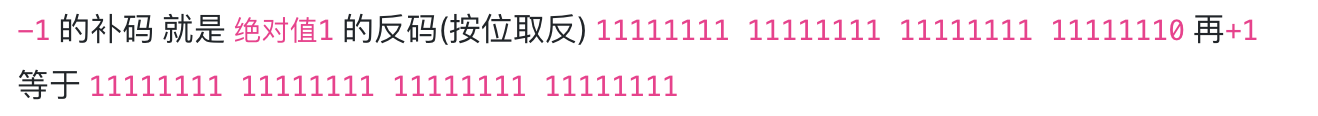
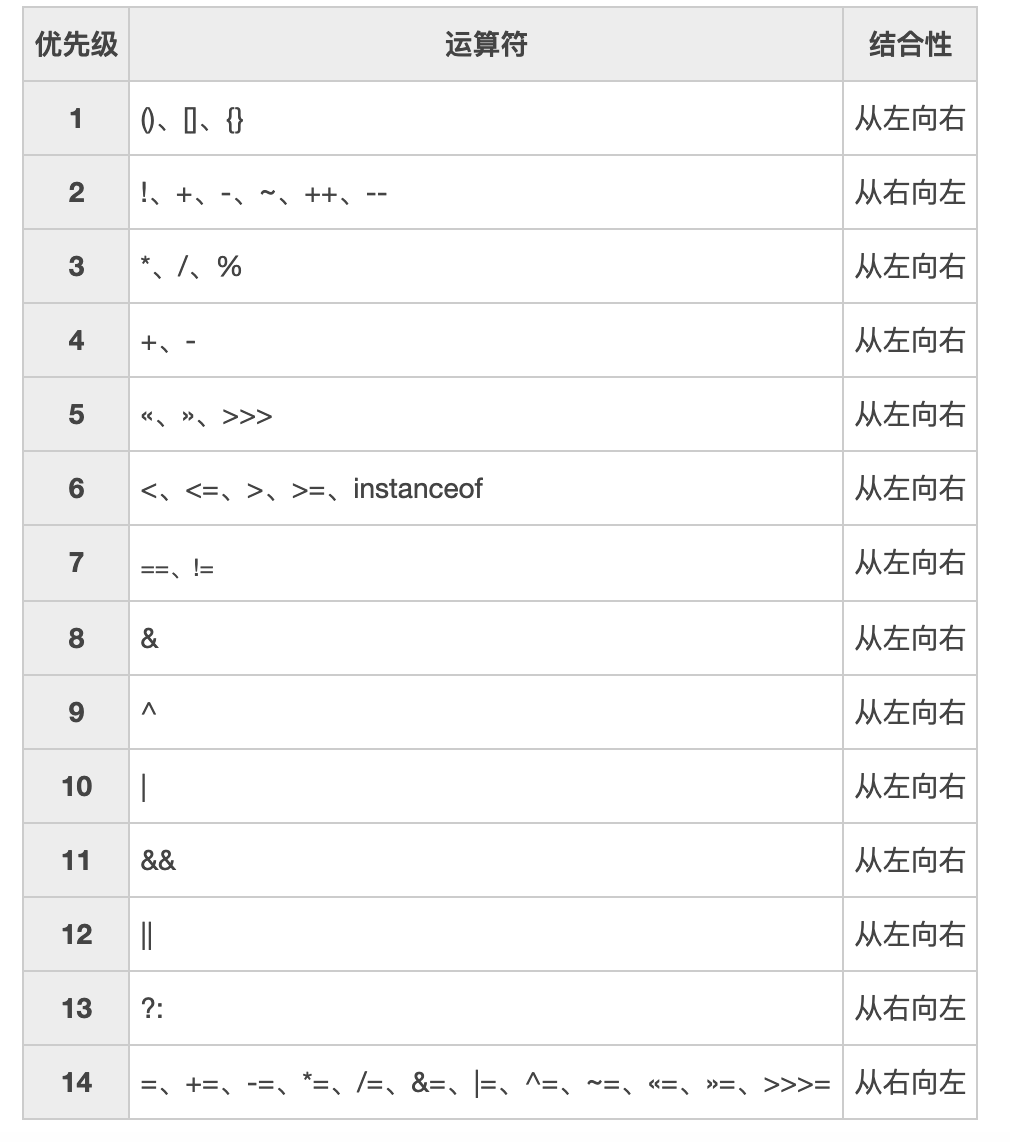
**今日所学：**

* join连接表时on和where的区别，其中on时连接表时的过滤条件，如果时left join，不管on是否符合，对会返回左表的全部数据，从而形成一个临时表；而where是对临时表进行条件判断，不符合条件的都会被过滤。
* java中int是由有正负之分，而内存里是没有的，只有二进制的形式，所以用补码表示，非负数的补码就是其本身，而负数的补码是其绝对值取反在+1



正好等于2^32+这个负数本身，就是存在计算机中的数值（补码）

* 顺便学习一下java的运算符优先级，注意位运算比<,==这些优先级都要小，所以在使用位运算是要加小括号



* 有两种关于时间的计算函数

1. datediff(日期1，日期2) :得到的结果是日期1-日期2，可以为正或为负
2. Timestampdiff(时间类型，日期1，日期2)：时间类型可以添加“day”，“hour”，”second“等关键词，得到的结果是日期2-日期1，与datediff（）相反。

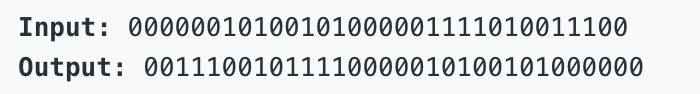
* 两表笛卡尔积：将两个表的每个数据都组合在一起。有两种形式

1. from table a,table where ...
2. from table a cross join table b on...

注意虽然两个方法都可以，但是效率明显是第二个高，因为，on是在建临时表之前就进行条件筛选，而第一个是先建完表在筛选查询结果，所以两者查询结果都相同，但是建的表格是不一样的。

#### 题目一

#### [190. Reverse Bits](https://leetcode-cn.com/problems/reverse-bits/)



本题使用位操作最简单快速，因为int本身是32位的（从0开始计），所以最大以为是31位，想要实现的是将第0位的数移到第31位，第10位得数移到第21位，所以先让n左移31-i，将那一位移到最小的位置然后和1相与，这样所有32位数只剩下最低位了，在左移i位，也就是对称的位置，即可。

源代码如下：

//位操作

        int ans=0;

        for(int i=31;i>=0;i--)

        {

            ans=ans|(((n>>(31-i))&1)<<i);

        }

        return ans;

#### 题目二

#### [196. Delete Duplicate Emails](https://leetcode-cn.com/problems/delete-duplicate-emails/)

本题可以先采用group by将相同email的分在一组，在挑选其中Id最小的，这里需要用到delete，

1.删除语句里不能使用要删除表的字段，所以需要先见一个临时表，不建的话会出错。

2.如果对表使用别名，则delete后面也要加如：delete a from Person a....

源代码如下：

delete

from Person

where Id not in (

    select temp.Id

    from(

    select min(Id) as Id

    from Person a

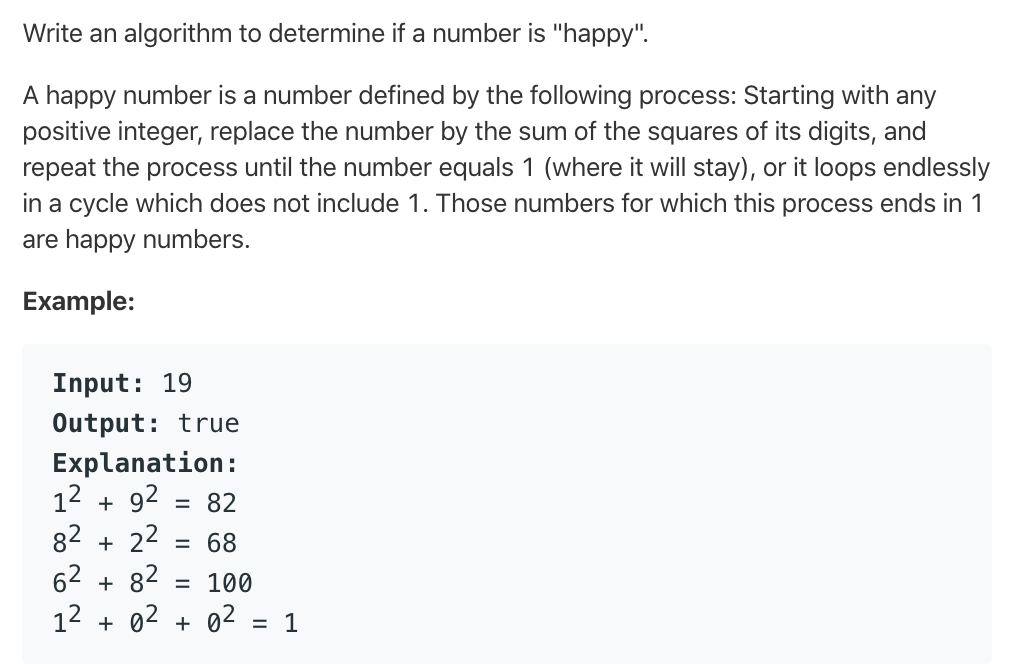
    group by(Email)

    ) as temp

)

#### 题目三

#### [202. Happy Number](https://leetcode-cn.com/problems/happy-number/)



对于这种有循环的题目，以后都要想到快慢指针，快指针移动两步，慢指针移动一步，跳出循环的时候判断是否为1即可。这里如果是happy number，则1就是循环，如果不是happy number 那么按照题目的意思 loops endlessly in a cycle ，所以迟早会两指针相遇。

源代码如下：

 public boolean isHappy(int n) {

        //题目中有循环，所以要想到快慢指针

        int slow=n,fast=n;

        do{

            slow=sumOfDigitsSquares(slow);

            fast=sumOfDigitsSquares(fast);

            fast=sumOfDigitsSquares(fast);

        }while(slow!=fast);

        return slow==1;

    }

    private int sumOfDigitsSquares(int n)

    {

        int sum=0;

        while(n!=0){

            int x=n%10;

            sum+=x\*x;

            n/=10;

        }

        return sum;

    }