배열 검색(Array Search)

Binary Search

정의

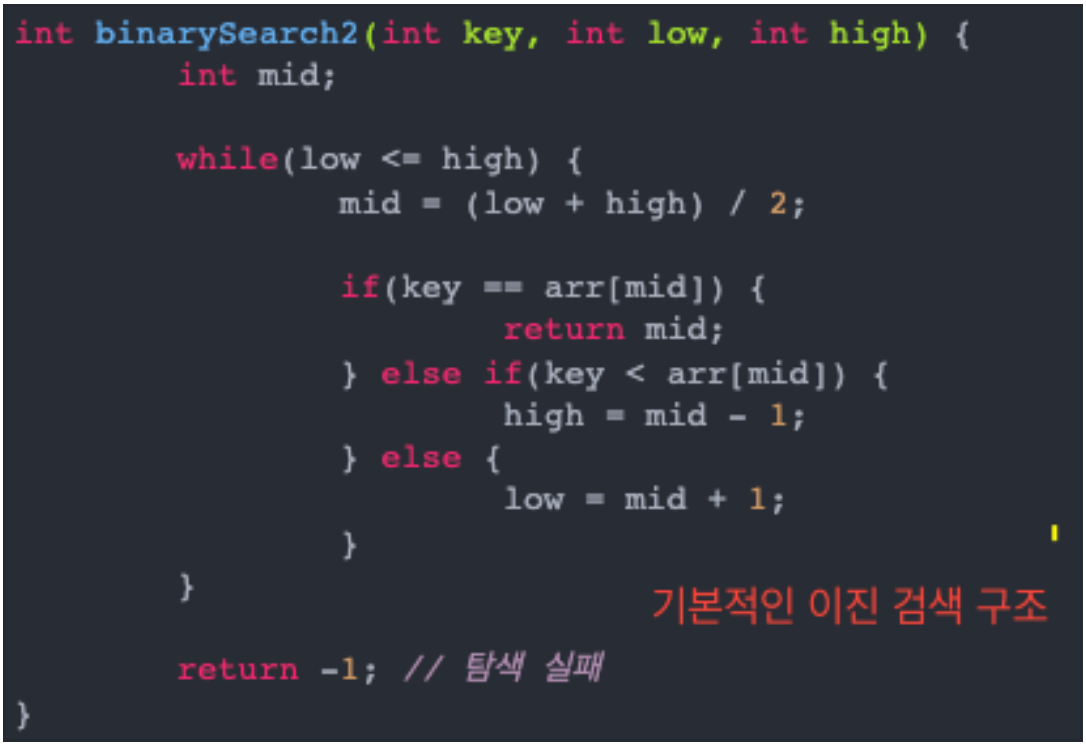
- 이진 검색은 요소가 오름차순 또는 내림차순으로 정렬된 배열에서 검색하는 알고리즘으로 배열의 중앙 값을 시작으로 key 값과 대소 비교를 통해 찾고자하는 요소에 접근한다.

- Arrays 클래스에 binarySearch 메서드로 구현되어 있으며, 제네렉 타입, 오브젝트 등 자바에서 제공하는 자료형 모두 사용 가능 하다는 특징을 가지고 있다.

- 이진 검색의 특성상 key 값과 같은 요소가 중복되어 배열에 존재 할 때, 앞에 위치한 요소를 배제 할 수 있다는 단점이 존재 한다. 이를 해결 하기 위해 while문 안에 inner for문을 넣어야 하는 등 요구 값이 커지기도 함

종료 조건

- 검색 할 값을 발견하지 못하고 이진 검색이 끝난 경우(실패)
- 검색 할 값과 같은 요소를 찾은 경우(성공)



```java
//--- 배열 a의 앞쪽 n개의 요소에서 key와 일치하는 요소를 이진검색 ---//
static int binSearchX(int[] a, int n, int key) {
    int pl = 0;          // 검색 범위 맨앞의 인덱스
    int pr = n - 1;      //    〃      맨끝의 인덱스

    do {
        int   pc = (pl + pr) / 2;    // 중앙요소의 인덱스
        if (a[pc] == key) {
            for ( ; pc > pl; pc--)  // key와 같은 맨앞의 요소를 검색합니다
                if (a[pc - 1] < key)
                    break;
            return pc;                              // 검색 성공
        } else if (a[pc] < key)
            pl = pc + 1;                          // 검색 범위를 앞쪽 절반으로 좁힘
        else
            pr = pc - 1;                          // 검색 범위를 뒤쪽 절반으로 좁힘
    } while (pl <= pr);

    return -1;       key값을 찾은 후 해당 인덱스      // 검색 실패
}                    이전까지 탐색하는 for문 추가
```

```java
public static int binarySearch(int[] a, int fromIndex, int toIndex,
                               int key) {
    rangeCheck(a.length, fromIndex, toIndex);
    return binarySearch0(a, fromIndex, toIndex, key);
}

// Like public version, but without range checks.
private static int binarySearch0(int[] a, int fromIndex, int toIndex,
                                 int key) {
    int low = fromIndex;
    int high = toIndex - 1;

    while (low <= high) {
        int mid = (low + high) >>> 1;
        int midVal = a[mid];

        if (midVal < key)
            low = mid + 1;                Arrays.binarySearch
        else if (midVal > key)
            high = mid - 1;
        else
            return mid; // key found
    }
    return -(low + 1);  // key not found.
}
```

추가

- Arrays.binarySearch에서 스트링 클래스를 사용 할 수 있는 이유는 스트링 클래스가 Comparable<T> 인터페이스와 compareTo 메서드를 구현하고 있기 때문이다.

```java
// 자연 정렬을 하려면 다음과 같은 패턴으로 클래스를 정의(예)
class A implements Comparable<A> {

    // 필드, 메소드 등

    public int compareTo(A c) {
        // this가 c보다 크면 양수를,
        // this가 c보다 작으면 음수를,
        // this가 c와 같으면 0을 반환합니다.
    }

    public boolean equals(Object c) {
        // this가 c와 같으면 true를,
        // this가 c와 같지 않으면 false를 반환합니다.
    }
}
```

- 자연 정렬이 되지 않은 배열에서 검색은 제네릭 메서드를 사용하는것인데, 이는 제네릭 메서드가 자료형에 구애를 받지 않기 때문에 가능하다. 다만, 배영 요소가 어떤 순서로 나열되어 있는지, 각 요소의 대소 관계를 어떻게 판단할 것인지 등은 binarySearch 메서드에 알려주어야 한다.

```java
// 클래스 X의 내부에서 COMPARATOR를 정의하는 방법(예)
import java.util.Comparator;

class X {
    // 필드, 메소드 등
    public static final Comparator<T> COMPARATOR = new Comp();

    private static class Comp implements Comparator<T> {
        public int compare(T d1, T d2) {
            // d1이 d2보다 크면 양의 값을,
            // d1이 d2보다 작으면 음의 값을,
            // d1이 d2와 같으면 0을 반환합니다.
        }
    }
}
```

- 보다 상세한 예제는 github/Algorithm/chap03/PhysExamSearch.java 참조