알고리즘

과제번호	05주차
날 짜	2018.10.11
학 번	201302395
이 름	류경빈

과제 1	loop invariant 프로그램 증명

◆ **loop invariant 의사 코드**

```
1   doLoopInvariant(A[0..N-1], x) {
2           low = 0
3           high = length(A) - 1
4           // 〈Initialization〉
5           // Loop Invariant : low 〈 high 성립
6           // Loop Invariant : A[low] 〈 key 〈 A[high] 성립
7           while ( high 〉= low ){
8                   // 〈Maintenance〉
9                   // Loop Invariant : low 〈 high 성립
10                  // Loop Invariant : A[low] 〈 key 〈 A[high] 성립
11                  mid = ( low + high ) / 2
12                  if ( x == A[mid])
13                          return mid
14                  else if ( x 〈 array[mid] )
15                          high = mid - 1
16                  else
17                          low = mid + 1
18          }
19          // 〈Termination〉
20          // Loop Invariant : low 〈 high 성립
21          // Loop Invariant : A[low] 〈 key 〈 A[high] 성립
22          return -( low + 1 )
23  }
```

◆ **binary search 알고리즘 실행 시간**

loop invariant를 이용하고 서로 다른 원소가 올림차순으로 정렬되어 있는 배열 A에 x라는 값이 A[1]부터 A[n-1]사이에 존재할 때, A에서 x를 가지고 있는 index를 log n의 실행 시간 안에 찾을 수 있는 방법은 binary search(이진 탐색)이다.

binary search는 divide and conquer 알고리즘으로 이루어져 있기 때문에 탐색 시간 성능은 'T = K * logN'으로 **O(logN)**이다.

◆ **binary search 알고리즘 loop invariant 조건 만족**

binary search에서 loop invariant 조건을 만족하는 경우

〈Initialization〉

Loop Invariant : low 〈 high 성립

Loop Invariant : A[low] 〈 key 〈 A[high] 성립

〈Maintenance〉

Loop Invariant : low 〈 high 성립

Loop Invariant : A[low] 〈 key 〈 A[high] 성립

〈Termination〉

Loop Invariant : low 〈 high 성립

Loop Invariant : A[low] 〈 key 〈 A[high] 성립

◆ 알고리즘 구현 결과 화면

.java × LoopInvariant.java ×

```java
public class LoopInvariant {

    private int [] array;
    private int key;

    public LoopInvariant(int [] array, int key){
        this.array = array;
        this.key = key;
    }

    public int doLoopInvariant(){
        int mid;
        int low = 0;
        int high = array.length-1;

        // <Initialization>
        // Loop Invariant : low < high 성립
        // Loop Invariant : A[low] < key < A[high] 성립
        while (high >= low){
            // <Maintenance>
            // Loop Invariant : low < high 성립
            // Loop Invariant : A[low] < key < A[high] 성립
            mid = (high + low)/2;

            if (key == array[mid]){
                return mid;
            }
            else if (key < array[mid]){
                high = mid-1;
            }
            else {
                low = mid+1;
            }
        }
        // <Termination>
        // Loop Invariant : low < high 성립
        // Loop Invariant : A[low] < key < A[high] 성립
        return -(low+1);
    }
}
```

main.java × LoopInvariant.java ×

```java
public class main {

    public static void main (String [] args){
        int [] A = {1,2,3,4,5,6,7,8,9,10};
        LoopInvariant loopInvariant = new LoopInvariant(A,  key: 3);
        System.out.println(loopInvariant.doLoopInvariant());
    }

}
```

main

main ×

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_71.jdk/Contents/Home/bin/java ... <1 internal call>
2

Process finished with exit code 0
```