JavaScript
이진 탐색 알고리즘
이진 탐색 문제 풀이 ②

이진 탐색 문제 풀이 ② | 코딩 테스트에서 자주 등장하는 이진 탐색 알고리즘 이해하기

강사 나동빈

JavaScript
이진 탐색 알고리즘
이진 탐색 문제 풀이 ②

숫자 카드 2: https://www.acmicpc.net/problem/10816

혼자 힘으로 풀어보기

문제 제목: 숫자 카드 2

문제 난이도: ★★☆☆☆

문제 유형: 이진 탐색

추천 풀이 시간: 40분

문제 해결 아이디어

- 다음과 같이 N개의 카드 데이터가 존재한다.

6	3	2	10	10	10	-10	-10	-10	7	3

- **이진 탐색**을 위해 먼저 데이터를 **오름차순 정렬**한다.
- 이 과정에서 $O(NlogN)$의 시간 복잡도가 요구된다.

-10	-10	-10	2	3	3	6	7	10	10	10

숫자 카드 2: https://www.acmicpc.net/problem/10816

문제 해결 아이디어

- 데이터의 수(N) 및 쿼리의 수(M)는 모두 **최대 50만**인 것을 알 수 있다.

-10	-10	-10	2	3	3	6	7	10	10	10

- 다음과 같이 쿼리가 있다고 해보자.

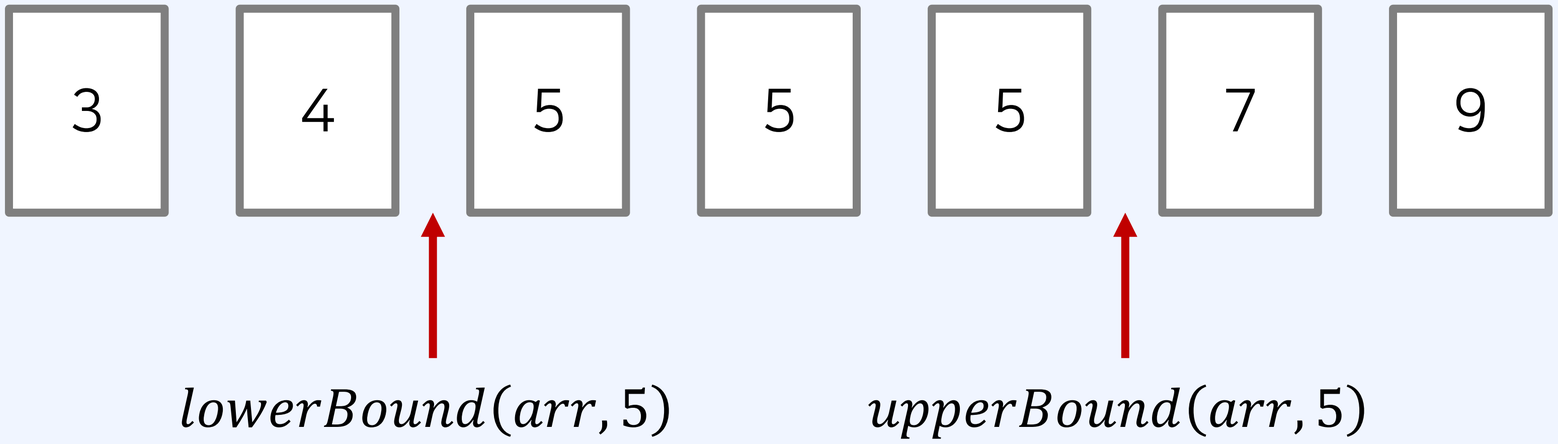
쿼리	10	9	-5	2	3	4	5	-10
개수	3개	0개	0개	1개	2개	0개	0개	3개

- 각 쿼리에 대하여 $O(logN)$으로 해결한다면, 전체 시간 복잡도는 $O(MlogN)$이다.

숫자 카드 2: https://www.acmicpc.net/problem/10816

문제 해결 아이디어

- 아래의 2가지 **이진 탐색 함수**가 제공하는 기능을 이해할 필요가 있다.
- $lowerBound(arr, x)$: 정렬된 순서를 유지하면서 배열 arr에 x를 넣을 가장 왼쪽 인덱스를 반환
- $upperBound(arr, x)$: 정렬된 순서를 유지하면서 배열 arr에 x를 넣을 가장 오른쪽 인덱스를 반환



숫자 카드 2: https://www.acmicpc.net/problem/10816

문제 해결 아이디어


```
// 정렬된 순서를 유지하면서 배열에 삽입할 가장 왼쪽 인덱스 반환
function lowerBound(arr, target, start, end) {
  while (start < end) {
    let mid = parseInt((start + end) / 2);
    if (arr[mid] >= target) end = mid; // 최대한 왼쪽으로 이동하기
    else start = mid + 1;
  }
  return end;
}

// 정렬된 순서를 유지하면서 배열에 삽입할 가장 오른쪽 인덱스 반환
function upperBound(arr, target, start, end) {
  while (start < end) {
    let mid = parseInt((start + end) / 2);
    if (arr[mid] > target) end = mid;
    else start = mid + 1; // 최대한 오른쪽으로 이동하기
  }
  return end;
}
```

숫자 카드 2: https://www.acmicpc.net/problem/10816

문제 해결 아이디어

- $countByRange()$: 정렬된 배열에서 값이 특정 범위에 속하는 원소의 개수를 계산한다.
- 앞서 정의한 $lowerBound()$ 함수와 $upperBound()$ 함수를 이용해 구현할 수 있다.

```
// 값이 [leftValue, rightValue]인 데이터의 개수를 반환하는 함수
function countByRange(arr, leftValue, rightValue) {
  // 유의: lowerBound와 upperBound는 end 변수의 값을 배열의 길이로 설정
  let rightIndex = upperBound(arr, rightValue, 0, arr.length);
  let leftIndex = lowerBound(arr, leftValue, 0, arr.length);
  return rightIndex - leftIndex;
}

// 배열 선언
let arr = [1, 2, 3, 3, 3, 3, 4, 4, 8, 9];
// 값이 4인 데이터 개수 출력
console.log(countByRange(arr, 4, 4));
// 값이 [-1, 3] 범위에 있는 데이터 개수 출력
console.log(countByRange(arr, -1, 3));
```

숫자 카드 2: https://www.acmicpc.net/problem/10816

문제 해결 아이디어

- 데이터의 수(N) 및 쿼리의 수(M)는 모두 **최대 50만**인 것을 알 수 있다.

-10	-10	-10	2	3	3	6	7	10	10	10

[예시] 값이 3인 데이터의 개수를 구하는 예시를 확인해 보자.

- $lowerBound(arr, 3)$: 4
- $upperBound(arr, 3)$: 6
- 따라서 값이 3인 데이터의 개수는 6 - 4 = 2이다.

숫자 카드 2: https://www.acmicpc.net/problem/10816

정답 코드 예시

JavaScript
이진 탐색
이진 탐색
문제 풀이 ②

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

let n = Number(input[0]); // 데이터의 수(N)
let arr = input[1].split(' ').map(Number);
let m = Number(input[2]); // 쿼리의 수(M)
let query = input[3].split(' ').map(Number);

arr.sort((a, b) => a - b); // 이진 탐색을 위한 오름차순 정렬

answer = '';
for (let i = 0; i < m; i++) {
  // 값이 query[i]인 데이터의 개수 계산
  let cnt = countByRange(arr, query[i], query[i]);
  answer += cnt + ' ';
}
console.log(answer);
```

혼자 힘으로 풀어보기

JavaScript
이진 탐색
이진 탐색
문제 풀이 ②

문제 제목: 병사 배치하기

문제 난이도: ★★☆☆☆

문제 유형: 이진 탐색, LIS

추천 풀이 시간: 50분

병사 배치하기: https://www.acmicpc.net/problem/18353

문제 해결 아이디어

JavaScript
이진 탐색
이진 탐색
문제 풀이 ②

- **문제 요구사항**: 전투력이 높은 병사가 앞쪽에 오도록 내림차순으로 배치한다.

1) 배치 과정에서는 특정한 위치에 있는 병사를 열외시키는 방법을 이용한다.
2) 그러면서도 남아 있는 병사의 수가 최대가 되도록 하는 것이 목표다.

문제 해결 아이디어

JavaScript
이진 탐색
이진 탐색
문제 풀이 ②

- 예를 들어 $N = 7$일 때 나열된 병사들의 전투력이 다음과 같다고 가정하자.

병사 번호	1	2	3	4	5	6	7
전투력	15	11	4	8	5	2	4

- 이때 3번 병사와 6번 병사를 열외시키면, 다음과 같이 남아 있는 병사의 수가 내림차순의 형태가 되며 5명이 된다. 이는 남아 있는 병사의 수가 최대가 되도록 하는 방법이다.

병사 번호	1	2	4	5	7
전투력	15	11	8	5	4

병사 배치하기: https://www.acmicpc.net/problem/18353

문제 해결 아이디어

JavaScript
이진 탐색
이진 탐색
문제 풀이 ②

- 본 문제는 특정한 수열에서 "**가장 긴 증가하는 부분 수열**"을 찾는 문제다.
  - Longest Increasing Subsequence (LIS) 알고리즘으로 소개된다.
  - **부분 수열**: 주어진 수열의 일부 항을 원래 순서대로 나열하여 얻을 수 있는 수열

4	2	5	8	4	11	15

4	2	5	8	4	11	15

Longest Increasing Subsequence (LIS): [2, 5, 8, 11, 15]

병사 배치하기: https://www.acmicpc.net/problem/18353

문제 해결 아이디어

- 이진 탐색을 활용하면 최악의 경우 시간 복잡도 $O(NlogN)$의 코드로 해결할 수 있다.
  - **아이디어**: 현재 원소가 가장 크다면 뒤에 삽입하고, 그렇지 않다면 최대한 왼쪽의 원소와 교체

	4	2	5	8	4	11	15
초기 상태	0						
$i = 0$	0	4					
$i = 1$	0	2					
$i = 2$	0	2	5				
$i = 3$	0	2	5	8			
$i = 4$	0	2	4	8			

(4이 가장 크므로 삽입)

(2를 최대한 왼쪽 원소와 교체)

(5가 가장 크므로 삽입)

(8이 가장 크므로 삽입)

(4를 최대한 왼쪽 원소와 교체)

⋮

병사 배치하기: https://www.acmicpc.net/problem/18353

JavaScript 이진 탐색
이진 탐색 문제 풀이 ②

정답 코드 예시

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

let n = Number(input[0]);
let arr = input[1].split(' ').map(Number);

arr.reverse(); // 순서를 뒤집어 최장 증가 부분 수열(LIS) 문제로 변환

let d = [0]; // LIS 배열
// 이진 탐색을 활용한 LIS 알고리즘 수행
for (x of arr) {
  if (d[d.length - 1] < x) { // 마지막 원소보다 현재 원소 x가 크다면 가장 뒤에 넣기
    d.push(x);
  }
  else { // x 이하인 원소가 있다면, 가능한 왼쪽에 있는 원소와 x를 교체
    let index = lowerBound(d, x, 0, d.length);
    d[index] = x;
  }
}

// 열외해야 하는 병사의 최소 수를 출력
console.log(n - (d.length - 1));
```

K번째 수: https://www.acmicpc.net/problem/1300

혼자 힘으로 풀어보기

JavaScript
이진 탐색
이진 탐색
문제 풀이 ②

문제 제목: K번째 수

문제 난이도: ★★★☆☆

문제 유형: 이진 탐색

추천 풀이 시간: 60분

K번째 수: https://www.acmicpc.net/problem/1300

문제 해결 아이디어

- 정렬 이후에 K번째 수를 구하는 것이 목표다. $K = 14$라고 가정해 보자. 답은 12다.

i \ j	1	2	3	4
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12
4	4	8	12	16

< 정렬된 값 >

[1, 2, 2, 3, 3, 4, 4, 4, 6, 6, 8, 8, 9, 12, 12, 16]

이분 탐색으로 정답 자체를 mid로 찾는다고 가정하자.

$\rightarrow$ mid가 K번째 수가 되려면?

"현재 mid보다 작거나 같은 데이터의 수가 K개 이상이 되는 조건"을 만족하는 mid 중에서 가장 작은 값을 구하면 된다.

K번째 수: https://www.acmicpc.net/problem/1300

문제 해결 아이디어

JavaScript
이진 탐색
이진 탐색
문제 풀이 ②

- 정렬 이후에 K번째 수를 구하는 것이 목표다. $K = 14$라고 가정해 보자. 답은 12다.

[초기 설정] $left = 1, right = 16$

i \ j	1	2	3	4
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12
4	4	8	12	16

< mid보다 작거나 같은 원소의 개수 >

$mid = 8$	$mid = 12$	$mid = 10$
4	4	4
4	4	4
2	4	3
2	3	2
$12 < 14$	$15 \geq 14$	$13 < 14$

...

K번째 수: https://www.acmicpc.net/problem/1300

정답 코드 예시

JavaScript
이진 탐색
이진 탐색
문제 풀이 ②

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

let n = Number(input[0]); // 배열의 크기(N)
let k = Number(input[1]); // 인덱스 K

let start = 1; // 배열에 존재할 수 있는 가장 작은 값
let end = 10 ** 10; // 배열에 존재할 수 있는 가장 큰 값

let result = 0;
while (start <= end) { // 이진 탐색 수행(반복적)
  let mid = parseInt((start + end) / 2); // 현재의 중간점
  let total = 0; // mid보다 작거나 같은 데이터의 개수
  for (let i = 1; i <= n; i++) { // 각 행마다 계산
    total += Math.min(parseInt(mid / i), n);
  }
  if (total >= k) { // mid보다 작거나 같은 데이터의 개수가 k 이상이라면
    result = mid; // result에 기록
    end = mid - 1;
  }
  // mid보다 작거나 같은 데이터의 개수가 k 미만이라면
  else start = mid + 1;
}
console.log(result);
```