2022-2 HI-ARC 중급스터디

1주차. 그리디

이지은 (leeju1013)

목차

1. 시간 복잡도

3 2747번 | 제출 | 맞힌 사람 | 숏코딩 | 재채점

피보나치 수 성공

3 브론즈 III

시간 제한	메모리 제한
1 초 (추가 시간 없음)	128 MB

제출 번호	아이디	문제	문제 제목	결과	메모리	시간	언어	코드 길이
40355570	leeju1013	2747	피보나치 수	시간 초과			C++14	362 B

1. 시간 복잡도 (Time Complexity)

- What?
  - 특정 알고리즘이 문제를 해결하는데 필요한 시간(연산의 횟수)

- How?
  - 빅-오(Big-O) 표기법
    : 알고리즘 최악의 실행시간 표기

1. 빅-오 표기법

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

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

    int n, sum=0; cin>>n;
    for(int i=1; i<=n; i++){
        sum += i;
    }
    cout<< sum;
    return 0;
}
```

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

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

    int n, sum=0; cin>>n;
    sum = n*(n+1)/2;
    cout<< sum;
    return 0;
}
```

O(N)

O(1)

O(1)

```
int n, ans=0, cin>>n;
ans = 3*n;
cout<< ans;
```

O(logN)

```
bool binary_search(int *arr, int len, int key){
    int l = 0, r = len-1, mid;
    while(l <= r) {
        mid = (l + r) / 2;
        if (arr[mid] == key) return true;
        else if (arr[mid] > key) r = mid - 1;
        else l = mid + 1;
    }
    return false;
}
```

O(N)

```
int n, ans=0, cin>>n;
for(int i=0; i<n; i++){
    ans += 3*n;
}
cout<< ans;
```

O(NlogN)

```
sort(arr, arr+n);
```

$O(N^2)$

```
int n, ans=0, cin>>n;
for(int i=1; i<=n; i++){
    for(int j=1; j<=n; j++){
        ans += i*j;
    }
}
cout<< ans;
```

$O(2^N)$

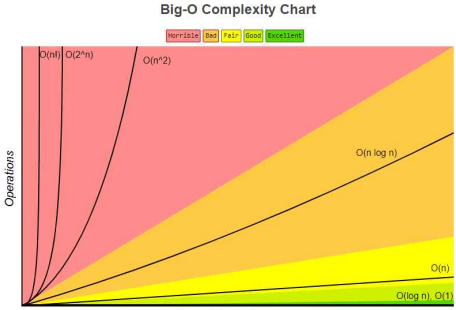
```
int fibo(int n){
    if(n <=1) return n;
    return fibo(n-2) + fibo(n-1);
}
```

1. 빅-오 표기법

- $O(1) < O(\log N) < O(N) < O(N \log N) < O(N^2) < O(2^N) < O(N!)$

  Fast <-- --> Slow

- O(3) -> O(1)
- O(100N + 3) -> O(N)
- $O(N^2 + 2N + 7)$ -> $O(N^2)$
- $O(2^N + 100N^2)$ -> $O(2^N)$
- $O(N! + 2^N + 1000)$ -> O(N!)



https://www.bigocheats

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
```

직관적으로 생각해보자.
동전 개수를 최소화하려면
가치가 높은 동전부터 최대한 많이 사용하면 된다.

4200 = 1000*4 + 100*2

반례가 있을까?
이 방법이 옳다고 증명할 수 있을까?

2. 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. ($1 \le N \le 10$, $1 \le K \le 100,000,000$)

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.

```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
```

주유소

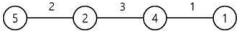
성공 서브태스크

실버 III

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	512 MB	39150	14982	11886	38.063%

문제

예를 들어, 이 나라에 다음 그림처럼 4개의 도시가 있다고 하자. 원 안에 있는 숫자는 그 도시에 있는 주유소의 리터당 가격이다. 도로 위에 있는 숫자는 도로의 길이를 표시한 것이다.



제일 왼쪽 도시에서 6리터의 기름을 넣고, 더 이상의 주유 없이 제일 오른쪽 도시까지 이동하면 총 비용은 30원이다. 만약 제일 왼쪽 도시에서 2리터의 기름을 넣고(2×5 = 10원) 다음 번 도시까지 이동한 후 3리터의 기름을 넣고(3×2 = 6원) 다음 도시에서 1리터의 기름을 넣어(1×4 = 4원) 제일 오른쪽 도시로 이동하면, 총 비용은 20원이다. 또 다른 방법으로 제일 왼쪽 도시에서 2리터의 기름을 넣고(2×5 = 10원) 다음 번 도시까지 이동한 후 4리터의 기름을 넣고(4×2 = 8원) 제일 오른쪽 도시까지 이동하면, 총 비용은 18원이다.

각 도시에 있는 주유소의 기름 가격과, 각 도시를 연결하는 도로의 길이를 입력으로 받아 제일 왼쪽 도시에서 제일 오른쪽 도시로 이동하는 최소의 비용을 계산하는 프로그램을 작성하시오.

입력

표준 입력으로 다음 정보가 주어진다. 첫 번째 줄에는 도시의 개수를 나타내는 정수 N(2 ≤ N ≤ 100,000)이 주어진다. 다음 줄에는 인접한 두 도시를 연결하는 도로의 길이가 제일 왼쪽 도로부터 N-1개의 자연수로 주어진다. 다음 줄에는 주유소의 리터당 가격이 제일 왼쪽 도시부터 순서대로 N개의 자연수로 주어진다. 제일 왼쪽 도시부터 제일 오른쪽 도시까지의 거리는 1이상 1,000,000,000 이하의 자연수이다. 리터당 가격은 1 이상 1,000,000,000 이하의 자연수이다.

출력

표준 출력으로 제일 왼쪽 도시에서 제일 오른쪽 도시로 가는 최소 비용을 출력한다.

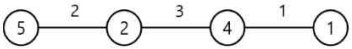
예제 입력 1 복사

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

예제 출력 1 복사

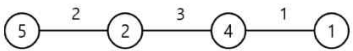
```
18
```

5원 * 6L = 30원



= 30원

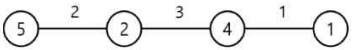
5원 * 2L = 10원



10+8 = 18원

2원 * 4L = 8원

5원 * 2L = 10원 4원 * 1L = 4원



10+6+4 = 20원

2원 * 3L = 6원

- 그리디?!

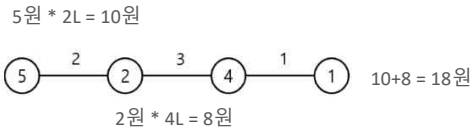
How?

1. 전체 문제를 작은 부분 문제로 나누기

2. 부분의 최적해를 찾기

3. 부분 최적해가 정당한지 증명하기

4. 구현하기

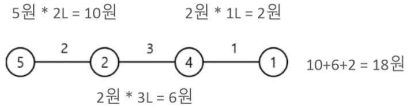


1. 전체 문제

맨 왼쪽 도시에서 맨 오른쪽 도시로 가기 위해 필요한 최소 비용 구하기

2. 부분의 최적해 찾기

여태까지 나온 가장 저렴한 기름으로 최대한 많이 이동하는 것



3. 정당성 증명하기 (귀류법)

반대 가정 1 - 가장 저렴한 기름을 제외한 임의의 기름으로, 최대한 많이 이동하기

반례 - (5원*2L) + (5원*3L) + (4원*1L) = 10+15+4 = 29원

반대 가정 2 - 가장 저렴한 기름으로, 적당히 이동하기

반례 - (5원*2L) + (2원*2L + 5원*1L) + (2원*1L) = 10+4+5+2 = 21원

-> 맨 처음 명제가 참임

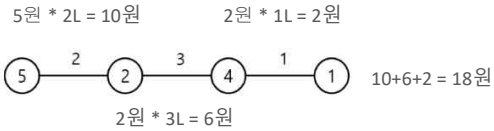
```cpp
#include <iostream>
using namespace std;
typedef long long ll;
#define NMAX 100000
int main() {
    ios_base::sync_with_stdio(0); cin.tie(0);

    int n; cin>>n;
    ll ans=0; //제일 왼쪽 도시에서 제일 오른쪽 도시로 이동하는 최소의 비용
    int road[NMAX], price[NMAX];

    for(int i=0; i<n-1; i++) cin>>road[i]; //인접한 두 도시를 연결하는 도로의 길이
    for(int i=0; i<n; i++) cin>>price[i]; //주유소의 리터당 가격

    int Pmin=price[0];
    for(int i=0; i<n-1; i++){
        Pmin=min(Pmin,price[i]);
        ans+=(ll)road[i]*Pmin;
    }
    cout<<ans;
    return 0;
}
```

예제 입력 1 복사

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

예제 출력 1 복사

```
18
```



쉬는 시간

1. 시간복잡도

2. 그리디
   - 11047. 동전0
   - 13305. 주유소
   - 1931. 회의실 배정

부록. 2차원 배열/벡터

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.

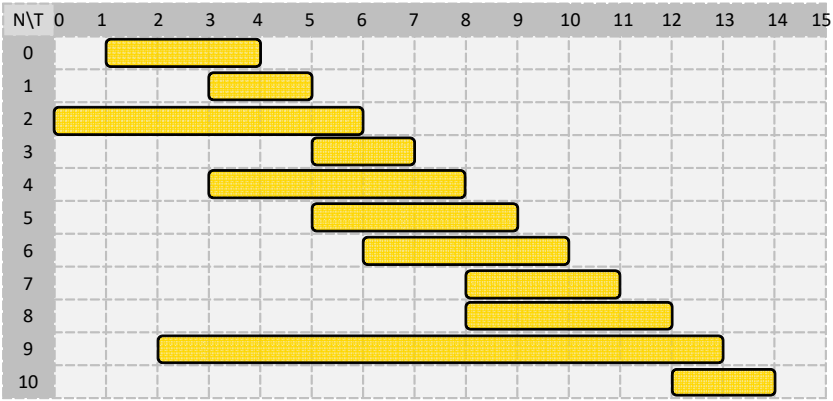
예제 입력 1 복사

```
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**



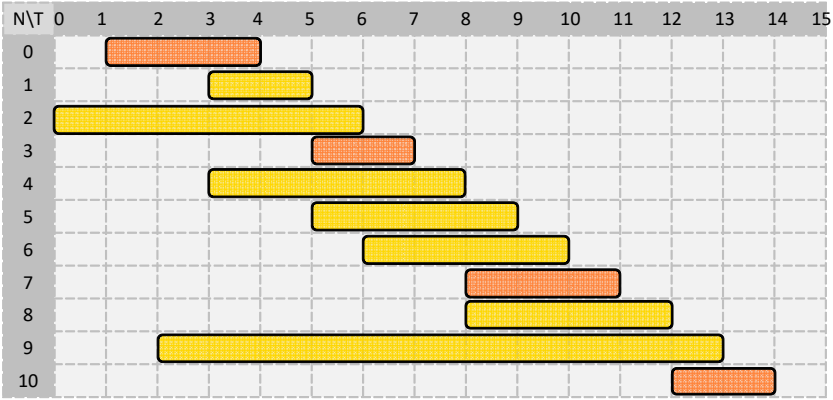
예제 입력 1 복사

```
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
```

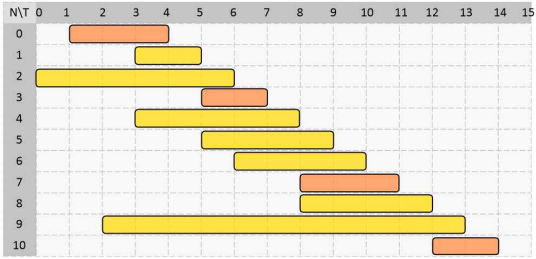
- 완전탐색?

  - 각 회의를 배정 한다/안 한다 : $O(2^N)$

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

  -> 시간 복잡도 $O(N * 2^N)$

  Nmax = 100,000



- 그리디?!

How?

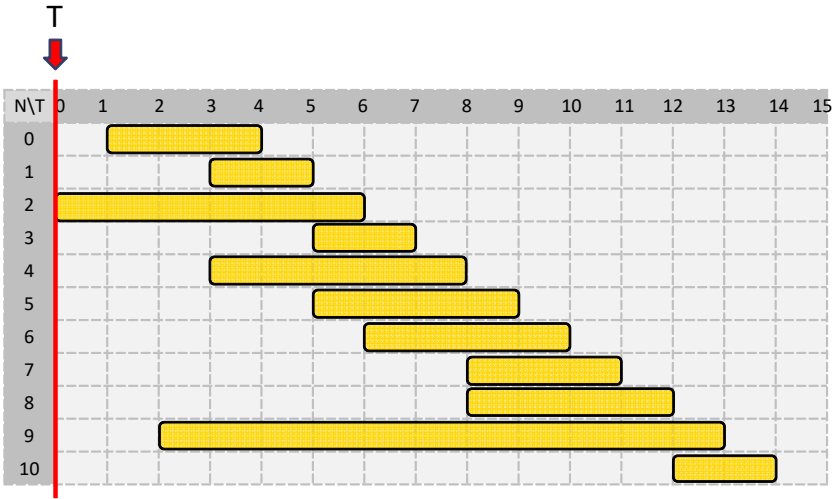
1. 전체 문제를 작은 부분 문제로 나누기

2. 부분의 최적해를 찾기

3. 부분 최적해가 정당한지 증명하기

4. 구현하기

1. 전체 문제를 작은 부분 문제로 나누기



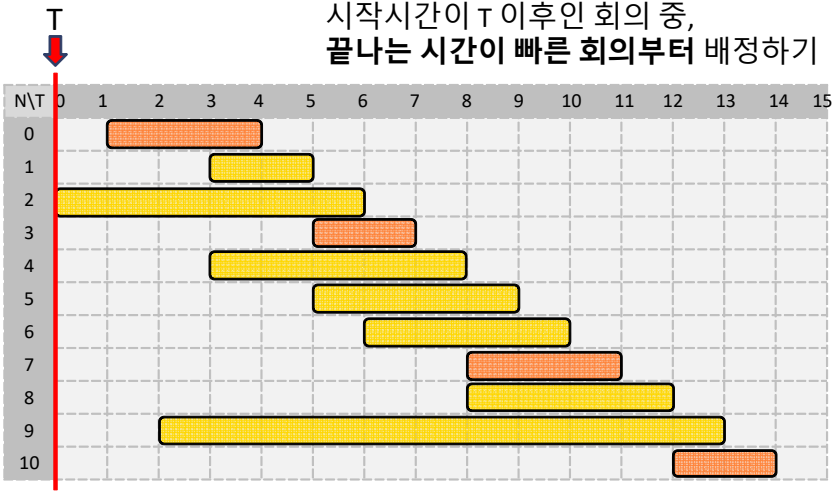
예제 입력 1 복사

```
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

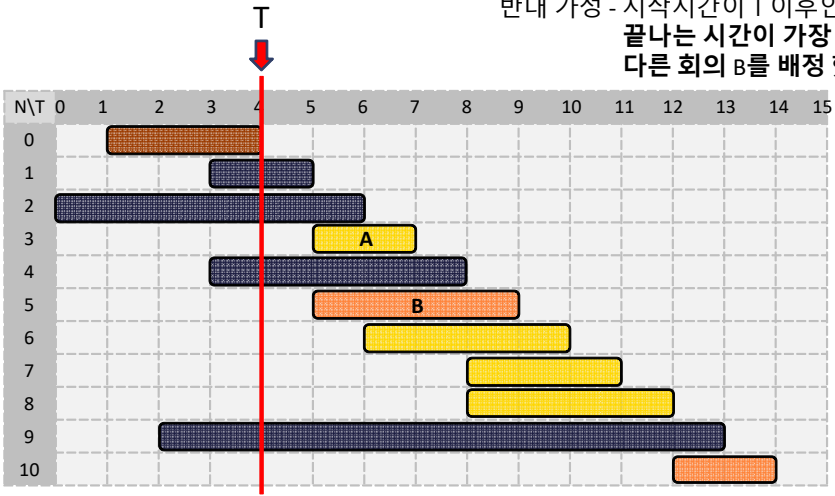
```
11
1 4
3 5
0 6
5 7
3 8
5 9
6 10
8 11
8 12
2 13
12 14
```

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

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

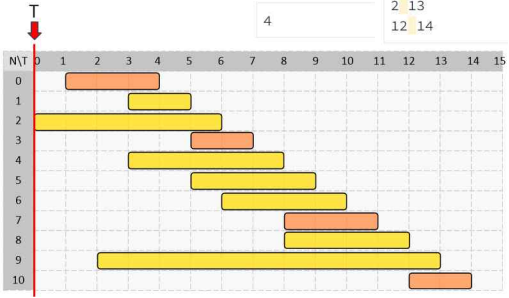
시작시간이 T 이후인 회의 중,
끝나는 시간이 빠른 회의부터 배정하기

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
```

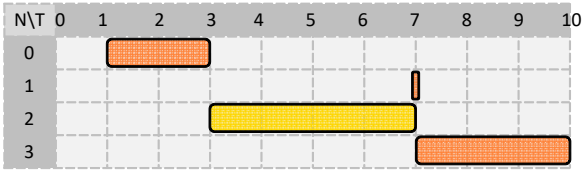



```cpp
#include <iostream>
#include <algorithm>
using namespace std;

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;
}
```

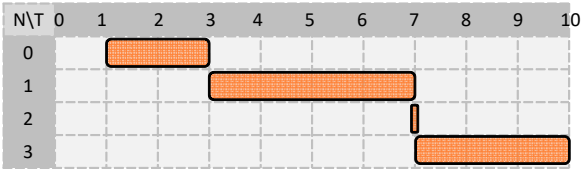
1931 틀렸습니다

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

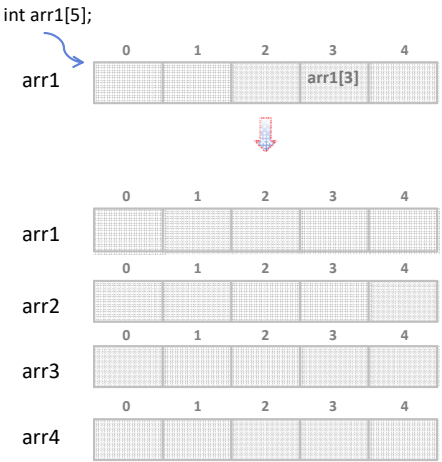
- When?
  - 부분 최적해를 반복적으로 취해서 답을 구할 수 있는 문제
    -> 부분 최적해들을 적용한 것이 전체 문제에서도 최적해인 경우

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

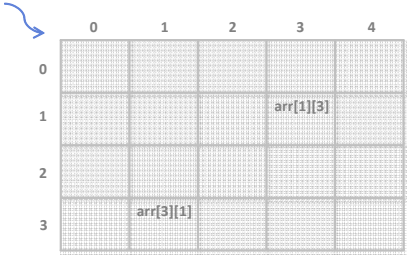
- 장점?
  - 탐색범위가 줄어서 빠른 시간에 풀 수 있음(시간 복잡도⬇)

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
  - 11047. 동전 0
  - 13305. 주유소
  - 1931. 회의실 배정
- 연습 문제
  - 1946. 신입 사원
  - 20921 그렇고 그런 사이
  - 23559. 밥
  - 11399. ATM
  - 16206. 롤케이크
  - 2847. 게임을 만든 동준이
  - 11501. 주식
  - 13413 오셀로 재배치
  - 19941. 햄버거 분배
  - 2167. 2차원 배열의 합
  - 1895. 필터
  - 19644. 좀비 떼가 기관총 진지에도 오다니
- 9월 28일(수요일) 저녁 6시 T동