**前言：**

又是一个痛苦的一天，感觉不仅被这个leetcode编辑器搞得头大（没事就红色的波浪线误导我），连一些常用的算法也卡壳，二分到现在边界还经常有错误。当然也学到了新的内容，如牛顿法求根号，一些新的函数，用法等。

* Java定义数组的形式 String[] str=new String[10],确定大小，分配内存然后初始化，才能进行赋值操作。
* String的trim()函数 可以自动去除掉字符串前后的空格
* String类型在一开始初始化以后就不能更改大小，其中的元素，如果想用charAt访问，也仅限于访问，但是可以改变整个字符串的引用如

所以如果想要改变字符串的值，可以使用StringBuffer，



String 和int的互相变换,int-->String 用Integer.toString() 。Sring->int用Integer.parseInt（）

Leetcode 67 Add Binary

这题本身就是大数相加，只不过变成了二进制的形式，在这里学习到了一种很简便的表达方式如下，

class Solution {

    public String addBinary(String a, String b) {

        //Integer.toBinaryString(Integer.parseInt(a,2)+Integer.parseInt(b,2));  这个方法不太行，看来是超过int范围了，还是用常规的解法吧

        StringBuffer ans=new StringBuffer();

        int i=a.length()-1;

        int j=b.length()-1;

        int carry=0,sum=0;

//其中这里相当于循环到max(i,j)次，对于下标已经小于0的数，则直接赋0，这样写很简便

        while(i>=0 || j>=0){

            sum=carry;

            sum+= i>=0?a.charAt(i)-'0':0;

            sum+= j>=0?b.charAt(j)-'0':0;

            ans.append(sum%2);

            carry=sum/2;

            i--;j--;

        }

        ans.append(carry>0?1:"");

        return ans.reverse().toString();

    }

}

Leetcode 69 Sqrt(x)

就是求x的近似平方根，这里用两种方法，

* 第一种是二分，对于x>1来说，x的根号都是<=x/2的，所以对于0和1可以单独讨论；对于其他的将left设为1，right为x/2 。如果最后的结果无法准确的保证是根号，那说明肯定会退出while循环，退出循环的条件就是l>r，选择l-1

public int mySqrt(int x) {

       //二分求解

       if(x==1)return 1;

       long l=1,r=x/2;

       while(l<=r)

       {

           long mid=(l+r)/2;

           if(mid\*mid==x) return (int)mid;

           else if(mid\*mid>x)r=mid-1;

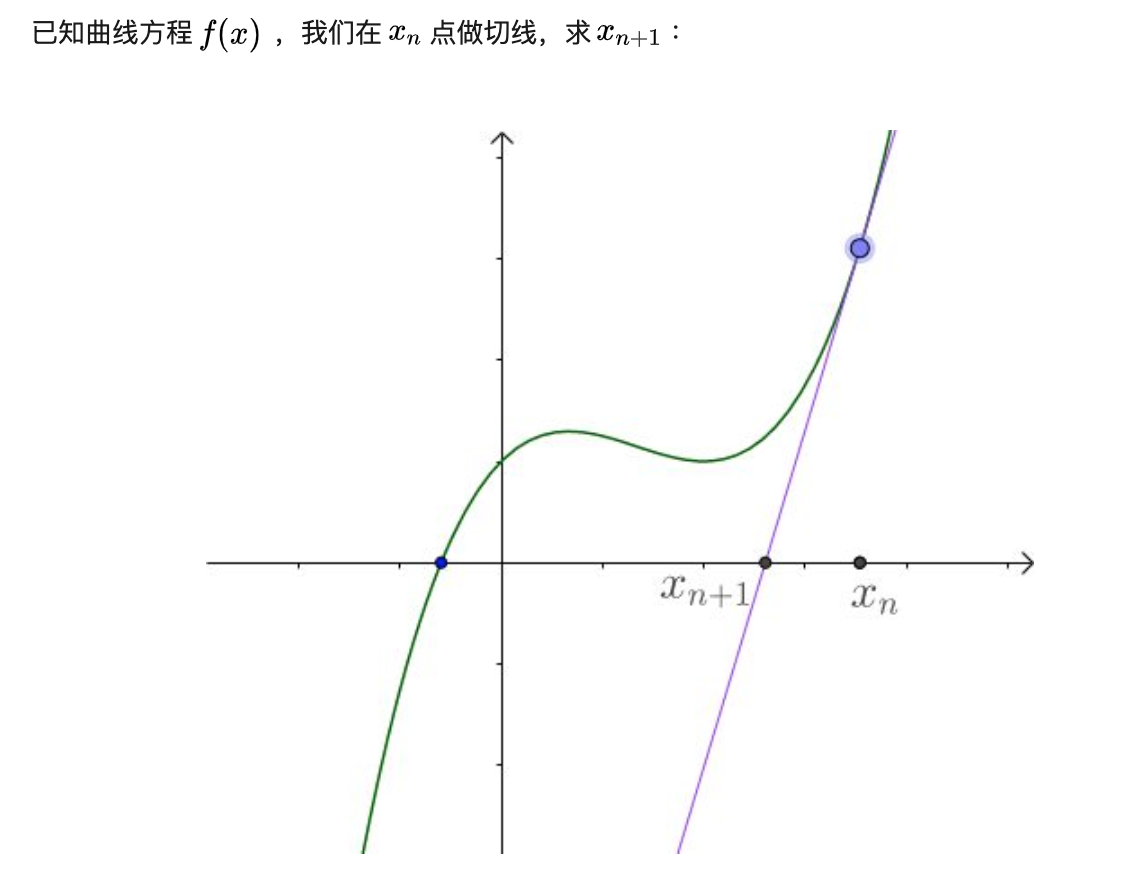
           else l=mid+1;

       }

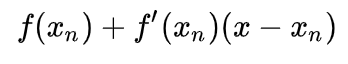
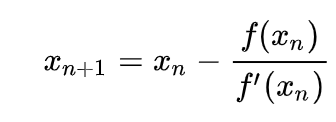
       return (int)(l-1);

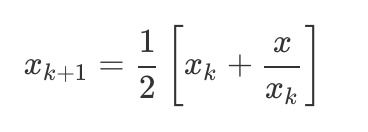
}

* 牛顿迭代法，根据指定精度求解平方根



如图所示，牛顿迭代法是用切线近似曲线，根据xn求切线方程，然后获得xn+1，然后用同样的方式迭代一直到收敛到曲线与x轴的交点为止。

xn点的切线方程为y=,与x轴交点就是令y=0,所以。对于本题求平方根，也就是x^2=n--->x^2-n与0的交点。F(x)=x^2-n,带入到上面的迭代式中，可得

,当xk+1和xk的差距小于给定值的时候就是近似解了，此题中为小于1.

    public int mySqrt(int x) {

       //牛顿法

       if(x<2)return x;

       double x1=x;

       double x2=(x1+x/x1)/2;

       while(Math.abs(x2-x1)>=1){

           x1=x2;

           x2=(x1+x/x1)/2;

           //System.out.println(x1+"  "+x2);

       }

       return (int)x2;

    }