二分

起初状态：

有思路，但不是二分法的思路，通过其他的思路比如：two pointers可以达到19/25分这样，但是始终不能AC。而且时间比较长，通常从读题到解题时间要40-60分钟！效率非常之低。而且也不能体会到这四个题目如何使用二分法思想。现总结如下：

题目大意：

A1010: 进制转换，给定两个字符串n1,n2.告诉其中一个是多少进制。让你给出另外一个数的进制，使得两个数相等。如果没有，输出“Impossible”

A1044: 给定一个数串，再给定一个数m,让你确定这个数串中是否存在一个连续的子串使得他们的和等于m。有的话就输出起始坐标-终点坐标（从1开始）。如果没有这样的子串，就输出总和大于m的最小子串。

A1048:给定一个数串，再给定一个数m，确定这个数串中是否存在两个数，使他们的总和等于m，输出这两个数，如果存在多个，就输出最小的那一个数

A1085:给定一个数串，找出一个最长子串使得：最大数字M,最小数字m，满足M<=m\*p。输出这个最长子串的长度。

二分法思想应用：

A1085: 递增排序。从左至右扫描，对其中的每一个数a[i]，在a[i+1]~a[n-1]内二分查找第一个超过a[i]\*p的数的位置j，这样j-i就是对位置i来说满足a[j]<=a[i]\*p的最远长度。取所有j-i的最大值即为所求的答案，时间复杂度为O(logn)

A1010: 将已经确定进制的数放在N1，将未确定进制的数放在N2，以便后面统一计算

将N1转换为十进制用long long进行存储。二分N2的进制，将N2转换为十进制，令其与N1的十进制进行比较：如果大于N1的十进制，说明N2的当前进制太大，应该往左子区间继续二分；如果小于N2的十进制，说明N2的当前进制太小，应往右子区间继续二分。当二分结束时就可判断是否存在。

A1044: 用Sum[i]=A[1]+……+A[i]。A[i]到A[j]的和即为Sum[j]-Sum[i-1]。Sum为严格单调增函数。要找出一个子串和为S，即从i(1<=i<=n)开始往后遍历，找出是否存在一个坐标j使得Sum[j]==S+Sum[i-1]。没有的话就找到第一个大于S+Sum[i-1]的位置j。输出i-j即可。

-----通过此题的不断排错，我对二分法的理解又增加了一个层次，虽然浪费了很长时间，但是我觉得值得。

A1048: 令int型a[]数组读入所有数，并排序，对每个a[i](0<=i<=n-1)，用二分法查找数组内是否存在m-a[i]，如果存在且下标不是i，则输出a[i]与m-a[i]。如果遍历完毕还没有发现匹配的一对，则输出“No Solution”

--------但是此题明显使用two poiniters的思想更简便一些。

综上：

1. 使用二分法必须有一个单调区间，如果题中没有明显的递增关系，也可以类似于A1044的思想构造一个。
2. 找出变量关系还是很重要的，比如A1044中就要找到第一个大于等于S+Sum[i-1]的位置j。A1085要寻找第一个超过a[i]\*p的数的位置j
3. 熟悉upper\_bound(first,last,x)和lower\_bound()函数的使用和实现。#include <algorithm>中的使用方法是：j = upper\_bound(Sum+i,Sum+n+1,Sum[i-1]+m) - Sum; 但是他的实现要了然于胸。