2022-1 HI-ARC 초급스터디

6주차. 그리디

이지은 (leeju1013)

목차

1. 그리디

부록. 2차원 배열/벡터

동전 0

성공

실버 III

시간 제한

1 초

예제 입력 1 복사

```
10 4200
1
5
10
50
100
500
1000
5000
10000
50000
```

예제 출력 1 복사

```
6
```

문제

준규가 가지고 있는 동전은 총 N종류이고, 각각의 동전을 매우 많이 가지고 있다.

동전을 적절히 사용해서 그 가치의 합을 K로 만들려고 한다. 이때 필요한 동전 개수의 최솟값을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 N과 K가 주어진다. ($1 \le N \le 10$, $1 \le K \le 100,000,000$)

둘째 줄부터 N개의 줄에 동전의 가치 A_i가 오름차순으로 주어진다. ($1 \le A_i \le 1,000,000$, $A_1 = 1$, $i \ge 2$인 경우에 A_i는 A_{i-1}의 배수)

출력

첫째 줄에 K원을 만드는데 필요한 동전 개수의 최솟값을 출력한다.

준규가 가지고 있는 동전은 총 N종류이고, 각각의 동전을 매우 많이 가지고 있다.

동전을 적절히 사용해서 그 가치의 합을 K로 만들려고 한다. 이때 필요한 동전 개수의 최솟값을 구하는 프로그램을 작성하시오.

첫째 줄에 N과 K가 주어진다. ($1 \le N \le 10$, $1 \le K \le 100,000,000$)

둘째 줄부터 N개의 줄에 동전의 가치 A_i가 오름차순으로 주어진다. ($1 \le A_i \le 1,000,000$, $A_1 = 1$, $i \ge 2$인 경우에 A_i는 A_{i-1}의 배수)

첫째 줄에 K원을 만드는데 필요한 동전 개수의 최솟값을 출력한다.

예제 입력 1 복사

```
10 4200
1
5
10
50
100
500
1000
5000
10000
50000
```

예제 출력 1 복사

```
6
```

1. DP 테이블 정의

   dp[i] = 가치의 합이 i일때 필요한 동전 개수 최솟값

2. 점화식 찾기

   dp[i] = min(dp[i-A1], dp[i-A2], ... , dp[i-An]) + 1

-> 시간 복잡도

   O(NK)

1. 그리디 (Greedy)

- What?

- 선택의 순간마다 당장 눈앞에 보이는 최적의 상황만을 쫓아 최종적인 해답에 도달하는 방법

- How?

- 전체 문제를 작은 부분 문제로 나누기
-> 부분의 최적해를 찾기 -> 부분 최적해가 정당한지 증명
-> 구현

1. 전체 문제

10, 50, 100, 500원 동전을 최소한으로 사용해서 K원 만들기

2. 부분의 최적해 찾기

가치가 높은 동전부터, 최대한 많이 사용하기

3. 정당성 증명하기 (귀류법)

반대 가정 1 - 가치 높은 동전을 제외한 임의의 동전으로, 최대한 많이 사용하기

반례 - k=1000인 경우, 100*10 보다 500*2 가 최적임

반대 가정 2 - 가치가 높은 동전부터, 적당히 남기면서 사용하기

반례 - k=1000인 경우, 500*1 + 100*3 + 50*3 + 10*5 보다 500*2 가 최적임

-> 맨 처음 명제가 참임

준규가 가지고 있는 동전은 총 N종류이고, 각각의 동전을 매우 많이 가지고 있다.

동전을 적절히 사용해서 그 가치의 합을 K로 만들려고 한다. 이때 필요한 동전 개수의 최솟값을 구하는 프로그램을 작성하시오.

첫째 줄에 N과 K가 주어진다. (1 ≤ N ≤ 10, 1 ≤ K ≤ 100,000,000)

둘째 줄부터 N개의 줄에 동전의 가치 A_i가 오름차순으로 주어진다. (1 ≤ A_i ≤ 1,000,000, A_1 = 1, i ≥ 2인 경우에 A_i는 A_{i-1}의 배수)

첫째 줄에 K원을 만드는데 필요한 동전 개수의 최솟값을 출력한다.

```cpp
#include <iostream>
using namespace std;

int A[11], n,k,ans;
int main(){

    cin >> n >> k;
    for (int i=0; i<n; i++)
        cin>> A[i]; //동전의 가치 Ai

    for (int i=n-1; i>=0; i--){ //가치가 높은 동전부터
        ans += k/A[i]; //최대한 많이 사용하기
        k %= A[i]; //만들어야하는 남은 돈 액수
    }
    cout << ans; //필요한 동전 개수의 최솟값
    return 0;
}
```

예제 입력 1 복사

```
10 4200
1
5
10
50
100
500
1000
5000
10000
50000
```

예제 출력 1 복사

```
6
```

회의실 배정

성공

실버 II

시간 제한	메모리 제한	제출	정답	맞힌 사람
2 초	128 MB	117566	36099	25757

문제

한 개의 회의실이 있는데 이를 사용하고자 하는 N개의 회의에 대하여 회의실 사용표를 만들려고 한다. 각 회의 I에 대해 시작시간과 끝나는 시간이 주어져 있고, 각 회의가 겹치지 않게 하면서 회의실을 사용할 수 있는 회의의 최대 개수를 찾아보자. 단, 회의는 한번 시작하면 중간에 중단될 수 없으며 한 회의가 끝나는 것과 동시에 다음 회의가 시작될 수 있다. 회의의 시작시간과 끝나는 시간이 같을 수도 있다. 이 경우에는 시작하자마자 끝나는 것으로 생각하면 된다.

입력

첫째 줄에 회의의 수 N(1 ≤ N ≤ 100,000)이 주어진다. 둘째 줄부터 N+1 줄까지 각 회의의 정보가 주어지는데 이것은 공백을 사이에 두고 회의의 시작시간과 끝나는 시간이 주어진다. 시작 시간과 끝나는 시간은 2^{31}-1보다 작거나 같은 자연수 또는 0이다.

출력

첫째 줄에 최대 사용할 수 있는 회의의 최대 개수를 출력한다.

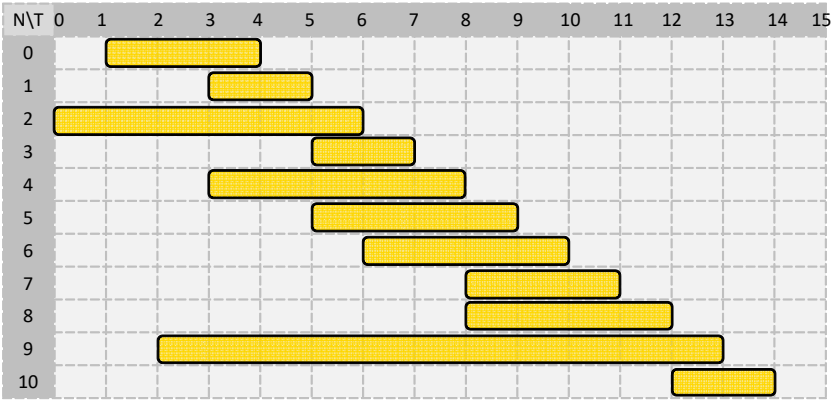
예제 입력 1 복사

```
11
1 4
3 5
0 6
5 7
3 8
5 9
6 10
8 11
8 12
2 13
12 14
```

예제 출력 1 복사

```
4
```

입력 : 회의의 수, 각 회의의 **시작시간**과 **끝나는 시간**
출력 : 각 회의가 겹치지 않게 하면서 회의실을 사용 할 수 있는 **회의의 최대 개수**



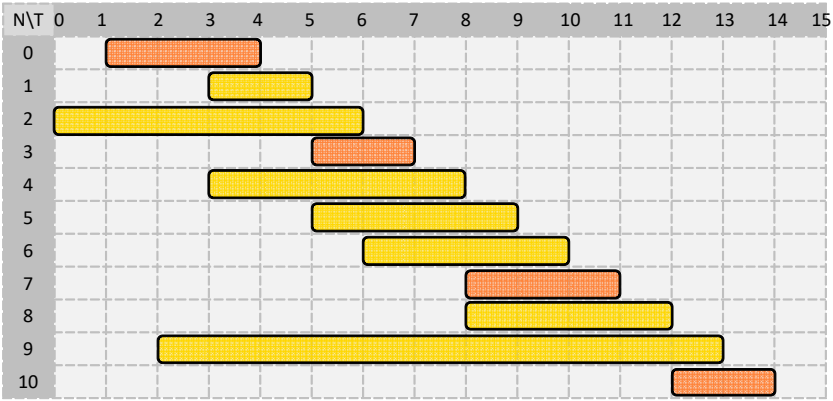
예제 입력 1 복사

```
11
1 4
3 5
0 6
5 7
3 8
5 9
6 10
8 11
8 12
2 13
12 14
```

예제 출력 1 복사

```
4
```

입력 : 회의의 수, 각 회의의 **시작시간**과 **끝나는 시간**
출력 : 각 회의가 겹치지 않게 하면서 회의실을 사용 할 수 있는 **회의의 최대 개수**



예제 입력 1 복사

```
11
1 4
3 5
0 6
5 7
3 8
5 9
6 10
8 11
8 12
2 13
12 14
```

예제 출력 1 복사

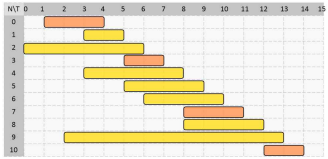
```
4
```

- 완전탐색?

  각 회의를 배정 한다/안 한다 : $O(2^N)$

  각 경우마다 겹치는 회의 없는지/회의 개수 총 몇 개인지 체크 : $O(N)$

-> 시간 복잡도 $O(N * 2^N)$



- DP?

1. DP 테이블 정의

   dp[i] = 마지막 회의가 i번째 회의일 때, 회의의 최대 개수

2. 점화식 찾기

   dp[i] = max(dp[j]) + 1

   (j번째 회의가 끝나는 시간 <= i번째 회의의 시작시간)

-> 시간 복잡도 $O(N^2)$

- 그리디?!

  How?

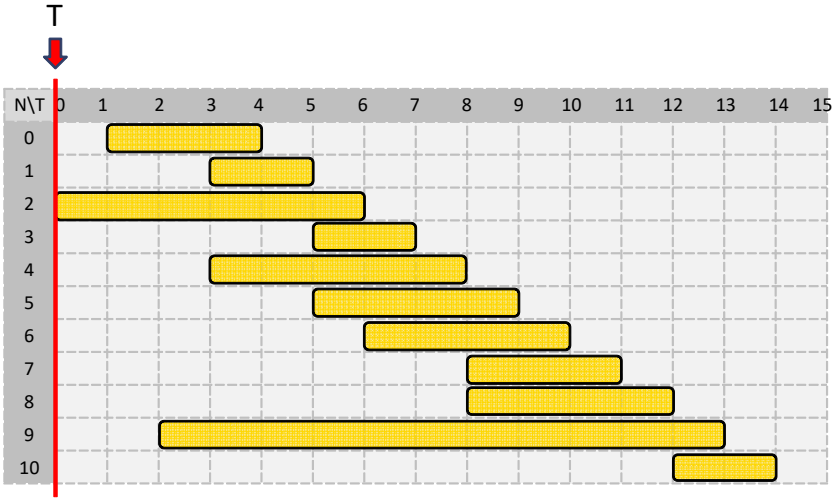
  1. 전체 문제를 작은 부분 문제로 나누기

  2. 부분의 최적해를 찾기

  3. 부분 최적해가 정당한지 증명하기

  4. 구현하기

1. 전체 문제를 작은 부분 문제로 나누기



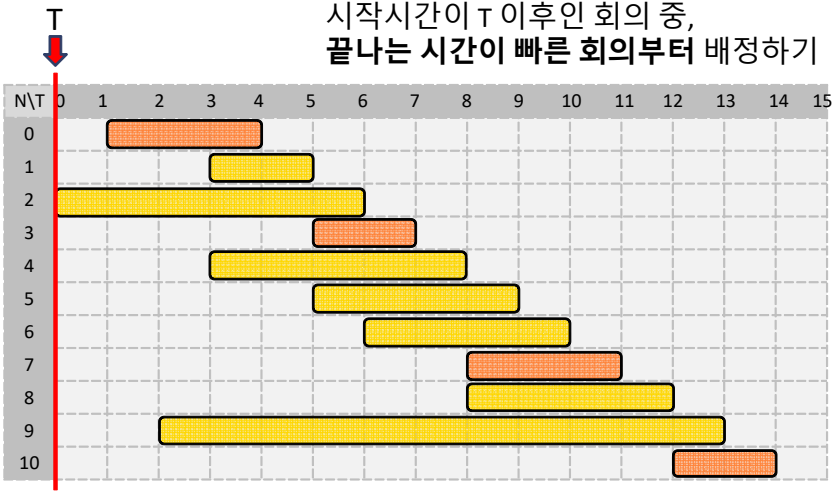
예제 입력 1 복사

```
11
1 4
3 5
0 6
5 7
3 8
5 9
6 10
8 11
8 12
2 13
12 14
```

예제 출력 1 복사

```
4
```

2. 부분의 최적해를 찾기



예제 입력 1 복사

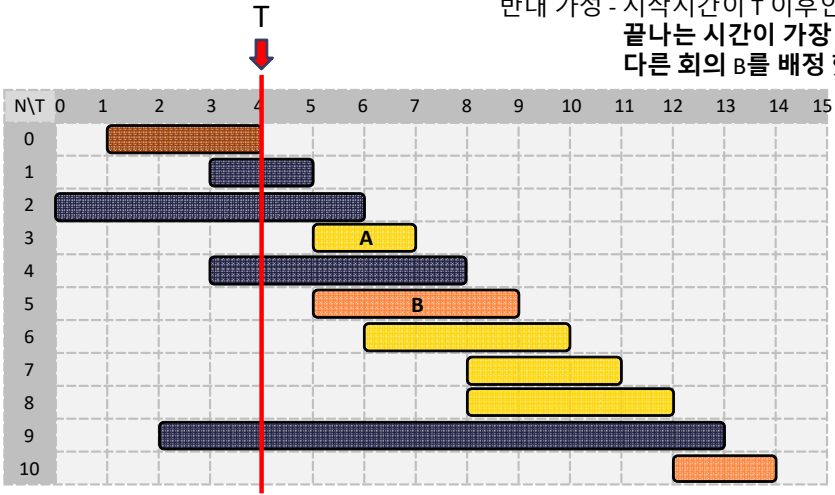
```
11
1 4
3 5
0 6
5 7
3 8
5 9
6 10
8 11
8 12
2 13
12 14
```

예제 출력 1 복사

```
4
```

3. 부분 최적해가 정당한지 증명하기

반대 가정 - 시작시간이 T 이후인 회의 중,
끝나는 시간이 가장 빠른 회의 A 말고
다른 회의 B를 배정 했을 때가 최적



4. 구현하기

시작시간이 T 이후인 회의 중,
끝나는 시간이 빠른 회의부터 배정하기

예제 입력 1 복사

```
11
1 4
3 5
0 6
5 7
3 8
5 9
6 10
8 11
8 12
2 13
12 14
```

예제 출력 1 복사

```
4
```

4. 구현하기

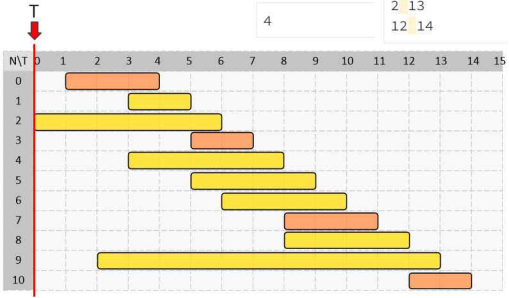
시작시간이 T 이후인 회의 중,
끝나는 시간이 빠른 회의부터 배정하기

예제 입력 1 복사

```
11
1 4
3 5
0 6
5 7
3 8
5 9
6 10
8 11
8 12
2 13
12 14
```

예제 출력 1 복사

```
4
```

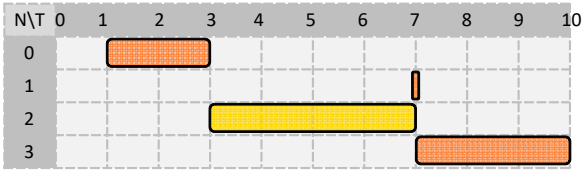



```cpp
#include <iostream>
#include <algorithm>
using namespace std;

struct meeting{
    int start,end;
}arr[100001];

bool cmp(const meeting& x, const meeting& y){
    return x.end < y.end;
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);

    int n; cin >> n;
    for(int i=0; i<n; i++){
        cin >> arr[i].start >> arr[i].end ;
    }
    sort(arr, arr+n,cmp);

    int t=0, ans=0;
    for (int i=0; i<n; i++){
        if (t <= arr[i].start){
            ans++;
            t = arr[i].end;
        }
    }
    cout << ans;
    return 0;
}
```

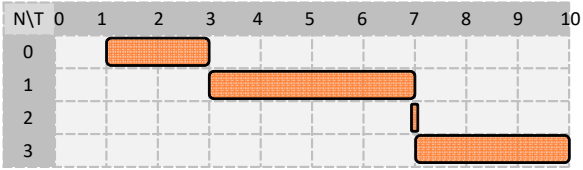
1931 틀렸습니다

문제

한 개의 회의실이 있는데 이를 사용하고자 하는 N개의 회의에 대하여 회의실 사용표를 만들려고 한다. 각 회의 I에 대해 시작시간과 끝나는 시간이 주어져 있고, 각 회의가 겹치지 않게 하면서 회의실을 사용할 수 있는 회의의 최대 개수를 찾아보자. 단, 회의는 한번 시작하면 중간에 중단될 수 없으며 한 회의가 끝나는 것과 동시에 다음 회의가 시작될 수 있다. 회의의 시작시간과 끝나는 시간이 같을 수도 있다. 이 경우에는 시작하자마자 끝나는 것으로 생각하면 된다.



끝나는시간이 빠른 순으로 정렬



끝나는시간이 빠른 순으로,
끝나는 시간이 같다면
시작하는 시간이 빠른 순으로 정렬

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

struct meeting{
    int start,end;
}arr[100001];

bool cmp(const meeting& x, const meeting& y){
    return x.end < y.end;
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);

    int n; cin >> n;
    for(int i=0; i<n; i++){
        cin >> arr[i].start >> arr[i].end ;
    }
    sort(arr, arr+n,cmp);

    int t=0, ans=0;
    for (int i=0; i<n; i++){
        if (t <= arr[i].start){
            ans++;
            t = arr[i].end;
        }
    }
    cout << ans;
    return 0;
}
```

끝나는시간이 빠른 순으로 정렬

1931 틀렸습니다

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

struct meeting{
    int start,end;
}arr[100001];

bool cmp(const meeting& x, const meeting& y){
    if (x.end != y.end)
        return x.end < y.end;
    else
        return x.start < y.start;
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);

    int n; cin >> n;
    for(int i=0; i<n; i++){
        cin >> arr[i].start >> arr[i].end ;
    }
    sort(arr, arr+n,cmp);

    int t=0, ans=0;
    for (int i=0; i<n; i++){
        if (t <= arr[i].start){
            ans++;
            t = arr[i].end;
        }
    }
    cout << ans;
    return 0;
}
```

끝나는시간이 빠른 순으로,
끝나는 시간이 같다면
시작하는 시간이 빠른 순으로 정렬

1931 맞았습니다!!

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);

    int n; cin >> n;
    pair<int,int> arr[100001];

    for(int i=0; i<n; i++){
        cin >> arr[i].second >> arr[i].first ; //{끝나는 시간, 시작시간}
    }
    sort(arr, arr+n);

    int t=0, ans=0;
    for (int i=0; i<n; i++){
        if (t <= arr[i].second){
            ans++;
            t = arr[i].first;
        }
    }
    cout << ans;
    return 0;
}
```

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL); co

    int n; cin >> n;
    vector<pair<int,int>> v;

    for(int i=0; i<n; i++){
        int start, end;
        cin>> start >> end;
        v.push_back({end, start});
    }
    sort(v.begin(), v.end());

    int t=0, ans=0;
    for (int i=0; i<n; i++){
        if (t <= v[i].second){
            ans++;
            t = v[i].first;
        }
    }
    cout << ans;
    return 0;
}
```

1. 그리디

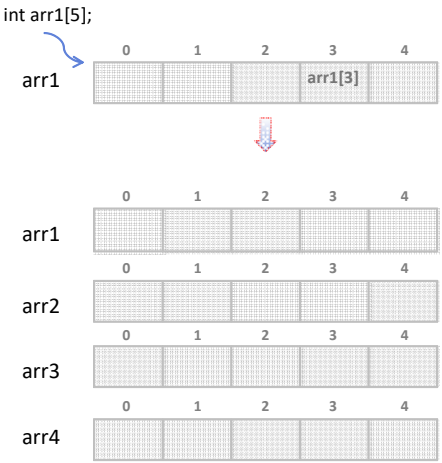
- When?
  - 부분 최적해를 반복적으로 취해서 답을 구할 수 있는 문제
    -> 부분 최적해들을 적용한 것이 전체 문제에서도 최적해인 경우

- 단점?
  - 부분 최적해를 떠올리는 것은 직관에 의존함,
    부분 최적해의 정당성을 증명해야함

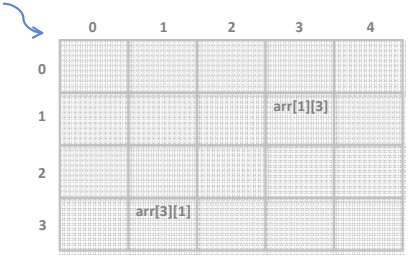
- 장점?
  - 탐색범위가 줄어서 빠른 시간에 풀 수 있음(시간 복잡도⬇)

부록. 2차원 배열

int arr1[5];



int arr[4][5];



부록. 2차원 배열

- 배열 선언

  int arr1[5], arr2[5], arr3[5], arr4[5]; => int arr[4][5];

- 배열 접근

  arr[0][0] ~ arr[3][4]

- 배열 초기화

  - 선언과 동시에 초기화
  - 전역 변수 배열 선언
  - 이중 for문

```cpp
#include <iostream>
using namespace std;

int main() {
    int arr[2][3] = {0,1,2,3,4,5};
    for(int i=0; i<2; i++){
        for(int j=0; j<3; j++){
            cout<<arr[i][j]<<" ";
        }
        cout<<'₩n';
    }
    return 0;
}
```

stdout

```
0 1 2
3 4 5
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int arr[2][3] = { {0,1,2}, {3,4,5} };
    for(int i=0; i<2; i++){
        for(int j=0; j<3; j++){
            cout<<arr[i][j]<<" ";
        }
        cout<<'₩n';
    }
    return 0;
}
```

stdout

```
0 1 2
3 4 5
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int arr[2][3] = {0,1,2,3};
    for(int i=0; i<2; i++){
        for(int j=0; j<3; j++){
            cout<<arr[i][j]<<" ";
        }
        cout<<'₩n';
    }
    return 0;
}
```

stdout

```
0 1 2
3 0 0
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int arr[2][3];
    for(int i=0; i<2; i++){
        for(int j=0; j<3; j++){
            arr[i][j] = i+j;
        }
    }

    for(int i=0; i<2; i++){
        for(int j=0; j<3; j++){
            cout<<arr[i][j]<<" ";
        }
        cout<<'₩n';
    }
    return 0;
}
```

stdout

```
0 1 2
1 2 3
```

부록. 2차원 벡터

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    vector<int> one; // 1차원 벡터

    one.push_back(1);
    one.push_back(2);
    one.push_back(3);

    for(auto &k :one)
        cout<< k << " ";

    return 0;
}
```

one

[0]	[1]	[2]
1	2	3

stdout

```
1 2 3
```

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    vector<int> two[5]; // 2차원 벡터

    two[0].push_back(1);
    two[3].push_back(2);
    two[3].push_back(3);
    two[4].push_back(4);
    two[4].push_back(5);
    two[4].push_back(6);

    for(int i=0; i<5; i++){
        cout<< i <<"행 : ";
        for(auto &k : two[i]) cout<< k << " ";
        cout<<'\n';
    }
    return 0;
}
```

two

	[0]	[1]	[2]
[0]	1		
[1]			
[2]			
[3]	2	3	
[4]	4	5	6

stdout

```
0행 : 1
1행 :
2행 :
3행 : 2 3
4행 : 4 5 6
```

감사합니다

- 필수 문제
  - 11047번 동전 0
  - 1931번 회의실 배정
- 연습 문제
  - 2585번 거스름돈
  - 11256번 사탕
  - 13305번 주유소
  - 2167번 2차원 배열의 합
  - 1895번 필터
- 5월 17일(화요일) 저녁 8시