first-bad-version

```
package algorithm.binarysearch;

/**
 * https://leetcode.com/problems/first-bad-version/
 *
 You are a product manager and currently leading a team to
 develop a new product. Unfortunately,
 the latest version of your product fails the quality check.
 Since each version is developed based on the previous version,
 all the versions after a bad version are also bad.

 Suppose you have n versions [1, 2, ..., n] and
 you want to find out the first bad one,
 which causes all the following ones to be bad.

 You are given an API bool isBadVersion(version)
 which will return whether version is bad.
 Implement a function to find the first bad version.
 You should minimize the number of calls to the API.

 * @author xiaobaoqiu  Date: 16-7-11 Time:  下午10:48
 */
public class FirstBadVersion {
    public static void main(String[] args) {
        int n = 2126753390;
        System.out.println(firstBadVersion(n));
    }

    /**
     * 思路：二分, n-1 good && n bad --> n
     * 极端情况：1 bad
     *
     * 36 ms
     * Your runtime beats 3.85% of java submissions.
     */
    public static int firstBadVersion(int n) {
        int low = 1, high = n, mid;
        while (low <= high) {
//            mid = (low + high) >> 1;  //over flow
            mid = (low & high) + ((low ^ high) >> 1);
            System.out.println(low + " -- " + mid + " -- " + high);
            boolean midV = isBadVersion(mid);
            if (!midV) low = mid + 1;
            else if (!isBadVersion(mid - 1)) return mid;    //n-1 good && n
            else high = mid -1;
        }

        return 1;
    }

    /* The isBadVersion API is defined in the parent class VersionControl.
    private static boolean isBadVersion(int version) {
        return version >= 1702766719;
```

```
        }
    }
```