1920. 수 찾기

출처	BoJ
실습/과제/자율	자율
문제유형	이분 탐색
날짜(처음 풀이)	@2024년 9월 25일
풀이 언어	Java

https://www.acmicpc.net/problem/1920

https://www.acmicpc.net/problem/1920

1. 문제 요약

N개의 정수 A[1], A[2], … , A[N]이 주어져 있을 때, 이 안에 X라는 정수가 존재하는지 알아내는 프로그램 작성하기.

2. 문제 입출력

입력

- 첫째 줄에 자연수 N이 주어진다.
- 두번째 줄에는 N개의 정수 A[1], A[2], … , A[N]이 주어진다.
- 세번째 줄에는 M이 주어진다.
- 마지막 줄에는 M개의 수들이 주어진다.

입력예시

```
5
4 1 5 2 3
5
1 3 7 9 5
```

출력

- M개의 줄에 답(주어진 수들이 A안에 존재하면 1, 아니면 0)을 출력한다.

출력예시

```
1
1
0
```

```
0
1
```

📣 3. 제한사항

- **시간** : 1초
- **메모리** : 128MB

- 1 ≤ N ≤ 100,000
- 1 ≤ M ≤ 100,000
- 모든 정수의 범위는 -2^31 보다 크거나 같고 2^31보다 작다.

🧐 4. 접근법

▼ 초기 접근법 - 단순 비교 (이분 탐색을 몰랐다고 가정)

1. N개의 숫자 중에서 가장 큰 수 찾기
2. M개의 수 반복문 돌며 가장 큰 수 보다 큰지 비교
    - 크면 return 0
    - 작으면 return 1

1. N개의 숫자 정렬 (arr)
2. M개의 수 반복문 돌며 이분탐색 진행
    - while (left <= right) (left = 0, right = arr.length - 1)
        - mid = (left + right) / 2
        - arr[mid] == num 이면 값을 찾은 것이니까 return true (= 1 출력)
        - arr[mid] < num 이면 left = mid + 1 (오른쪽으로 이동)
        - arr[mid] > num 이면 right = mid - 1 (왼쪽으로 이동)
    - 값을 찾지 못하면 return false (= 0 출력)

💻 5. 코드

▼ 이전 코드

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Arrays;
import java.util.StringTokenizer;
```

```java
public class Main {
    static int N, M;
    static int maxnum;
    static int[] nNums, mNums;
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st;

        N = Integer.parseInt(br.readLine());
        nNums = new int[N];

        st = new StringTokenizer(br.readLine());
        for (int n = 0; n < N; n++) {
            nNums[n] = Integer.parseInt(st.nextToken());
        }

        Arrays.sort(nNums);
        maxnum = nNums[N - 1];

        M = Integer.parseInt(br.readLine());
        mNums = new int[M];

        st = new StringTokenizer(br.readLine());
        for (int m = 0; m < M; m++) {
            mNums[m] = Integer.parseInt(st.nextToken());
        }

        for (int i = 0; i < M; i++) {
            if (mNums[i] > maxnum) {
                System.out.println(0);
            } else {
                System.out.println(1);
            }
        }

    }
}
```

▼ 최종 코드

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Arrays;
import java.util.StringTokenizer;

public class Main {
    static int N, M;
    static int[] nNums, mNums;

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st;

        N = Integer.parseInt(br.readLine());
        nNums = new int[N];

        st = new StringTokenizer(br.readLine());
        for (int n = 0; n < N; n++) {
```

```
            nNums[n] = Integer.parseInt(st.nextToken());
        }

        Arrays.sort(nNums);

        M = Integer.parseInt(br.readLine());
        mNums = new int[M];

        st = new StringTokenizer(br.readLine());
        for (int m = 0; m < M; m++) {
            mNums[m] = Integer.parseInt(st.nextToken());
        }

        for (int i = 0; i < M; i++) {
            if (binarySearch(nNums, mNums[i])) {
                System.out.println(1); // 찾으면 1
            } else {
                System.out.println(0); // 못 찾으면 0
            }
        }
    }

    public static boolean binarySearch(int[] arr, int num) {
        int left = 0;
        int right = arr.length - 1;

        while (left <= right) {
            int mid = (left + right) / 2;
            if (arr[mid] == num) {
                return true; // 값을 찾음
            } else if (arr[mid] < num) {
                left = mid + 1; // 중간값이 찾는 값보다 작으면 오른쪽으로 이동
            } else {
                right = mid - 1; // 중간값이 찾는 값보다 크면 왼쪽으로 이동
            }
        }
        return false; // 값을 찾지 못함
    }
}
```