2019년 알고리즘

- HW 07 -

제출일자	2019.10.24.
이 름	장수훈
학 번	201402414
분 반	01

1

```java
public static void main(String[] args) throws IOException {
    // TODO Auto-generated method stub
    FileReader fr = new FileReader("C:\\Users\\micke\\eclipse-workspace\\ALG\\src\\data07_a.txt");
    BufferedReader br = new BufferedReader(fr);
    String read = br.readLine();
    String[] arrString = new String[1000000];
    arrString = read.split(", ");
    int[] array = new int[1000000];

    for (int i = 0; i < arrString.length; i++) {
        array[i] = Integer.parseInt(arrString[i]);
    }
    array = Sort(array,arrString.length);

    Scanner input = new Scanner(System.in);
    System.out.print("찾을 숫자를 입력하시오 : ");
    int temp = input.nextInt();
    int result = binarySearch(array, arrString.length, temp);
    if (result == -1) {
        System.out.println("배열에 존재하지 않습니다.");
    } else {
        System.out.println(temp + "는 " + result + "번째 인덱스에 존재합니다.");
    }
}

public static int[] Sort(int[] A, int size) {
    for (int j = 1; j < size; j++) {
        int key = A[j];
        int i = j - 1;

        while (i >= 0 && A[i] > key) {
            A[i + 1] = A[i];
            i = i - 1;
        }
        A[i + 1] = key;
    }
    return A;

}
```

```java
public static int binarySearch(int[] arr, int size, int n) {
    int first = 0;
    int last = size - 1;
    int mid = 0;
    while (first <= last) {
        mid = (first + last) / 2;
        if (arr[mid] == n) {
            return mid;
        }
        if (arr[mid] > n) {
            last = mid - 1;
        } else {
            first = mid + 1;
        }
    }
    return -1;
}
```

Loop-invariant

1. Initialization

첫 반복 전 만약 n이 배열A에 있으면 n은 A[first...last]에 있다.

2. Maintenance

k번째 반복 되었을 때 n < 배열 mid index값 일 때 A[first' ... mid-1] 아닐땐 A[mid+1 ... list'] 에 있다.

3. Termination

A[mid] = n 일 때 A[first ... last]에 있다.

2

```
public static int search(int[] A, int sizea, int[] B, int sizeb) {
    int start_a = 0;
    int start_b = 0;
    int last_a = sizea - 1;
    int last_b = sizeb - 1;

    while ((last_a - start_a != 1) && (last_b - start_b != 1)) {
        int mid_a = (start_a + last_a) / 2;
        int mid_b = (start_b + last_b) / 2;
        if (A[mid_a] > B[mid_b]) {
            last_a = mid_a;
            start_b = mid_b;
        }
        if (A[mid_a] < B[mid_b]) {
            start_a = mid_a;
            last_b = mid_b + 1;
        }
    }
    if (A[last_a] < B[last_b]) {
        return A[last_a];
    } else {
        return A[last_b];
    }
}
```

Loop-invariant

1. Initialization

loop들어간 시점에 A,B 배열은 모두 정렬되어있다. 변수들은 위에 선언되어 있는대로 설정되어 loop가 돈다.

2. Maintenance

k번째 반복 시 A배열과 B배열이 이진탐색처럼 k번 반으로 나뉘어 (A,B 배열은 아직도 정렬되어 있다) A의 가운데 인덱스, B의 가운데 인덱스의 크기비교 조건을 따져 이후 k+1번째도 조건에 만족시키게 반복한다.

3. Termination

loop를 종료후 가운데값을 반환한다. 배열의 인덱스가 0~1999 의 중앙 값 999번째 인덱스를 반환한다. 결과는 알고리즘의 목적에 부합한다.