add-binary

```
package algorithm.math;

/**
 * https://leetcode.com/problems/add-binary/
 *
 Given two binary strings, return their sum (also a binary string).

 For example,
 a = "11"
 b = "1"
 Return "100".

 * @author xiaobaoqiu  Date: 16-7-8 Time:  下午10:23
 */
public class AddBinary {
    public static void main(String[] args) {
        String a = "1010", b = "1011";
        System.out.println(addBinary(a, b));    //10101
    }

    /**
     * 3 ms
     * Your runtime beats 89.98% of java submissions
     */
    public static String addBinary(String a, String b) {
        StringBuilder builder = new StringBuilder();
        int i = a.length() - 1, j = b.length() - 1;
        int carry = 0, temp;
        for (; i >= 0 && j >= 0; i--, j--) {
            temp = a.charAt(i) - '0' + b.charAt(j) - '0' + carry;
            carry = 0;
            if (temp >= 2) {carry = 1;temp -= 2;}
            builder.append((char)(temp + '0'));
        }
        while (i >= 0) {
            temp = a.charAt(i--) - '0' + carry;
            carry = 0;
            if (temp >= 2) {carry = 1;temp -= 2;}
            builder.append((char)(temp + '0'));
        }
        while (j >= 0) {
            temp = b.charAt(j--) - '0' + carry;
            carry = 0;
            if (temp >= 2) {carry = 1;temp -= 2;}
            builder.append((char)(temp + '0'));
        }
        if (carry == 1) builder.append('1');
        return builder.reverse().toString();
    }
}
```

add-digits

```
package algorithm.math;

/**
 * https://leetcode.com/problems/add-digits/
 *
 Given a non-negative integer num, repeatedly add all its digits until the

 For example:

 Given num = 38, the process is like: 3 + 8 = 11, 1 + 1 = 2. Since 2 has or

 Follow up:
 Could you do it without any loop/recursion in O(1) runtime?
 *
 * @author xiaobaoqiu  Date: 16-5-17 Time:  下午9:22
 */
public class AddDigits {
    public static void main(String[] args) {
        for(int i = 0; i< 100; i++)
            System.out.println(i + " --> " + addDigits_1(i));
    }

    /**
     * 2 ms
     * Your runtime beats 20.66% of java submissions.
     *
     * 规律:
     * 0 -> 0
     *
     * 1 -> 1
     * ...
     * 9 --> 9
     *
     * 10 -> 1
     * 11 -> 2
     * 12 -> 3
     * 13 -> 4
     * 14 -> 5
     * 15 -> 6
     * 16 -> 7
     * 17 -> 8
     * 18 -> 9
     *
     * 19 -> 1
     * 20 -> 2
     * 21 -> 3
     * ...
     * 27 -> 9
     *
     * 28 -> 1
     *
     * 即 9 为一个循环节
     */
    public static int addDigits_1(int num) {
        if (num == 0) return 0;
        int ret = num % 9;
        return (ret == 0) ? 9 : ret;
    }
}
```

bulb-switcher

```
package algorithm.math;

/**
 * https://leetcode.com/problems/bulb-switcher/
 *
 There are n bulbs that are initially off.
 You first turn on all the bulbs.
 Then, you turn off every second bulb.
 On the third round, you toggle every third bulb (turning on if it's off o
 For the ith round, you toggle every i bulb.
 For the nth round, you only toggle the last bulb. Find how many bulbs are

 Example:

 Given n = 3.

 At first, the three bulbs are [off, off, off].
 After first round, the three bulbs are [on, on, on].
 After second round, the three bulbs are [on, off, on].
 After third round, the three bulbs are [on, off, off].

 So you should return 1, because there is only one bulb is on.

 * @author xiaobaoqiu  Date: 16-5-22 Time:  下午6:40
 */
public class BulbSwitcher {
    public static void main(String[] args) {
//        int n = 2;
//        int n = 4;
//        int n = 100;
        int n = 100000000;
//        int n = Integer.MAX_VALUE - 1;
        System.out.println(bulbSwitch(n));
    }

    /**
     * 第 i 个被翻转的次数为: 1~i中因子的个数，如果个数为奇数，则是打开，偶数则是关闭
     * 问题转化为 求i的因子的个数
     *
     * 其实更进一步，我们压根不需要知道具体的因子个数，我们只需要知道因子个数的奇偶性
     * 这个就很简单，完全平方数的因子个数为奇数 (如100)
     *
     *
     * 这个解法是对的，结果也是对的，但是超时～～～～
     */
    public static int bulbSwitch(int n) {
        int ret = 1;
        int sqrt;
        for (int i = 2; i <= n; i++) {
            sqrt = (int)Math.sqrt(i);
            if (sqrt * sqrt == i) ret += 1;
        }
        return ret;
    }
    /**
     * 其实问题转化为求 1～n 中完全平方数的个数
     * 这个结果就是：sqrt(n)
     * 因为 1～sqrt(n) 中的每个数x，其平方都是完全平方数
     *
     * 0 ms
     * Your runtime beats 26.04% of java submissions
     */
    public static int bulbSwitch_1(int n) {
        return (int)Math.sqrt(n);
    }
}
```

climbing-stairs

```
package algorithm.math;

/**
 * https://leetcode.com/problems/climbing-stairs/
 *
 You are climbing a stair case. It takes n steps to reach to the top.

 Each time you can either climb 1 or 2 steps.
 In how many distinct ways can you climb to the top?

 * @author xiaobaoqiu  Date: 16-5-27 Time:  下午11:25
 */
public class ClimbingStairs {
    public static void main(String[] args) {

    }

    /**
     * f(1) --> 1
     * f(2) --> 2
     * f(3) --> 先走1步,剩下两步的走法就是 f(2),或者先走两步,剩下的走法就是 f(1)
     * ...同理
     * f(n) --> 先走1步,剩下两步的走法就是 f(n-1),或者先走两步,剩下的走法就是 f(n-2
     *
     * 因此: f(n) = f(n-1) + f(n-2)
     *
     * 0 ms
     * Your runtime beats 13.04% of java submissions
     */
    public int climbStairs(int n) {
        if (n == 1) return 1;
        if (n == 2) return 2;
        int prepre = 1, pre = 2, ret = 0;
        for (int i = 3; i<= n; i++) {
            ret = pre + prepre;
            prepre = pre;
            pre = ret;
        }

        return ret;
    }
}
```

excel-sheet-column-number

```
package algorithm.math;

/**
 * https://leetcode.com/problems/excel-sheet-column-number/
 *
 Related to question Excel Sheet Column Title

 Given a column title as appear in an Excel sheet, return its corresponding

 For example:

 A -> 1
 B -> 2
 C -> 3
 ...
 Z -> 26
 AA -> 27
 AB -> 28

 * @author xiaobaoqiu  Date: 16-5-22 Time:  上午11:19
 */
public class ExcelSheetColumnNumber {
    public static void main(String[] args) {
        String s = "AA";
        System.out.println(titleToNumber(s));
    }

    /**
     * 相当于 26 进制的数
     *
     * 2 ms
     * Your runtime beats 76.30% of java submission
     */
    public static int titleToNumber(String s) {
        int num = 0, base = 1;
        for (int i = s.length() - 1; i >= 0; i--) {
            num += (s.charAt(i) - 'A' + 1) * base;
            base *= 26;
        }
        return num;
    }
}
```

excel-sheet-column-title

```
package algorithm.math;

/**
 * https://leetcode.com/problems/excel-sheet-column-title/
 *
 Given a positive integer,
 return its corresponding column title as appear in an Excel sheet.

 For example:

 1 -> A
 2 -> B
 3 -> C
 ...
 26 -> Z
 27 -> AA
 28 -> AB

 * @author xiaobaoqiu  Date: 16-7-11 Time:  下午11:16
 */
public class ExcelSheetColumnTitle {
    public static void main(String[] args) {
        int[] cases = new int[] {1, 2, 25, 26, 27, 28};
        for (int v : cases) {
            System.out.println(v + " --> " + convertToTitle(v));
        }
    }

    /**
     * 思路：26进制
     * 1 --> A
     * 2 --> B
     * 26 --> Z
     *
     * 0 ms
     * Your runtime beats 9.50% of java submissions
     */
    public static String convertToTitle(int n) {
        StringBuilder builder = new StringBuilder();
        do {
            builder.append((char) ('A' + ((n - 1) % 26)));
            n = (n - 1) / 26;
        } while (n > 0);
        return builder.reverse().toString();
    }
}
```

factorial-trailing-zeroes

```
package algorithm.math;

/**
 * https://leetcode.com/problems/factorial-trailing-zeroes/
 *
 Given an integer n, return the number of trailing zeroes in n!.

 Note: Your solution should be in logarithmic time complexity.

 * @author xiaobaoqiu  Date: 16-6-28 Time:  下午10:13
 */
public class FactorialTrailingZeroes {
    public static void main(String[] args) {
        int n = 6;
        System.out.println(fn(n));
        System.out.println(trailingZeroes(n));

        n = 10;
        System.out.println(fn(n));
        System.out.println(trailingZeroes(n));
    }

    /**
     * 1-n 中 5 的因子个数
     * 如 6! 中 还有一个 5，则其末尾有一个 0
     *
     * 2 ms
     * Your runtime beats 4.30% of java submissions.
     */
    public static int trailingZeroes(int n) {
        int count = 0,next;
        while(n >= 5) {
            next = n/5;
            count += next;
            n = next;
        }
        return count;
    }

    public static int fn(int n) {
        int s = 1;
        while(n > 1) {
            s *= n;
            n--;
        }
        return s;
    }
}
```

happy-number

```java
package algorithm.math;

import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

/**
 * https://leetcode.com/problems/happy-number/
 * <p/>
 * <p/>
 * Write an algorithm to determine if a number is "happy".
 * <p/>
 * A happy number is a number defined by the following process:
 * Starting with any positive integer, replace the number by the sum of the
 * and repeat the process until the number equals 1 (where it will stay),
 * or it loops endlessly in a cycle which does not include 1.
 * Those numbers for which this process ends in 1 are happy numbers.
 * <p/>
 * Example: 19 is a happy number
 * <p/>
 * 1^2 + 9^2 = 82
 * 8^2 + 2^2 = 68
 * 6^2 + 8^2 = 100
 * 1^2 + 0^2 + 0^2 = 1
 *
 * @author xiaobaoqiu  Date: 16-5-27 Time:  下午10:44
 */
public class HappyNumber {
    public static void main(String[] args) {
        for (int i = 1; i< 100; i++) {
            if (isHappy(i)) System.out.println(i);
        }
    }

    /**
     * 模拟
     * 6 ms
     * Your runtime beats 26.98% of java submissions
     */
    public static boolean isHappy(int n) {
        Set<Integer> history = new HashSet<Integer>();
        while (n != 1) {
            int sum = 0, last = 0;
            while (n > 0) {
                last = n % 10;
                sum += (last * last);
                n /= 10;
            }
            n = sum;
            if(history.contains(n)) break;
            history.add(n);
        }

        return n == 1;
    }

    /**
     * https://en.wikipedia.org/wiki/Happy_number
     * 任何非 happy 的数,最终都会进到一个循环中
     *
     * 2 ms
     * Your runtime beats 84.07% of java submissions
     */
    public static boolean isHappy_1(int n) {
        int[] breaker = new int[] {4, 16, 20, 37, 42, 58, 89, 145};
        while (n != 1) {
            int sum = 0, last = 0;
            while (n > 0) {
                last = n % 10;
                sum += (last * last);
                n /= 10;
            }
            n = sum;
            if(Arrays.binarySearch(breaker, n) >= 0) break;
        }

        return n == 1;
    }
}
```

palindrome-number

```
package algorithm.math;

/**
 * https://leetcode.com/problems/palindrome-number/
 *
 Determine whether an integer is a palindrome. Do this without extra space

 click to show spoilers.

 Some hints:
 Could negative integers be palindromes? (ie, -1)

 If you are thinking of converting the integer to string, note the restrict

 You could also try reversing an integer. However, if you have solved the p

 There is a more generic way of solving this problem.

 * @author xiaobaoqiu  Date: 16-6-28 Time:  下午10:51
 */
public class PalindromeNumber {
    public static void main(String[] args) {
        System.out.println((int)Math.log10(101));
        int x = 654321;
//        System.out.println(isPalindrome_1(x));
//        x = 1;
//        System.out.println(isPalindrome_1(x));
//        x = 101;
//        System.out.println(isPalindrome_1(x));
//        x = 1000021;
//        System.out.println(isPalindrome_1(x));
//        x = 10;
//        System.out.println(isPalindrome_1(x));
        x = 121;
        System.out.println(isPalindrome_1(x));
    }

    /**
     * 15 ms
     * Your runtime beats 21.84% of java submissions.
     */
    public static boolean isPalindrome(int x) {
        if (x < 0) return false;

        int i = 0, j = (int) Math.log10(x);
        int iBase = 1, jBase = (int)Math.pow(10, j);
        while( i < j) {
            //i-th dight
            int iDight = (x / iBase) % 10;

            //j-th dight
            int jDight = (x / jBase) % 10;

            if (iDight != jDight) return false;
            i++;
            j--;
            iBase *= 10;
            jBase /= 10;
        }
        return true;
    }

    /**
     * 构造 高半部分 和 低半部分
     *
     * 11 ms
     * Your runtime beats 76.50% of java submissions
     */
    public static boolean isPalindrome_1(int x) {
        if (x < 0) return false;
        if (x < 10) return true;
        if (x < 100) return x % 11 == 0;
        if (x % 10 == 0) return false;

        int low = 0;
        while (x > low) {
            low = low * 10 + x % 10;
            x /= 10;
        }
        return (x == low) || (x == low / 10);
    }
}
```

power-of-three

```
package algorithm.math;

import java.util.Arrays;
import java.util.HashSet;

/**
 * https://leetcode.com/problems/power-of-three/
 *
 Given an integer, write a function to determine if it is a power of three

 Follow up:
 Could you do it without using any loop / recursion?

 * @author xiaobaoqiu  Date: 16-5-26 Time:  下午9:03
 */
public class PowerOfThree {
    public static void main(String[] args) {
        System.out.println((int) Math.pow(3, (int)(Math.log(Integer.MAX_VAL
        int n = 27;
//        System.out.println(isPowerOfThree(n));
        System.out.println(isPowerOfThree_1(n));

    }

    /**
     * 循环
     * log(n)
     *
     * 20 ms
     * Your runtime beats 20.47% of java submissions
     */
    public static boolean isPowerOfThree(int n) {
        if (n <= 0) return false;
        while (n % 3 == 0) n /= 3;
        return n == 1;
    }

    /**
     * int范围内的 3 的幂 的最大值
     * (int) Math.pow(3, (int)(Math.log(Integer.MAX_VALUE)/Math.log(3.0)))
     *
     * 则 3 的幂一定是这个数的因子
     *
     * 19 ms
     * Your runtime beats 33.19% of java submissions.
     */
    public static boolean isPowerOfThree_1(int n) {
        return n > 0 && n < 1162261467 && 1162261467 % n == 0;
    }

    /**
     * 列出所有的 3的幂
     *
     * 20 ms
     * Your runtime beats 20.47% of java submissions
     */
    public static boolean isPowerOfThree_2(int n) {
        int[] allPowerOfThree = new int[]{1, 3, 9, 27, 81, 243, 729, 2187,
        return Arrays.binarySearch(allPowerOfThree, n) >= 0;
    }

    /**
     * 55 ms
     */
    public static boolean isPowerOfThree_3(int n) {
        HashSet<Integer> set = new HashSet<Integer>(Arrays.asList(1, 3, 9,
        return set.contains(n);
    }

    /**
     * 3进制
     *
     * 83 ms
     */
    public static boolean isPowerOfThree_4(int n) {
        return Integer.toString(n, 3).matches("10*");
    }

    /**
     * 19ms
     */
    public static boolean isPowerOfThree_5(int n) {
        switch (n) {
            case 1:
            case 3:
            case 9:
            case 27:
            case 81:
            case 243:
            case 729:
            case 2187:
            case 6561:
            case 19683:
            case 59049:
            case 177147:
            case 531441:
            case 1594323:
            case 4782969:
            case 14348907:
            case 43046721:
            case 129140163:
            case 387420489:
            case 1162261467:
                return true;
            default:
                return false;
        }
    }
}
```

rectangle-area

```
package algorithm.math;

/**
 * https://leetcode.com/problems/rectangle-area/
 *
 Find the total area covered by two rectilinear rectangles in a 2D plane.

 Each rectangle is defined by its bottom left corner and top right corner a

 https://leetcode.com/static/images/problemset/rectangle_area.png

 Assume that the total area is never beyond the maximum possible value of i

 * @author xiaobaoqiu  Date: 16-7-4 Time:  下午10:45
 */
public class RectangleArea {
    public static void main(String[] args) {

    }

    public int computeArea(int A, int B, int C, int D, int E, int F, int G,
        int base = (C - A) * (D - B) + (G - E) * (H - F);
        //不相交
        if (C <= E || A >= G || B >= H || D <= F) {
            return base;
        }

        //TODO
        return 0;
    }
}
```

reverse-integer

```
package algorithm.math;

import org.omg.CORBA.INTERNAL;

/**
 * https://leetcode.com/problems/reverse-integer/
 *
 Reverse digits of an integer.

 Example1: x = 123, return 321
 Example2: x = -123, return -321

 click to show spoilers.

 Have you thought about this?
 Here are some good questions to ask before coding.
 Bonus points for you if you have already thought through this!

 If the integer's last digit is 0, what should the output be?
 ie, cases such as 10, 100.

 Did you notice that the reversed integer might overflow?
 Assume the input is a 32-bit integer,
 then the reverse of 1000000003 overflows.
 How should you handle such cases?

 For the purpose of this problem,
 assume that your function returns 0 when the reversed integer overflows.

 * @author xiaobaoqiu  Date: 16-7-8 Time:  下午11:45
 */
public class ReverseInteger {
    public static void main(String[] args) {
        int cases[] = new int[] {1, 10, 100, 1001, 0, -1, -101, Integer.MA
        for (int c : cases) {
            System.out.println(c + " -->" + reverse(c));
        }
    }

    /**
     * 2 ms
     * Your runtime beats 49.00% of java submissions.
     */
    public static int reverse(int x) {
        long v = 0;
        while (x != 0) {
            int d = x % 10;
            x /= 10;
            v = v * 10 + d;
        }
        if (v > Integer.MAX_VALUE || v < Integer.MIN_VALUE) return 0;
        return (int)v;
    }
}
```

string-to-integer-atoi

```
package algorithm.math;

/**
 * https://leetcode.com/problems/string-to-integer-atoi/
 *
 Implement atoi to convert a string to an integer.

 * @author xiaobaoqiu  Date: 16-7-12 Time:  下午10:34
 */
public class StringToIntegerAtoi {
    public static void main(String[] args) {
        String[] cases = new String[] {
//                "123",      //123
//                "-123",     //-123
//                "00123",    //123
//                "+-2",      //0
//                "b11228552307",     //0
                "9223372036854775809", //2147483647
        };
        for (String v : cases) {
            System.out.println(v + " --> " + myAtoi(v));
        }
    }

    /**
     * trim
     * first char is digit or + or -
     * fail fast while result out of integer range
     * ignore ended invalid char
     *
     * 5 ms
     * Your runtime beats 16.48% of java submissions
     */
    public static int myAtoi(String str) {
        if (str == null || str.isEmpty()) return 0;
        str = str.trim();
        if (str.isEmpty()) return 0;

        int index = 0;
        if (str.charAt(index) < '0' && str.charAt(index) > '9' && str.charA

        long sum = 0;
        boolean negative = false;
        if (str.charAt(index) == '+' || str.charAt(index) == '-') {
            if (str.charAt(index) == '-') negative = true;
            ++index;
        }
        while (index < str.length() && str.charAt(index) >= '0' && str.char
            int d = str.charAt(index) - '0';
            sum = sum * 10 + d;
            if (!negative && sum > Integer.MAX_VALUE) return Integer.MAX_VA
            else if (negative && -sum < Integer.MIN_VALUE) return Integer.M
            ++index;
        }
        if (negative) sum = -sum;
        return (int) sum;
    }
}
```

ugly-number

```
package algorithm.math;

/**
 * https://leetcode.com/problems/ugly-number/
 *
 Write a program to check whether a given number is an ugly number.

 Ugly numbers are positive numbers whose prime factors only include 2, 3, 5

 Note that 1 is typically treated as an ugly number.

 * @author xiaobaoqiu  Date: 16-5-27 Time:  下午10:39
 */
public class UglyNumber {
    public static void main(String[] args) {
        System.out.println(isUgly(6));
        System.out.println(isUgly(17));
    }

    /**
     * 2 ms
     * Your runtime beats 17.06% of java submissions
     */
    public static boolean isUgly(int num) {
        if (num == 0) return false;
        while (num % 2 == 0) num /= 2;
        while (num % 3 == 0) num /= 3;
        while (num % 5 == 0) num /= 5;
        return num == 1;
    }
}
```