

1주차: 다양한 문제 해결 기법

브루트 포스, 그리디, 분할정복

신지섭

■ 앞으로의 일정

- 1주차: 브루트 포스, 그리디, 분할정복
- 2주차: 이분탐색, LIS(최장 증가 부분 수열)
- 3주차: BFS, DFS, 백트래킹
- 4주차(미정): 복습 || 최단 경로(다익스트라, 벨만-포드, 플로이드-와샬)

브루트 포스 Brute Force



■ Brute: 짐승, 야만적인, 무식한

■ Force: 힘

브루트 포스(**완전 탐색**)는 이름처럼 무차별적으로 **모든 경우의 수**를 탐색합니다.

이 알고리즘의 강력한 장점은 **예외없이 100% 정답**만을 출력합니다.

~~구현만 잘 한다면....~~

그 대신, 시간이 오래 걸린다는 단점이 있습니다.

Brute Force 예

⇒ 선형탐색: 1부터 n 까지 하나씩 탐색해 원하는 값 x 를 찾는 알고리즘

⇒ BFS, DFS: 비선형 자료구조를 탐색하는 알고리즘

(나중에 배울 내용입니다!!)

Brute Force Algorithm 문제풀이

일단 풀어보자!

백준 2309
일곱 난쟁이

2309 일곱 난쟁이

문제: 9명 중에 선택한 7명의 합이 100이 되면 된다

옵션1: 9명 중에 7명을 선택하는 것(7중 for문)

옵션2: 9명 중에 2명을 선택하는 것(2중 for문)

=> 더 간단한 것은?

전체 키 합 - 100 = x

9명 중에 뽑은 두명의 키의 합이 x가 되면 나머지 7명 출력!

※시간 복잡도: $9C2 == 9C2$

일곱 난쟁이 코드

```
1 #include <iostream>
2 #include <algorithm>
3 using namespace std;
4
5 int k[15]; //배열은 넉넉히
6
7 int main() {
8     //속도 향상
9     ios_base::sync_with_stdio(false);
10    cin.tie(NULL);
11    cout.tie(NULL);
12    int sum = 0;
13    for (int i = 0; i < 9; i++) {
14        cin >> k[i]; //입력 받기
15        sum += k[i];
16    }
17    for (int i = 0; i < 8; i++) {
18        for (int j = i + 1; j < 9; j++) {
19            if (k[i] + k[j] == sum - 100) {
20                k[i] = 1000;
21                k[j] = 1000;
22                // 키를 가장 크게 만들어
23                // 맨 뒤에 정렬됨
24                sort(k, k + 9); // 오름차순 정렬
25                for (int i = 0; i < 7; i++) {
26                    cout << k[i] << "\n";
27                }
28                // 출력 후 종료
29                return 0;
30            }
31        }
32    }
33    return 0;
34 }
```

■ 그리디 Greedy

탐욕 알고리즘, 욕심쟁이 알고리즘으로 불림

미래를 생각하지 않고 **각 단계에서 가장 최선의 선택을 하는 기법**

=> 각 단계에서 최선의 선택이 전체적으로 최선일때 사용하면 됩니다!!

그리디를 이용할 수 있는 경우에는 다른 알고리즘보다 **빠른 시간** 내에 풀 수 있다.

=> 다시 말하면, **그리디로 풀 수 있는 문제는 다른 방법으로도 풀 수 있다**

직관적인 간단한 방법이 있을 것 같고 시간제한이 작다 => 그리디를 의심해라!

Greedy Algorithm 문제풀이

문제가 어렵지 않으니
일단 풀어보자!

백준 5585
거스름돈

5585 거스름돈

문제: 거스름돈을 500엔, 100엔, 50엔, 10엔, 5엔, 1엔짜리 화폐로 주는데
잔돈의 개수를 최소로~!

잔돈이 배수 관계에 놓여 있다

=> 예를 들어, 50엔을 거슬러 줄 때 10엔짜리 5개보다 50엔짜리 1개가 이득!!

단위가 큰 잔돈을 먼저 준다!

거스름돈 코드

```
1  #include <iostream>
2  using namespace std;
3
4  int coin[6] = { 500, 100, 50 ,10 ,5, 1 }; //동전 배열
5
6  int main() {
7      ios_base::sync_with_stdio(false);
8      cin.tie(NULL);
9      cout.tie(NULL);
10     int money;
11     int an = 0;
12     cin >> money;
13     money = 1000 - money;
14     for (int i = 0; i < 6; i++) { // 큰 동전 순으로
15         an += (money / coin[i]);
16         money %= coin[i];
17     }
18     cout << an;
19     return 0;
20 }
```

Greedy Algorithm 문제풀이

**백준 1931
회의실 배정**

1931 회의실 배정

문제: 하나의 회의실을 이용해 겹치지 않게 최대한 많은 수의 회의를 잡는 수를 구하자

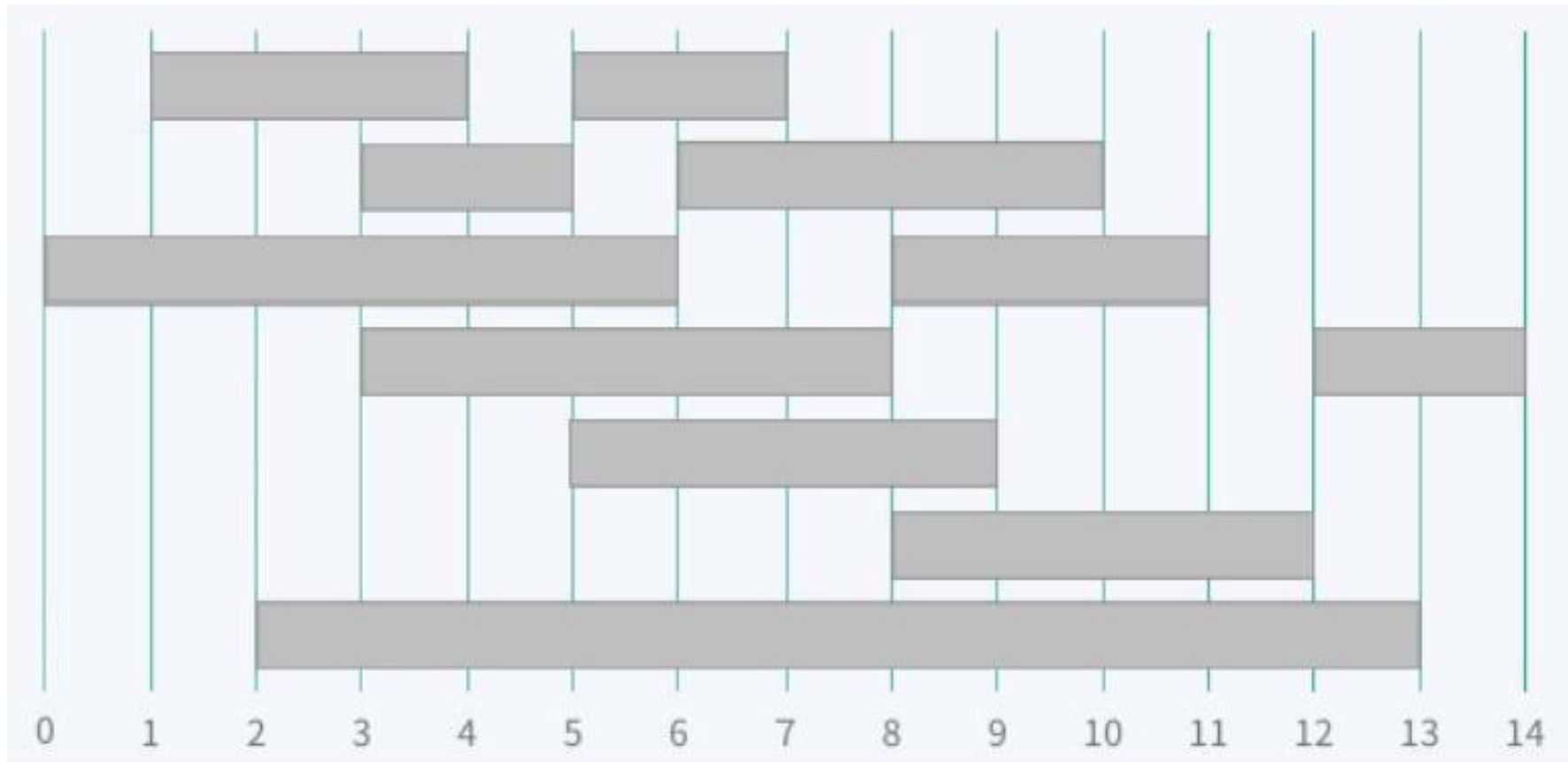
※ 문제에 주어진 조건

1. 회의가 한번 시작하면 중단되지 않는다.
2. 한 회의가 끝나는 것과 동시에 다음회의가 시작될 수 있다.
3. 회의의 시작시간과 종료시간이 같을 수가 있다!

=> 이를 만족하는 최대 회의 진행 가능 수를 출력해주면 된다!

1931 회의실 배정

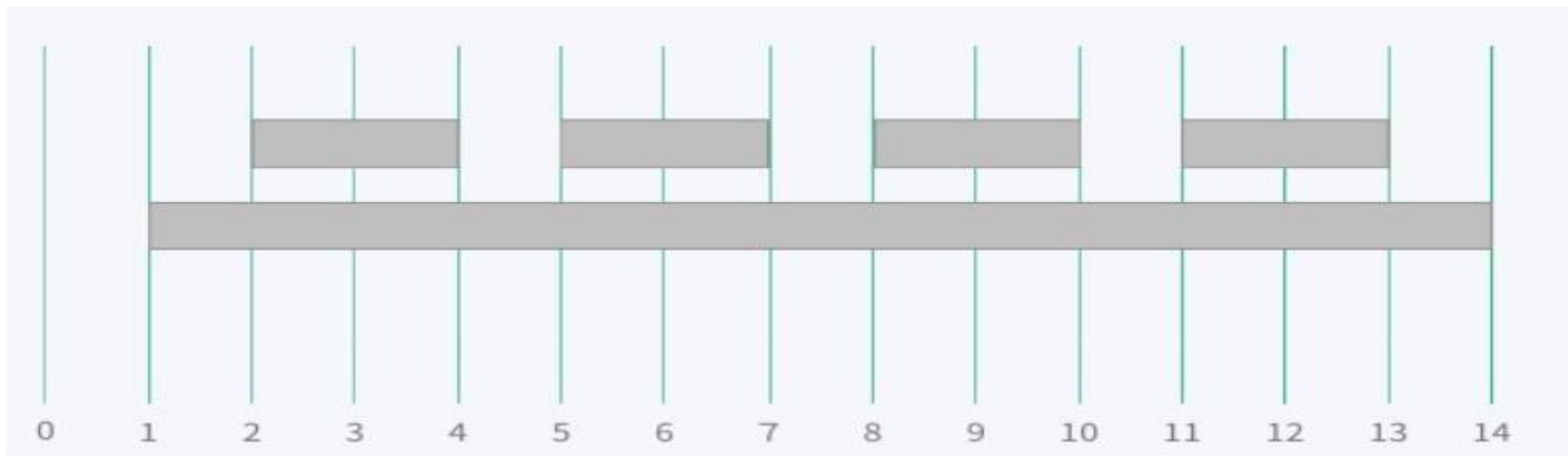
현재 문제 예제에 나온 입력 결과에 따라 회의 진행을 막대 그림으로 나타내면 다음과 같다.



1931 회의실 배정

1. 회의를 일찍 시작하는 순으로 접근하면?

가장 처음 생각나는 방법이다. 일찍 시작하는 순으로 정렬하면 구할 수 있지 않을까?
하지만 아래 그림과 같은 반례가 존재한다.

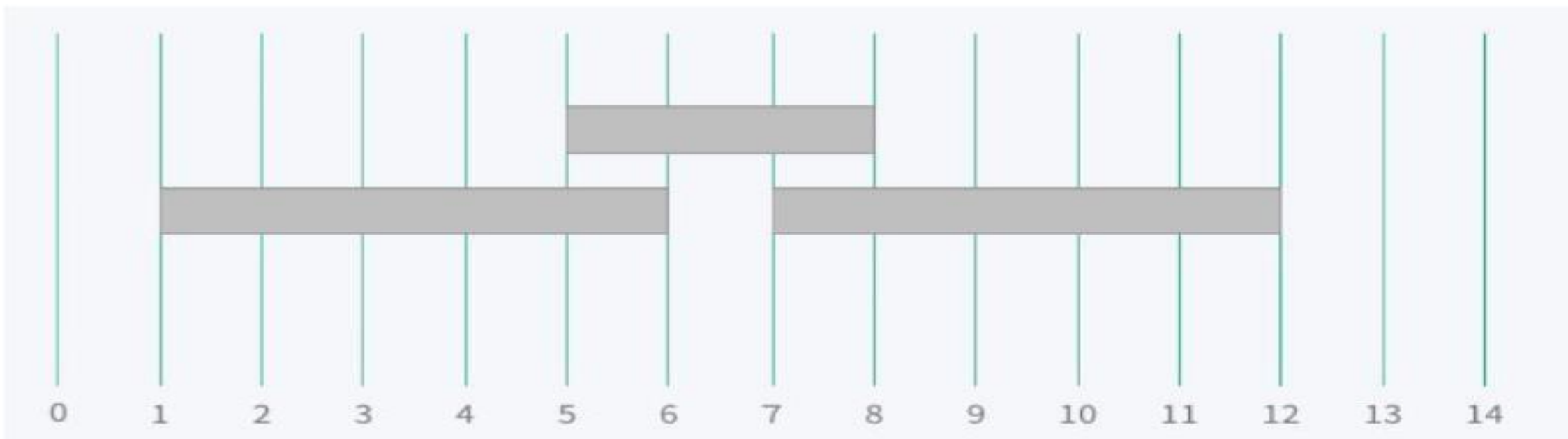


일찍 시작했지만, **종료시간이 늦어서** 중간에 빨리 끝나는 회의가 있으면 최대 회의 수를 구할 수 없다.

1931 회의실 배정

2. 그렇다면 회의가 짧은 순으로 구하면?!

위에서 존재했던 반례를 이용해 짧은 순으로 정렬하면 가능할까 싶지만, 또 반례가 존재한다.

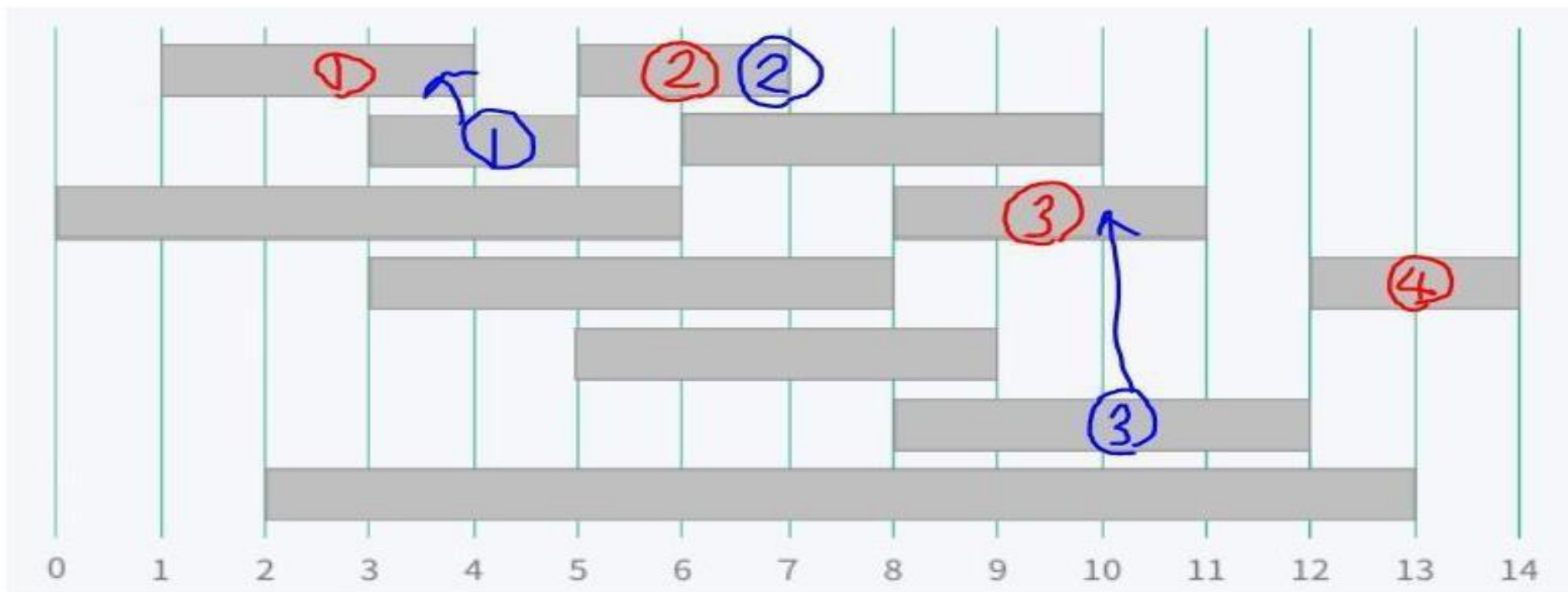


위 그림과 같이 존재한다면, 아무리 짧은 회의 더라도 최대 회의 수를 구할 수 없다.

1931 회의실 배정

3. 일찍 끝나는 회의를 기준으로 잡으면?!

회의가 일찍 끝나는 기준을 잡으면 ,회의를 **선택할 수 있는 범위**가 넓어진다.
일찍 끝나면 더 회의를 많이 진행할 수 있기 때문이다.



1931 회의실 배경

몇 가지 의문을 가져봅시다.

1. 과연 예외는 없을까?
2. 그냥 단순히 내가 감으로 푸는게 아닐까?
3. 진짜 빨리 끝나는 순으로 정렬했을 때 그것이 전체적으로 최선의 선택이라고 단정짓기에는 좀 이른 감이 있지 않을까?

“이게 맞아?”

1931 회의실 배정 - 정당성 증명

1. 탐욕적 선택 속성

동적 계획법처럼 답의 모든 부분을 고려하지 않고 탐욕적으로만 선택하더라도 최적해를 구할 수 있는 속성

◆ 가장 종료 시간이 빠른 회의 s_{\min} 을 포함하는 최적해가 반드시 존재한다.

증명)

S 의 최적해 중 s_{\min} 을 포함하지 않는 답이 있다면 이 답의 목록에서 첫번째로 개최되는 회의를 지운다. 그리고 s_{\min} 을 대신 추가해서 새로운 목록을 만든다.

s_{\min} 은 S 에서 가장 일찍 끝나는 회의이기 때문에 지워진 회의는 s_{\min} 보다 일찍 끝날 수 없다.

두번째 회의와 s_{\min} 이 겹치는 일이 없으며, 새로 만든 목록도 최적해 중 하나가 된다.

1931 회의실 배정 - 정당성 증명

2. 최적 부분 구조

항상 **최적의 선택만을 내려서 전체 문제의 최적해**를 얻을 수 있음을 보여야한다!
즉, **부분문제의 최적해**에서 **전체 문제의 최적해**를 만들 수 있음을 보여야한다!

모든 회의를 **종료시간의 오름차순으로 정렬**해 두고

정렬 된 배열의 첫 번째 회의는 무조건 선택해도 된다!! (아까 전에 증명했어용 ☺)

그 후 정렬 된 배열을 순회하면서 **첫 번째 회의와 겹치지 않는 회의**를 찾습니다.

회의들은 오름차순으로 정렬 되어있기에

겹치지 않는 회의를 찾자마자 **나머지를 보지 않고 선택해도 된다.**

회의실 배정 코드

```
1 #include <iostream>
2 #include <algorithm>
3 using namespace std;
4 int n;
5 pair<int, int> meeting[100005]; // 두 개의 값을 담을 수 있는 구조
6
7 bool com(pair<int, int>a, pair<int, int> b) {
8     if (a.second == b.second) // 끝나는 시간이 같으면
9         return a.first < b.first; // 시작하는 시간으로 오름차순
10    return a.second < b.second; // 끝나는 시간으로 오름차순
11 }
12
13 int main() {
```

```
13 int main() {
14     ios_base::sync_with_stdio(false);
15     cin.tie(NULL);
16     cout.tie(NULL);
17     cin >> n;
18     for (int i = 0; i < n; i++) {
19         int a, b;
20         cin >> a >> b;
21         meeting[i] = { a, b }; // {시작시간, 끝시간}
22     }
23     sort(meeting, meeting + n, com); // com이라는 방법으로 정렬
24     // 끝시간 오름차순, 끝시간이 같으면 시작시간이 이른거 먼저
25
26     int an = 1;
27     int before = 0; // 전 회의의 인덱스번호
28     for (int i = 1; i < n; i++) {
29         if (meeting[i].first >= meeting[before].second) { // 전 회의가 끝나고 새로운 회의가 시작할 때
30             before = i;
31             an++;
32         }
33     }
34     cout << an;
35     return 0;
36 }
```

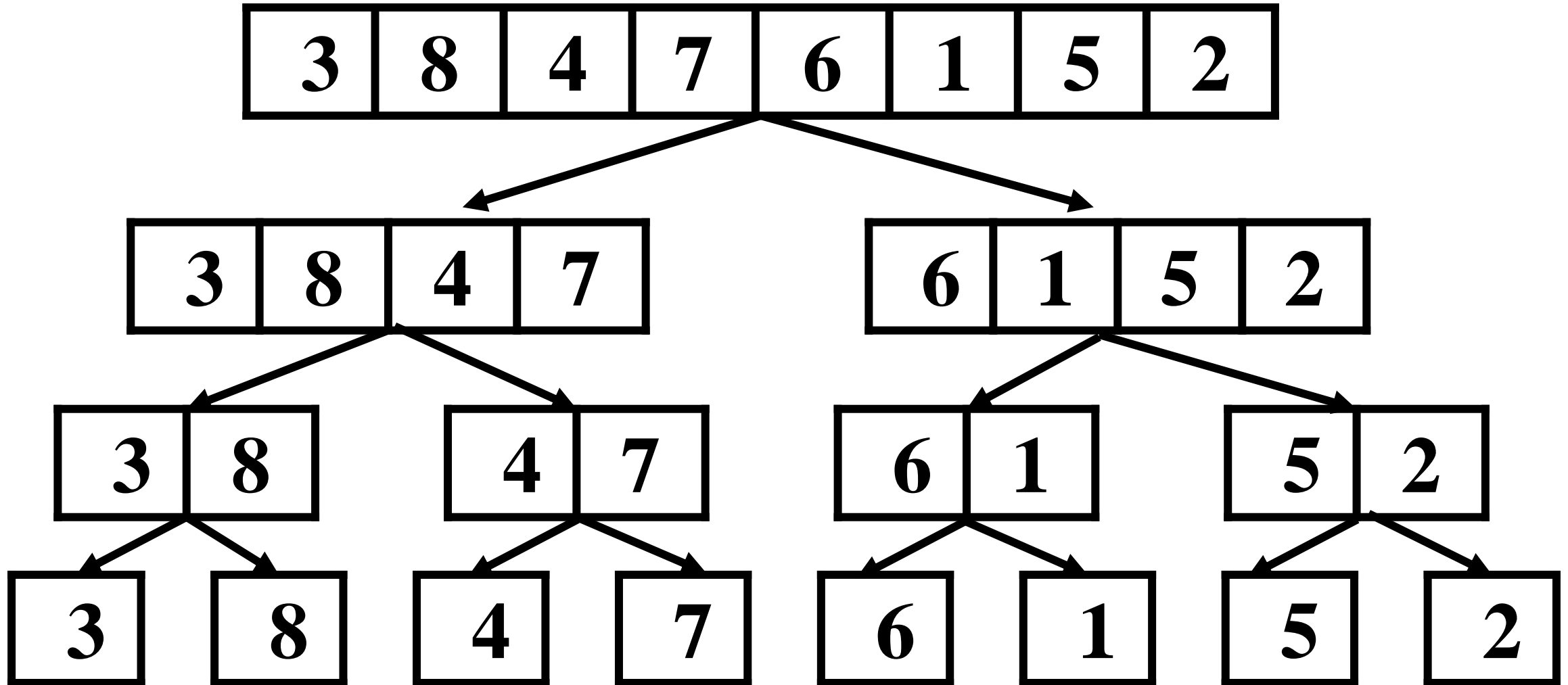
■ 분할 정복 Divide and Conquer

①문제를 나눌 수 없을 때까지 나누고 ②나눈 문제 각각을 풀면서 ③다시 합병해 푸는 방법

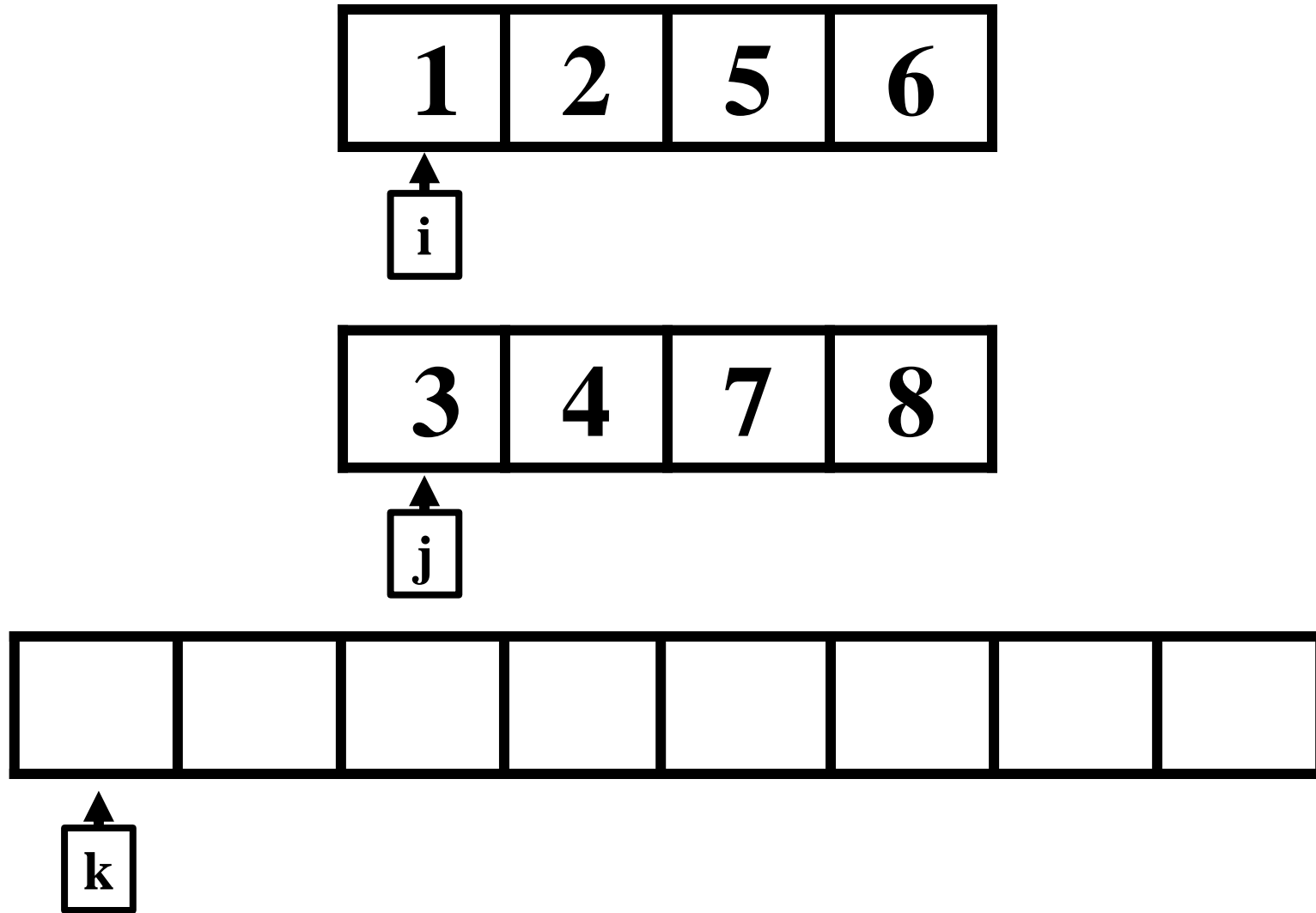
알고리즘 설계 과정

- ① Divide : 문제가 분할이 가능한 경우, 2개 이상의 문제로 나눈다.
- ② Conquer : 나누어진 문제가 여전히 분할이 가능하면, 또 다시 Divide를 수행한다. 분할이 안될 경우 그 문제를 푼다(기저조건)
- ③ Combine : Conquer한 문제들을 통합하여 문제의 답을 얻는다.

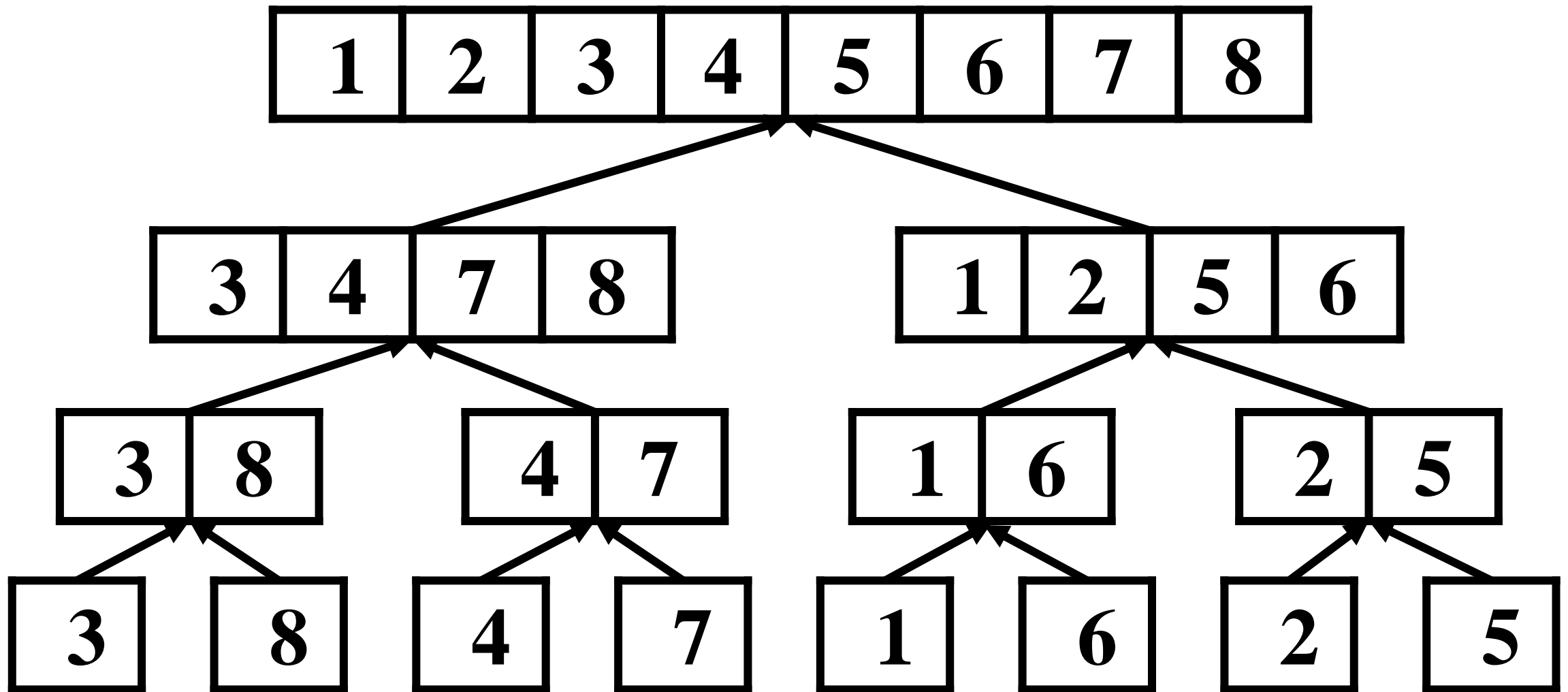
분할 정복 예시 – 병합 정렬(Merge sort)



분할 정복 예시 – 병합 정렬(Merge sort)



분할 정복 예시 - 병합 정렬(Merge sort)



병합정렬 코드

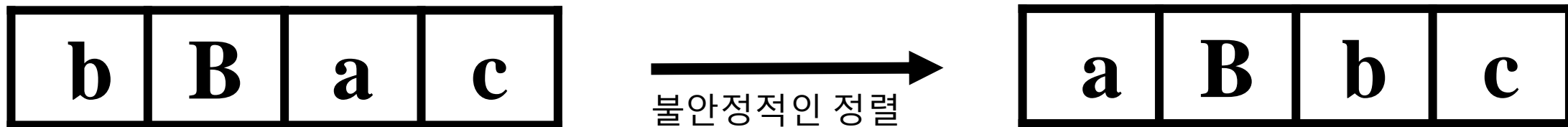
```
void mergeSort(int start, int end) {  
    if (start < end) {  
        int mid = (start + end) / 2;  
        mergeSort(start, mid); //집합을 두개로 나눔  
        mergeSort(mid + 1, end);  
        merge(start, mid, end); // 나눈 두개의 집합 합치기  
    }  
}
```

```
void merge(int start, int mid, int end) {  
    int i = start;  
    int j = mid + 1;  
    int k = start;  
    while (i ≤ mid && j ≤ end) {  
        if (num[i] ≤ num[j]) { // 첫번째 배열값이 두번째 배열보다 작은 경우  
            sorted[k] = num[i];  
            i++; // index 증가  
        }  
        else { // 두번째 배열값이 첫번째 배열보다 작은 경우  
            sorted[k] = num[j];  
            j++; // index 증가  
        }  
        k++;  
    }  
    if (i > mid) { //첫번째 배열을 모두 넣었을 때  
        for (int t = j; t ≤ end; t++) {  
            sorted[k] = num[t];  
            k++;  
        }  
    }  
    else { //두번째 배열을 모두 넣었을 때  
        for (int t = i; t ≤ mid; t++) {  
            sorted[k] = num[t];  
            k++;  
        }  
    }  
    for (int t = start; t ≤ end; t++) {  
        num[t] = sorted[t];  
    } // 다시 대입  
}
```

※시간 복잡도: $n \log n$

정렬의 안정성

안정적인 정렬: 같은 값을 가진 데이터의 순서가 정렬 후에도 바뀌지 않고 그대로 유지 되는 정렬



이름	최선	평균	최악	메모리	안정
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	O
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	O
Heap sort	$O(n)$	$O(n \log n)$	$O(n \log n)$	$O(1)$	X
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(n)$	O
Quick sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$O(\log n)$	X

Divide and Conquer Algorithm 문제풀이

다들 풀어보셨죠?

백준 1074
Z

1074 Z

문제: Z자로 지그재그 방문한다. r행 c열을 몇 번째로 방문할까?

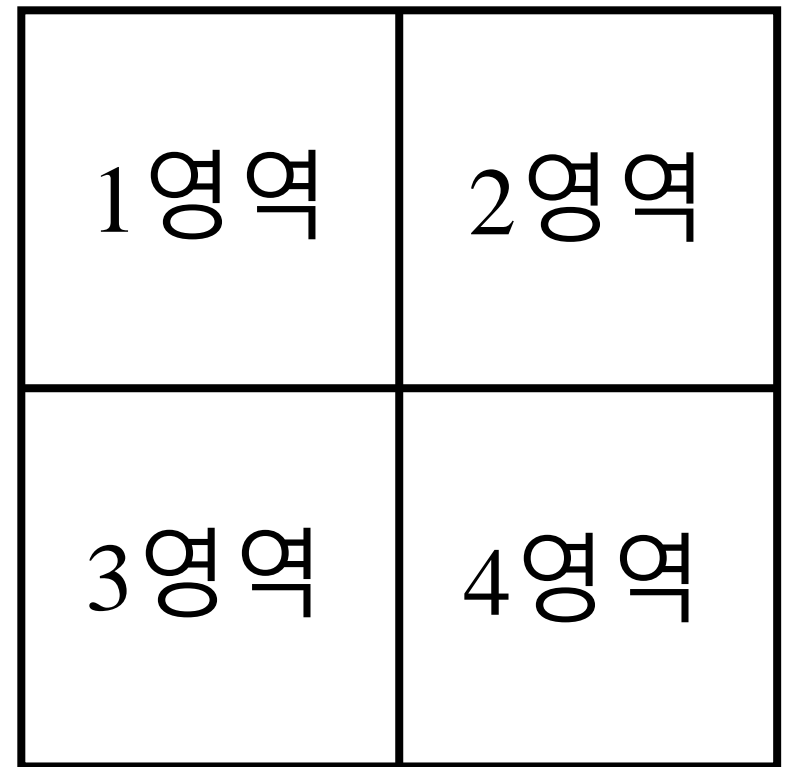
divide

=> 영역을 4개로 쪼개서 생각하자!!

영역을 쪼갤 수 없을 때(1x1)까지

Conquer

- 1영역: 영역 안에서 방문한 횟수
- 2영역: 영역 안에서 방문한 횟수 + 1영역의 수
- 3영역: 영역 안에서 방문한 횟수 + 1, 2영역의 수
- 4영역: 영역 안에서 방문한 횟수 + 1, 2, 3영역의 수



Z 코드

```
int z(int len, int x, int y) { //len은 4분면의 한 변의 길이
    if (len == 0) { //영역을 쪼갤 수 없을 때
        int sum = 0;
        if (x == 1)
            sum++;
        if (y == 1)
            sum += 2;
        return sum;
    }
    else {
        if (len ≥ x && len ≥ y) // 1영역
            return z(len / 2, x, y);
        else if (len < x && len ≥ y) //2영역
            return z(len / 2, x - len - 1, y) + pow(len + 1, 2);
        else if (len ≥ x && len < y) // 3영역
            return z(len / 2, x, y - len - 1) + (2 * pow(len + 1, 2));
        else return z(len / 2, x - len - 1, y - len - 1) + (3 * pow(len + 1, 2)); //4영역
    }
}
```

Problem Set – 브루트 포스

4

1920 수 찾기

5

1018 체스판 다시 칠하기


5


7568 덩치


5


1759 암호 만들기


Problem Set - 그리디

 1026 보물

 11399 ATM

 14241 슬라임 합치기

 1541 잃어버린 괄호

 2437 저울

Problem Set – 분할 정복



1629 곱셈



2630 색종이 만들기



1992 쿼드트리



1517 버블 소트

Problem Set – 도전!



5520 The Clocks



14927 전구 끄기



1725 히스토그램