给定一个表示**为非空**数字数组的非负整数，再加上一个整数。

您可以假设该整数不包含任何前导零，除了数字0本身。

数字被存储为使得最高有效数字位于列表的头部。

就是，数组里存一个数，我们把这个数+1.

首先，因为是当作数字处理，所以肯定是从个位数加+1，然后考虑是否进位的情况，所以我们需要从数组的末尾加起，这道题需要处理的情况分为两种

1.当数字不全为9的时候

说明当前大小的数组是够用的，我们遇到9进位，该位为0，在离该位最近的小于9的位置进位即可。进位完成后返回数组，结束。

例如

1，2，3，4 + 1 = 1，2，3，5 //末尾小于9，末尾+1

1，2，3，9 + 1 = 1，2，4，0 //当该位为9，直接改0，然后在离该位最近的小于9的位置进位，前边位等于9做同样处理。

其实我们可以总结为，从最后一位开始向前遍历，如果该位等于9，直接更改为0，在前边小于9的位置进位。进位完成后，结束。

2.当数字全是9的时候

当前大小的数组是不够存的，需要新建一个数组。

例如

9，9，9 + 1 = 1，0，0，0

时间复杂度O(n),空间复杂度O(n)

代码

public static int[] plusOne(int[] digits){

for(int i = digits.length-1;i>=0;i--){

if(digits[i]==9) //该位等于9，改为0，隐试进位，原本为10，进位1，变成0

digits[i]=0;

else {

//因为我们这里的进位和加数都是1，所以加进位和加和可以归纳为一种情况

//该位不等于9，直接+1进位，结束

digits[i]+=1;

return digits;

}

}

//如果digits数组全是9的情况，for循环内是没有返回的

//99999+1=100000，数组长度是不够的

//需要新建一个length+1的数组，首位为1，其他都为0

int[] newDigits = new int[digits.length+1];

newDigits[0]=1;

return newDigits;

}