2021 신촌 연합 여름캠프 초급반 7회차

분할정복 & 이분탐색

서강대학교 박재형

분할정복이란?

- Divide and Conquer

- 주어진 문제를 둘 이상의 부분 문제로 나누어 푸는 방법

- 거의 같은 크기로 부분 문제를 나눔

- 재귀 호출 사용

분할정복이란?

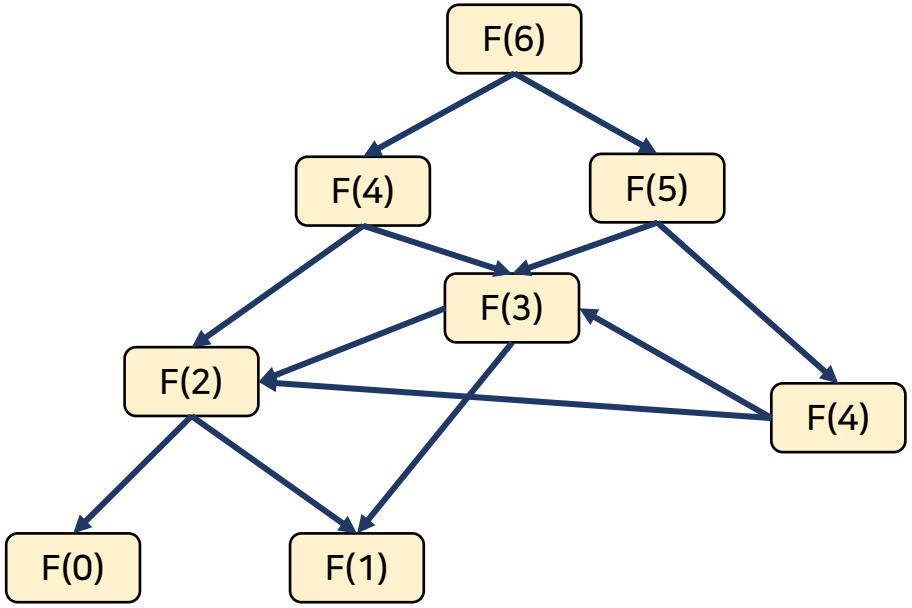
1) Divide : 부분 문제로 나눌 수 있는 경우 2개 이상의 문제로 나눈다.

2) Conquer : 더 이상 나눌 수 없는 경우 현재 문제를 해결(정복)한다.

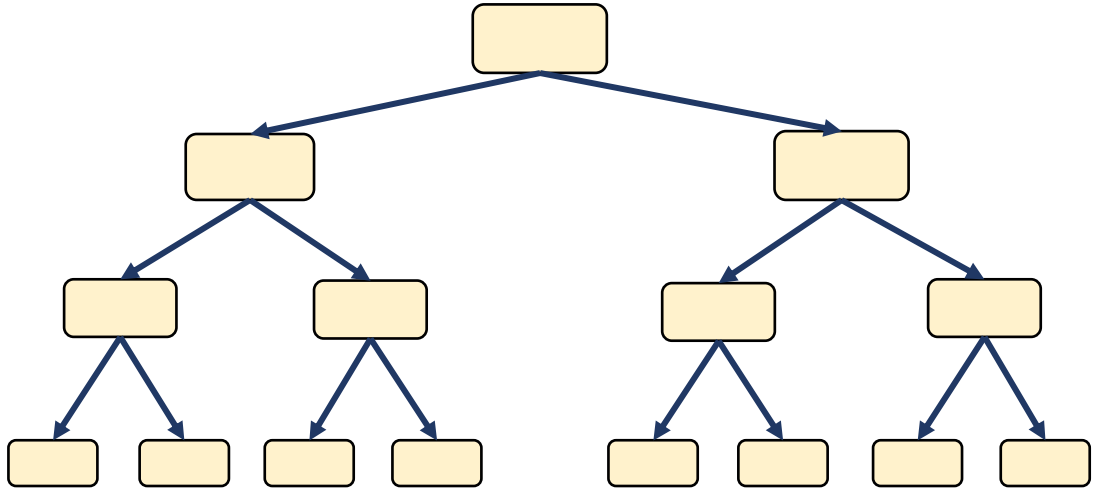
3) Combine : 해결된 부분 문제들을 합쳐서 기본 문제를 해결한다.

vs DP

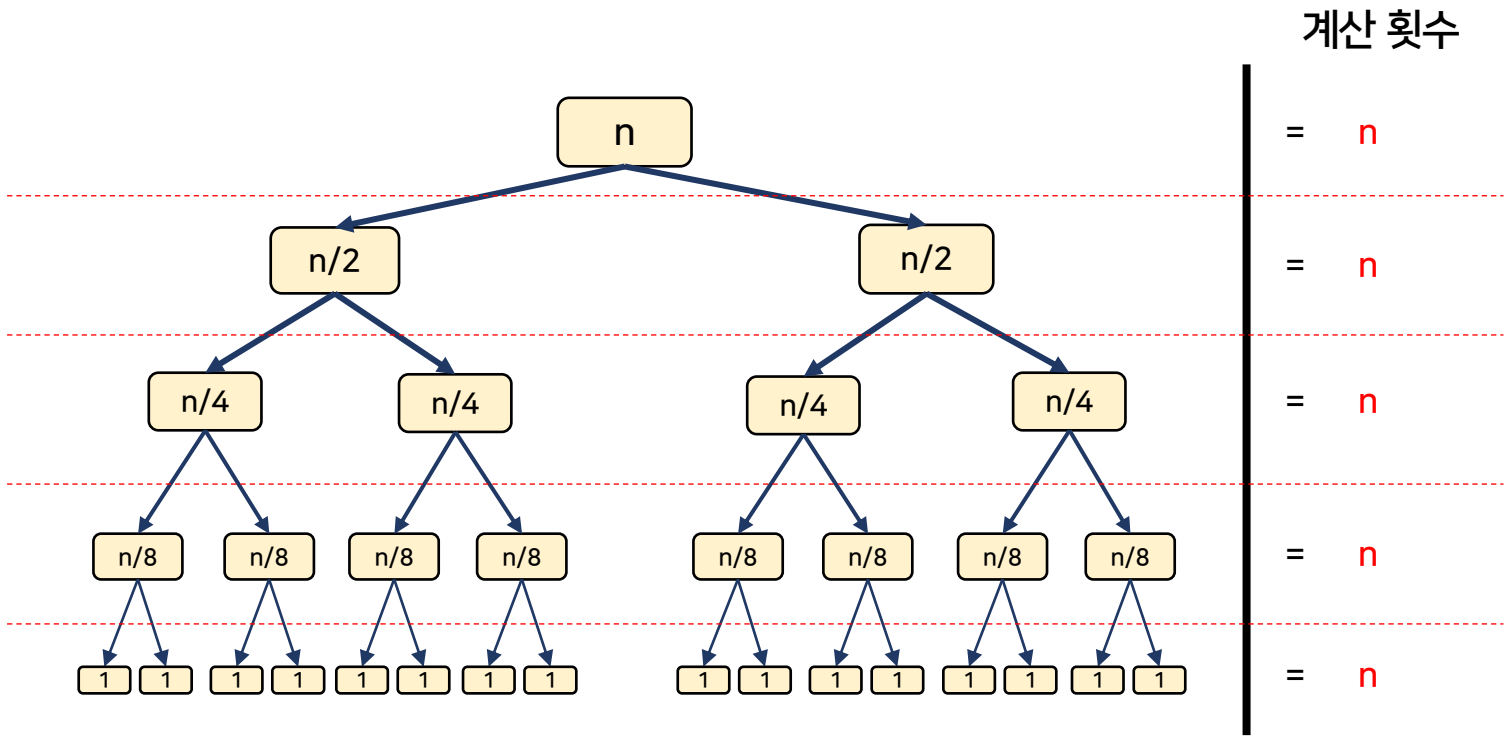
<DP>



<Divide and Conquer>



시간복잡도



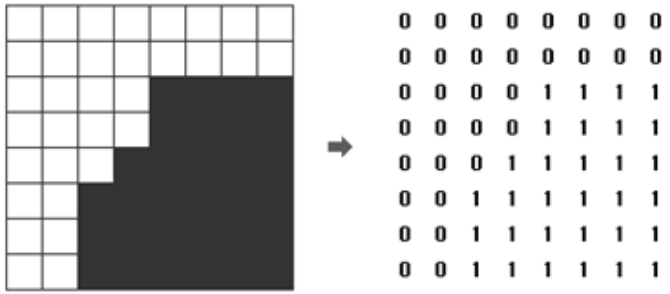
O(nlogn)

BOJ 1992 (쿼드트리)

문제

흑백 영상을 압축하여 표현하는 데이터 구조로 쿼드 트리(Quad Tree)라는 방법이 있다. 흰 점을 나타내는 0과 검은 점을 나타내는 1로만 이루어진 영상(2차원 배열)에서 같은 숫자의 점들이 한 곳에 많이 몰려있으면, 쿼드 트리에서는 이를 압축하여 간단히 표현할 수 있다.

주어진 영상이 모두 0으로만 되어 있으면 압축 결과는 "0"이 되고, 모두 1로만 되어 있으면 압축 결과는 "1"이 된다. 만약 0과 1이 섞여 있으면 전체를 한 번에 나타내지를 못하고, 왼쪽 위, 오른쪽 위, 왼쪽 아래, 오른쪽 아래, 이렇게 4개의 영상으로 나누어 압축하게 되며, 이 4개의 영역을 압축한 결과를 차례대로 괄호 안에 묶어서 표현한다

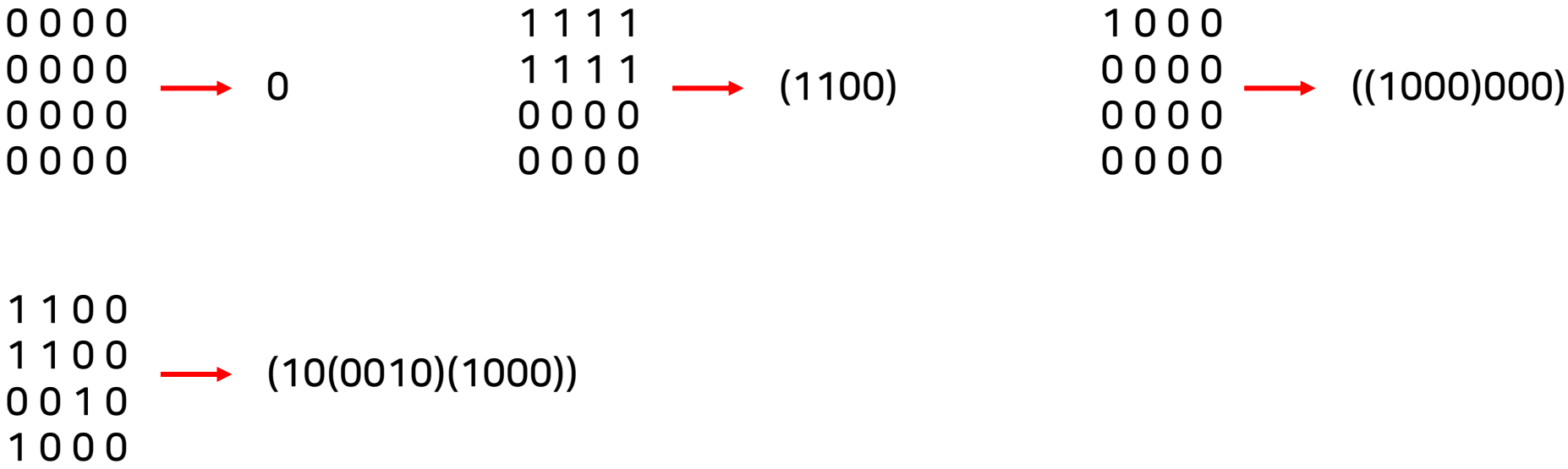


```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1
0 0 0 0 1 1 1 1
0 0 0 1 1 1 1 1
0 0 1 1 1 1 1 1
0 0 1 1 1 1 1 1
0 0 1 1 1 1 1 1
```

위 그림에서 왼쪽의 영상은 오른쪽의 배열과 같이 숫자로 주어지며, 이 영상을 쿼드 트리 구조를 이용하여 압축하면 "`(0(0011)(0(0111)01)1)`"로 표현된다. N ×N 크기의 영상이 주어질 때, 이 영상을 압축한 결과를 출력하는 프로그램을 작성하시오.

BOJ 1992 (쿼드트리)

- 출력 예시 설명

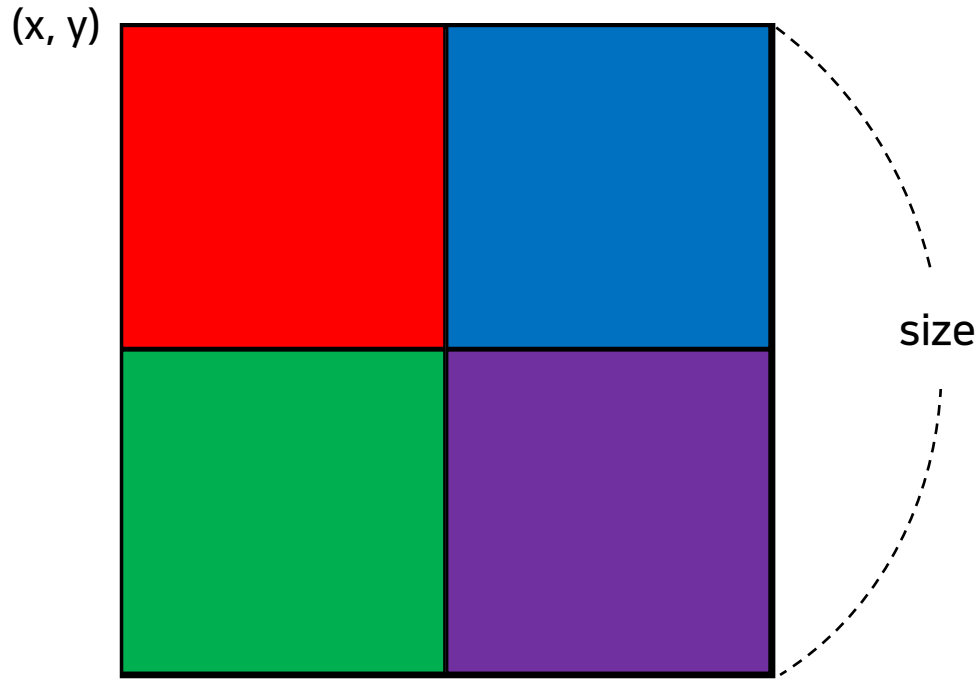


BOJ 1992 (쿼드트리)

- 4개의 영상으로 나누어 압축 → 4개의 부분 문제로 분할

(x, y, size)

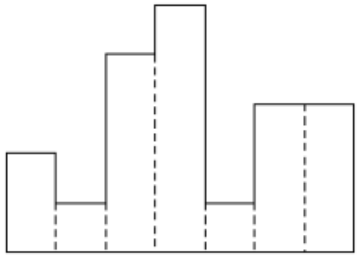
- (x, y, size/2)
- (x, y + size/2, size/2)
- (x + size/2, y, size/2)
- (x + size/2, y + size/2, size/2)



BOJ 1725 (히스토그램)

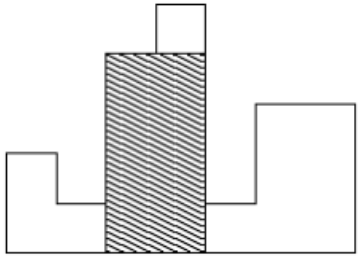
문제

히스토그램에 대해서 알고 있는가? 히스토그램은 아래와 같은 막대그래프를 말한다.



각 칸의 간격은 일정하고, 높이는 어떤 정수로 주어진다. 위 그림의 경우 높이가 각각 2 1 4 5 1 3 3이다.

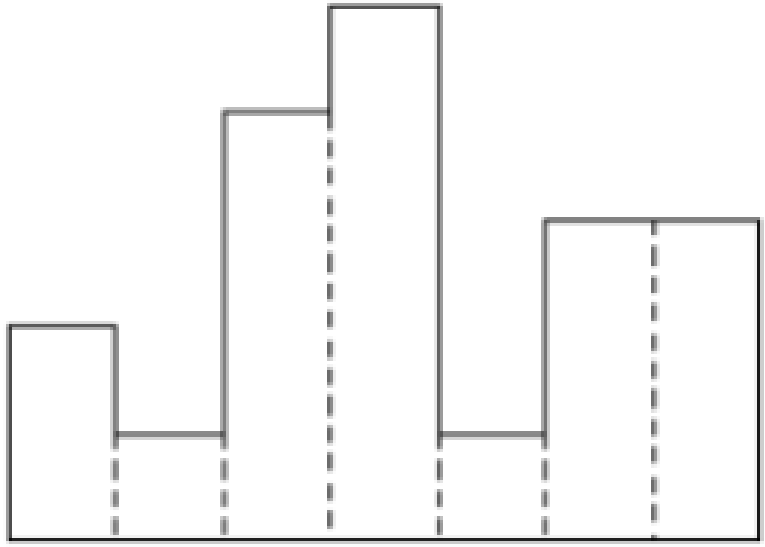
이러한 히스토그램의 내부에 가장 넓이가 큰 직사각형을 그리려고 한다. 아래 그림의 빗금 친 부분이 그 예이다. 이 직사각형의 밑변은 항상 히스토그램의 아랫변에 평행하게 그려져야 한다.



주어진 히스토그램에 대해, 가장 큰 직사각형의 넓이를 구하는 프로그램을 작성하시오.

BOJ 1725 (히스토그램)

- ✓ Naive 풀이 (N = 100,000)
- ✓ 시간 복잡도



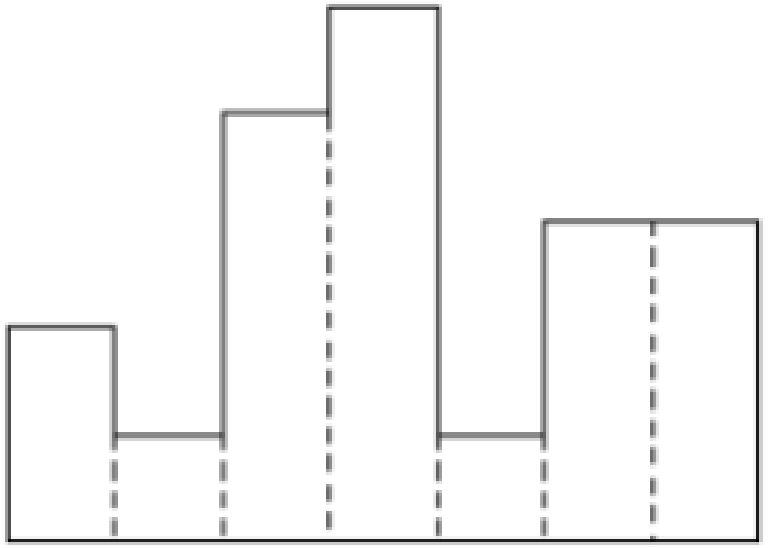
BOJ 1725 (히스토그램)

- ✓ Before) 현재 사각형이 왼쪽 끝인 직사각형으로 고려
- ✓ 현재 사각형을 포함하는 직사각형으로 생각한다면?

  현재 k번째 사각형 -> [1 , k] x [k , n] 개의 직사각형

  → $O(N^2)$

- ✓ 시간복잡도를 줄여보자! → $O(N)$



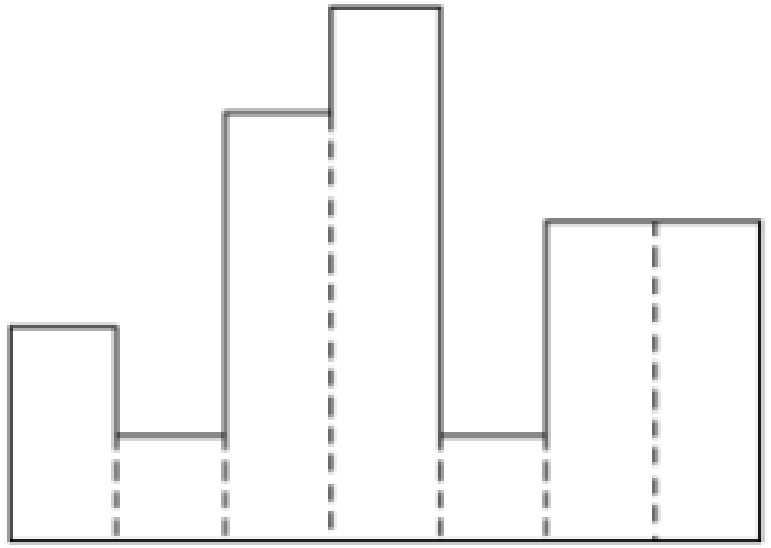
BOJ 1725 (히스토그램)

1. 현재 사각형을 포함하는 직사각형 (Conquer)

2. 현재 사각형을 포함하지 않는 직사각형 (Divide)

i) 왼쪽 : [1, k-1]

ii) 오른쪽 : [k+1, n]



이분 탐색

- 탐색 범위를 두 부분으로 나눠가면서 원하는 원소를 찾는 방법

- 탐색 전, 원소가 정렬되어 있어야 함

1	2	4	5	6	9	10	10	11	14

이분 탐색

1. 탐색 범위 설정 ex) [s, e]

2. 탐색 지점(mid) 설정 후 찾는 값과 비교 ex) mid = (s + e)/2

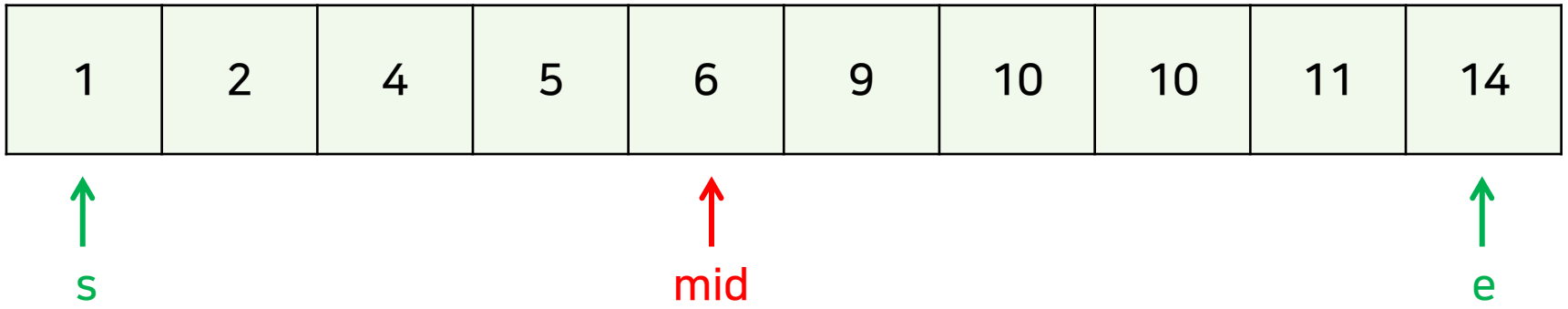
3. mid가 더 크면 [1, mid-1] 에서 다시 탐색, 더 작으면 [mid+1, e]에서 다시 탐색

4. s > e가 될 때까지 2~3 과정 반복

1	2	4	5	6	9	10	10	11	14

이분 탐색

Q. '5'의 위치



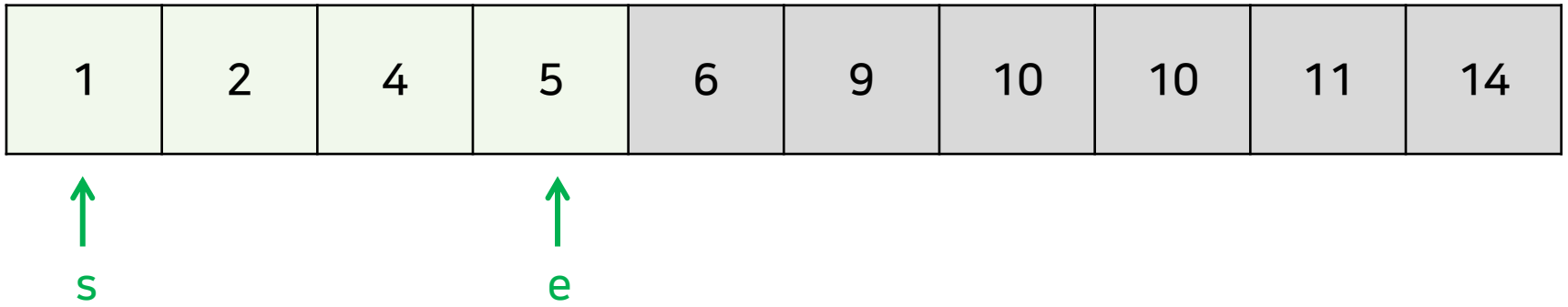
s = 1
e = 10
mid = (s + e)/2 = 5

ans = ?

A[mid] > 5

이분 탐색

Q. '5'의 위치

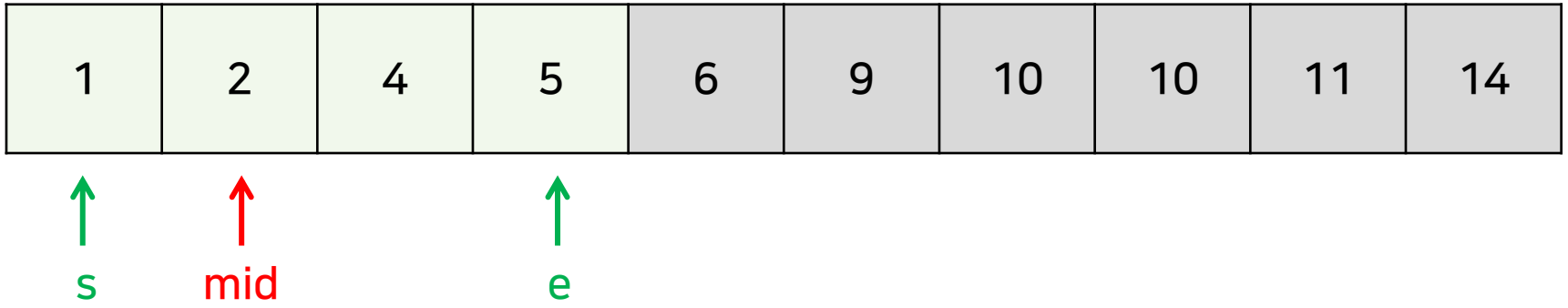


s = 1
e = 4
mid = (s + e)/2 =

ans = ?

이분 탐색

Q. '5'의 위치



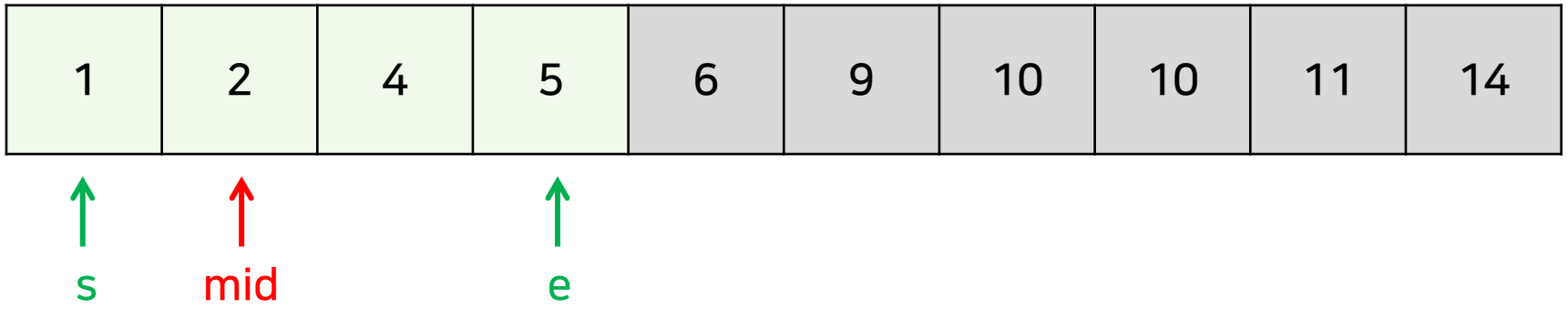
s = 1
e = 4
mid = (s + e)/2 = 2

ans = ?

A[mid] ≤ 5

이분 탐색

Q. '5'의 위치



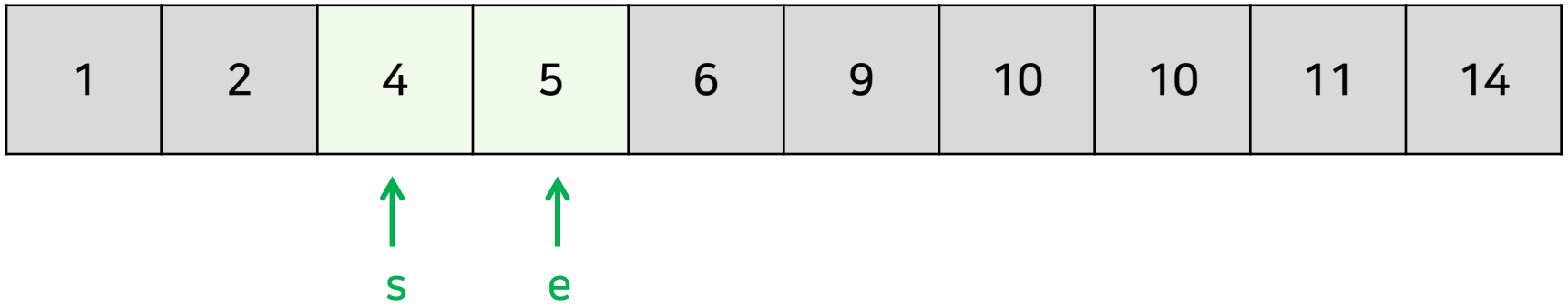
s = 1
e = 4
mid = (s + e)/2 = 2

ans = 2

A[mid] ≤ 5

이분 탐색

Q. '5'의 위치

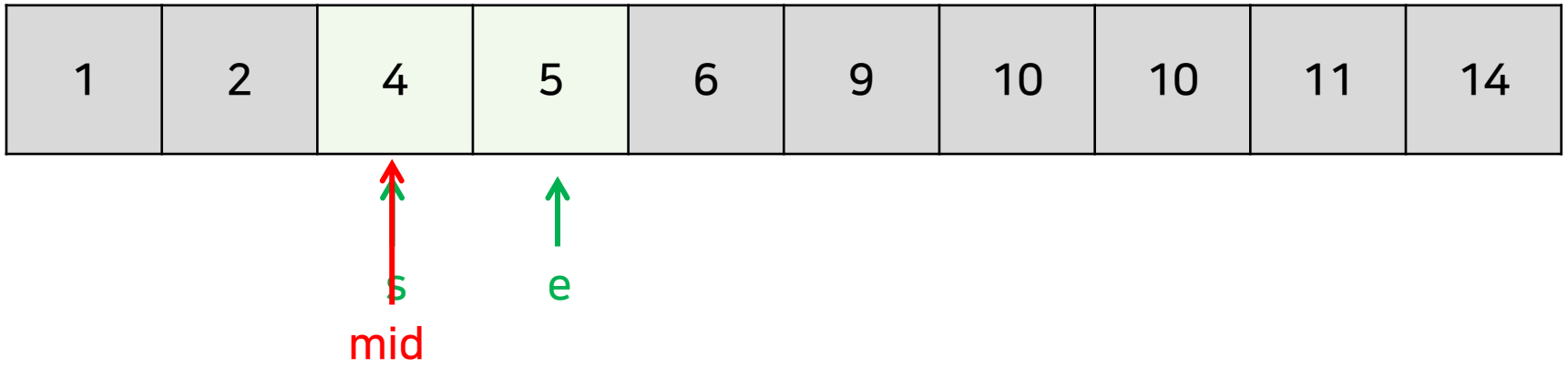


s = 3
e = 4
mid = (s + e)/2 =

ans = 2

이분 탐색

Q. '5'의 위치



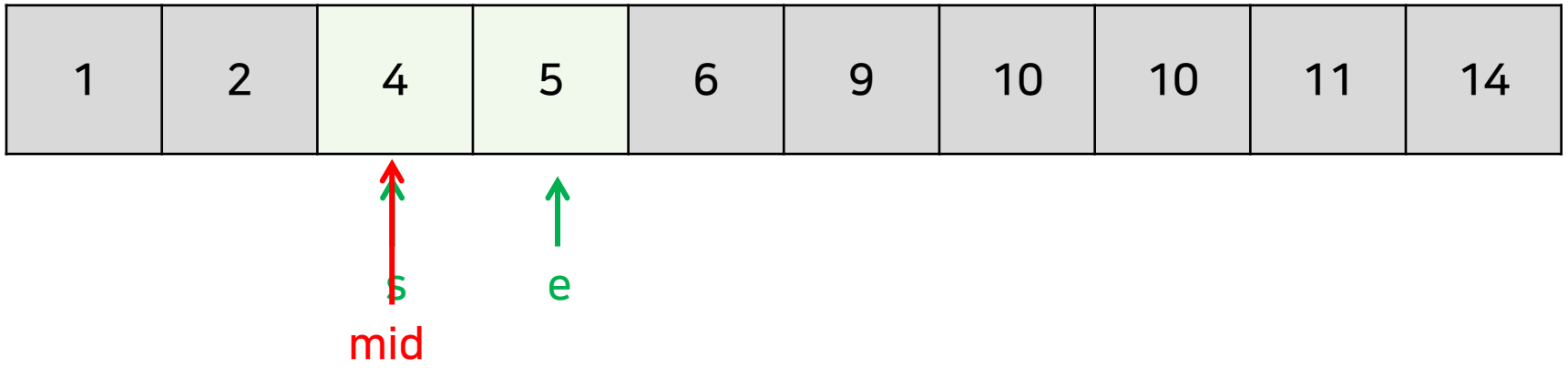
s = 3
e = 4
mid = (s + e)/2 = 3

ans = 2

A[mid] ≤ 5

이분 탐색

Q. '5'의 위치



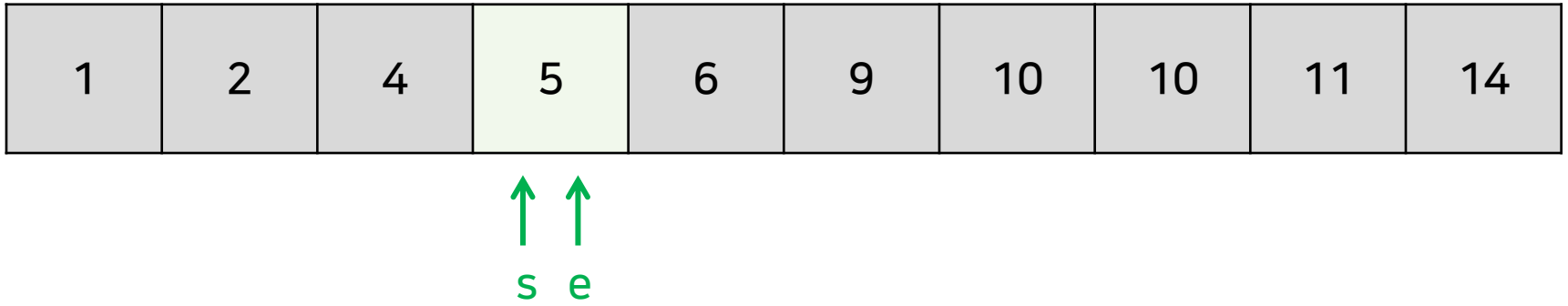
s = 3
e = 4
mid = (s + e)/2 = 3

ans = 3

A[mid] ≤ 5

이분 탐색

Q. '5'의 위치

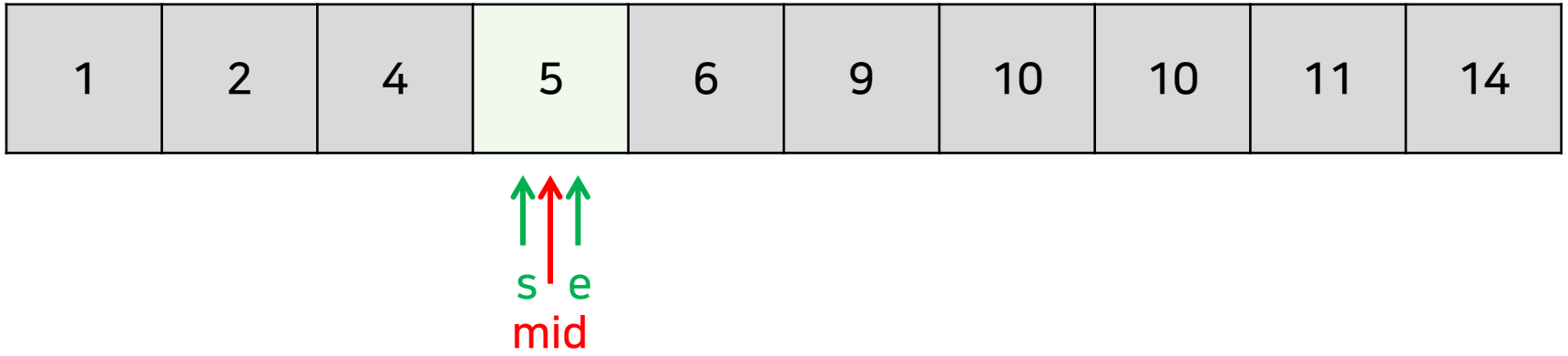


s = 4
e = 4
mid = (s + e)/2 =

ans = 3

이분 탐색

Q. '5'의 위치



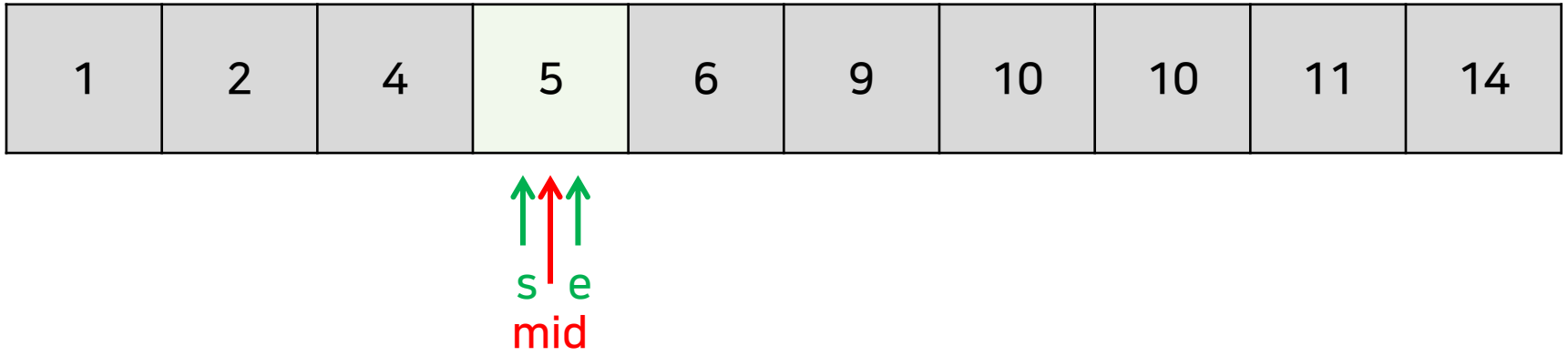
s = 4
e = 4
mid = (s + e)/2 = 4

ans = 3

A[mid] ≤ 5

이분 탐색

Q. '5'의 위치



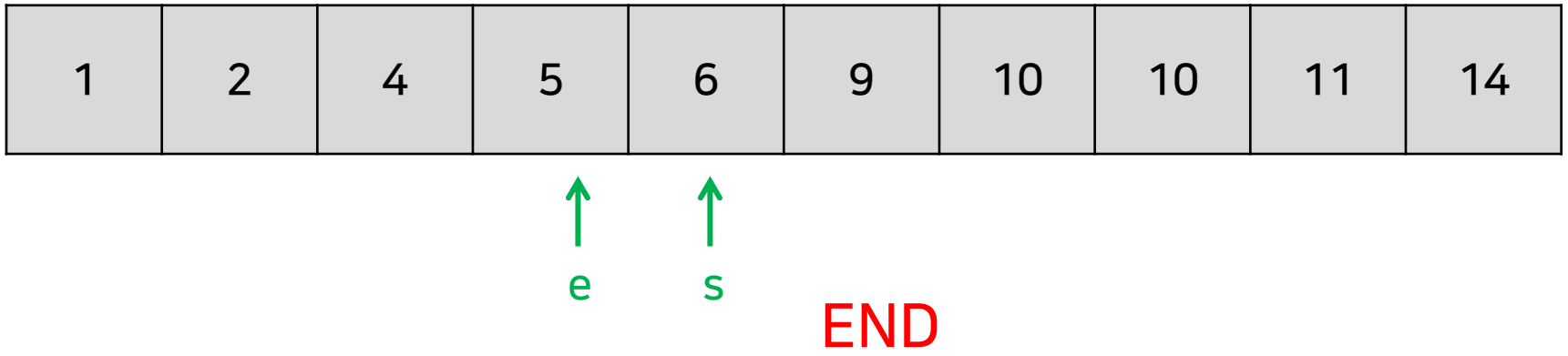
s = 4
e = 4
mid = (s + e)/2 = 4

ans = 4

A[mid] ≤ 5

이분 탐색

Q. '5'의 위치



s = 5
e = 4
mid = (s + e)/2 =

ans = 4

이분 탐색

[Source Code]

```
int s = 1;
int e = n;
while (s <= e) {
    int mid = (s + e) / 2;
    if (A[mid] >= k) {
        ans = min(ans, mid);
        e = mid - 1;
    }
    else {
        s = mid + 1;
    }
}
```

Q. 원하는 값이 존재하지 않는 경우는?

```
int s = 1;
int e = n;
while (s <= e) {
    int mid = (s + e) / 2;
    if (A[mid] > k) {
        e = mid - 1;
    }
    else if (A[mid] < k) {
        s = mid + 1;
    }
    else {
        ans = mid;
        break;
    }
}
```

O(logn)

BOJ 10815 (숫자 카드)

문제

숫자 카드는 정수 하나가 적혀져 있는 카드이다. 상근이는 숫자 카드 N개를 가지고 있다. 정수 M개가 주어졌을 때, 이 수가 적혀있는 숫자 카드를 상근이가 가지고 있는지 아닌지를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 상근이가 가지고 있는 숫자 카드의 개수 N(1 ≤ N ≤ 500,000)이 주어진다. 둘째 줄에는 숫자 카드에 적혀있는 정수가 주어진다. 숫자 카드에 적혀있는 수는 -10,000,000보다 크거나 같고, 10,000,000보다 작거나 같다. 두 숫자 카드에 같은 수가 적혀있는 경우는 없다.

셋째 줄에는 M(1 ≤ M ≤ 500,000)이 주어진다. 넷째 줄에는 상근이가 가지고 있는 숫자 카드인지 아닌지를 구해야 할 M개의 정수가 주어지며, 이 수는 공백으로 구분되어져 있다. 이 수도 -10,000,000보다 크거나 같고, 10,000,000보다 작거나 같다

BOJ 10815 (숫자 카드)

- Naive 풀이 $O(N^2)$

- 원하는 값이 존재하는지 찾고 싶다

- 정렬만 되어 있다면 이분탐색(O(logn))으로 해결 가능

- 정렬 + 이분탐색 = O(nlogn) + O(logn) = O(nlogn)

Binary Search in C++

[lower_bound / upper_bound]

```
template< class ForwardIt, class T >
ForwardIt lower_bound( ForwardIt first, ForwardIt last, const T& value );

template< class ForwardIt, class T >
ForwardIt upper_bound( ForwardIt first, ForwardIt last, const T& value );
```

[binary_search]

```
template< class ForwardIt, class T >
bool binary_search( ForwardIt first, ForwardIt last, const T& value );
```

Binary Search in C++

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

int main(void) {
    vector<int> v = { 1, 2, 4, 8, 9, 10 };
    vector<int>::iterator it;
    it = lower_bound(v.begin(), v.end(), 3);
    cout << *it; //4
    it = lower_bound(v.begin(), v.end(), 12);
    cout << *it; //ERROR (it == v.end())
    it = upper_bound(v.begin(), v.end(), 4);
    cout << *it; //8
    int x = lower_bound(v.begin(), v.end(), 3) - v.begin();
    cout << x; //2
    cout << binary_search(v.begin(), v.end(), 5); // 0 (False)
}
```

BOJ 2805 (나무 자르기)

문제

상근이는 나무 M미터가 필요하다. 근처에 나무를 구입할 곳이 모두 망해버렸기 때문에, 정부에 벌목 허가를 요청했다. 정부는 상근이네 집 근처의 나무 한 줄에 대한 벌목 허가를 내주었고, 상근이는 새로 구입한 목재절단기를 이용해서 나무를 구할것이다.

목재절단기는 다음과 같이 동작한다. 먼저, 상근이는 절단기에 높이 H를 지정해야 한다. 높이를 지정하면 톱날이 땅으로부터 H미터 위로 올라간다. 그 다음, 한 줄에 연속해있는 나무를 모두 절단해버린다. 따라서, 높이가 H보다 큰 나무는 H 위의 부분이 잘릴 것이고, 낮은 나무는 잘리지 않을 것이다. 예를 들어, 한 줄에 연속해있는 나무의 높이가 20, 15, 10, 17이라고 하자. 상근이가 높이를 15로 지정했다면, 나무를 자른 뒤의 높이는 15, 15, 10, 15가 될 것이고, 상근이는 길이가 5인 나무와 2인 나무를 들고 집에 갈 것이다. (총 7미터를 집에 들고 간다) 절단기에 설정할 수 있는 높이는 양의 정수 또는 0이다.

상근이는 환경에 매우 관심이 많기 때문에, 나무를 필요한 만큼만 집으로 가져가려고 한다. 이때, 적어도 M미터의 나무를 집에 가져가기 위해서 절단기에 설정할 수 있는 높이의 최댓값을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 나무의 수 N과 상근이가 집으로 가져가려고 하는 나무의 길이 M이 주어진다. ($1 \le N \le 1,000,000$, $1 \le M \le 2,000,000,000$)

둘째 줄에는 나무의 높이가 주어진다. 나무의 높이의 합은 항상 M보다 크거나 같기 때문에, 상근이는 집에 필요한 나무를 항상 가져갈 수 있다. 높이는 1,000,000,000보다 작거나 같은 양의 정수 또는 0이다.

BOJ 2805 (나무 자르기)

- Naive 풀이

- N = 100만, H (나무 높이) = 10억

- 시간 복잡도 = O(NH) TLE

BOJ 2805 (나무 자르기)

<아이디어>

- 특정 높이(k)를 선택했을 때 M 이상을 얻을 수 있다면 k 미만의 높이도 모두 가능하다

- k가 낮아질수록 얻는 나무의 양 ↑ / 높아질수록 ↓

= 내림차순 정렬된 상태

→ 파라메트릭 서치 (Parametric Search)

Parametric Search

❖ 파라메트릭 서치(Parametric Search)란?

- 최적화 문제(최대, 최소)를 결정 문제(참, 거짓)로 바꿔 푸는 기법

ex) 나무 자르기

Q. 적어도 M미터의 나무를 집에 가져가기 위한 절단기의 높이의 최댓값은? (최적화 문제)

Q. 절단기의 높이가 K일 때 적어도 M미터의 나무를 집에 가져갈 수 있는가? (결정 문제)

Parametric Search

❖ 파라메트릭 서치(Parametric Search)란?

- 주어진 상황을 매개변수화 시켜 정의한 함수에 대입

- 이분탐색과 유사

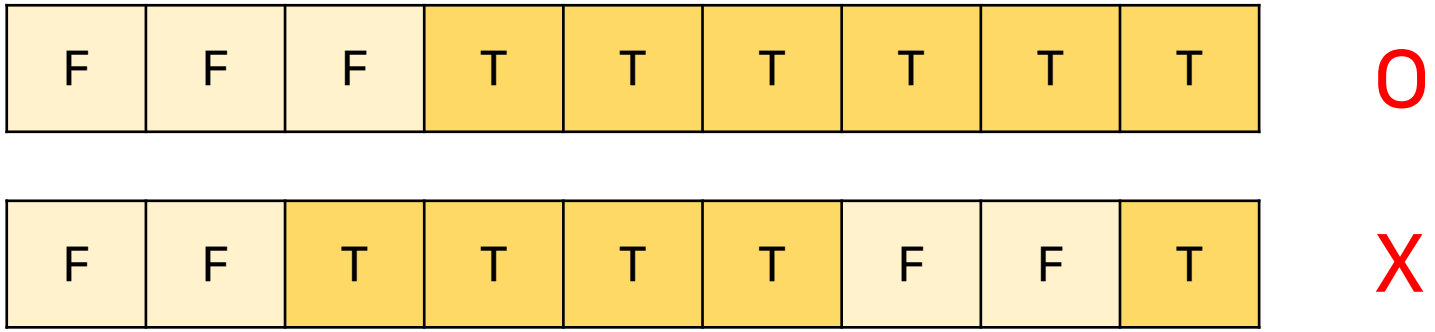
차이점
- 이분탐색 : 원하는 값이 존재하는가? 존재한다면 어디 있는가?
- 파라메트릭 서치 : 조건을 만족하는 값 중 최소/최대가 어디 있는가?

Parametric Search

<주의할 점>

- 최대 or 최소를 구하는 형태여야 한다

- 단조성이 있어야 한다



BOJ 2805 (나무 자르기)

```
int s = 0;
int e = 1000000000;
int ans = 0;
while (s <= e) {
    int mid = (s + e) / 2;
    if (sol(mid)) {
        ans = max(ans, mid);
        s = mid + 1;
    }
    else e = mid - 1;
}
cout << ans;
```

```
bool sol(int mid) {
    long long sum = 0;
    for (int i = 1; i <= n; i++) {
        sum += max(0, h[i] - mid);
    }
    if (sum >= m) return true;
    else return false;
}
```

BOJ 2343 (기타 레슨)

문제

강토는 자신의 기타 레슨 동영상을 블루레이로 만들어 판매하려고 한다. 블루레이에는 총 N개의 레슨이 들어가는데, 블루레이를 녹화할 때, 레슨의 순서가 바뀌면 안 된다. 순서가 뒤바뀌는 경우에는 레슨의 흐름이 끊겨, 학생들이 대혼란에 빠질 수 있기 때문이다. 즉, i번 레슨과 j번 레슨을 같은 블루레이에 녹화하려면 i와 j 사이의 모든 레슨도 같은 블루레이에 녹화해야 한다.

강토는 이 블루레이가 얼마나 팔릴지 아직 알 수 없기 때문에, 블루레이의 개수를 가급적 줄이려고 한다. 오랜 고민 끝에 강토는 M개의 블루레이에 모든 기타 레슨 동영상을 녹화하기로 했다. 이때, 블루레이의 크기(녹화 가능한 길이)를 최소로 하려고 한다. 단, M개의 블루레이는 모두 같은 크기이어야 한다.

강토의 각 레슨의 길이가 분 단위(자연수)로 주어진다. 이때, 가능한 블루레이의 크기 중 최소를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 레슨의 수 N ($1 \le N \le 100,000$)과 M ($1 \le M \le N$)이 주어진다. 다음 줄에는 강토의 기타 레슨의 길이가 레슨 순서대로 분 단위로(자연수)로 주어진다. 각 레슨의 길이는 10,000분을 넘지 않는다.

BOJ 2343 (기타 레슨)

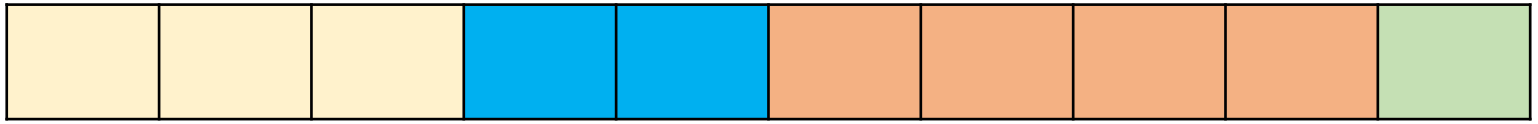
- Naive 풀이

- N = 100,000개의 레슨, M(≤ N)개의 블루레이

- 레슨의 순서가 바뀌면 안됨 = 반드시 연속되도록

→ N개의 레슨을 M개의 연속된 구간으로 나누는 모든 경우 중 최소

ex) N = 10, M = 4



BOJ 2343 (기타 레슨)

- 블루레이의 크기를 배열의 인덱스처럼 생각해보자!

- 블루레이의 특정 크기(k)에 대해서 레슨이 모두 저장되는가?

  - 반드시 연속되어야 하는 조건 → 순서대로 넣어보면서 확인

- 저장된다면, k보다 큰 크기에 대해서도 항상 가능 (오름차순 형태)

→ Parametric Search

필수 / 연습문제

[필수문제]

[BOJ 1725] 히스토그램

[BOJ 2343] 기타 레슨

[BOJ 1074] Z

[BOJ 1493] 박스 채우기

[BOJ 18113] 그르다 김가놈

[BOJ 1477] 휴게소 세우기

[연습문제]

[BOJ 1030] 프렉탈 평면

★ [BOJ 14637] Need for Speed

[BOJ 3079] 입국 심사

[BOJ 12846] 무서운 아르바이트

[BOJ 5904] Moo 게임

[BOJ 1300] K번째 수

[BOJ 2110] 공유기 설치