이분 탐색(binary search)

'이분 탐색'은 이미 정렬된 배열에서 특정한 원소가 포함되어 있는지 아닌지를 빠르게 판단할 수 있는 기법이다. 판단 순서는 다음과 같다. 값 A 를 찾고자 할 때, 값이 포함된 것으로 예상되는 범위의 가운데 값을 먼저 확인한다. 그 다음, A 와 가운데 원소를 비교하며 A 가 있을 수 없는 곳은 탐색하지 않고 지나갑니다. 정렬된 배열이 주어졌을 때 15 라는 원소가 어디에 있는지 이분 탐색으로 찾아보면,

1	3	5	6	11	12	15	17	19	20

가장 처음에는 11 을 기준으로 15 와 비교를 해보자

1	3	5	6	11	12	15	17	19	20

이때 11 은 15 보다 작은 수이므로 11 보다 작은 부분에서는 15 를 찾을 필요가 없다.

~~1~~	~~3~~	~~5~~	~~6~~	~~11~~	12	15	17	19	20

다음은 17 과 15 를 비교해보고, 17 은 15 보다 큰 수이므로 17 보다 큰 부분에서는 15 를 찾을 필요가 없다.

~~1~~	~~3~~	~~5~~	~~6~~	~~11~~	12	15	~~17~~	~~19~~	~~20~~

다음 역시 12 와 15 를 비교해보고 필요 없는 부분을 삭제하면,.

~~1~~	~~3~~	~~5~~	~~6~~	~~11~~	~~12~~	15	~~17~~	~~19~~	~~20~~

드디어 찾고자 하던 15 를 찾을 수 있다. 이분 탐색은 정렬이 이미 되어있는 경우 $O(logN)$의 시간 복잡도를 가지며, 정렬이 되어있지 않은 경우 정렬을 위해 O(NlogN)의 시간 복잡도를 갖게 됩니다. 다음은 C++을 이용한 이분 탐색의 예제 코드이다.

```swift
func binary_search(_ arr: [Int], _ data: Int) -> Int {
    var left = 0, right = arr.count - 1

    while left <= right {
        let mid = (left + right) / 2
        if arr[mid] == data {
            return mid
        }
        else if arr[mid] < data {
            left = mid + 1
        }
        else {
            right = mid - 1
        }
    }
    return -1
}
```

이분 탐색은 정렬된 배열에서 특정한 원소를 찾기 위해 사용될 뿐만 아니라, 어떤 문제를 풀며 **답을 직접 구하기는 어렵지만, 답을 가정할 수 있고 해당 답이 맞는지 틀린지 확인하는 문제에서 자주 사용**된다.