JooHyun – Lee (comkiwer)

TS자유무역협정

Hancom Education Co. Ltd.

Problem

Problem

TS자유무역협정

자유 무역 협정을 맺은 나라 A와 나라 B가 있다.

나라 A에 numA개의 기업이 있고, 나라 B에 numB개의 기업이 있다.

같은 나라에 있는 기업 간에는 동맹을 맺을 수 있다.

기업 1과 기업 2가 서로 동맹을 맺으면

기업 1 또는 기업 2와 동맹인 모든 기업들이 서로 동맹이 된다.

예로 기업 1과 기업 2가 동맹이고 기업 3과 기업 4가 동맹일 때

기업 1과 기업 3이 동맹을 맺으면

기업 2와 기업 3도 동맹이 되고 기업 2와 기업 4도 동맹이 된다.

Problem

서로 다른 나라의 두 기업 간에는 공동 프로젝트를 개시할 수 있다.
또한 공동 프로젝트를 종료할 수도 있다.

같은 나라에 있는 두 기업 간에 분쟁이 발생할 수 있다.
이 경우 신속하고 공정한 조정을 위해서 타국 기업이 개입한다.
자국 기업의 경우 복잡한 이해관계가 얽혀 섣불리 발언하기 힘들기 때문에
타국 기업에서 중재를 맡아야 한다.

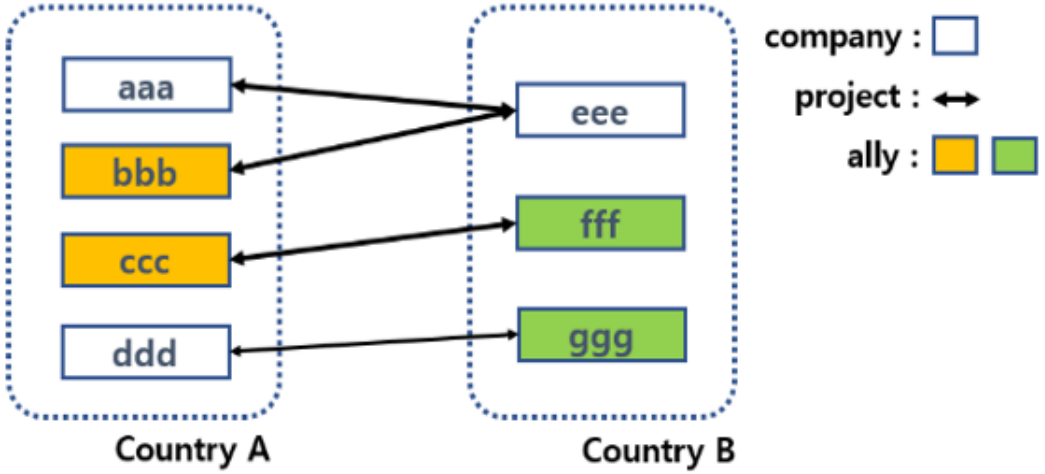
Problem

타국 기업이 중재를 맡기 위해서 다음 두 조건을 모두 만족해야 한다.
어느 한 조건이라도 만족하지 못하면 중재할 수 없다.
편의상 분쟁이 발생한 기업을 기업 1, 2이라고 하자.

① **기업 1 또는 기업 1의 동맹인 기업과 공동 프로젝트를 하고 있는 기업 또는 그 기업과 동맹인 기업**

② **기업 2 또는 기업 2의 동맹인 기업과 공동 프로젝트를 하고 있는 기업 또는 그 기업과 동맹인 기업**

예로, [Fig. 1]과 같은 경우를 생각해 보자.

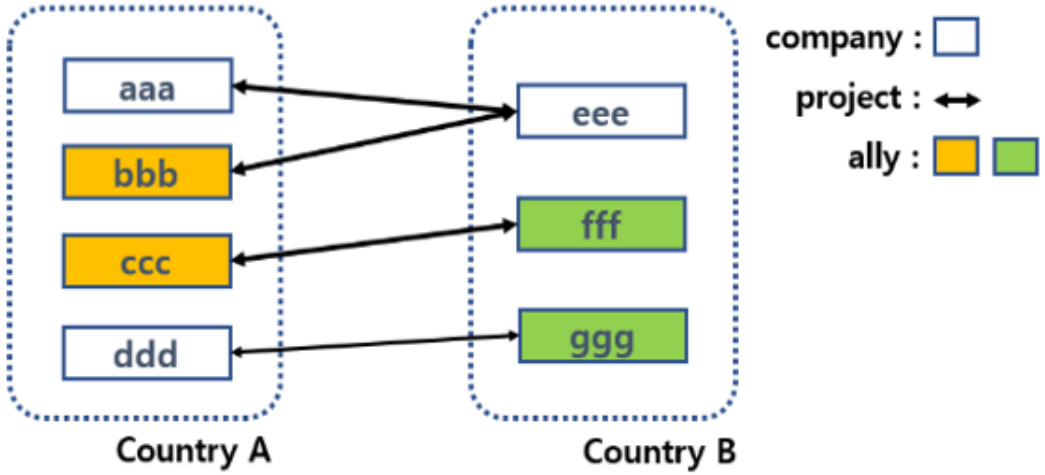


[Fig. 1]

Problem

TS자유무역협정

나라 A에는 기업 aaa, bbb, ccc, ddd가 있고
나라 B에는 기업 eee, fff, ggg가 있다.
기업 bbb와 기업 ccc는 서로 동맹이고 기업 fff와 기업 ggg는 서로 동맹이다.
검은 색 선은 공동 프로젝트를 하고 있다는 것을 뜻한다.

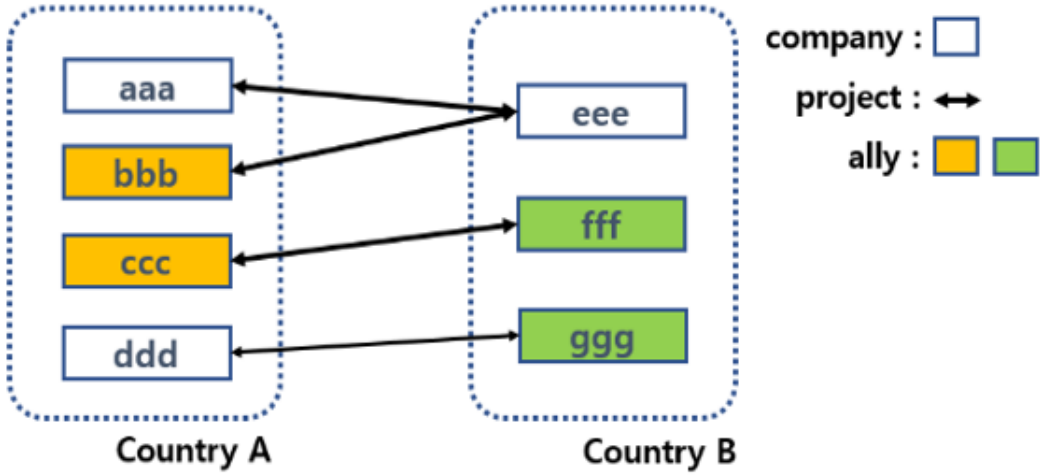


[Fig. 1]

Problem

만약 기업 aaa와 기업 bbb 간에 분쟁이 발생하면
기업 eee만 그 분쟁을 중재할 수 있다.

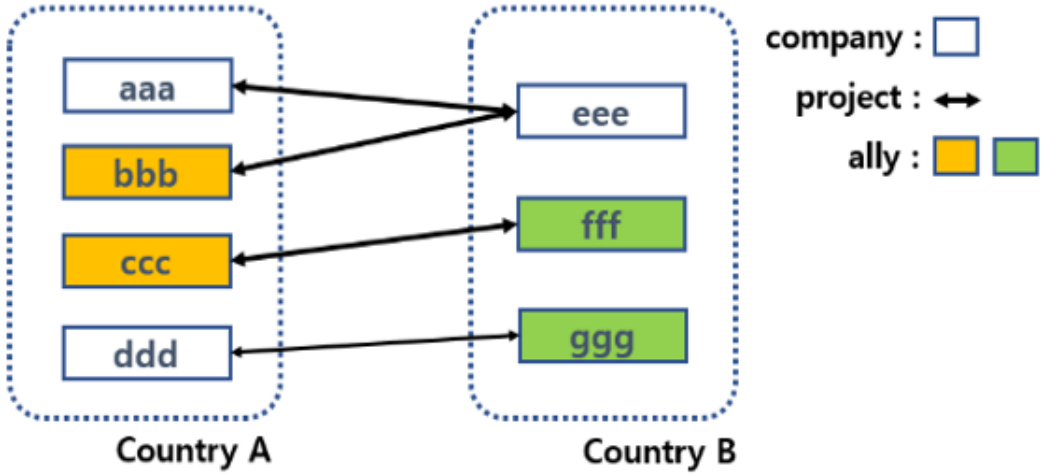
기업 eee는 기업 aaa와 공동 프로젝트를 하고 있고
기업 bbb와 공동 프로젝트를 하고 있기 때문이다.



[Fig. 1]

Problem

만약 기업 bbb와 기업 ddd 간에 분쟁이 발생하면
기업 fff와 기업 ggg만 그 분쟁을 중재할 수 있다.
기업 fff는 기업 bbb의 동맹인 기업 ccc와 공동 프로젝트를 하고 있고
기업 fff의 동맹인 기업 ggg는 기업 ddd와 공동 프로젝트를 하고 있기 때문이다.
기업 ggg도 마찬가지이다.



[Fig. 1]

Problem

TS자유무역협정

※ 아래 API 설명을 참조하여 각 함수를 구현하라.

```
void init(int numA, char listA[][], int numB, char listB[][])
```

테스트 케이스에 대한 초기화하는 함수로 각 테스트 케이스의 맨 처음에 1회 호출된다.

나라 A에 numA개의 기업이 있고 나라 A에 있는 기업의 이름은
listA[0] ~ listA[numA - 1]으로 주어진다.
나라 B에 numB개의 기업이 있고 나라 B에 있는 기업의 이름은
listB[0] ~ listB[numB - 1]으로 주어진다.

listA[i], listB[j]는 영어 소문자로 구성된 문자열이고 길이는 1 이상 10 이하이다.
문자열 끝에는 '\0'가 있고 길이에 포함되지 않는다.
(0 ≤ i ≤ numA - 1, 0 ≤ j ≤ numB - 1)

listA[], listB[]로 주어지는 회사 이름 중에 같은 이름은 없다.
즉, 모두 서로 다른 이름이다.

Problem

TS자유무역협정

```
void init(int numA, char listA[][], int numB, char listB[][])
```

(계속)

초기에는 어떤 기업도 서로 동맹인 기업이 없다.
또한 공동 프로젝트를 하고 있는 기업도 없다.

Parameters

numA : 나라 A에 있는 기업의 수 (2 ≤ numA ≤ 500)

listA[] : 나라 A에 있는 기업의 이름 목록
(1 ≤ |listA[]| ≤ 10, |a|는 문자열 a의 길이를 뜻한다.)

numB : 나라 B에 있는 기업의 수 (2 ≤ numB ≤ 500)

listB[] : 나라 B에 있는 기업의 이름 목록 (1 ≤ |listB[]| ≤ 10)

Problem

TS자유무역협정

```
void startProject(char mCompayA[], char mCompanyB[])
```

나라 A의 기업 mCompanyA와 나라 B의 기업 mCompanyB는 공동 프로젝트를 개시한다.

mCompanyA는 나라 A의 기업 이름임을 보장한다. mCompanyB는 나라 B의 기업 이름임을 보장한다.

함수 호출 시 기업 mCompanyA와 기업 mCompanyB는 공동 프로젝트를 진행하고 있지 않음을 보장한다.

Parameters

mCompanyA : 나라 A의 기업 이름 ($1 \leq |mCompanyA| \leq 10$)

mCompanyB : 나라 B의 기업 이름 ($1 \leq |mCompanyB| \leq 10$)

Problem

TS자유무역협정

```
void finishProject(char mCompanyA[], char mCompanyB[])
```

나라 A의 기업 mCompanyA와 나라 B의 기업 mCompanyB가 진행하고 있는 공동 프로젝트가 종료된다.

mCompanyA는 나라 A의 기업 이름임을 보장한다. mCompanyB는 나라 B의 기업 이름임을 보장한다.

함수 호출 시 기업 mCompanyA와 기업 mCompanyB는 공동 프로젝트를 진행하고 있음을 보장한다.

Parameters

mCompanyA : 나라 A의 기업 이름 ($1 \le |mCompanyA| \le 10$)

mCompanyB : 나라 B의 기업 이름 ($1 \le |mCompanyB| \le 10$)

Problem

TS자유무역협정

```
void ally(char mCompany1[], char mCompany2[])
```

기업 mCompany1과 기업 mCompany2가 서로 동맹을 맺는다.

mCompany1과 mCompany2는 같은 나라의 기업 이름임을 보장한다. mCompany1과 mCompany2는 서로 다른 이름이다.

기업 mCompany1과 기업 mCompany2가 이미 동맹일 수도 있다.

이미 호출된 mCompany1, mCompany2의 쌍에 대해서 다시 호출되는 경우는 없다.

Parameters

mCompany1 : 기업의 이름 ($1 \le |mCompany1| \le 10$)

mCompany2 : 기업의 이름 ($1 \le |mCompany2| \le 10$, $mCompany1 \neq mCompany2$)

Problem

TS자유무역협정

```
int conflict(char mCompany1[], char mCompany2[])
```

기업 mCompany1과 기업 mCompany2 간에 분쟁이 발생했을 때,

분쟁에 개입하여 중재할 수 있는 다른 나라의 기업의 수를 반환한다.

분쟁에 개입하여 중재를 맡을 수 있는 기업의 조건은 본문 설명을 참조하라.

mCompany1과 mCompany2는 같은 나라의 기업 이름임을 보장한다.

mCompany1과 mCompany2는 서로 다른 이름이다.

함수 호출 시 기업 mCompany1과 기업 mCompany2는 동맹이 아님을 보장한다.

Parameters

mCompany1 : 기업의 이름 ($1 \le |mCompany1| \le 10$)

mCompany2 : 기업의 이름 ($1 \le |mCompany2| \le 10$, $mCompany1 \neq mCompany2$)

Returns

분쟁에 개입하여 중재할 수 있는 다른 나라 기업의 수

Problem

TS자유무역협정

[제약사항]

1. 나라 A, B의 기업의 수 numA, numB는 2 이상 500 이하의 정수이다.
2. 각 나라의 기업 이름은 영어 소문자로 구성되어 있고 길이는 1 이상 10 이하이다.
3. 각 나라의 기업 이름의 문자열의 끝에는 '\0'인 값이 저장된다.
4. 각 테스트 케이스에서 conflict() 함수의 호출 횟수는 5,000 이하이다.
5. 각 테스트 케이스에서 모든 함수의 호출 횟수 총합은 10,000 이하이다.

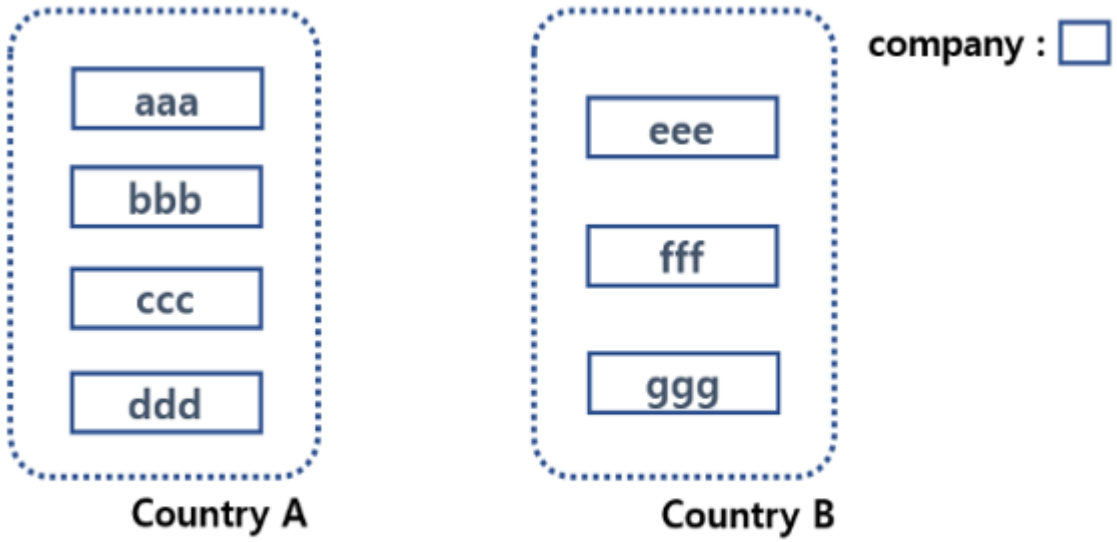
Problem analysis

Problem analysis : 예제

TS자유무역협정

[Order #1] 테스트 케이스의 초기 상태이다.

Order	Function	Description	Return	Figure
1	init(4, {"aaa", "bbb", "ccc", "ddd"}, 3, {"eee", "fff", "ggg"})	나라 A에 기업 aaa, bbb, ccc, ddd가 있고 나라 B에 기업 eee, fff, ggg가 있다.		[Fig. 2]



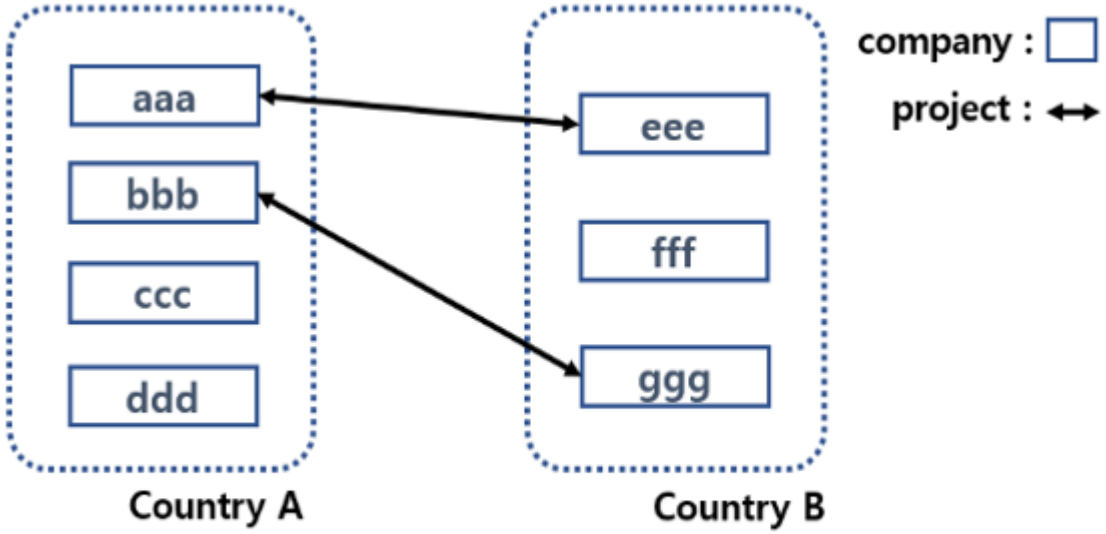
[Fig. 2]

Problem analysis : 예제

[Order #4] 기업 ccc는 어느 타국 기업과 공동 프로젝트를 진행하고 있지 않다.

따라서, 기업 bbb와 기업 ccc 간에 분쟁이 발생하면 중재할 수 있는 기업은 없다.

Order	Function	Description	Return	Figure
2	startProject("aaa", "eee")	기업 aaa와 기업 eee는 공동 프로젝트를 시작한다.		
3	startProject("bbb", "ggg")	기업 bbb와 기업 ggg가 공동 프로젝트를 시작한다.		
4	conflict("bbb", "ccc")	기업 bbb와 기업 ccc 간에 분쟁이 발생할 경우 중재할 수 있는 타국 기업은 없다.	0	[Fig. 3]



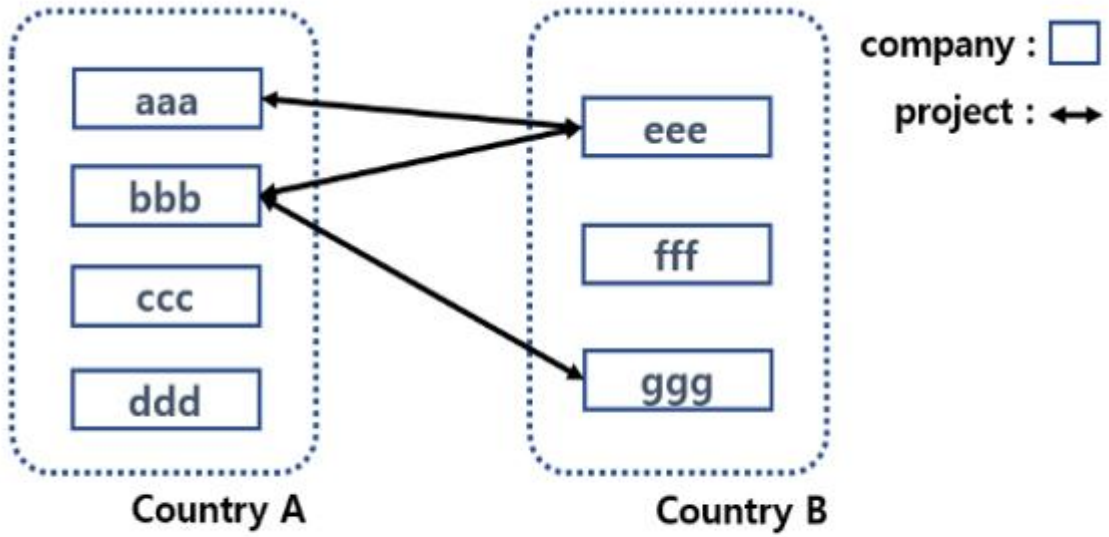
[Fig. 3]

Problem analysis : 예제

TS자유무역협정

[Order #6] 기업 eee는 기업 aaa, bbb와 각각 공동 프로젝트를 진행하고 있다.
따라서, 기업 eee는 기업 aaa와 기업 bbb 간에 분쟁이 발생하면 중재할 수 있다.

Order	Function	Description	Return	Figure
5	startProject("bbb", "eee")	기업 bbb와 기업 eee는 공동 프로젝트를 시작한다.		
6	conflict("aaa", "bbb")	기업 aaa와 기업 bbb 간에 분쟁이 발생할 경우 중재할 수 있는 타국 기업은 기업 eee이다.	1	[Fig. 4]



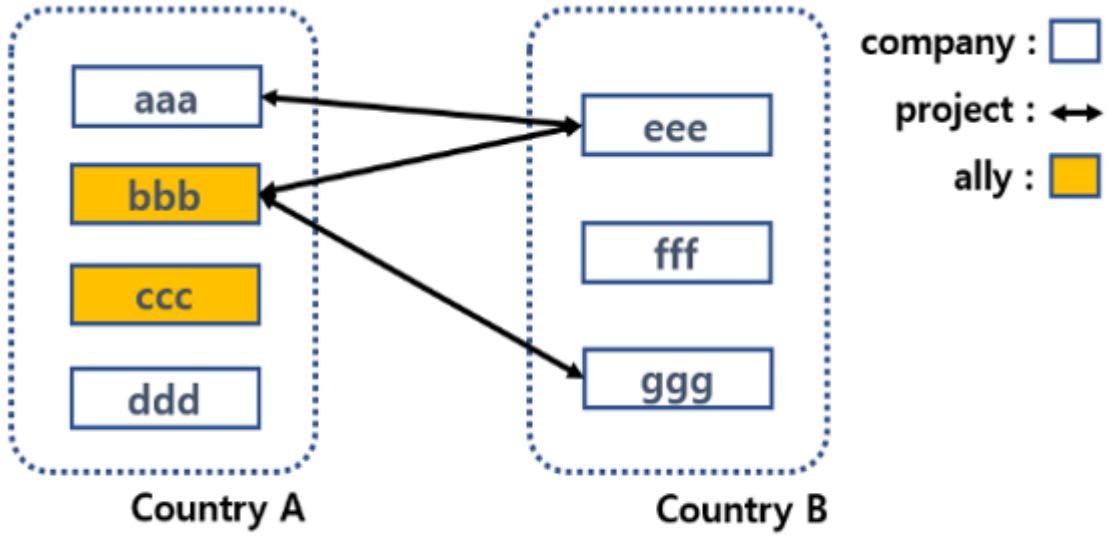
[Fig. 4]

Problem analysis : 예제

TS자유무역협정

[Order #8] 기업 bbb와 기업 ccc는 동맹이다.
그러므로, 기업 eee는 기업 aaa와 기업 ccc 간에 분쟁이 발생하면 중재할 수 있다.
[Order #9] 기업 bbb는 기업 eee, ggg와 각각 공동 프로젝트를 진행하고 있다.
따라서, 기업 bbb와 그와 동맹 기업인 ccc는 기업 eee와 기업 ggg간에 분쟁이 발생하면 중재할 수 있다.

Order	Function	Description	Return	Figure
7	ally("bbb","ccc")	기업 bbb와 기업 ccc가 동맹을 맺는다.		
8	conflict("aaa", "ccc")	기업 aaa와 기업 ccc 간에 분쟁이 발생할 경우 중재할 수 있는 타국 기업은 기업 eee이다.	1	[Fig. 5]
9	conflict("eee","ggg")	기업 eee와 기업 ggg 간에 분쟁이 발생할 경우 중재할 수 있는 타국 기업은 기업 bbb, ccc이다.	2	[Fig. 5]



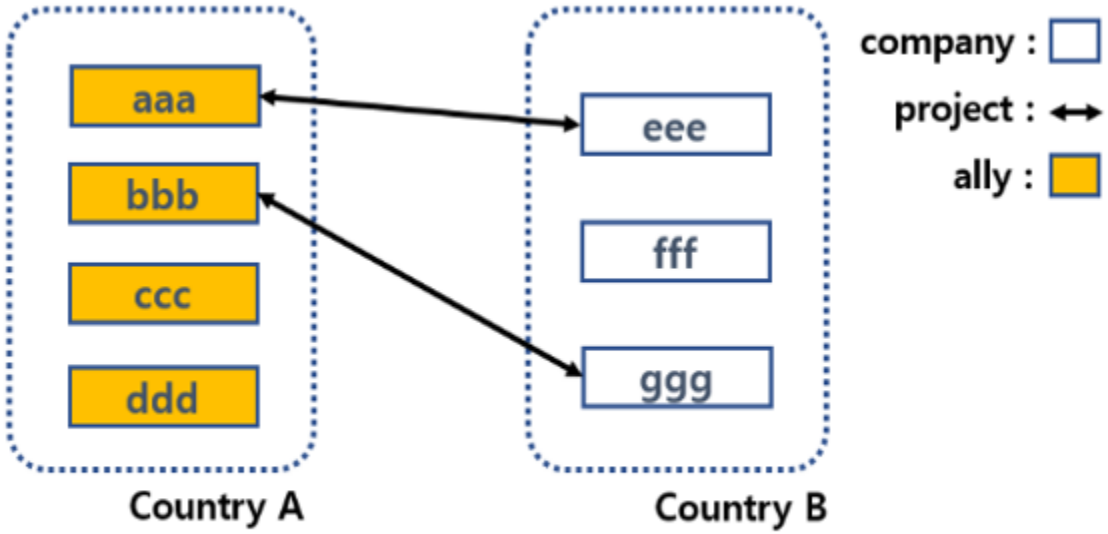
[Fig. 5]

Problem analysis : 예제

TS자유무역협정

[Order #13] 기업 aaa, bbb, ccc, ddd가 동맹이다.
기업 aaa는 기업 eee와 공동 프로젝트를 진행하고 있고 기업 bbb는 기업 ggg와 공동 프로젝트를 진행하고 있다.
따라서, 기업 aaa, bbb, ccc, ddd는 서로 동맹이므로 기업 eee와 기업 ggg 간에 분쟁이 발생하면 중재할 수 있다.

Order	Function	Description	Return	Figure
10	ally("aaa","ddd")	기업 aaa와 기업 ddd가 동맹을 맺는다.		
11	ally("bbb", "aaa")	기업 bbb와 기업 aaa가 동맹을 맺는다. 기업 aaa, bbb, ccc, ddd 모두 서로 동맹 기업이다.		
12	finishProject("bbb", "eee")	기업 bbb와 기업 eee가 진행하고 있는 프로젝트가 종료된다.		
13	conflict("eee", "ggg")	기업 eee와 기업 ggg가 분쟁이 발생할 경우 중재할 수 있는 타국 기업은 기업 aaa, bbb, ccc, dd이다.	4	[Fig. 6]



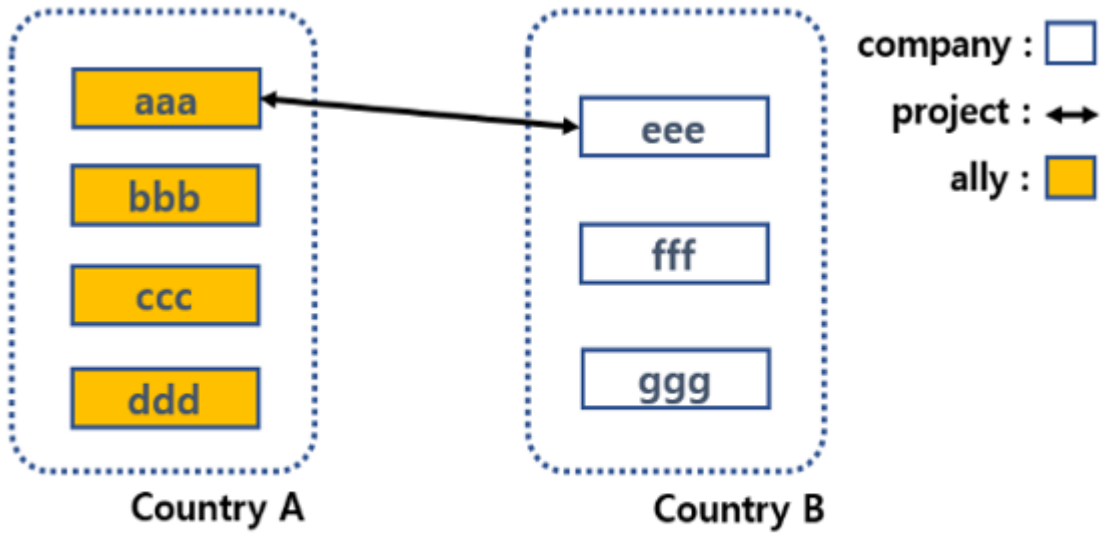
[Fig. 6]

Problem analysis : 예제

TS자유무역협정

[Order #15] 기업 bbb와 기업 ggg 간에 진행되었던 공동 프로젝트가 종료되었다.
기업 ggg와 공동 프로젝트를 진행하는 타국 기업이 없으므로
기업 eee와 기업 ggg 간에 분쟁이 발생할 경우 어느 타국 기업도 중재할 수 없다.

Order	Function	Description	Return	Figure
14	finishProject("bbb", "ggg")	기업 bbb와 기업 ggg가 진행하고 있는 프로젝트가 종료된다.		
15	conflict("eee", "ggg")	기업 eee와 기업 ggg가 분쟁이 발생할 경우 중재할 수 있는 타국 기업은 없다.	0	[Fig. 7]



[Fig. 7]

Problem analysis : 제약 조건 및 API

TS자유무역협정

- 두 나라가 있으며 각 나라의 기업 수는 최대 500이므로
  최대 1,000개의 기업이 주어질 수 있다.
  회사 이름이 문자열로 주어지므로 인덱스 기반 프로그래밍을 위하여
  renumbering hash등이 필요할 수 있다.

- 공동프로젝트는 다른 나라 기업과 이루어진다.
- 동맹은 같은 나라 기업과 이루어진다.
- 분쟁은 같은 나라 기업과 발생한다.

- 핵심 함수는 conflict 함수이다.
  이를 통해 프로그램의 정확성을 측정한다.
  공동프로젝트의 시작과 종료 및 동맹 설정이 conflict에서 사용된다.

Problem analysis

TS자유무역협정

- conflict(aid, bid) 호출시에 다음 작업이 이루어진다.
  - ✓ aid기업과 동맹인 모든 기업들과 공동 프로젝트를 수행하는 다른 나라의 기업들 xrr을 구한다.
  - ✓ Bid기업과 동맹인 모든 기업들과 공동 프로젝트를 수행하는 다른 나라의 기업들 yrr을 구한다.
  - ✓ xrr에도 포함되고 yrr에도 포함되는 기업의 수를 구하여 반환한다.

- aid와 동맹인 기업을 어떻게 관리할 수 있을까?
- aid와 동맹인 기업들과 공동 프로젝트를 수행하는 다른 나라 기업들을 어떻게 관리할 수 있을까?

Solution sketch

Solution sketch

TS자유무역협정

- 회사 이름이 문자열로 주어지므로 인덱스 기반 프로그래밍을 위하여 unordered_map<string, int> hmap; 을 사용할 수 있다.

- 동맹관계는 union find를 이용하여 관리될 수 있다.
  동맹관계인 기업을 트리로 관리하고 각 그룹의 개수를
  트리의 루트 기업과 함께 관리하여 문제해결에 이용할 수 있다.

- 공동프로젝트 회사들은 1000*1000배열로 다룰 수 있다.

```
int group[LM], cnt[LM];        // group[i]: i 회사의 그룹번호, cnt[i]: i그룹의 회사수
int partner[LM][LM];           // partner[i][j]: i회사와 공동프로젝트를 수행하는 j회사수 표시
```

Solution sketch

- 동맹관계로 인하여 두 그룹이 합쳐질 때
  동맹관계기업수 cnt[] 를 업데이트 한다.
  동맹의 루트 기업에 공동프로젝트를 갖는 기업들의 수를
  업데이트 한다.

```
const int LM = 1005;            // 전체 회사수는 1000
int an, bn;                     // an: A국가의 회사수, bn-an: B국가의 회사수
int group[LM], cnt[LM];         // group[i]: i 회사의 그룹번호, cnt[i]: i그룹의 회사수
int partner[LM][LM];            // partner[i][j]: i회사와 공동프로젝트를 수행하는 j회사수 표시
int visited[LM], vn;            // conflict함수에서 사용
unordered_map<string, int> hmap;  // <string:company_name, int:renumbering_id>

int Find(int n) {                   // 그룹의 루트 찾기 : 경로압축이 사용된다.
    if (group[n] == n) return n;
    return group[n] = Find(group[n]);
}
```

Solution sketch

각 API함수별 할 일을 정리해 보자.

```
void init(int numA, char listA[][], int numB, char listB[][])
```

전역 변수 및 해시 테이블 등을 초기화 한다.

```
an = numA, bn = numA + numB;                          // an: A나라 회사수, bn: 전체 회사수
hmap.clear();                                         // renumbering hash 초기화
int i, j = 0;
for (i = 0; i < numA; ++i) hmap[listA[i]] = ++j;      // A나라 renumbering
for (i = 0; i < numB; ++i) hmap[listB[i]] = ++j;      // B나라 renumbering
for (i = 1; i <= bn; ++i) group[i] = i, cnt[i] = 1; // 그룹번호, 그룹 수 초기화
memset(partner, 0, sizeof(partner));
```

Solution sketch

TS자유무역협정

void startProject(char mCompayA[], char mCompanyB[])

: 각 기업의 속한 동맹의 루트를 구한다.

: 루트 기업 사이에 공동 프로젝트를 시작한다.

```
// aid와 bid가 공동 프로젝트를 수행한다는 것은
// aroot그룹 전체와 broot그룹 전체가 하는 것과 같다.
int aid = hmap[mCompanyA], bid = hmap[mCompanyB];
int aroot = Find(aid), broot = Find(bid);
partner[aroot][broot]++, partner[broot][aroot]++;
```

Solution sketch

TS자유무역협정

void finishProject(char mCompanyA[], char mCompanyB[])

: 각 기업의 속한 동맹의 루트를 구한다.

: 루트 기업 사이에 공동 프로젝트를 종료한다.

```
// aid와 bid가 공동 프로젝트를 종료한다는 것은
// aroot그룹 전체와 broot그룹 전체가 종료하는 것과 같다.
int aid = hmap[mCompanyA], bid = hmap[mCompanyB];
int aroot = Find(aid), broot = Find(bid);
partner[aroot][broot]--;
partner[broot][aroot]--;
```

Solution sketch

TS자유무역협정

```
void ally(char mCompany1[], char mCompany2[])
```

: 각 기업이 속한 루트 기업을 구하고 두 그룹을 합친다.

: 동맹 기업의 수를 업데이트한다.

: 공동 프로젝트 기업을 업데이트 한다.

```
// aid와 bid가 동맹 관계를 맺는다는 것은 각각이 속한 그룹 vs 그룹으로 동맹관계를 맺는 것이다.
int cid1 = hmap[mCompany1], cid2 = hmap[mCompany2];
int root1 = Find(cid1), root2 = Find(cid2);
if (root1 == root2) return;     // 이미 동맹관계인 경우

group[root2] = root1;
cnt[root1] += cnt[root2], cnt[root2] = 0;

for (int i = 1; i <= bn; ++i) {
    partner[root1][i] += partner[root2][i];       // 상호 대칭 업데이트
    partner[i][root1] += partner[root2][i];       // 상호 대칭 업데이트
    partner[root2][i] = partner[i][root2] = 0;    // 이전 기록은 삭제
}
```

Solution sketch

TS자유무역협정

```
int conflict(char mCompany1[], char mCompany2[])
```

: 각 기업이 속한 그룹의 루트 기업 root1, root2를 구한다.
: root1과 공동프로젝트를 수행하면서 root2와 공동 프로젝트를 수행하는 기업의 수를 구하여 반환한다.

```
int cid1 = hmap[mCompany1], cid2 = hmap[mCompany2];
int root1 = Find(cid1), root2 = Find(cid2);
int ret = 0, tg = ++vn;
for (int i = 1; i <= bn; ++i) if (partner[root1][i])
    visited[Find(i)] = tg;     // root1의 파트너 표시

for (int i = 1; i <= bn; ++i) if (partner[root2][i]) {
    int k = Find(i);
    if (visited[k] == tg) {    // root1의 파트너이고 root2의 파트너인데 처음 확인된 경우라면
        visited[k] = tg - 1;   // 이미 확인해 본 것으로 표시
        ret+=cnt[k];           // 중재가능한 회사수 업데이트
    }
}
return ret;
```

Solution sketch

TS자유무역협정

[Summay]

- 식별자로 문자열이 주어진 경우 인덱스 기반 프로그래밍을 위하여 unordered<string, int> hmap; 을 사용할 수 있다.

- union find 알고리즘을 문제해결에 사용할 수 있다.

- 최근 자주 등장하는 알고리즘이므로 반드시 익혀 두자.

Code example

Code example

TS자유무역협정

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

const int LM = 1005;                // 전체 회사수는 1000
int an, bn;                         // an: A국가의 회사수, bn-an: B국가의 회사수
int group[LM], cnt[LM];             // group[i]: i 회사의 그룹번호, cnt[i]: i그룹의 회사수
int partner[LM][LM];                // partner[i][j]: i그룹과 공동프로젝트를 수행하는 j회사수 표시
int visited[LM], vn;
unordered_map<string, int> hmap;    // <string:company_name, int:renumbering_id>

int Find(int n) {
    if (group[n] == n) return n;
    return group[n] = Find(group[n]);
}

void init(int numA, char listA[500][11], int numB, char listB[500][11]) {
    an = numA, bn = numA + numB;                            // an: A나라 회사수, bn: 전체 회사수
    hmap.clear();                                           // renumbering hash 초기화
    int i, j = 0;
    for (i = 0; i < numA; ++i) hmap[listA[i]] = ++j;        // A나라 renumbering
    for (i = 0; i < numB; ++i) hmap[listB[i]] = ++j;        // B나라 renumbering
    for (i = 1; i <= bn; ++i) group[i] = i, cnt[i] = 1; // 그룹번호, 그룹 수 초기화
    memset(partner, 0, sizeof(partner));
}
```

Code example

TS자유무역협정

```
void startProject(char mCompanyA[11], char mCompanyB[11]) {
    int aid = hmap[mCompanyA], bid = hmap[mCompanyB]; // aid와 bid가 공동 프로젝트를 수행한다는 것은
    int aroot = Find(aid), broot = Find(bid);  // aroot그룹 전체와 broot그룹 전체가 하는 것과 같다.
    partner[aroot][broot]++, partner[broot][aroot]++;
}

void finishProject(char mCompanyA[11], char mCompanyB[11]) {
    int aid = hmap[mCompanyA], bid = hmap[mCompanyB]; // aid와 bid가 공동 프로젝트를 종료한다는 것은
    int aroot = Find(aid), broot = Find(bid); // aroot그룹 전체와 broot그룹 전체가 종료하는 것과 같다.
    partner[aroot][broot]--;
    partner[broot][aroot]--;
}
```

Code example

TS자유무역협정

```
void ally(char mCompany1[11], char mCompany2[11]) {
    int cid1 = hmap[mCompany1], cid2 = hmap[mCompany2]; // aid와 bid가 동맹 관계를 맺는다는 것은
    int root1 = Find(cid1), root2 = Find(cid2); // aroot그룹 전체와 broot그룹 전체가 동맹을 맺는 것과 같다.
    if (root1 == root2) return;    // 이미 동맹관계인 경우

    group[root2] = root1;
    cnt[root1] += cnt[root2], cnt[root2] = 0;

    for (int i = 1; i <= bn; ++i) {
        partner[root1][i] += partner[root2][i];      // 상호 대칭 업데이트
        partner[i][root1] += partner[root2][i];      // 상호 대칭 업데이트
        partner[root2][i] = partner[i][root2] = 0;   // 이전 기록은 삭제
    }
}
```

Code example

TS자유무역협정

```
int conflict(char mCompany1[11], char mCompany2[11]) {
    int cid1 = hmap[mCompany1], cid2 = hmap[mCompany2];
    int root1 = Find(cid1), root2 = Find(cid2);
    int ret = 0, tg = ++vn;
    for (int i = 1; i <= bn; ++i) if (partner[root1][i])
        visited[Find(i)] = tg;    // root1의 파트너 표시

    for (int i = 1; i <= bn; ++i) if (partner[root2][i]) {
        int k = Find(i);
        if (visited[k] == tg) {   // root1의 파트너이고 root2의 파트너인데 처음 확인된 경우라면
            visited[k] = tg - 1;  // 이미 확인해 본 것으로 표시
            ret+=cnt[k];          // 중재가능한 회사수 업데이트
        }
    }
    return ret;
}
```

Thank you.