알고리즘 실습 보고서

-Loop Invariant-

전공 :컴퓨터공학과

분반 : 05반

학번 :201701988

이름 :김수빈

1. 실행 환경

본 실습은 Windows10 64bit, jdk 1.8.0_221, Eclipse EE가 설치된 환경에서 실행되었다

2. 과제 설명

- 출제된 과제에 대한 설명

이번 첫 번째 과제는 하나의 배열에서 X값을 찾아내는 pseudo code 작성이며, Loop invariant를 정의하여 자신의 알고리즘을 증명해야 한다.

두 번째 과제는 data06_a.txt와 data06_b.txt 파일을 읽어 각각을 다른 array로 저장한다.
서로 다른 database A와 B에서 A와 B의 중간 값을 찾아내 반환하는 것이다.
이 때, 두 개의 배열을 병합하지 않으며 시간 복잡도는 $O(\lg n)$로 제한한다.

3. 문제 해결 방법

과제 1.

아래 알고리즘의 PRE-CONDITION과 POST-CONDITION은 다음과 같다.
PRE-CONDITION : 배열 A는 오름차순 정렬되어 있고, X는 배열 A 안에 존재한다.
POST-CONDITION : 배열 A의 인덱스 s 에 있는 값은 X이다.
LOOP INVARIANT : 배열 A의 인덱스 p와 q 사이에 X가 존재한다.

<pseudo code>
배열 A에서 X를 찾는 알고리즘

```
FIND_X (A, X)
  p <- 0
  q <- N - 1
  s <- (p+q)/2

  WHILE (A[s]!=X){
        IF A[s] < X
             p <- s+1
        ELSE
             q <- s-1
        ENDIF
        s <- (p+q)/2
  }
RETURN s
```

Initialization

- pre-condition이 배열 A 내에 X가 존재한다는 것으로, 배열 A의 인덱스 0과 인덱스 N-1 사이에는 X가 존재한다는 것이 성립한다. 따라서 loop invariant는 참이다.

Maintenance

- 반복문 내 'p <- s+1' 또는 'q <- s-1' 가 X가 배열 A의 인덱스 p와 q 사이에 존재한다는 Invariant를 깨트릴 수 있지만, 그 전에 조건문으로 X가 A[s]보다 큰지, 작은지 확인하기 때문에 Invariant를 깨트리지 않는다.

즉, X는 인덱스 p와 q 사이에 있다는 loop invariant에서, p와 q의 중앙 인덱스 s의 값보다 X가 작은 경우 X는 p와 s-1 사이에 있는 것으로 loop invariant는 참이다.

Termination

- 반복문 탈출 조건은 A[s]=X인 경우이다. s는 p와 q의 중앙값으로 p와 q 사이에 X가 있다는 loop invariant를 만족한다.

$\{P\} \rightarrow I$
$WHILE\ (B)\{$
$\quad B \wedge I$
$\quad S$
$\quad I$
$\}$
$\neg B \wedge I$
$\neg B \wedge I \rightarrow \{R\}$

PRE-CONDITION으로부터 Invariant를 만족하는 Initial Assignment(p=0, q=N-1, s=(p+q)/2)를 한 후에, 조건 B { $A[s] \neq X$ }를 만족하면 반복문 안으로 들어가게 된다. Invariant와 조건 B를 만족하므로 반복문 안에서 초기 상태는 $B \wedge I$ 이다. 이때, S 연산 부분에서는 p = s+1 또는 q = s-1를 할 때 Invariant가 깨지게 되는데, 조건문 A[s] < X 검사를 통해 Invariant가 깨지는 것을 막아준다. A[s] < X 일 때는 X가 s+1와 q 사이에 있는 것이므로 p를 s+1로 이동시키고, A[s] > X일 때는 X가 p 와 s-1 사이에 있는 것으로 q를 s-1로 이동시키고 s를 다시 구한다. 따라서 p와 q 사이에 X가 있다는 Invariant는 깨지지 않으나, 조건 B를 다시 만족할지는 모르는 상태인 I 가 된다.
반복문을 빠져 나온다는 것은 조건 B를 만족하지 않는다는 것으로 $\neg B \wedge I$인 상태가 되며 이 상태는 POST CONDITION을 만족해야 한다.

조건 $\neg B$ 는 $\{A[s] = X\ (s = \frac{p+q}{2})\}$ 이고

Invariant I는 $\{p \leqq r \wedge r \leqq q, (A[r] = X)\}$ 이므로

$\neg B \wedge I$는 $\{r = s\ (s = \frac{p+q}{2}, A[r] = X)\}$ 가 된다. 따라서 A[s]=X가 성립한다.

과제 2.

data06_a.txt 파일과 data06_b.txt 파일을 읽어 split을 통해 ', ' 단위로 잘라 Integer로 파싱하여 각각 databaseA와 databaseB에 저장한다. 두 배열을 매개변수로 넣어 find_Median_loop 메소드를 호출한다. $O(\lg n)$의 조건을 맞추기 위해 이진 탐색을 수행한다.

PRE-CONDITION : 배열 databaseA와 배열 databaseB는 각각 오름차순 정렬되어 있고, databaseA와 databaseB의 중간 값은 M이고, 중간 값 M의 인덱스는 m이다.
$\{databaseA[m] = M \vee databaseB[m] = M\}$

POST-CONDITION : 반복문을 빠져나온 k에 대해서 A[k]는 M이다. $\{A[k] = M\}$

LOOP INVARIANT : M은 databaseA 배열의 인덱스 AStart 이상 AStart+N-1 이하 내의값과 databaseB 배열의 인덱스 BStart이상 BStart+N-1 이하 내의 값의 중간 값이다.

Initialization

- pre-condition으로부터

Invariant를 만족하도록 하는 Initial Assignment는 AStart=0, BStart=0,
N = (databaseA.length+databaseB.length)/2 이다.
M은 databaseA 배열의 인덱스 0 이상 N-1 이하 내의 값과 databaseB 배열의 인덱스 0 이상 N-1 이하 내의 값의 중간 값이다. 따라서 loop invariant가 성립한다.

Maintenance

- 반복문 내 `N = N- N/2` 가 M은 databaseA 배열의 인덱스 AStart 이상 AStart+N-1 이하 내의 값과 databaseB 배열의 인덱스 BStart 이상 BStart+N-1 이하 내의 값의 중간 값이라는 Invariant를 깨트릴 수 있지만, 그 전에 조건문으로 databaseA 인덱스 Astart부터 AStart+N-1 내, 중간 값 AStart+N/2-1과 databaseB 인덱스 Bstart부터 BStart+N-1 내 중간 값 BStart+N/2-1을 비교하여 아래 연산을 수행하므로 invariant가 깨지지 않는다.

A쪽이 더 클 경우 중간 값 M은 AStart+N/2 ~ AStart+N-1와 (A쪽을 반쪽 쪼갠 뒷부분) BStart ~ BStart+N-N/2-1 의 중간 값이므로, BStart = BStart+N/2을 먼저 수행하여 invariant를 유지하고

B쪽이 더 클 경우 중간 값 M은 BStart+N/2 ~ BStart+N-1 와(B쪽을 반쪽 쪼갠 뒷부분) AStart ~ AStart+N-N/2-1 사이에 있기 때문에 AStart = AStart+N/2을 먼저 수행하여 invariant를 유지한다.

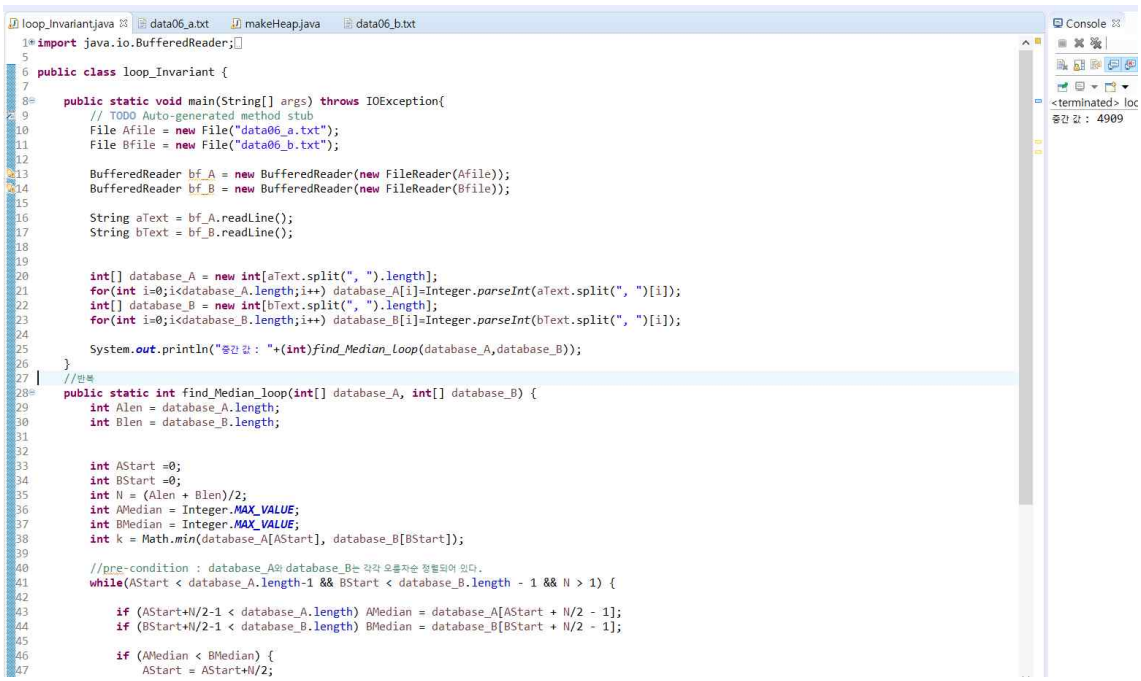
위 연산을 하지 않는 다면 M은 databaseA 배열의 인덱스 AStart 이상 AStart+N-1 이하 내의 값과 databaseB 배열의 인덱스 BStart 이상 BStart+N-1 이하 내의 값의 중간 값이라는 invariant가 databaseA 배열의 인덱스 AStart 이상 AStart+N/2—1 (A쪽을 반쪽 쪼갠 앞부분) 이하 내의 값과 databaseB 배열의 인덱스 BStart 이상 BStart+N/2-1 이하 내의 값(B쪽을 반쪽 쪼갠 앞부분) 의 중간 값으로 변형되어 invariant가 깨진다.

Termination

- 반복문을 빠져 나온 k에 대해서 post condition을 만족하므로 A[k]는 M이다. $\{A[k] = M\}$
반복문을 빠져나오려면 AStart 또는 BStart가 배열의 끝 인덱스거나 N이 1이 되는 경우이다.

반복문 안에서 마지막에 AStart가 배열의 끝 인덱스보다 크거나 같은 경우 k는
databaseB의 BStart+N-1 값이 되며, 따라서 이 경우에 invariant는 유지된다.
BStart가 배열의 끝 인덱스보다 크거나 같은 경우 k는
databaseA의 AStart+N-1 값이 되며, 따라서 이 경우에도 invariant는 유지된다.
N이 1이 되는 경우에는 AStart ~ AStart 사이 또는 BStart~BStart이므로 databaseB의 BStart 또는 databaseA의 AStart 값이 된다. 따라서 loop invariant는 만족된다.

4. 결과 화면




```java
import java.io.BufferedReader;

public class loop_Invariant {

    public static void main(String[] args) throws IOException{
        // TODO Auto-generated method stub
        File Afile = new File("data06_a.txt");
        File Bfile = new File("data06_b.txt");

        BufferedReader bf_A = new BufferedReader(new FileReader(Afile));
        BufferedReader bf_B = new BufferedReader(new FileReader(Bfile));

        String aText = bf_A.readLine();
        String bText = bf_B.readLine();


        int[] database_A = new int[aText.split(", ").length];
        for(int i=0;i<database_A.length;i++) database_A[i]=Integer.parseInt(aText.split(", ")[i]);
        int[] database_B = new int[bText.split(", ").length];
        for(int i=0;i<database_B.length;i++) database_B[i]=Integer.parseInt(bText.split(", ")[i]);

        System.out.println("중간 값 : "+(int)find_Median_loop(database_A,database_B));
    }
    //반복
    public static int find_Median_loop(int[] database_A, int[] database_B) {
        int Alen = database_A.length;
        int Blen = database_B.length;


        int AStart =0;
        int BStart =0;
        int N = (Alen + Blen)/2;
        int AMedian = Integer.MAX_VALUE;
        int BMedian = Integer.MAX_VALUE;
        int k = Math.min(database_A[AStart], database_B[BStart]);

        //pre-condition : database_A와 database_B는 각각 오름차순 정렬되어 있다.
        while(AStart < database_A.length-1 && BStart < database_B.length - 1 && N > 1) {

            if (AStart+N/2-1 < database_A.length) AMedian = database_A[AStart + N/2 - 1];
            if (BStart+N/2-1 < database_B.length) BMedian = database_B[BStart + N/2 - 1];

            if (AMedian < BMedian) {
                AStart = AStart+N/2;
```

5. 느낀점 및 고찰

평소에 고려하지 않고 프로그래밍을 해 와서 그런지, invariant를 설정하는 부분도 어려웠고, 증명이 너무 어려웠습니다. 시간 복잡도 제한이 있는 부분도 어렵게 느껴졌습니다.