常规题目解题报告

Problem1

题目链接：https://vjudge.net/contest/147658#problem/A

题目大意：给在[1,limit]得到一些数使得他们的二进制，从高位到低位第一个 1 出现的位置的和等于 sum

数据范围：*sum*, *limit* (1 ≤ *sum*, *limit* ≤ 100000)

解题思路：考虑到数目不是太大的情况下，先是将[1,limit]范围内的每个数求一下转换为二进制的时候第一个 1 的位置，然后再遍历一遍他们相加是否为 sum，之后比较综合是否为 sum，是则输出个数和具体数字，不是则输出-1

在这之中，有一个新的方法更方便的去转换 bitset 类去取二进制

在代码中先是添加头文件#include <bitset>

然后在代码中的体现是这个

```
bitset<18> t; //声明一个含有 18 位的 t 二进制数组
t = i; //将循环的每个数 i 传递给 t
string str;//string 类
str = t.to_string();//bitset 类成员函数 to_string()，作用是将 t 转化成一个二进制的字符串
```

Problem 2

题目链接:https://vjudge.net/contest/147658#problem/G

题目大意:给你一组字符串 str，每个字符串背后代替着不同二进制，要求出这个字

符串后的组成的数关于 1000003 的模

数据范围:1<=str<=100

解题思路:

一开始想的是暴力求，逐一去判断每个字符串代表的什么，然后用一个数组去承接每一个二进制，之后每位挨个去转十进制求模，结果 WA 两发，我就决定去换，我发现每个二进制是 8-15 任意一个数，这样的话，我只要把他们按照大小顺序排好，然后，连续的前一个*16，后一个根据数目转换十进制，然后每次循环求模

算法题目解题报告

Problem 1

题目链接:https://vjudge.net/contest/147661#problem/G

题目大意：给你 T 组字符串 str，问你每个字符串的子字符串是回文串的最小个数

数据范围:1<=str<=1000

涉及算法:DP(动态规划)

动态规划算法通常基于一个递推公式及一个或多个初始状态，当前子问题的解将由上一次子问题的解推出，使用动态规划来解题只需要多项式时间复杂度，因此他比回溯法，暴力法等要快许多

大致思路是：

1.要先找到初状态和末状态，通过将此题的问题不断往下化小变成子问题，直至无法化小，那便是初状态

2.找到状态转移方程，就是通过这个子问题，是如何推到上一个子问题的一个状态，这是 DP 的最关键的一个地方，也是重点

3.根据不同的问题，会附加不同的其他算法，这是根据实际情况有所附加

解题思路:

当字符串长度为 i 的时候，要求 i 的时候最小的回文个数，就应该在 1-i 的范围内 j 的最小个数，根据 j 的改变，不断更新 i 的最小个数，当 i=0 时，最小长度为 0 这就是他的初始状态了，状态转移方程就是刚刚说到那些东西