Chapter. 02

알고리즘

이분 탐색 응용편
Parametric Search

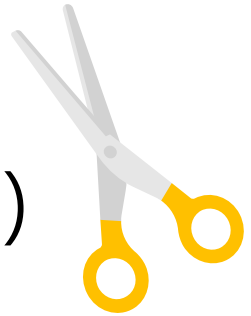
FAST CAMPUS
ONLINE
알고리즘 공채 대비반!

강사. 류호석

Life Changing Education

Chapter. 02

알고리즘
매개 변수 탐색 (Parametric Search)



FAST CAMPUS
ONLINE
류호석 강사.

I 이분 탐색(Binary Search)

이분 탐색(Binary Search)

무엇: 정렬이 보장되는 배열에서 기준 x 를 가지고 범위를 **"이분"** 하면서 탐색하는 방법

$\Rightarrow$ **시간 복잡도: $O(\log N)$**

***X* = 63**	10	11	18	19	38	58	72	87	92

I 매개 변수 탐색(Parametric Search)

매개 변수 탐색(Parametric Search) ← 이분 탐색의 아이디어!

배열이 0 과 1 만 존재하며 오름차순 인건 보장되지만, 전체 배열은 모른다.

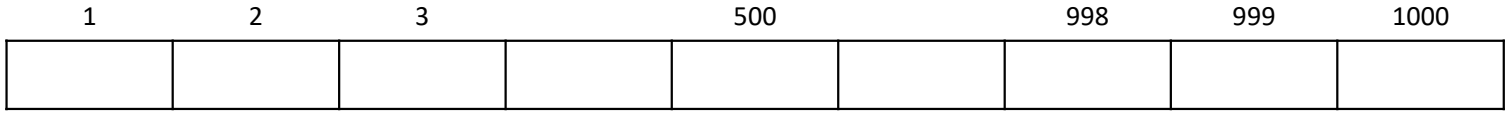
특정 인덱스의 값을 $O(T)$에 계산 가능할 때, 여기서 0 과 1 의 경계를 찾아야 한다면?

	1	2	3				998	999	1000
***X* = 1**	?	?	?		...		?	?	?

예) Up-Down 게임!

- A 가 1 ~ 1000 사이의 어떤 자연수를 선택
- B 는 A 한테 “생각한 숫자가 X 이상이야?” 라는 질문 가능
- A는 맞으면 1, 아니면 0 이라고 대답
- 최소 횟수로 질문하려면?

I 매개 변수 탐색(Parametric Search)



1부터 1,000까지 전부

“생각한 숫자가 1 이상이야?”

“생각한 숫자가 2 이상이야?”

“생각한 숫자가 3 이상이야?”

…

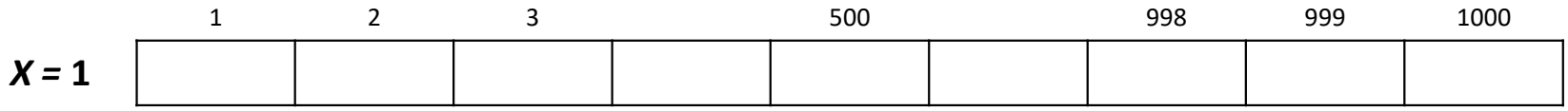
“생각한 숫자가 1000 이상이야?”

➔ 마지막 Yes인 대답이 정답!

[L=1, R=1000] 에서 이분 탐색!

➔ 정답이 가능한 구간 [L, R]을 좁혀 나가기!

| 매개 변수 탐색(Parametric Search)



1부터 1,000까지 전부

"생각한 숫자가 1 이상이야?"

"생각한 숫자가 2 이상이야?"

"생각한 숫자가 3 이상이야?"

...

"생각한 숫자가 1000 이상이야?"

$O(T \times 1000)$

➔ 처음 Yes인 대답이 정답!

[L=1, R=1000] 에서 이분 탐색!

$O(T \times \log 1000)$
$= O(T \times 10)$

➔ 정답이 가능한 구간 [L, R]을 좁혀 나가기!

I 매개 변수 탐색(Parametric Search)

<핵심>

1. 정답을 **"매개 변수(Parameter)"**로 만들고 **Yes/No 문제(결정 문제)**로 바꿔 보기
2. 모든 값에 대해서 Yes/No 를 채웠다고 생각했을 때, **정렬된 상태인가**?
3. Yes/No 결정하는 문제를 **풀기**!

문제를 거꾸로 푸는 것이기 때문에 통찰력을 요구합니다.

최근 코테에 굉장히 빈도 높게 나오기 때문에 중요하며, 훈련이 많이 필요한 알고리즘입니다.

I 매개 변수 탐색(Parametric Search) – 자주 하는 실수

1위 매개 변수에 대한 결정이 Nooooooooo Yesssssssss 꼴이 아닌데 이분 탐색 하는 경우!

2위 L, R, M, Result 변수의 정의를 헷갈려서 부등호 등을 잘못 쓰는 경우!

3위 L, R 범위를 잘못 설정하거나 Result의 초기값을 잘못 설정하는 경우!

I 매개 변수 탐색(Parametric Search) – 꿀팁

<키워드>

~~의 **최댓값**을 구하시오

~~의 **최솟값**을 구하시오

=> Parametric Search 접근을 시도해볼 가치가 있다!

I 매개 변수 탐색(Parametric Search) – 많은 연습 필수!!!

- BOJ 2805 – 나무 자르기
- BOJ 1654 – 랜선 자르기
- BOJ 2512 – 예산
- BOJ 2110 – 공유기 설치
- BOJ 2343 – 기타 레슨
- BOJ 6236 – 용돈 관리
- BOJ 13702 – 이상한 술집
- BOJ 17266 – 어두운 굴다리

- BOJ 1300 – K 번째 수 (고난이도)
- BOJ 1637 – 날카로운 눈 (고난이도)

BOJ 2805 – 나무 자르기

난이도: 2

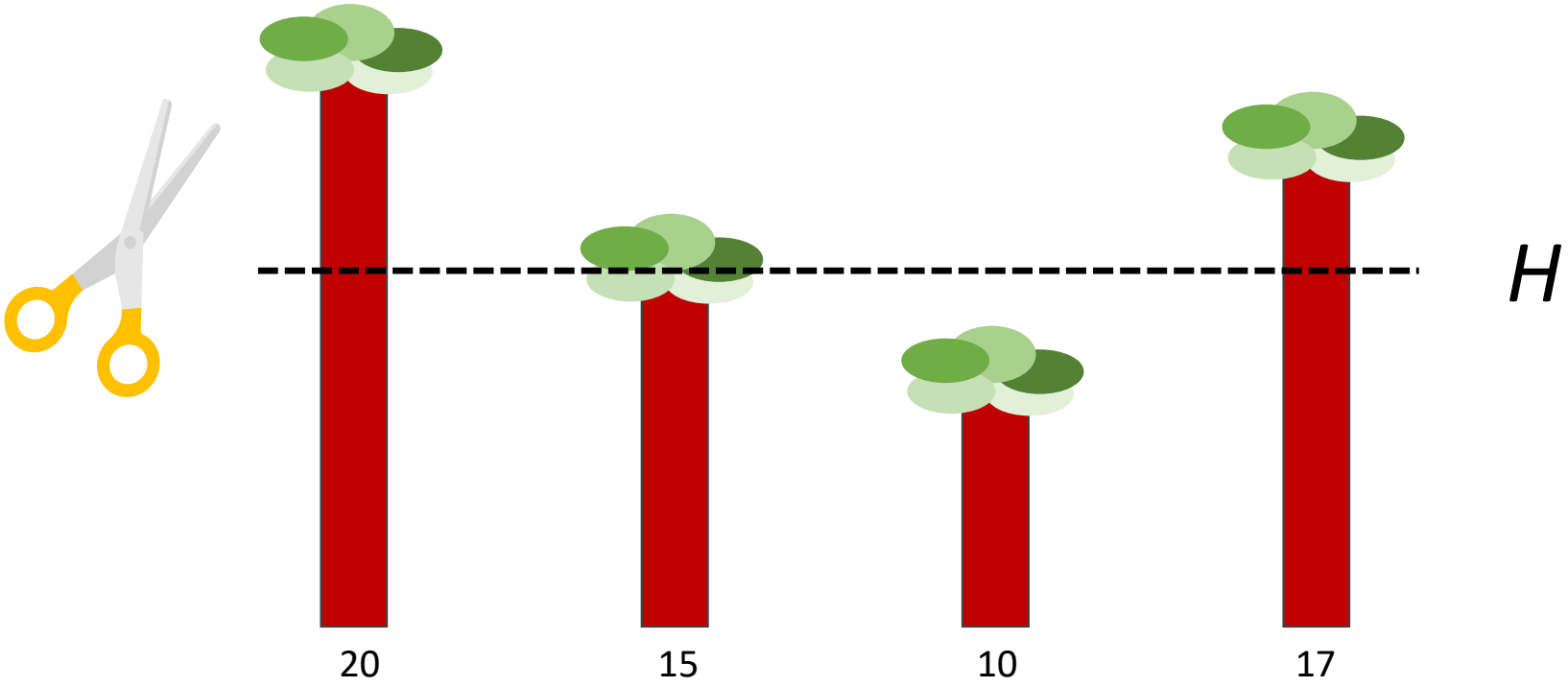
$1 \leq N \leq 100$만

$1 \leq M \leq 20$억

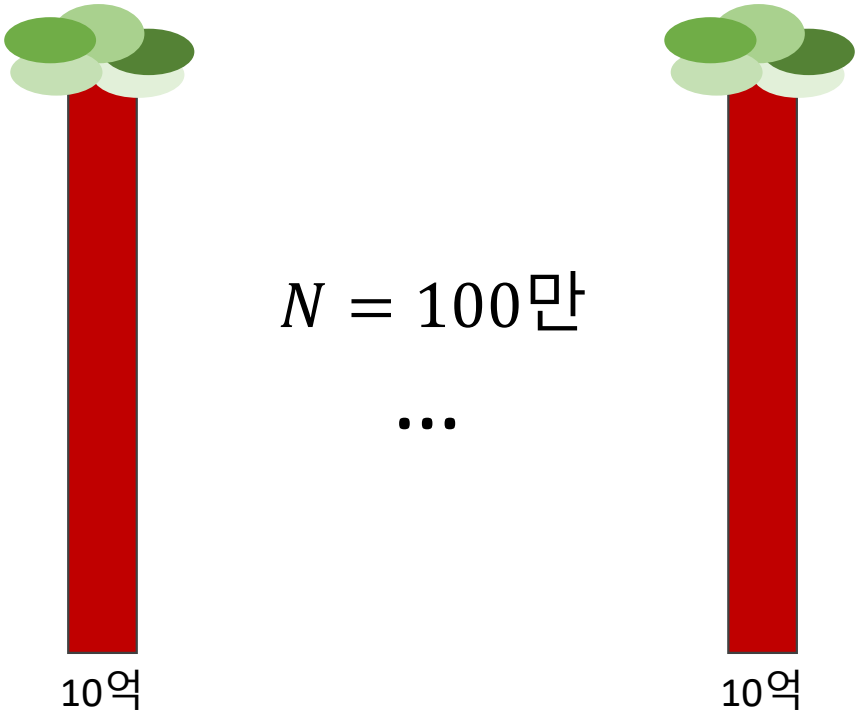
$0 \leq$ 각 나무 높이 ≤ 10억

나무 개수 $N = 4$

필요한 나무의 길이 $M = 7$



I 문제 파악하기 – 정답의 최대치



나무 개수 $N = 100$만
필요한 나무의 길이 $M = 20$억

1. 정답의 범위: $0 \sim 10$억

2. 잘린 나무의 길이 합
   ≤ 나무 높이의 총합
   = 100만×10억 = 10^{15}

즉, 계산 과정 중의 변수 타입은 *long* 으로!

I 문제 파악하기 – **키워드** 체크하기

적어도 M미터의 나무를 집에 가져가기 위해서 절단기에

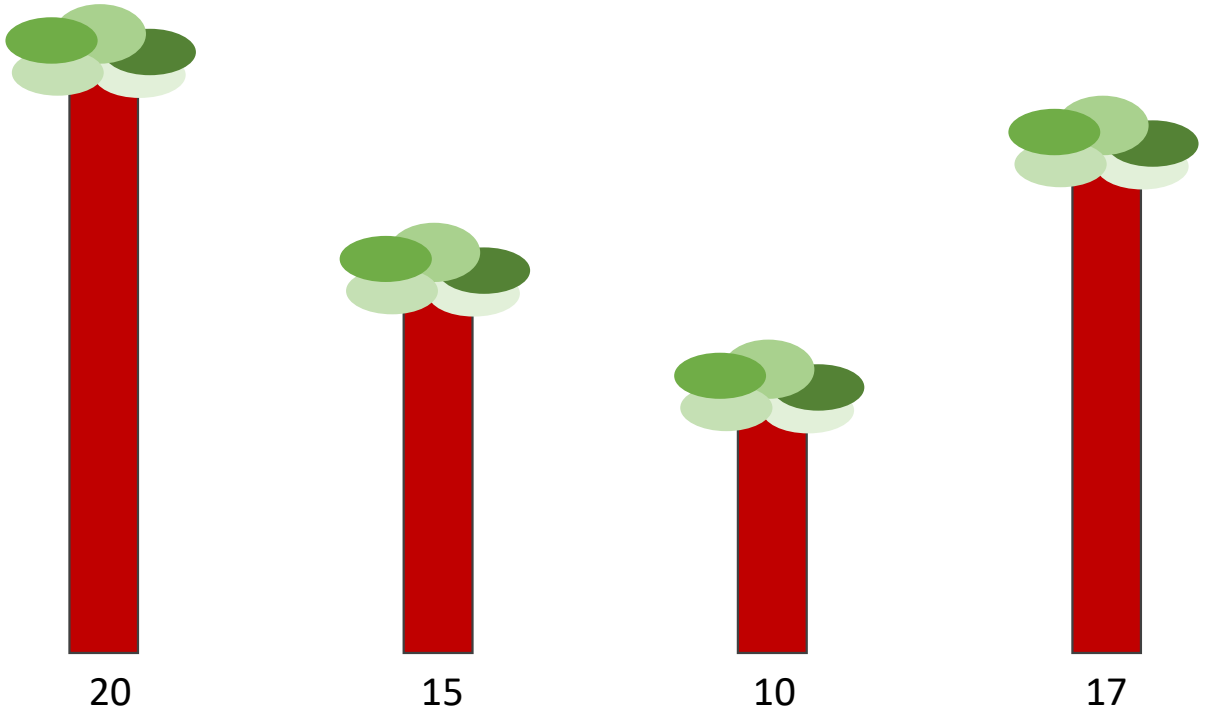
설정할 수 있는 높이의 최댓값을 구하는 프로그램을 작성하시오.

I 접근 – 매개 변수 만들어 보기

원래 문제: 원하는 길이 M 만큼을 얻을 수 있는 최대 높이(H_{ans})는 얼마인가?

뒤집은 문제: 어떤 높이(H) 로 잘랐을 때, 원하는 길이 M 만큼을 얻을 수 있는가? **Yes/No**

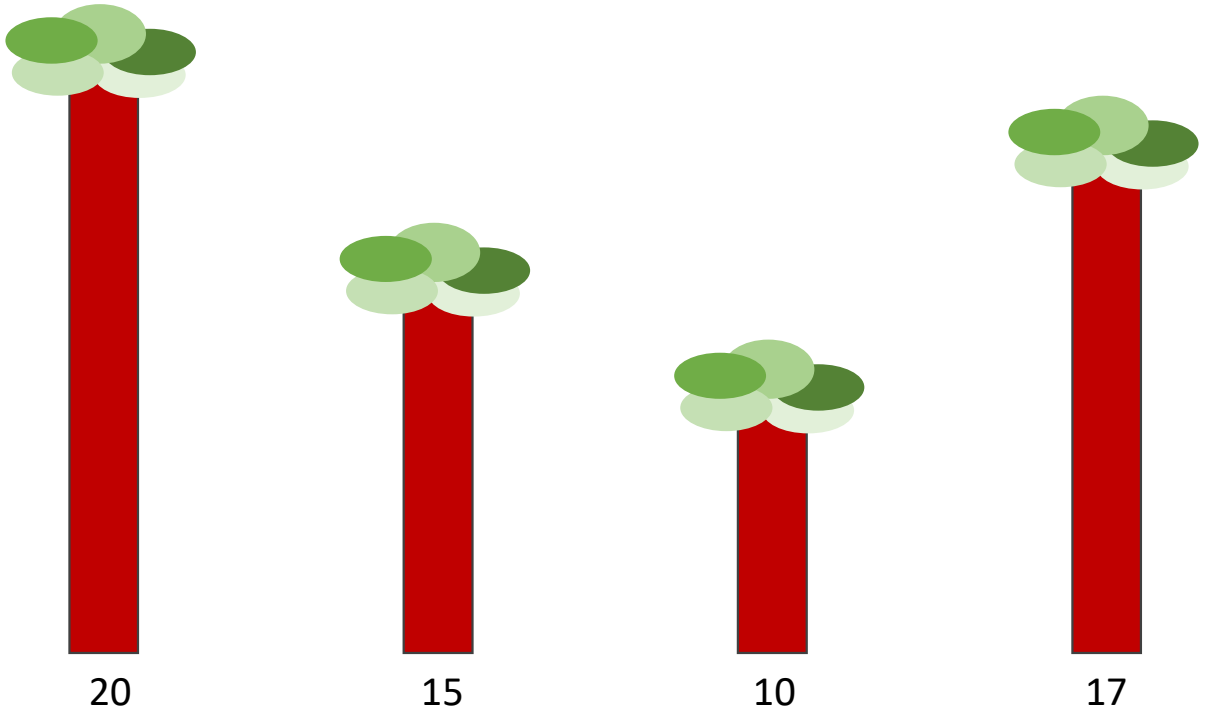
Easier?



I 접근 – 매개 변수 만들어 보기

뒤집은 문제: 어떤 높이(H)로 잘랐을 때, 원하는 길이 M 만큼을 얻을 수 있는가? **Yes/No**

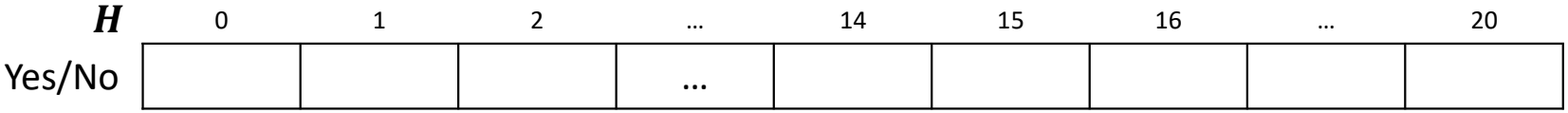
시간 복잡도: $O(N)$



I 접근 – 뒤집은 문제를 써먹기

뒤집은 문제: 어떤 높이(H) 로 잘랐을 때, 원하는 길이 M 만큼을 얻을 수 있는가? **Yes/No**

나무 길이: [20, 15, 10, 17]



<핵심>

1. 정답을 “매개 변수(Parameter)”로 만들고 Yes/No 문제(결정 문제)로 바꿔 보기
2. 모든 값에 대해서 Yes/No 를 채웠다고 생각했을 때, 정렬된 상태인가?
3. Yes/No 결정하는 문제를 풀기!

I 접근 – 뒤집은 문제를 써먹기

뒤집은 문제: 어떤 높이(H) 로 잘랐을 때, 원하는 길이 M 만큼을 얻을 수 있는가? **Yes/No**

시간 복잡도: $O(N)$

나무 길이: [20, 15, 10, 17]

H	0	1	2	…	14	15	16	…	20
Yes/No	?	?	?	…	?	?	?	?	?

원래 문제: 뒤집은 문제를 모든 H 마다 (0~20억) 해보면 마지막 Yes가 정답

시간 복잡도: $O(\text{뒤집은 문제} \times \log 20\text{억}) = O(N \times \log X) \cong N \times 31$

I 시간, 공간 복잡도 계산하기

1. H를 정해서 결정 문제 한 번 풀기 => $O(N)$

2. 정답의 범위를 이분 탐색하면서 풀기
   => $O(\log X)$ 번 반복할 것

3. 총 시간 복잡도: $O(N \log X)$

I 구현


```
static boolean determination(int H) {
    // H 높이로 나무들을 잘랐을 때, M 만큼을 얻을 수 있으면 true, 없으면 false를 return하는 함수
}

static void pro() {
    long L = 0, R = 2000000000, ans = 0;
    // [L ... R] 범위 안에 정답이 존재한다!
    // 이분 탐색과 determination 문제를 이용해서 answer를 빠르게 구하자!
    System.out.println(ans);
}
```

I 연습 문제

- BOJ 1654 – 랜선 자르기
- BOJ 2512 – 예산

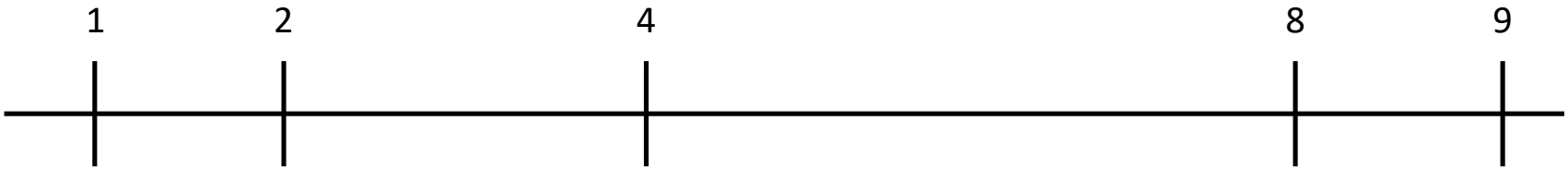
이외의 추천 문제가 추가되면 Github 자료에 코드 업로드

BOJ 2110 – 공유기 설치

난이도: 3

$2 \leq N \leq 200{,}000$
$2 \leq C \leq N$
$1 \leq$ 좌표 $x_i \leq 10$억

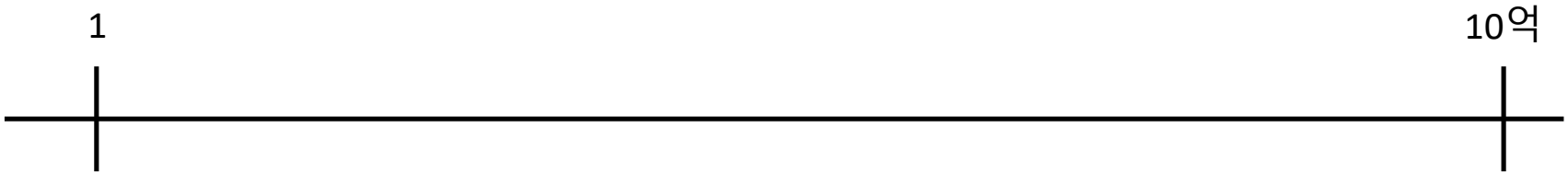
집의 개수 $N = 5$
공유기의 개수 $C = 3$



- C개의 공유기를 몇몇 집에 설치
- 인접한 공유기 사이의 거리 최대화하기

I 문제 파악하기 – 정답의 최대치

집의 개수 $N = 2$
공유기의 개수 $C = 2$



제일 멀리 설치 해보면 정답은 10억 이하 => Integer

I 문제 파악하기 – **키워드** 체크하기

C개의 공유기를 N개의 집에 적당히 설치해서,

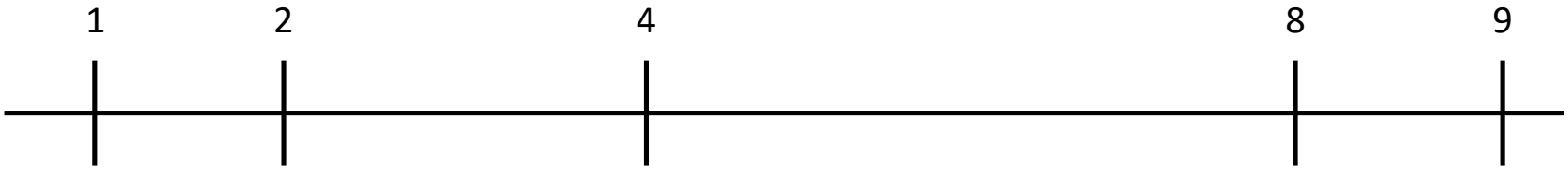
가장 인접한 두 공유기 사이의 거리를 최대로 하는 프로그램을 작성하시오.

I 접근 – 매개 변수 만들어 보기

원래 문제: C개의 공유기를 설치했을 때, 최대 인접 거리(D_{ans})는 얼마인가?

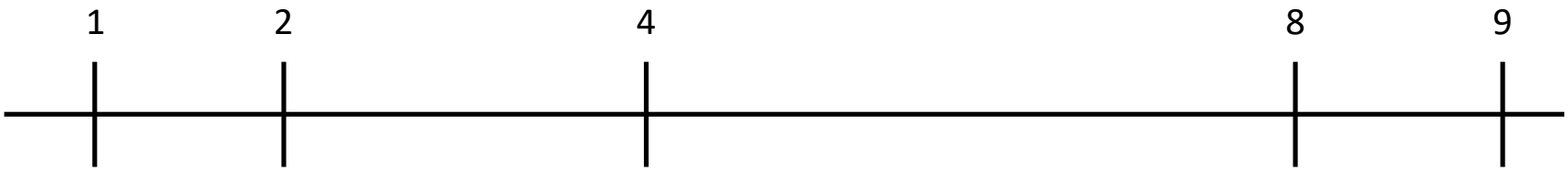
뒤집은 문제: 어떤 거리(D) 만큼은 거리를 둘 때, 공유기 C개를 설치할 수 있는가? **Yes/No**

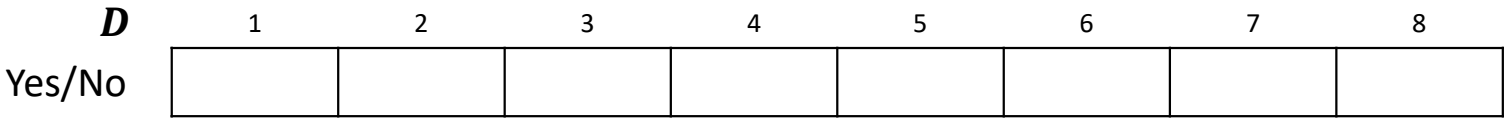
Easier?



I 접근 – 뒤집은 문제를 써먹기

뒤집은 문제: 어떤 거리(D) 만큼은 거리를 둘 때, 공유기 C개를 설치할 수 있는가? **Yes/No**





어떤 거리(D) 만큼은 거리를 둘 때, 왼쪽 집부터 되는 대로 전부 설치해보기!

I 접근 – 뒤집은 문제를 써먹기

뒤집은 문제: 어떤 거리(D) 만큼은 거리를 둘 때, 공유기 C개를 설치할 수 있는가? **Yes/No**

시간 복잡도: $O(N)$

D	1	2	3	4	5	6	7	8
Yes/No								

원래 문제: 뒤집은 문제를 모든 D 마다 (1~10억) 해보면 마지막 Yes가 정답

시간 복잡도: $O(\text{뒤집은 문제} \times \log 10\text{억}) = O(N \times \log X) \cong N \times 30$

I 시간, 공간 복잡도 계산하기

1. 주어진 집들을 정렬하기 => $O(N \log N)$
2. D를 정해서 결정 문제 한 번 풀기 => $O(N)$
3. 정답의 범위를 이분 탐색하면서 풀기
   => $O(\log X)$ 번 반복할 것
4. 총 시간 복잡도: $O(N \log N + N \log X)$

I 구현

```
static boolean determination(int D) {
    // D 만큼의 거리 차이를 둔다면 C 개 만큼의 공유기를 설치할 수 있는가?

    // 제일 왼쪽 집부터 가능한 많이 설치해보자!
    // D 만큼의 거리를 두면서 최대로 설치한 개수와 C 를 비교하자.
    int cnt = 1, last = A[1];
    return cnt >= C;
}

static void pro() {
    // determination을 빠르게 하기 위해서 정렬해주자.

    int L = 1, R = 1000000000, ans = 0;
    // [L ... R] 범위 안에 정답이 존재한다!
    // 이분 탐색과 determination 문제를 이용해서 answer를 빠르게 구하자!
    System.out.println(ans);
}
```

I 연습 문제

- BOJ 2343 – 기타 레슨
- BOJ 6236 – 용돈 관리

이외의 추천 문제가 추가되면 Github 자료에 코드 업로드