<내풀이>

```
for ( int i = 0 ; i < fromId.size()  ; i++ )   //from 만큼 확인 (1<50)이라 시간ok

    if (bottles [fromId[i]] + bottles [toId[i]] {  //옮길 병의 용량보다
                    < capacities [toId[i]])          합친 물의 양이 작다면
                                                     물긴다
        bottles [toId[i]] += bottles [fromId[i]]
        continue                                  → 물끼리 더함
}

bottles [toId[i]] = capacities [toId[i]]   //용량을 초과한다면
bottles [fromId[i]] -=                     // 병은 꽉 차고
        capacities [toId[i]] - bottles [toId[i]]  // 남은 양만큼 빼준다
```

틀린부분

```
for ( int i = 0 ; i < fromId.size()  ; i++ )

    if (bottles [fromId[i]] + bottles [toId[i]] {  물을 꽉 채워도 된다!!
                    <= capacities [toId[i]])
        bottles [toId[i]] += bottles [fromId[i]]   to는 계산했는데
        continue                                   from은 계산 안해줌
}

bottles [toId[i]] = capacities [toId[i]]
bottles [fromId[i]] -=
        capacities [toId[i]] - bottles [toId[i]]
```

toId[i] 이런식으로 인덱스 적는게 어렵다.

<해답1>

```
for (int i=0   i < fromId . size()  i++ )

    int f = fromId [i]    //인덱스 분리 - 실수 방지하기 좋겠다
    int t = toId [i]
    int space = capacities [t] - bottles [t]  //남은 용량 분리

    if (space >= bottles [f])   용량이 남으면

        int vol = bottles [f]
        bottles [t] += vol      옮길양을 부어주고
        bottles [f] = 0         다 옮겨서 0으로 초기화
        continue

    int vol = space          남은 양만큼 옮긴다
    bottles [t] += vol
    bottles [f] -= vol
```

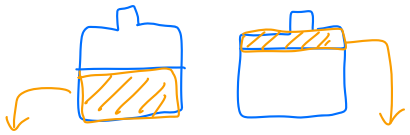
<해답2>

```
for(int i = 0  i < fromId.size()  i++)

    int f = fromId[i]
    int t = toId[i]

    int vol = min(bottles[f], capacities[t] - bottles[t])

    bottles[f] -= vol
    bottles[t] += vol
```



옮길양 자체를 비교

해서 옮긴다!!!

<해답3>

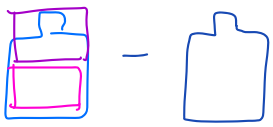
```
for ( int i=0  i < fromId.size()  i++ )

    int sum = bottles[fromId[i]] + bottles[toId[i]]
    bottles[toId[i]] = min ( sum, capacities[toId[i]] )
    bottles[fromId[i]] = sum - bottles[toId[i]]
```

이미 다 옮겨놓음

용량만큼 줄인다

종합에서 옮긴만큼 뺀다 ;;



<내풀이>

```
map<string, int>  m

for ( int  i = 0    i < first.size()  i++ )
    m[first[i]]++
    m[second[i]]++

int ans = -1
for ( auto& [key, value] : m )
    ans = max ( value , ans )

return ans
```

각 문자열이 몇 번 들어왔는지 센다

최댓값 찾음

출력

<해답1>

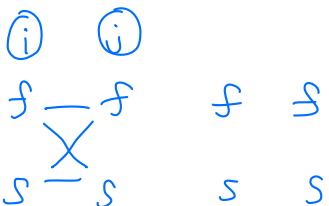
```
int ans = 0

for( int i = 0    i < first.size()  i++ )

    int f = 0
    int s = 0

    for ( j = 0    j < first.size    j++ )

        if( first[i] == first[j] )  f++
        if (first[i] == second[j])  f++
        if ( second[i] == first[j] )  s++
```



```
if (second[i] == second[j]) s+f
```

모든 경우 전부 비교..!

```
ans = max(f, ans)
ans = max(s, ans)
```

<해답2>

```
map<string, int> dic

for (int i = 0  i < first.size()  i++)
        dic[first[i]] = 0
        dic[second[i]] = 0

for (int i = 0  i < first.size()  i++)
        dic[first[i]]++
        dic[second[i]]++

int ans = 0
map<siring, int>:: iterator it

for( it = dic.begin()  it != dic.end()  it++)
      ans = max(ans, it -> second)

return ans
```

---

2 5 10 : 100

1 5 10 → 15 11 → 55

↓

16 10 : 60

가장 작은걸 증가시키는게 유리

2^{62} : long long 사용

```
Sort ( numbers.begin(), numbers.end )

numbers[0] ++

i64 ans = 1
for ( int i = 0  i < numbers.size  i++)
      ans *= numbers[i]

return ans
```

```
for ( int i = 0  i< numbers.size() i++ )
```

i : 1증가시킬 인덱스

```
    long long temp = 1
    for ( int j = 0   j < numbers.size()  j++)
```

```
            if ( i == j )
                temp *= numbers[j] + 1
            else
                temp *= numbers[j]

        ans = max(ans, temp)

return ans
```

) i번째라면 증가시키기