Lower/Upper bound

Binary Search

- algorithm 헤더에 있음
- 오름차순으로 정렬된 배열/리스트에서 사용
- 찾으려는 값이 있으면 true, 없으면 false를 리턴
- 사용법 : binary_search(시작주소, 끝주소, 찾으려는 값)

10815_숫자 카드

문제

숫자 카드는 정수 하나가 적혀져 있는 카드이다. 상근이는 숫자 카드 N개를 가지고 있다. 정수 M개가 주어졌을 때, 이 수가 적혀있는 숫자 카드를 상근이가 가지고 있는지 아닌지를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 상근이가 가지고 있는 숫자 카드의 개수 N(1 ≤ N ≤ 500,000)이 주어진다. 둘째 줄에는 숫자 카드에 적혀있는 정수가 주어진다. 숫자 카드에 적혀있는 수는 -10,000,000보다 크거나 같고, 10,000,000보다 작거나 같다. 두 숫자 카드에 같은 수가 적혀있는 경우는 없다.

셋째 줄에는 M(1 ≤ M ≤ 500,000)이 주어진다. 넷째 줄에는 상근이가 가지고 있는 숫자 카드인지 아닌지를 구해야 할 M개의 정수가 주어지며, 이 수는 공백으로 구분되어져 있다. 이 수도 -10,000,000보다 크거나 같고, 10,000,000보다 작거나 같다

출력

첫째 줄에 입력으로 주어진 M개의 수에 대해서, 각 수가 적힌 숫자 카드를 상근이가 가지고 있으면 1을, 아니면 0을 공백으로 구분해 출력한다.

10815_숫자 카드

```
cin >> n;
for (int i = 0; i < n; i++) {
    int k;
    cin >> k;
    v.push_back(k);
}
sort(v.begin(), v.end());
```

입력 받고 정렬

```
cin >> m;
while (m--) {
    int k;
    cin >> k;
    cout << binary_search(v.begin(), v.end(), k) << ' ';
}
return 0;
```

binary_search 사용

upper_bound

- 이진탐색(Binary Search)기반의 탐색법
- 이진탐색(Binary Search)기반이므로 배열이나 리스트가 오름차순으로 정렬 되어있어야 함
- upper_bound는 key값을 초과하는 가장 첫 번째 원소의 위치를 구함
- 사용법 : upper_bound(시작 주소, 끝 주소, key 값)
- 리스트의 값이 key값을 가장 먼저 초과하는 주소 값을 리턴
- 찾는 값이 없을 경우 끝 주소를 리턴(벡터의 경우 v.end())

lower_bound

- 이진탐색(Binary Search)기반의 탐색법
- 이진탐색(Binary Search)기반이므로 배열이나 리스트가 오름차순으로 정렬 되어있어야 함
- lower_bound는 key과 같거나 큰 가장 첫 번째 원소의 위치를 구함
- 사용법 : lower_bound(시작 주소, 끝 주소, key 값)
- 리스트의 값이 key과 같거나 큰 최초 주소 값을 리턴
- 찾는 값이 없을 경우 끝 주소를 리턴(벡터의 경우 v.end())

10816_숫자 카드

문제

숫자 카드는 정수 하나가 적혀져 있는 카드이다. 상근이는 숫자 카드 N개를 가지고 있다. 정수 M개가 주어졌을 때, 이 수가 적혀있는 숫자 카드를 상근이가 몇 개 가지고 있는지 구하는 프로그램을 작성하시오.

입력

첫째 줄에 상근이가 가지고 있는 숫자 카드의 개수 N(1 ≤ N ≤ 500,000)이 주어진다. 둘째 줄에는 숫자 카드에 적혀있는 정수가 주어진다. 숫자 카드에 적혀있는 수는 -10,000,000보다 크거나 같고, 10,000,000보다 작거나 같다.

셋째 줄에는 M(1 ≤ M ≤ 500,000)이 주어진다. 넷째 줄에는 상근이가 몇 개 가지고 있는 숫자 카드인지 구해야 할 M개의 정수가 주어지며, 이 수는 공백으로 구분되어져 있다. 이 수도 -10,000,000보다 크거나 같고, 10,000,000보다 작거나 같다.

출력

첫째 줄에 입력으로 주어진 M개의 수에 대해서, 각 수가 적힌 숫자 카드를 상근이가 몇 개 가지고 있는지를 공백으로 구분해 출력한다.

10816_숫자 카드

- 몇장이 있는지 어떻게 알까?
- 예시

예제 입력 1 복사

```
10
6 3 2 10 10 10 -10 -10 7 3
8
10 9 -5 2 3 4 5 -10
```

10816_숫자 카드

- upper_bound – lower_bound == 찾고자 하는 숫자 카드의 개수!!!

```
for(int i=0;i<m;i++){
    int tmp;
    cin >> tmp;
    int x = upper_bound(arr, arr+n, tmp) - lower_bound(arr, arr+n, tmp);

    cout << x << " ";
}
```

lower/upper에서 index 찾기

- Lower / upper bound는 주소값을 리턴함 – 인덱스로 찾고싶으면 어떡하지?
- sol : lower_bound(시작주소, 끝주소, key값) – 시작주소
- Ex) lower_bound(v.begin(), v.end(), key) – v.begin() == 내가 찾은 값에 해당하는 벡터 내부의 인덱스