코테 스터디

4주차 - 2024 / 05 / 21

이분 탐색 이해하기

이분 탐색과 파라메트릭 서치

> 이분 탐색이란

- 업-다운 게임 생각해보기

> 이분 탐색이란

- 한 사람이 1~100 중 하나의 수를 생각함
- 다른 사람이 수 1개를 얘기함
- 그러면 대답으로 생각한 수보다 큰지 작은 지 얘기함

- 생각한 수를 맞춰보기

> 이분 탐색이란

- 몇 번의 시도만에 찾을 수 있을까?

> 이분 탐색이란

- 최대 7번 만에 찾을 수 있는 방법이 있음

> 이분 탐색이란

- 가능한 수 범위에서 중앙값을 얘기함
- 업-다운을 들으면 불가능해지는 범위를 제외함
- 수가 확정될 때까지 이 과정을 반복하기

> 이분 탐색이란

- 업-다운을 한 번 들을 때마다 수 범위가 절반으로 줄어듬
- 업-다운을 N번 들을 때마다 수 범위가 $\frac{1}{2^N}$으로 줄어듬

> 이분 탐색이란

- 수 하나를 찾기 위해선 $\log_2 N$번을 해야함
- $6 < \log_2 100 < 7$이므로 최대 7번 물어보면 됨

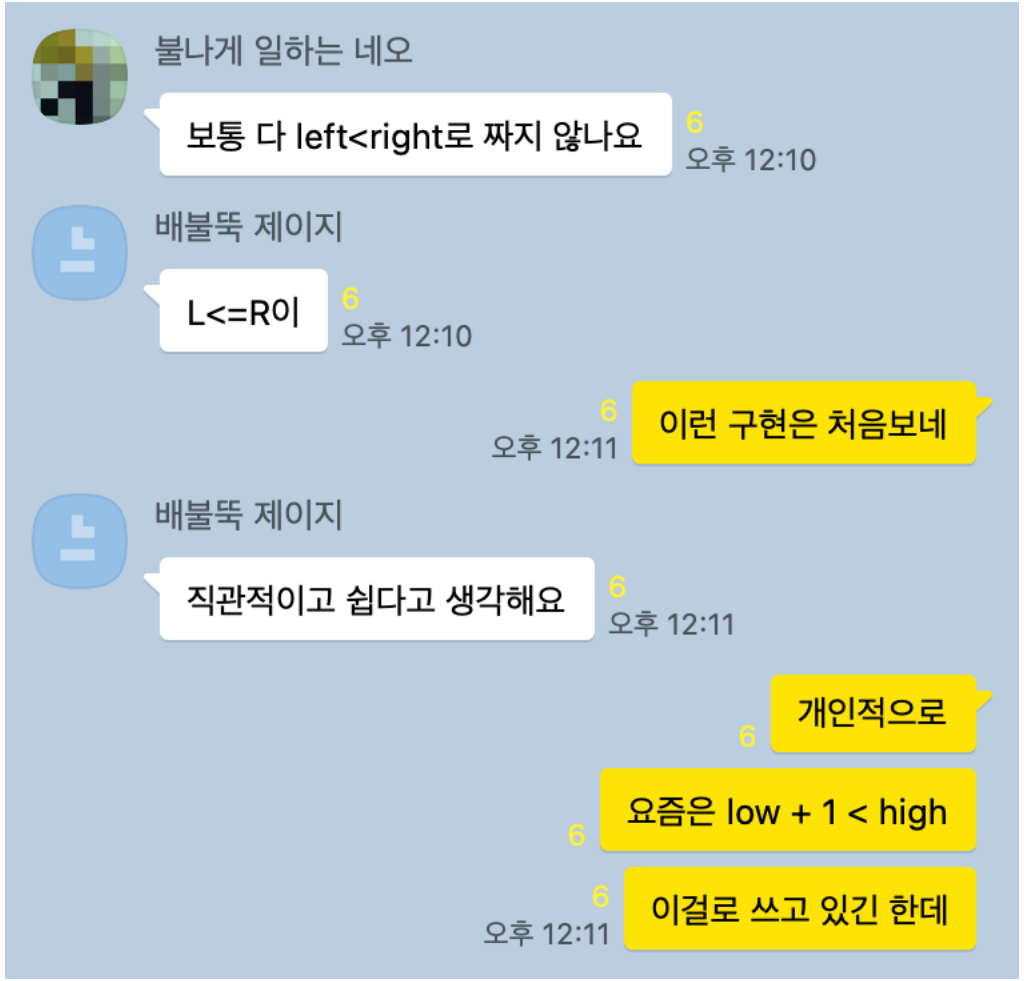
> 그래서 이분 탐색이란

- 리스트에서 매번 구간을 절반으로 줄이면서 원소를 찾는 알고리즘
- 리스트가 정렬되어 있을 때만 사용할 수 있음

> 그래서 이분 탐색이란

- 앞에서 이야기한 업-다운 게임의 전략과 방법이 같음

> 이분 탐색 구현해보기



- 많은 구현 방법이 있고

  실수할 여지가 매우 많음

> 이분 탐색 구현해보기

Ex) 1~10 까지의 수가 있을 때 6이라는 숫자 찾기

> 이분 탐색 구현해보기

1. 범위 설정하기

- 숫자의 범위가 1~10이므로 다음과 같이 범위의 초기값을 설정
  - low = 1 – 1 = 0, high = 10 + 1 = 11

→ 이렇게 되면 구간의 범위는 (low, high) 와 같음 (low < {구간} < high)

2. 구간의 중간 위치 찾기

- 현재 구간의 low, high를 기준으로 중간 위치인 mid를 구함
  - mid = (low + high) / 2 = (0 + 11) / 2 = 5

3. 중간값 체크

- mid위치에 있는 값이 찾는 값인지 확인하기

3. 중간값 체크

- mid위치에 있는 값이 찾는 값이 아니라면 대소 비교 하기
  - 만약 찾는 값이 더 크면 low = mid
  - 만약 찾는 값이 더 작으면 high = mid

→ 이렇게 되면 찾는 값이 있을 수 있는 후보 구간의 범위는

그대로 (low, high) 와 같음 (low < {구간} < high)

4. 반복하기

- low + 1 < high 를 만족하는 동안 2, 3번 과정을 반복하기

> 이분 탐색 구현해보기

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

int main() {
    int v[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int num = 6;

    // 1. 범위 설정하기
    int low = 1; int high = 10;
    low -= 1; high += 1;

    while (low + 1 < high) {
        // 2. 구간의 중간 위치 찾기
        int mid = (low + high) / 2;

        // 3. 중간값 체크
        if (v[mid] == num) { break; } // 찾았을 때

        if (v[mid] < num) low = mid;
        if (v[mid] > num) high = mid;
    }

    return 0;
}
```

> 다른 이분 탐색 구현 방법

- https://witch.work/posts/binary-search-next-step
- 다양한 이분 탐색 구현 방법을 상세히 설명해놓음

> 이분 탐색 연습하기

- BOJ 1920번 수 찾기
  - https://www.acmicpc.net/problem/1920

> 이분 탐색 연습하기

수 찾기 성공

★

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	266650	83137	55110	30.130%

문제

N개의 정수 A[1], A[2], ..., A[N]이 주어져 있을 때, 이 안에 X라는 정수가 존재하는지 알아내는 프로그램을 작성하시오.

입력

첫째 줄에 자연수 N(1 ≤ N ≤ 100,000)이 주어진다. 다음 줄에는 N개의 정수 A[1], A[2], ..., A[N]이 주어진다. 다음 줄에는 M(1 ≤ M ≤ 100,000)이 주어진다. 다음 줄에는 M개의 수들이 주어지는데, 이 수들이 A안에 존재하는지 알아내면 된다. 모든 정수의 범위는 -2^{31} 보다 크거나 같고 2^{31}보다 작다.

출력

M개의 줄에 답을 출력한다. 존재하면 1을, 존재하지 않으면 0을 출력한다.

> 이분 탐색 연습하기

```cpp
int n, v[1010101];

cin >> n;
for (int i = 0; i < n; i++) cin >> v[i];

// 수열 정렬하기
sort(v, v + n);

int tc; cin >> tc;

while (tc--) {
    int num;
    cin >> num;

    // 수를 찾았는 지 확인하기 위한 변수
    bool ok = false;

    // 이분 탐색하기
    int low = 0; int high = n - 1;
    low -= 1; high += 1;

    while (low + 1 < high) {
        int mid = (low + high) / 2;

        if (v[mid] == num) {
            ok = true;
            break;
        }

        if (v[mid] < num) low = mid;
        if (v[mid] > num) high = mid;
    }

    cout << (ok ? 1 : 0) << '\n';
}
```

> 이분 탐색 연습하기

- **백준 1920** 수 찾기
  수열에 원소가 있는 지 찾는 문제 1
- **백준 10815** 숫자 카드
  수열에 원소가 있는 지 찾는 문제 2
- **백준 2776** 암기왕
  수열에 원소가 있는 지 찾는 문제 3

질문 시간

편하게 질문해주세요

> 파라메트릭 서치 이해하기

- 이분 탐색을 이용하여 최적화 문제를 결정 문제로 바꾸어 해결하는 기법

> 파라메트릭 서치 이해하기

- 최적화 문제?
  - 이 조건을 만들 수 있는 최소한의 / 최대한의 x 구하기

- 결정 문제?
  - 답이 True / False 두 가지로만 나뉘는 문제
  - 보통 이분 탐색에서는 parameter가 1개

> 파라메트릭 서치 이해하기

- 파라메트릭 서치를 사용하기 위해선 문제가 다음 조건을 만족해야 함
  - 리스트의 원소들이 결정 문제에 대한 결과가 이분적이어야함

FFFFFFFFTT / TTTTTFFFFF → 이분적임

FFFFFTTTFFF / TTFFFFFFT → 이분적이지 않음

> 파라메트릭 서치 이해하기

- Ex) 수열 V = [2, 7, 1, 9, 10] 에서 5 이상인 가장 작은 수 구하기

> 파라메트릭 서치 이해하기

- 문제 상황 분석하기
- V[i] >= 5를 만족하는 V[i] 중 가장 작은 수 구하기

> 파라메트릭 서치 이해하기

- 문제 상황 분석하기
- V[i] >= 5를 만족하는 V[i] 중 가장 작은 수 구하기

최적화 문제

> 파라메트릭 서치 이해하기

- 결정 문제로 바꿀 방법?

> 파라메트릭 서치 이해하기

- 일단 정렬해보기
- V = [1, 2, 7, 9, 10]

> 파라메트릭 서치 이해하기

- 결정 문제의 조건 함수 → Check(x) = V[i] >= x
- V = [1, 2, 7, 9, 10] → [F, F, T, T, T]

> 파라메트릭 서치 이해하기

- V = [1, 2, 7, 9, 10] → [F, F, T, T, T]
- 결정 문제에 대해 이분적이게 됨

> 파라메트릭 서치 이해하기

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

int main() {
    int v[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int num = 6;

    // 1. 범위 설정하기
    int low = 1; int high = 10;
    low -= 1; high += 1;

    while (low + 1 < high) {
        // 2. 구간의 중간 위치 찾기
        int mid = (low + high) / 2;

        // 3. 중간값 체크
        if (v[mid] == num) { break; } // 찾았을 때

        if (v[mid] < num) low = mid;
        if (v[mid] > num) high = mid;
    }

    return 0;
}
```

> 파라메트릭 서치 이해하기

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

bool Check(int x) {
    if (x >= 5) return true;
    else return false;
}

int main() {
    int v[] = {1, 2, 7, 9, 10};

    // 배열의 크기로 범위 설정
    int low = 1; int high = 5;
    low--; high++;

    while (low + 1 < high) {
        int mid = (low + high) / 2;

        if (Check(v[mid]) == true) high = mid;
        else low = mid;
    }

    return 0;
}
```

> 여담

- C++ 에서는 다음과 같은 구현된 함수들을 이미 사용할 수 있음

  binary_search(), lower_bound(), upper_bound()

> 파라메트릭 서치 연습하기

- **백준 10816** 숫자 카드 2
  lower_bound와 upper_bound를 이용하는 문제
- **백준 2805** 나무 자르기
  최적화 문제를 결정 문제로 바꿔보는 연습 문제
- **백준 2343** 기타 레슨
  최적화 문제를 결정 문제로 바꿔보는 연습 문제 2

질문 시간

편하게 질문해주세요

수고하셨습니다

질문 받습니다