

JavaScript

이진 탐색 알고리즘

이진 탐색 문제 풀이 ①

이진 탐색 문제 풀이 ① | 코딩 테스트에서 자주 등장하는 이진 탐색 알고리즘 이해하기

강사 나동빈

JavaScript

이진 탐색 알고리즘

이진 탐색 문제 풀이 ①

JavaScript 이진 탐색
이진 탐색 문제 풀이 ①

혼자 힘으로 풀어보기

JavaScript
이진 탐색
이진 탐색
문제 풀이 ①

문제 제목: 예산

문제 난이도: ★★☆☆☆

문제 유형: 이진 탐색, 파라메트릭 서치

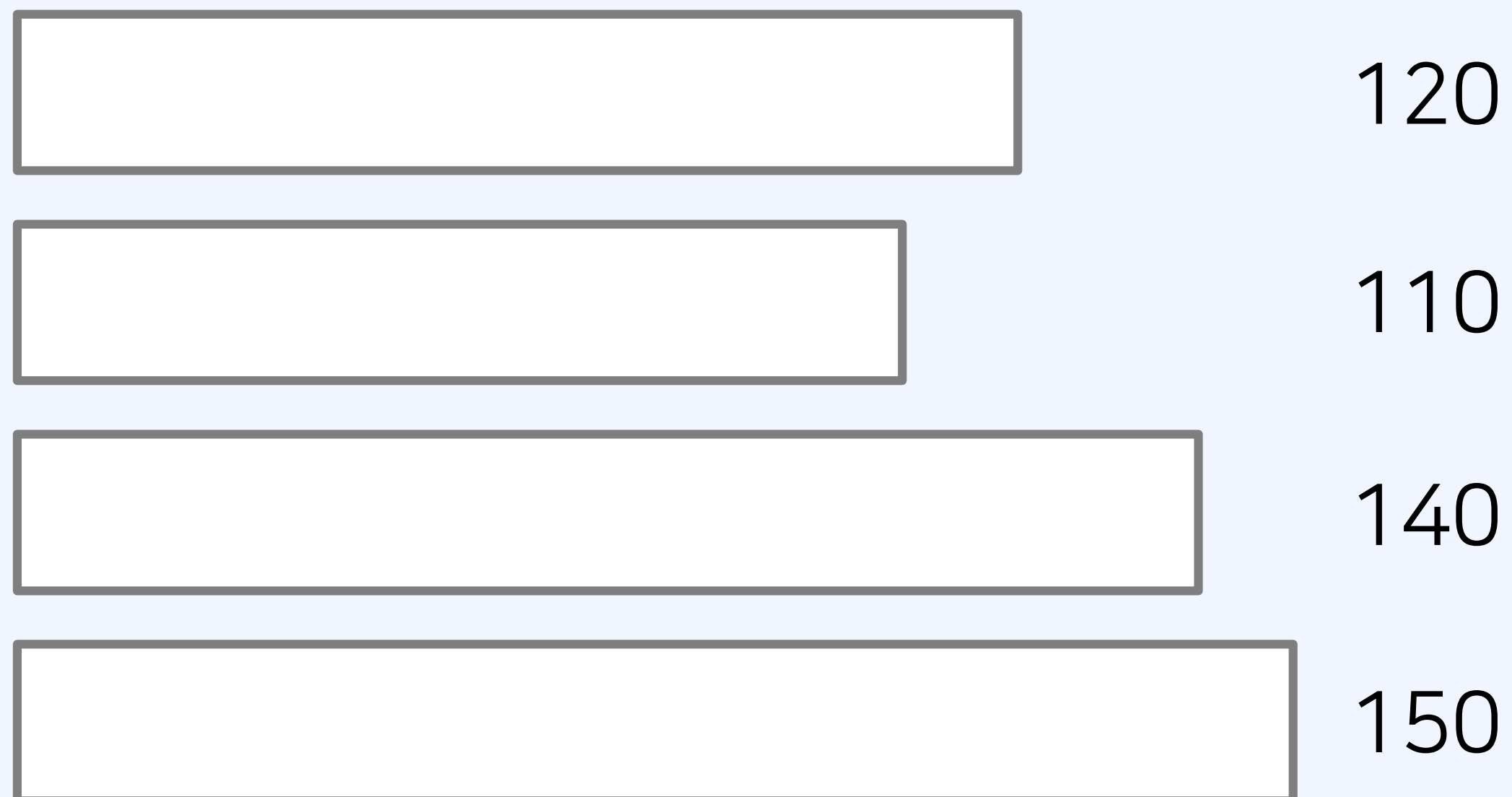
추천 풀이 시간: 30분

JavaScript 이진 탐색
이진 탐색 문제 풀이 ①

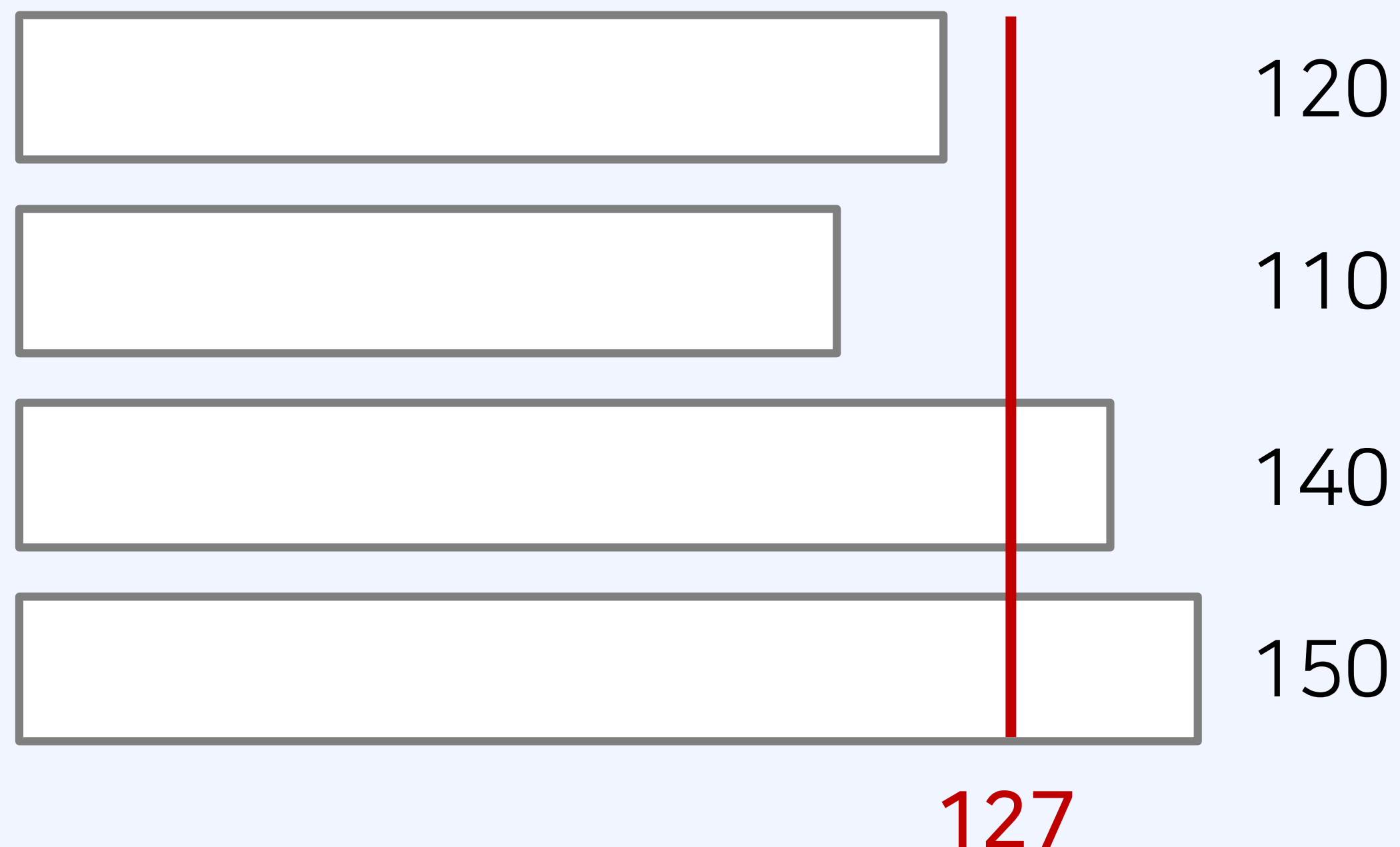
문제 해결 아이디어

JavaScript
이진 탐색
이진 탐색
문제 풀이 ①

- 문제 요구사항: 적절한 **상한 금액**을 찾는 것이 문제의 목표다.
- 전체 국가 예산이 485이고, 4개의 지방 예산 요청이 120, 110, 140, 150이라고 하자.

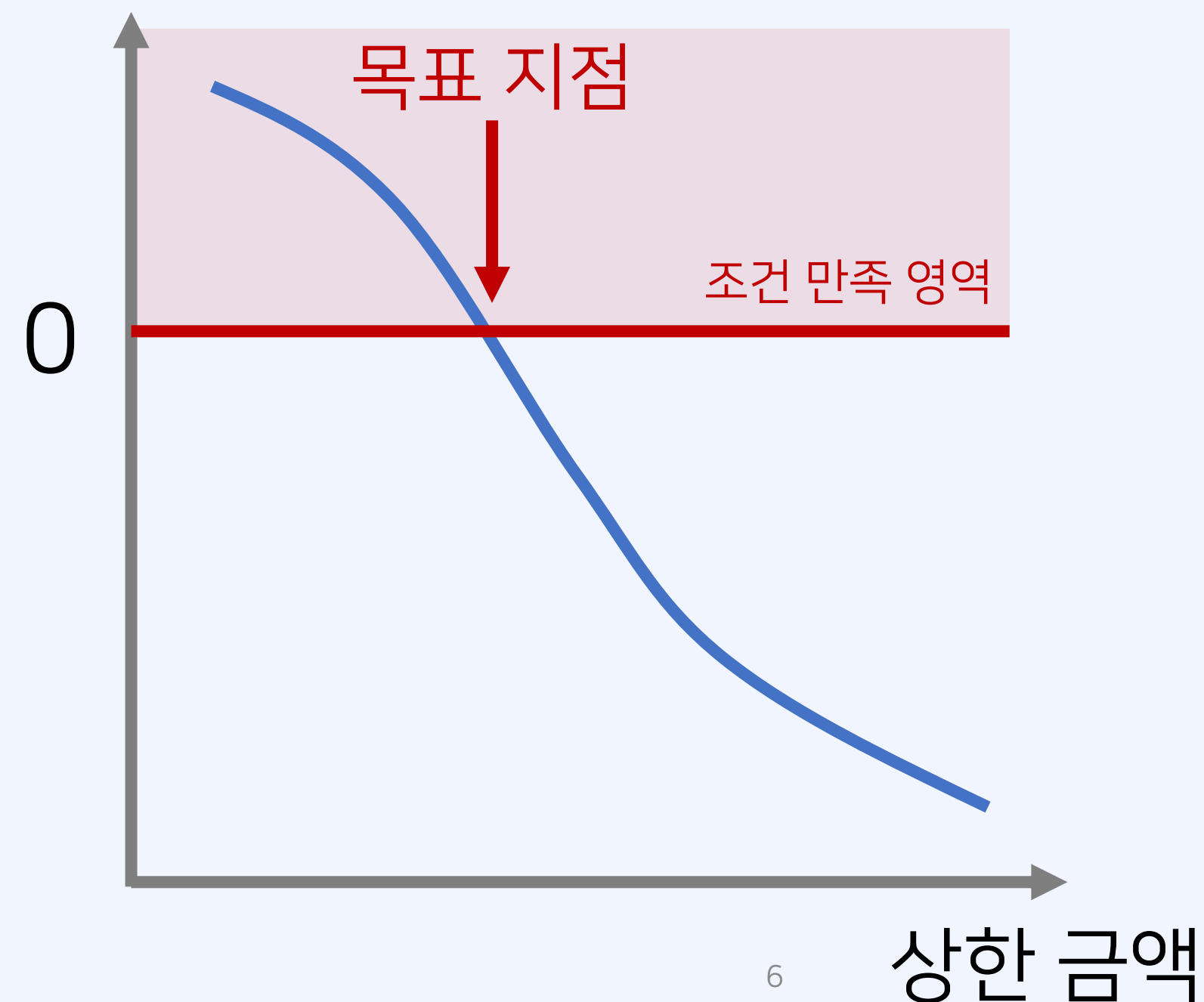


- 문제 요구사항: 적절한 **상한 금액**을 찾는 것이 문제의 목표다.
- 전체 국가 예산이 485이고, 4개의 지방 예산 요청이 120, 110, 140, 150이라고 하자.
- 상한 금액이 127인 경우, 배정 금액의 합이 $120 + 110 + 127 + 127 = 484$ 이다.



- 문제 요구사항: 적절한 **상한 금액**을 찾는 것이 문제의 목표다.
 1. 배정된 총 예산이 조건을 만족한다면, 상한 금액을 증가(최대화가 목표)시킨다.
 2. 배정된 총 예산이 조건을 만족하지 못한다면, 상한 금액을 감소시킨다.

배정을 다 하고도 남은 예산



JavaScript 이진 탐색 이진 탐색 문제 풀이 ①

정답 코드 예시

```
let fs = require('fs');
let input = fs.readFileSync('/dev/stdin').toString().split('\n');

let n = Number(input[0].split(' ')[0]); // 지방의 수(N)
let arr = input[1].split(' ').map(Number); // 각 지방의 예산 요청
let m = Number(input[2]); // 총 예산(M)

let start = 1; // 이진 탐색을 위한 시작점(start)과 끝점(end) 설정
let end = arr.reduce((a, b) => Math.max(a, b));

let result = 0;
while (start <= end) { // 이진 탐색 수행(반복문)
    let mid = parseInt((start + end) / 2); // 현재의 중간점(상한액)
    let total = 0; // 배정된 예산의 총액 계산
    for (x of arr) { // 각 지방에서 요청한 예산을 하나씩 확인하며
        total += Math.min(mid, x); // 예산 배정
    }
    if (total <= m) { // 조건을 만족한다면, 상한액(최대화가 목표)을 증가시키기
        result = mid;
        start = mid + 1;
    }
    else { // 조건을 만족하지 못한다면, 상한액을 감소시키기
        end = mid - 1;
    }
}
console.log(result);
```

JavaScript 이진 탐색

이진 탐색
문제 풀이 ①

JavaScript 이진 탐색
이진 탐색 문제 풀이 ①

혼자 힘으로 풀어보기

JavaScript
이진 탐색
이진 탐색
문제 풀이 ①

문제 제목: 나무 자르기

문제 난이도: ★★☆☆☆

문제 유형: 이진 탐색, 파라메트릭 서치

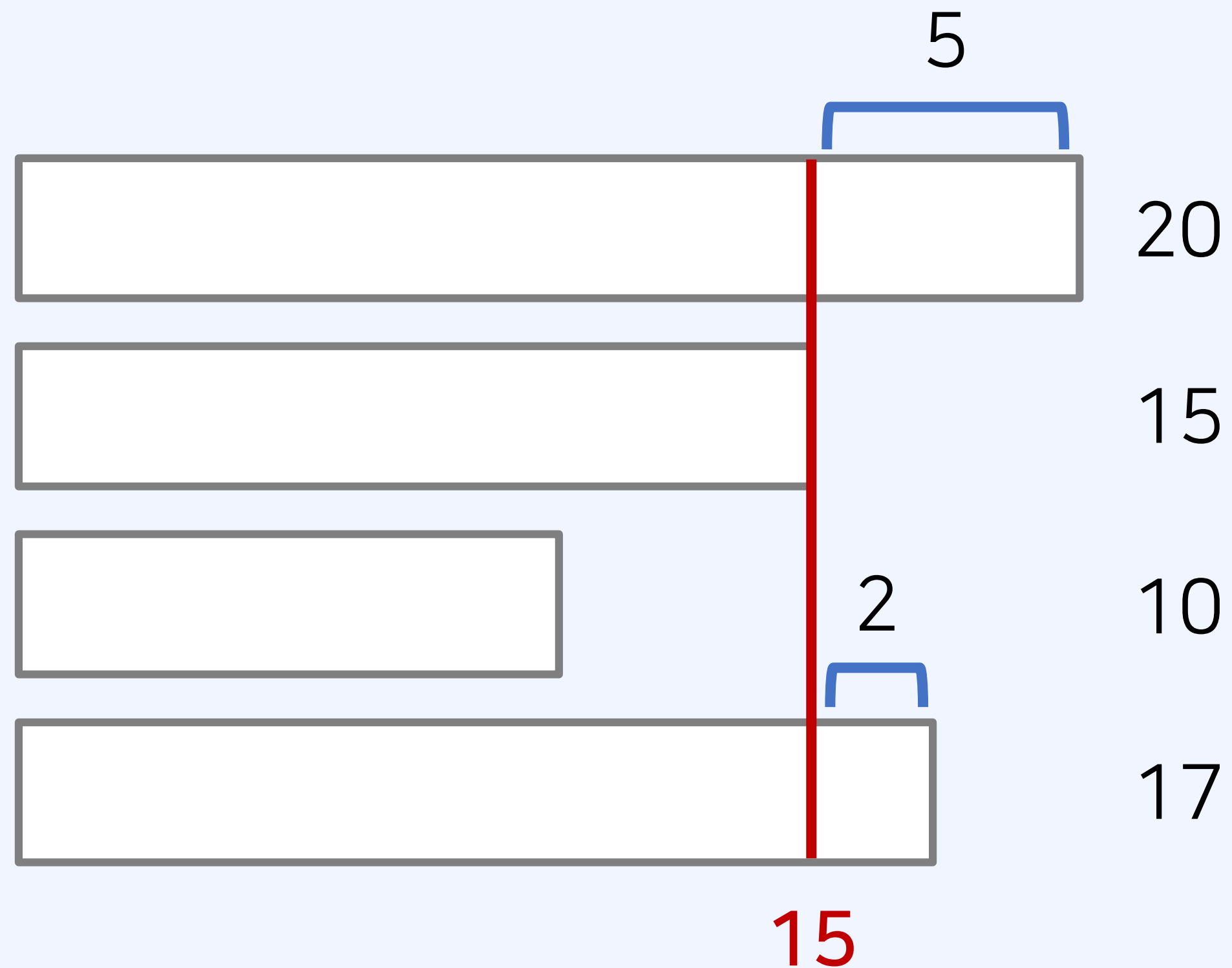
추천 풀이 시간: 40분

JavaScript 이진 탐색
이진 탐색 문제 풀이 ①

문제 해결 아이디어

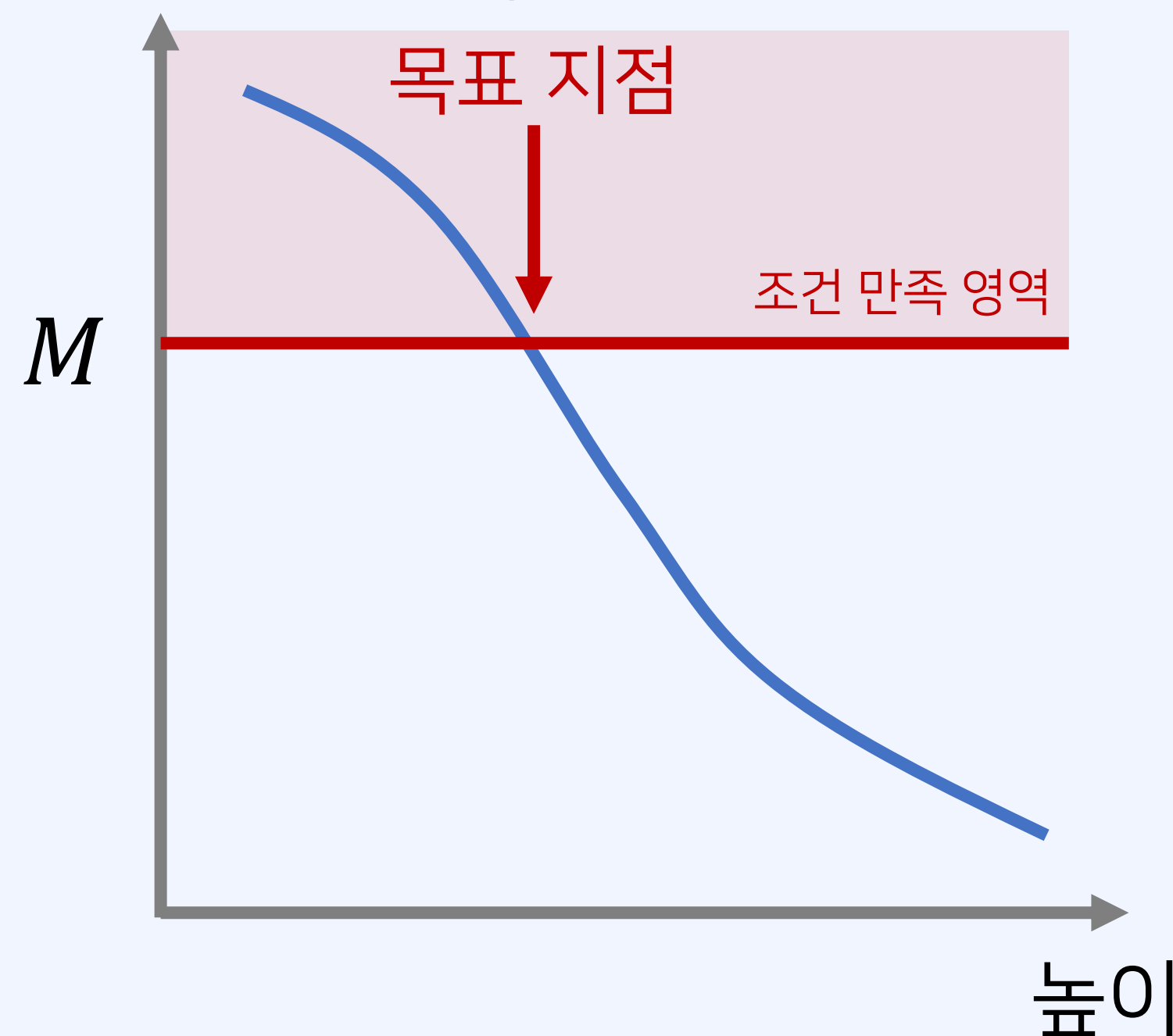
JavaScript
이진 탐색
이진 탐색
문제 풀이 ①

- 나무 자르기 문제의 목표: 적절한 높이(height) 값을 찾기
- **높이**를 15로 설정한 경우, 총 7만큼의 나무를 얻을 수 있다.



1. 절단기의 높이가 올라가는 경우: 일반적으로 얻을 수 있는 나무의 양이 감소한다.
2. 절단기의 높이가 내려가는 경우: 일반적으로 얻을 수 있는 나무의 양이 증가한다.

얻을 수 있는 나무의 양



M 이상의 나무를 얻을 수 있는 조건을 만족하는
높이의 최댓값을 구하는 문제다.
→ 파라메트릭 서치를 이용한다.

JavaScript 이진 탐색 이진 탐색 문제 풀이 ①

정답 코드 예시

JavaScript 이진 탐색

이진 탐색
문제 풀이 ①

```
let fs = require('fs');
let input = fs.readFileSync('/dev/stdin').toString().split('\n');

let n = Number(input[0].split(' ')[0]); // 나무의 수(N)
let m = Number(input[0].split(' ')[1]); // 가져갈 나무의 길이(M)
let arr = input[1].split(' ').map(Number); // 각 나무의 높이

let start = 0; // 이진 탐색을 위한 시작점(start)과 끝점(end) 설정
let end = arr.reduce((a, b) => Math.max(a, b));

let result = 0;
while (start <= end) { // 이진 탐색 수행(반복문)
    let mid = parseInt((start + end) / 2); // 현재의 중간점(높이)
    let total = 0; // mid로 잘랐을 때 얻을 수 있는 나무의 양 계산
    for (x of arr) if (x > mid) total += x - mid;
    if (total < m) end = mid - 1; // 나무의 양이 부족한 경우 더 많이 자르기(높이 줄이기)
    else { // 나무의 양이 충분한 경우 덜 자르기(높이 키우기)
        result = mid; // 최대한 덜 잘랐을 때가 정답이므로, result에 기록
        start = mid + 1;
    }
}
console.log(result);
```

JavaScript 이진 탐색
이진 탐색 문제 풀이 ①

혼자 힘으로 풀어보기

JavaScript
이진 탐색
이진 탐색
문제 풀이 ①

문제 제목: 랜선 자르기

문제 난이도: ★★☆☆☆

문제 유형: 이진 탐색, 파라메트릭 서치

추천 풀이 시간: 40분





JavaScript 이진 탐색
이진 탐색 문제 풀이 ①

문제 해결 아이디어

JavaScript
이진 탐색
이진 탐색
문제 풀이 ①

[문제의 요구사항] 랜선의 개수 N 개 이상을 얻을 수 있는 길이의 최댓값 구하기

1. **길이**를 키우면, 얻을 수 있는 랜선의 수는 감소한다.
2. **길이**를 줄이면, 얻을 수 있는 랜선의 수는 증가한다.

		길이: 200	길이: 100
	802	4개	8개
	743	3개	7개
	457	2개	4개
	539	2개	5개
		↓	↓
		11개	24개

JavaScript 이진 탐색 이진 탐색 문제 풀이 ①

정답 코드 예시

JavaScript 이진 탐색

이진 탐색
문제 풀이 ①

```
let fs = require('fs');
let input = fs.readFileSync('/dev/stdin').toString().split('\n');

let k = Number(input[0].split(' ')[0]); // 가지고 있는 랜선의 개수(K)
let n = Number(input[0].split(' ')[1]); // 필요한 랜선의 개수(N)

let arr = []; // 각 랜선의 길이
for (let i = 1; i <= k; i++) arr.push(Number(input[i]));

let start = 1; // 이진 탐색을 위한 시작점(start)과 끝점(end) 설정
let end = arr.reduce((a, b) => Math.max(a, b));

let result = 0;
while (start <= end) { // 이진 탐색 수행(반복문)
    let mid = parseInt((start + end) / 2); // 현재의 중간점(길이)
    let total = 0; // 가지고 있는 각 랜선을 잘라서 길이가 mid인 랜선을 몇 개 만들 수 있는지
    for (x of arr) total += parseInt(x / mid);
    if (total < n) end = mid - 1; // 만들 수 있는 랜선의 개수가 부족한 경우 길이 줄이기
    else { // 만들 수 있는 랜선의 개수가 충분한 경우 길이 늘이기
        result = mid; // 최대 길이를 찾아야 하므로, result에 기록
        start = mid + 1;
    }
}
console.log(result);
```