Algorithm
Binary Search

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Linear Search

Linear Search는 가장 단순한 방식의 탐색 방법입니다.

배열의 요소를 처음부터 끝까지 순차적으로 순회하며 원하는 요소를 찾는 방식입니다.

최악의 경우는 모든 배열을 순회하고 나서야 값을 찾거나 찾지 못할 수 있습니다.

따라서 시간복잡도가 O(n)이 될 수 있는 알고리즘입니다.

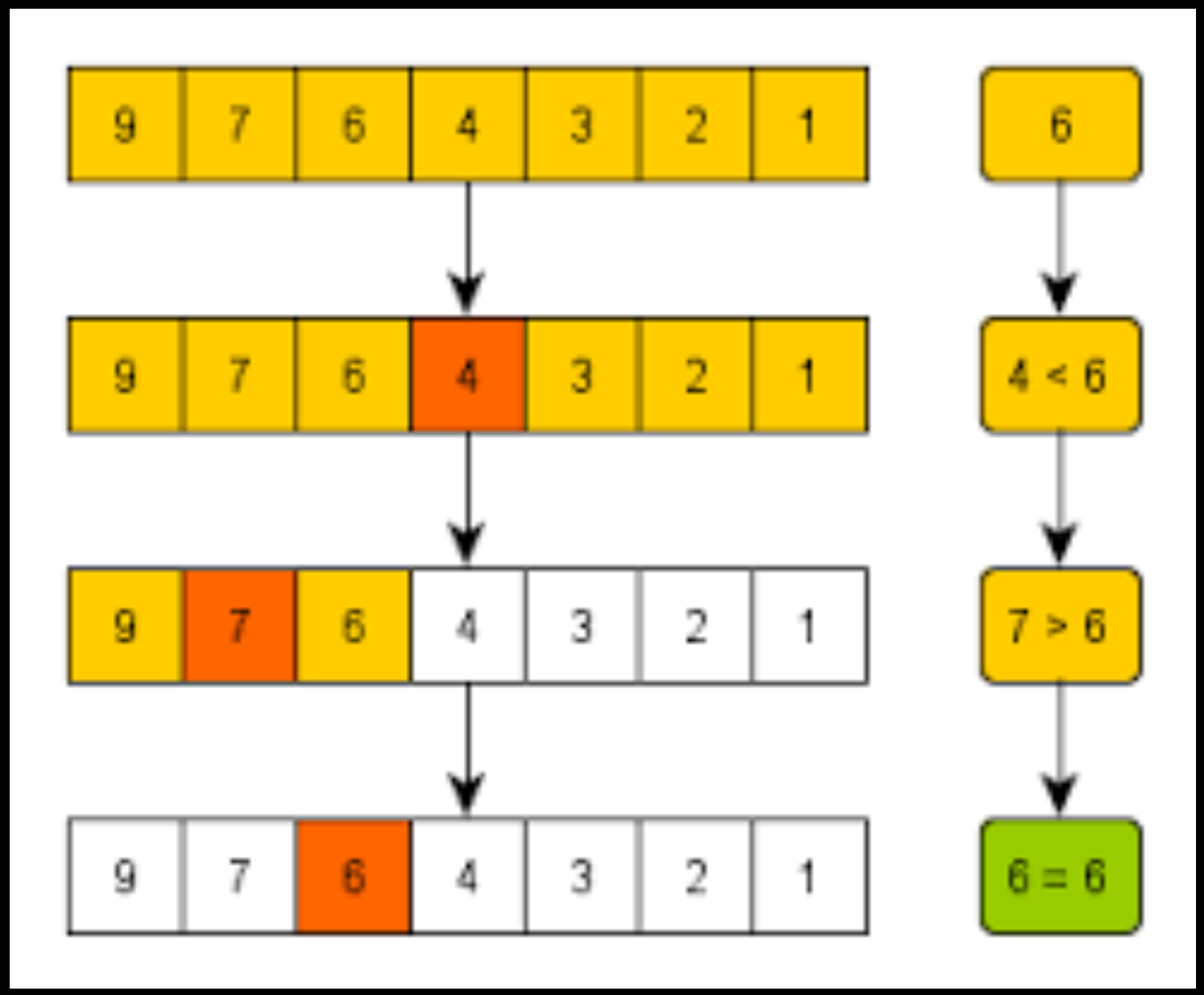
Binary Search

Binary Search이란 이진 탐색으로서 단순한 선형 탐색(Linear Search)와 달리 절반씩 배열을 줄여나가면서 찾는 탐색 알고리즘입니다.

단 이진 탐색 알고리즘을 실행하기 위해서는 배열의 요소가 정렬이 되어있어야 하는 단점이 있습니다.

Concept

이진 탐색의 과정을 보면 아래와 같은 방식으로 탐색을 합니다.



1. 배열의 중앙 값(middle index)을 찾습니다.
2. 찾고자하는 값과 중앙값을 비교하여 작으면 왼쪽, 크면 오른쪽의 요소들의 배열의 중앙값을 다시 찾습니다.
3. 현재 중앙값의 요소와 찾고자하는 값을 계속 비교하며 찾을때까지 2번의 과정을 반복합니다.

Features

이진 탐색은 아래와 같은 특징을 가진 알고리즘입니다.

1. 기본적으로 배열을 절반으로 나누면서 탐색을 하는 방식
2. 탐색 전에 배열이 정렬이 되어 있어야 함
3. 시간복잡도는 O(logn)의 속도를 가짐
4. 알고리즘의 복잡도가 높지 않으며 효율이 선형 탐색보다 좋음

Implementation

Swift를 활용하여 이진 탐색 알고리즘을 살펴보겠습니다.

아래는 기본적으로 오른차순으로 정렬된 배열을 기준으로하여 탐색하는 알고리즘이니다.

```swift
func binarySearch<T: Comparable>(array: [T], item: T) -> Int {
    var low = 0
    var high = array.count - 1

    while low <= high {
        let mid = (low + high) / 2
        let guess = array[mid]
        if guess == item {
            return mid
        } else if guess > item {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }

    return -1
}
```

Implementation

아래와 같이 재귀 호출 방식으로도 이진 탐색 알고리즘을 구현할 수 있습니다.

```
func binarySearch(first: Int, last: Int, target: Int, array: [Int]) -> Int {
    if first > last {
        return -1
    }
    let middle = (first + last) / 2

    if target == array[middle] {
        return middle
    } else {
        if target < array[middle] {
            return binarySearch(first: first, last: middle-1, target: target,
array: array)
        } else {
            return binarySearch(first: middle+1, last: last, target: target,
array: array)
        }
    }
}
```

Implementation

```
let numbers = [-1, 0, 1, 2, 5, 13, 15, 20, 45, 51, 59, 68, 77]

print("Index : \(binarySearch(array: numbers, item: 1))") // Index : 2
print("Index : \(binarySearch(array: numbers, item: 68))") // Index : 11
print("Index : \(binarySearch(array: numbers, item: 99))") // Index : -1

print("Index : \(binarySearch(first: 0, last: numbers.count, target: 1, array:
numbers))") // Index : 2
```

References

[1] [Swift] Swift로 Binary Search(이진 탐색 알고리즘)를 구현해보자 : https://eunjin3786.tistory.com/32

[2] Play With Code: Binary Search In Swift : https://learnappmaking.com/binary-search-swift-how-to/

[3] Swift, Algorithm, linear Search & Binary Search : https://devmjun.github.io/archive/BinarySearch

[4] How to implement Binary Search in Swift? : https://medium.com/@notestomyself/how-to-implement-binary-search-in-swift-6867840302ed

[5] Divide and Conquer with Binary Search in Swift : https://mikebuss.com/2016/04/21/binary-search/

References

[6] Binary Search Array Extension in Swift : https://agostini.tech/2017/01/30/binary-search-array-extension-in-swift/

[7] Swift Algorithms - Binary Search : https://www.seemuapps.com/swift-algorithms-binary-search

[8] 이진 검색(Binary Search) : https://kka7.tistory.com/77

[9] Algorithm) 이진 탐색(Binary Search) in Swift : https://atelier-chez-moi.tistory.com/88

[10] [스위프트:알고리즘] 이진 탐색[1 / 3]: Binary Search: 이진 탐색이 뭐야? : https://the-brain-of-sic2.tistory.com/42

References

[11] Swift로 자료구조, 알고리즘 공부하기 (1) - Binary Search : https://kor45cw.tistory.com/1

Thank you!