

자료 구조 Lab 013 :

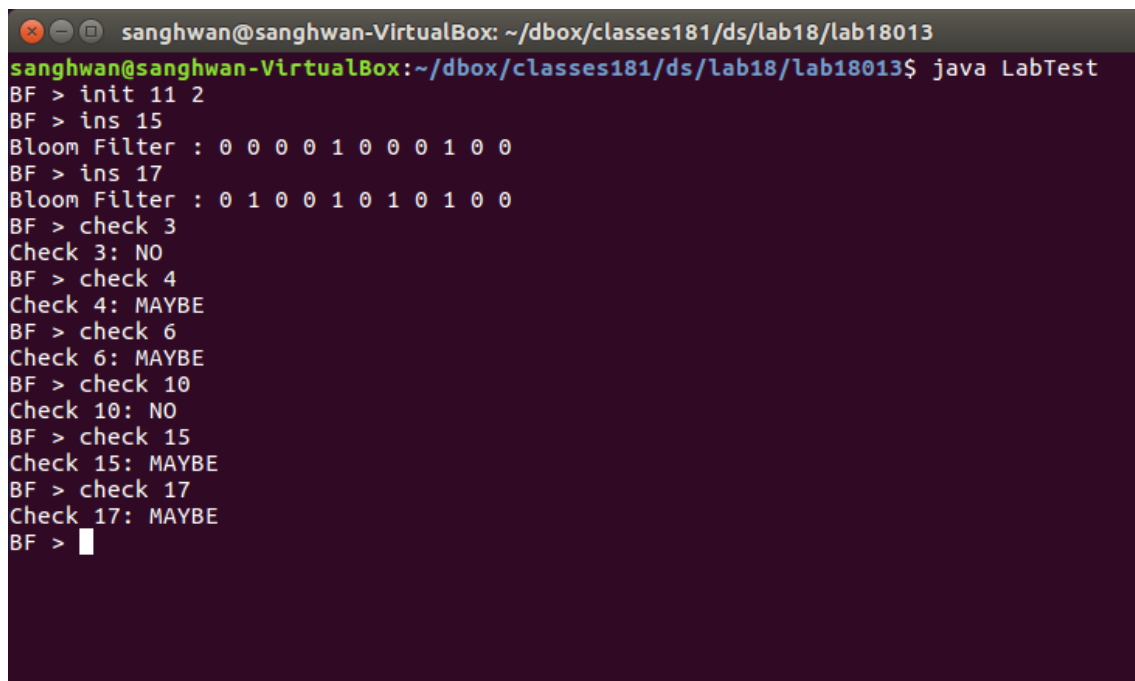
lab013.zip 파일 : LabTest.java lab013.java lab.in lab.out

제출

lab013.java 를 학번.java 로 변경하여 이 파일 한 개만 제출할 것.

이번 실습은 Bloom Filter의 핵심 기능을 구현하는 것이다. 사용자로부터 Bloom Filter의 크기 m 과 hash function의 수 h 를 입력 받은 다음, 미리 정해져 있는 hash function을 활용하여 Bloom filter의 bit를 설정하고, 사용자로부터의 질문에 대답한다.

수행 예는 다음과 같다.

A terminal window titled 'sanghwan@sanghwan-VirtualBox: ~/dbbox/classes181/ds/lab18/lab18013' shows the execution of a Java program 'LabTest'. The user enters commands to initialize the Bloom Filter with m=11 and h=2, insert keys 15 and 17, and check for the presence of keys 3, 4, 6, 10, 15, and 17. The output shows the bit patterns for the filter and the results of the checks: NO for 3 and 10, and MAYBE for 4, 6, 15, and 17.

```
sanghwan@sanghwan-VirtualBox: ~/dbbox/classes181/ds/lab18/lab18013
sanghwan@sanghwan-VirtualBox:~/dbbox/classes181/ds/lab18/lab18013$ java LabTest
BF > init 11 2
BF > ins 15
Bloom Filter : 0 0 0 0 1 0 0 0 1 0 0
BF > ins 17
Bloom Filter : 0 1 0 0 1 0 1 0 1 0 0
BF > check 3
Check 3: NO
BF > check 4
Check 4: MAYBE
BF > check 6
Check 6: MAYBE
BF > check 10
Check 10: NO
BF > check 15
Check 15: MAYBE
BF > check 17
Check 17: MAYBE
BF > 
```

사용자가 사용하는 명령어의 syntax는 다음과 같다. Main() 함수에 정의되어 있다.

- `init m h`

m 은 Bloom filter의 메모리 크기이고, h 는 해시 함수의 수이다.

- `ins key`

Bloom filter에 주어진 key를 입력한다.

- check key

key를 Bloom filter로 탐색을 하여 MAYBE나 NO의 대답을 한다.

이 내용을 구현하기 위해 다음 두 함수를 구현해야 한다.

- `Void Insert(int key);`

주어진 key를 바탕으로 h 개의 해시 함수를 실행하여 나온 비트를 1로 설정한다. 이 Bloom filter의 메모리는 `bfArray[]` 배열로 구현하였고, 이 배열의 한 원소가 하나의 비트를 의미한다. 즉 `bfArray[0]`은 0번 비트를 의미하고, `bfArray[1]`은 1번 비트를 의미한다.

- `Bool Check(int key);`

주어진 key가 존재하는지를 검사하여 NO 인 경우는 false를 return하고, MAYBE 인 경우는 true를 return 한다.

해시 함수 `f()`는 `BloomFilter` 클래스에 이미 정의되어 있다.

```
int f(int h, int k) {return (k * (h + 1)) % m; }
```

`f0(key)`는 `f(0, key)`를 호출하면 되고, `f1(key)`는 `f(1, key)`를 호출하면 된다. 즉, h개의 해시 함수는 각각 `f(0, key)`, `f(1, key)`, ..., `f(h-1, key)`가 된다.