- 시간 복잡도 : $O(\log N)$ (물론 정렬된 컨테이너에서만 올바르게 동작).

- binary_search() 함수도 존재한다.
  but, 리턴 값이 bool이다. (있는지 체크만 가능).

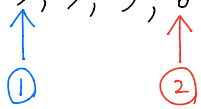
- lower_bound(), upper_bound()의 원래 목적은
  컨테이너의 정렬상태를 깨지 않고 원소를 삽입할 수 있는 위치를 찾아주는 것.
  (원소의 위치를 찾아주는 것이 아니다 !! ⇒ 주의).
  * 해당 위치에 insert되어도 원래의 컨테이너의 정렬을 깨지 않는것.
  ⇒ find() 계열의 함수는 탐색 실패시 end()의 iterator 값을 반환.

- equal_range() 함수는 lower, upper의 반환 값인 iterator쌍을 pair 형태로 반환. → first가 lower.
  ⇒ 두 iterator 값의 차이는 원소의 개수를 나타낸다.

ex) {1, 2, 2, 2, 3, 3, 3, 6, 7, 8} 에서

```
auto i = equal_range(v.begin(), v.end(), 3);
cout << i.first - v.begin() << '\n'; // 4
cout << i.second - v.begin() << '\n'; // 7.
```

{1, 2, 2, 2, 3, 3, 3, 6, 7, 8} 에서



① 3이라는 값을 lower (정렬을 깨지 않는 가장 앞자리)로 찾으면
[4]의 iterator 반환.

② upper (정렬을 깨지 않는 가장 뒷자리)로 찾으면
[7]의 iterator 반환 ([6]에 넣으면 마지막에 위치하면 3이 뒤로 밀려남).

ex2) 위 컨테이너에서 4를 찾으면. first, second 모두 3 다음의 위치의 iterator 반환. (정렬을 깨지 않는 가장 앞/뒤를 반환하므로).
[7].