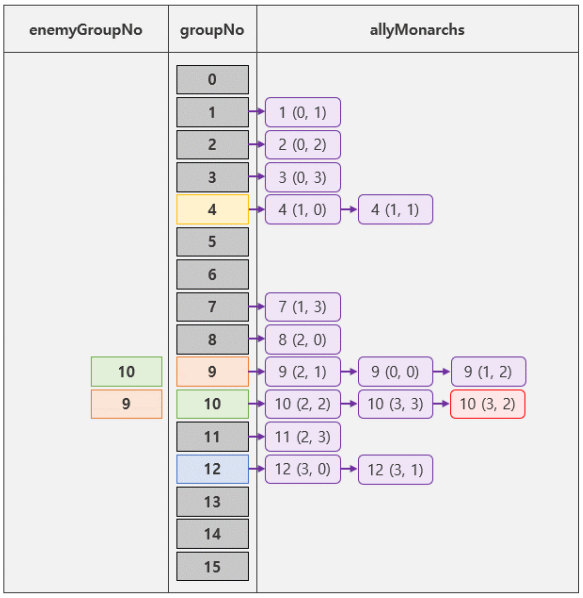
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	lingtong	caopi
yuanshu	machao	guanyu	gaoren
weiyan	huangzhong	**igija**	xuhuang

nodeMap 그룹번호 (x좌표, y좌표)

9 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	4 (1, 1)	9 (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	12 (3, 1)	10 (3, 2)	10 (3, 3)

groupList



16. attack(huangzhong, igija, jjukkumi)

[16] attack(huangzhong, igija, jjukkumi)

soldierMap 병사 수

18	5	12	9
9	12	10	11
9	8	7 ->**4**	10
5	7 ->**4**	12 ->**(19-3)**	8 -> **4**

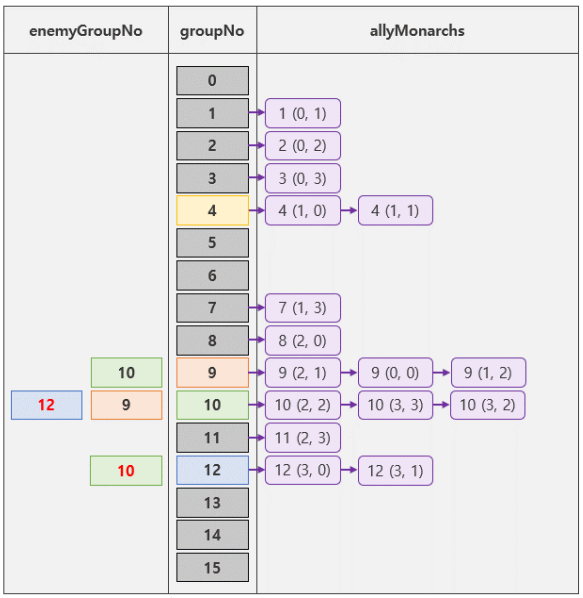
monarchNameMap 군주의 이름

chengpu	sunce	huanggai	cagzhi
liuzhang	xiahouyuan	lingtong	caopi
yuanshu	machao	guanyu	gaoren
weiyan	**huangzhong**	**igija**	xuhuang

nodeMap 그룹번호 (x좌표, y좌표)

9 (0, 0)	1 (0, 1)	2 (0, 2)	3 (0, 3)
4 (1, 0)	4 (1, 1)	9 (1, 2)	7 (1, 3)
8 (2, 0)	9 (2, 1)	10 (2, 2)	11 (2, 3)
12 (3, 0)	12 (3, 1)	10 (3, 2)	10 (3, 3)

groupList



이후 생략

제가 만든 코드는 이러한 방식으로 진행되었습니다. 코드에 대해 궁금한 사항이나 기타 의견이 있다면, 댓글 남겨주시면 답변 드리겠습니다.
긴 글 읽어주셔서 감사합니다. 수료까지 다들 화이팅하세요!

---

* 코드 전문입니다. (UserSolution.java)

```
import java.util.ArrayList;
import java.util.List;

/**
 * 문제 내용: 삼국지 게임을 구현
 *
 * 전략
 * - 모든 영토들을 초기화(init)하는 과정에서 각 영토들에 그룹 번호를 부여한다.
 * - 이 그룹 번호를 지칭하는 노드(IndexNode)에는 그룹 번화 동맹국, 적대국들의 정보가 담겨있다.
```

```
 * - A,B가 동맹을 하면 A쪽으로 그룹 번호를 통일시킨다.
 *   - B의 그룹번호에 들어있는 모든 동맹국들을 통으로 A에 옮긴다.
 *   - B의 그룹번호에 들어있는 모든 적대국들을 통으로 A의 적대국 목록에 옮긴다.
 *      - 이 때, B를 적대국으로 지정한 모든 그룹번호들에 A라는 적대국을 추가한다. (서로가 적으로 인지하게)
 * - A가 B영토를 침략해서 성공한다면, B를 떼어내서 그룹번호를 A로 바꾸고 군주 이름을 바꾸고 A로 넘어간다.
 *
 * @autho HeejoPark
 */
class IndexNode{
    int groupNo;    //그룹번호
    List<Integer> enemyGroupNo; //해당 동맹의 적들
    Monarchs allyMonarchs;  //해당 그룹 번호를 가진 동맹국들

    IndexNode(int groupNo){
        this.groupNo = groupNo;
        this.allyMonarchs = new Monarchs(-1, -1, groupNo);
        this.enemyGroupNo = new ArrayList<>();
    }
}
class Monarchs{ //군주의 정보
    int groupNo;    //그룹 번호
    int x;  //영토의 x 위치
    int y;  //영토의 y 위치
    Monarchs next;  //다음 군주로의 연결 링크

    Monarchs(int x, int y, int groupNo){
        this.x = x;
        this.y = y;
        this.groupNo = groupNo;
        this.next = null;
    }
}
class UserSolution {
    Monarchs[][] nodeMap;    //군주 노드
```

```
    IndexNode[] groupList;   //군주들의 그룹번호 정보
    int[][] soldierMap;      //해당 영토의 병사 수
    String[][] monarchNameMap;  //해당 군주의 이름
    int N;
    void init(int N, int mSoldier[][], char mMonarch[][][])
    {
        this.N = N;
        groupList = new IndexNode[N*N];
        nodeMap = new Monarchs[N][N];
        soldierMap = new int[N][N];
        monarchNameMap = new String[N][N];
        for(int i =0; i<N; i++){
            for(int j = 0; j<N; j++){
                int groupNo = (i * N) + j;
                nodeMap[i][j] = new Monarchs(i, j, groupNo);    //노드 생성
                groupList[groupNo] = new IndexNode(groupNo);
//그룹리스트 초기화
                groupList[groupNo].allyMonarchs = new Monarchs(-1, -1,
groupNo);    //동맹리스트 초기화
                groupList[groupNo].allyMonarchs.next = nodeMap[i][j];
//해당 군주 동맹에 넣기
                soldierMap[i][j] = mSoldier[i][j];  //군사 수 입력받기
                monarchNameMap[i][j] = charToString(mMonarch[i][j]);
//군주 이름 입력받기
            }
        }
    }
    void destroy()
    {
    }
    int ally(char mMonarchA[], char mMonarchB[])
    {
        //해당 군주의 노드를 찾기
        Monarchs monarchA = findNodeOfMonarch(mMonarchA);
        Monarchs monarchB = findNodeOfMonarch(mMonarchB);

        //군주가 같으면 -1 반환
        if(monarchA == monarchB){
            return -1;
        }
```

```
        //군주들이 이미 동맹관계라면 -1 반환
        if(isAlreadyAlly(monarchA, monarchB)){
            return -1;
        }
        //군주들이 이미 적대관계라면 -2 반환
        if(isAlreadyEnemy(monarchA, monarchB)){
            return -2;
        }
        //두 군주는 동맹관계가 된다.
        setAlly(monarchA, monarchB);
        return 1;
    }
    int attack(char mMonarchA[], char mMonarchB[], char mGeneral[])
    {
        //군주 노드를 찾기
        Monarchs monarchA = findNodeOfMonarch(mMonarchA);
        Monarchs monarchB = findNodeOfMonarch(mMonarchB);

        //만약 두 군주가 동맹관계라면 -1 반환
        if(isAlreadyAlly(monarchA, monarchB)){
            return -1;
        }

        //monarchB 주변에 monarchA 의 동맹국이 없다면 -2 반환
        boolean isMonarchAAllyNearMonarchB = false;
        findAllys : for(int i =-1; i<=1; i++){
            for(int j =-1; j<=1; j++){
                if(i==0 && j==0){
                    continue;
                }
                int nextX = monarchB.x + i;
                int nextY = monarchB.y + j;
                if(nextX < 0 || nextX >= N || nextY < 0 || nextY >=
N){ //배열 밖을 벗어나면 패스
                    continue;
                }
                //해당 위치가 A 와 동맹국이라면 주변에 동맹국이 존재하는
것이다.
                if(isAlreadyAlly(nodeMap[nextX][nextY], monarchA)){
                    isMonarchAAllyNearMonarchB = true;
```

```
                    break findAllys;
                }
            }
        }
        if(!isMonarchAAllyNearMonarchB){
            return -2;
        }

        //전투 발생. 각 군주의 모든 노드에 적대관계를 표시
        setEnemy(monarchA, monarchB);

        //monarchA 가 속한 동맹팀이 지역 monarchB 를 친다.
        int aSoldierSum = 0;
        int bSoldierSum = 0;
        for(int i =-1; i<=1; i++){
            for(int j =-1; j<=1; j++){
                if(i==0 && j==0){   //공격 대상인 monarchB 자기 자신은 모든
병력을 동원한다.
                    bSoldierSum += soldierMap[monarchB.x][monarchB.y];
                    continue;
                }
                int nextX = monarchB.x + i;
                int nextY = monarchB.y + j;
                if(nextX < 0 || nextX >= N || nextY < 0 || nextY >=
N){ //배열 밖을 벗어나면 패스
                    continue;
                }
                //해당 위치가 A 와 동맹국이라면 a 쪽에 병력을 지원한다.
                if(isAlreadyAlly(nodeMap[nextX][nextY], monarchA)){
                    aSoldierSum += (soldierMap[nextX][nextY] / 2);
                    soldierMap[nextX][nextY] -= (soldierMap[nextX][nextY] /
2);
                }
                //해당 위치가 B 와 동맹국이라면 b 쪽에 병력을 지원한다.
                else if(isAlreadyAlly(nodeMap[nextX][nextY], monarchB)){
                    bSoldierSum += (soldierMap[nextX][nextY] / 2);
                    soldierMap[nextX][nextY] -= (soldierMap[nextX][nextY] /
2);
                }
                //어느 쪽도 아니라면 패스
            }
```

```
        }
        //전투 결과 반영
        soldierMap[monarchB.x][monarchB.y] = Math.abs(aSoldierSum -
bSoldierSum);
        //점령 성공했다면
        if(aSoldierSum > bSoldierSum){
            //monarchB 에 관련된 정보를 지운다.
            Monarchs curIndex =
groupList[monarchB.groupNo].allyMonarchs.next;
            Monarchs prevIndex = groupList[monarchB.groupNo].allyMonarchs;
            while(curIndex.next != null){
                //해당 노드를 찾았다면
                if(curIndex.x == monarchB.x && curIndex.y == monarchB.y){
                    break;
                }
                curIndex = curIndex.next;
                prevIndex = prevIndex.next;
            }
            //찾은 노드를 B 에서 끊는다.
            prevIndex.next = curIndex.next;
            //A 의 영토로 변경하고, mGeneral 이 군주가 된다.
            curIndex.groupNo = monarchA.groupNo;
            String str = charToString(mGeneral);
            monarchNameMap[curIndex.x][curIndex.y] = str;
            //A 에 집어넣는다.
            Monarchs newIndex = groupList[monarchA.groupNo].allyMonarchs;
            while(newIndex.next != null){
                newIndex = newIndex.next;
            }
            newIndex.next = curIndex;
            curIndex.next = null;
            return 1;
        }
        //점령 실패
        else{
            return 0;
        }
    }


    int recruit(char mMonarch[], int mNum, int mOption)
```

```
    {
        int sum = 0;
        Monarchs monarch = findNodeOfMonarch(mMonarch);
        switch(mOption){
            case 0: //0번 옵션이면 해당 군주에만 병사를 추가한다.
                soldierMap[monarch.x][monarch.y] += mNum;    //해당 군주에
병사 추가
                sum += soldierMap[monarch.x][monarch.y];      //추가된
병사를 포함하여 병사 수를 출력
                break;
            case 1: //1번 옵셥이면 모든 동맹의 영토에 병사를 추가한다.
                int groupNo = monarch.groupNo;
                Monarchs increaseIndex = groupList[groupNo].allyMonarchs;
                while(increaseIndex.next != null){
                    //모든 동맹들을 하나씩 돌아가며 병사 추가
                    increaseIndex = increaseIndex.next;
                    soldierMap[increaseIndex.x][increaseIndex.y] += mNum;
                    sum += soldierMap[increaseIndex.x][increaseIndex.y];
                }
                break;
            default:
                //ERROR: 잘못된 옵션 번호 입력
                sum = -1;
                break;
        }
        return sum;
    }

    /**
     * 서로를 동맹국으로 설정한다.
     * @param monarchA
     * @param monarchB
     */
    void setAlly(Monarchs monarchA, Monarchs monarchB){
        int unionGroupNo = monarchA.groupNo;     //합칠 그룹 번호
        int removeGroupNo = monarchB.groupNo;    //사라질 그룹 번호
        Monarchs unionIndex = groupList[unionGroupNo].allyMonarchs; //합칠
그룹
        while(unionIndex.next != null){
```

```
            unionIndex = unionIndex.next;   //합칠 그룹 노드의 맨 끝으로
이동
        }
        Monarchs removeIndex = groupList[removeGroupNo].allyMonarchs;
//옮겨질 노드들

        while(removeIndex.next != null){
            removeIndex = removeIndex.next; //monarch와 동맹인 모든
군주들의
            removeIndex.groupNo = unionGroupNo; //그룹번호를 monarchA의
그룹번호로 바꾼다.
        }

        //monarchB와 동맹인 모든 군주들을 monarchA에 편입시킨다.
        removeIndex = groupList[removeGroupNo].allyMonarchs;
//monarchB를 포함한 모든 동맹들
        unionIndex.next = removeIndex.next; //monarchA쪽에 넣는다

        //적의 정보들도 서로 공유한다.
        for(int i =0; i<groupList[removeGroupNo].enemyGroupNo.size(); i++){

groupList[unionGroupNo].enemyGroupNo.add(groupList[removeGroupNo].enemyGrou
pNo.get(i)); //합쳐진 그룹번호에 기존 적국 명단을 추가한다.

groupList[groupList[removeGroupNo].enemyGroupNo.get(i)].enemyGroupNo.add(un
ionGroupNo); //적국들은 합쳐진 새로운 그룹번호를 적으로 저장한다.
        }
    }

    /**
     * 두 군주를 적으로 표기한다.
     * @param monarchA
     * @param monarchB
     */
    void setEnemy(Monarchs monarchA, Monarchs monarchB){
        if(!isAlreadyEnemy(monarchA, monarchB)) {
            groupList[monarchA.groupNo].enemyGroupNo.add(monarchB.groupNo);
            groupList[monarchB.groupNo].enemyGroupNo.add(monarchA.groupNo);
        }
```

```
    }

    /**
     * 두 군주가 이미 동맹인지를 판단한다.
     * @param monarchA
     * @param monarchB
     * @return 동맹국 여부
     */
    boolean isAlreadyAlly(Monarchs monarchA, Monarchs monarchB){
        if(monarchA.groupNo == monarchB.groupNo){
            return true;
        }
        else {
            return false;
        }
    }

    /**
     * 두 군주가 이미 적인지를 판단한다.
     * @param monarchA
     * @param monarchB
     * @return 적국 여부
     */
    boolean isAlreadyEnemy(Monarchs monarchA, Monarchs monarchB){
        for(int i =0; i<groupList[monarchA.groupNo].enemyGroupNo.size();
i++){
            if(groupList[monarchA.groupNo].enemyGroupNo.get(i) ==
monarchB.groupNo){
                return true;
            }
        }
        return false;
    }

    /**
     * 군주의 이름을 가지고 해당 군주가 있는 위치를 찾는다.
     * @param mMonarch
     * @return 군주의 좌표(x,y)
     */
    int[] findLocationOfMonarch(char mMonarch[]){
        String str = charToString(mMonarch);
```

```
        for(int i = 0; i<N; i++){
            for (int j =0; j<N; j++){
                //이름이 일치한다면
                if(str.equals(monarchNameMap[i][j])){
                    return new int[]{i, j};
                }
            }
        }
        //ERROR
        return new int[]{-1, -1};
    }

    /**
     * 해당 군주를 상징하는 노드를 찾는다.
     * @param mMonarch
     * @return 군주의 노드
     */
    Monarchs findNodeOfMonarch(char mMonarch[]){
        int[] locMonarch = findLocationOfMonarch(mMonarch);
        return nodeMap[locMonarch[0]][locMonarch[1]];
    }

    /**
     * char[] 타입을 String 타입으로 변경
     * @param mMonarch
     * @return (String) mMonarch
     */
    String charToString(char mMonarch[]) {
        String str = "";
        for (int i = 0; i < mMonarch.length; i++) {
            //알파벳부분이 아니라면 종료
            if(mMonarch[i] < 'a' || mMonarch[i] > 'z'){
                break;
            }
            str += mMonarch[i];
        }
        return str;
    }
}
```