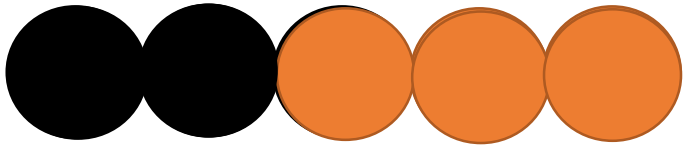
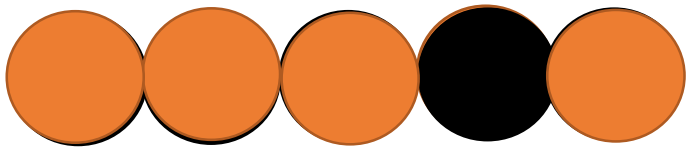
湖南大学新生杯解题分析

A题：AFei Loves Magic





题意：阿飛在找到一堆排成一条直线的魔法石，然后触发了魔法阵，他被石化，魔法石开始以1m/s匀速向两端移动，移动到直线两端，魔法石就会消失。魔法石相碰就会改变方向继续以1m/s移动。给定直线长度L，魔法石总数以及初始位置，初始运动方向（方向只有0→L或者L→0）。求t秒后，直线上剩余魔法石数量，然后再加1，输出这个结果。

解题思路：不考虑碰撞，直接根据初始坐标，方向，时间和速度求出每一个魔法石的最终坐标，最终坐标在（0,L)的魔法石数量就是剩下的魔法石数量

B题：BearBaby like Sleeping（搜索bfs）

- 睡觉是little bearBaby的最爱，因为长沙冬天的湿冷实在是太难受了，有一天早上，little bearBaby
- 一不小心睡过头了，完了，迟到的后果是挂科。你是最聪明的人工智能
- ，little bearBaby现在请你帮助他算出最短需要多少时间到达教学楼。
- 学校地图由n*m的网格组成，每个单元格要么是空地，要么是建筑物（不能通过），little bearBaby的寝室在（1，1）
- 教学楼在（x,y)，他只能往下或者往右走，每走一步需要花费1分钟。保证一定可到达教学楼。

- Input：
- 第一行有两个正整数n,m(1<=n,m<=100),n表示行，m表示列
- 接下来有两个正整数x，y(1<=x<=n,1<=y<=m)表示教学楼的坐标
- 接下来是n行m列的地图, 0表示空地，1表示障碍物

- Output:
- 输出最短的时间到达教学楼

```
void bfs()//走迷宫
{
    queue<P> que;//定义一个队列
    que.push(P(sx,sy));//将起点放进队列中
    d[sx][sy]=0;//到起点步数为0

    while(que.size())//题目告诉一定有解，不用担心死循环
    {
        P p=que.front();//定义当前队列顶端坐标 为P
        que.pop();//将P弹出 @
        if(p.first==gx&&p.second==gy) break;//判断该坐标是否是终点，是的话结束
        for(int i=0;i<4;i++)//使坐标不断向四个方向移动
        {
            int x=p.first+dx[i];
            int y=p.second+dy[i];
            if(0<=x&&x<N&&0<=y&&y<=M&&maze[x][y]!=1&&d[x][y]==0)//当前坐标某方位是否可移动判断
            //如果四个方位都不行回到@步骤不断将顶点位置弹出，直到返回上一有多个出口的位置
            {                                                    //如果坐标当前某方向可移动
                que.push(P(x,y));                                //则将该位置该点坐标放入队列中
                d[x][y]=d[p.first][p.second]+1;                   //记录该位置的最短步数=上一位置到达步数+1
            }
        }
    }
}
```

C题：Sleep Kaguya

题意：给定一个数列F{}，F[1]=F[2]=1,F[n]=F[n-1]+F[n-2],现在给定k，求F[k+1]*F[k+1]-F[k]*F[k+2]（0<k<1e18）

题解:$F[n+1]*F[n+1]-F[n]*F[n+2]=(-1)^2$

1.n=1时，结论显然成立

2. 假设n=k时结论成立，即$F[k+1]*F[k+1]-F[k]*F[k+2]=(-1)^2$

则 $F[k+2]*F[k+2]=F[k+2]*F[k]+F[k+2]*F[k+1]$

$=F[k+1]^2-(-1)^2+F[k+2]*F[k+1]$

$=F[k+1]*(F[k+1]+F[k+2])+(-1)^{k+1}$

$=F[k+1]*F[k+3]+(-1)^{k+1}$

故n=k+1时结论仍然成立

复杂度：O(1)

D题： Dandan’s Lunch by-Dandan

题意

现有n个人在比赛，每个都有一个面包以及一个解题数目Si。当一个人(A)发现自己的解题数目大于另一个人(B)的解题数目，且A拥有的面包的大小小于B拥有的面包的大小时，A会与B交换面包。Dandan也是参赛者之一，现在给出他以及其它所有人的解题数目以及拥有的面包的信息，问Dandan最终(任何两个人都不能再交换面包时)能获得的面包的大小。

面包由三个点(x1,y1,x2,y2,x3,y3)描述，其大小为由这三个点组成的三角形的面积的两倍。

数据范围：

所有数据均为整数，且:

0<n<=1e5;

0<=Si<1e9，且对于i≠j，有Si!=Sj;

-1e8<x1,y1,x2,y2,x3,y3<1e8，且所有的面包的大小不相等;

时间限制：1s

空间限制：64M

示例输入及输出

输入

```
4
3 0 0 1 0 0 1
1 0 0 2 0 0 2
2 0 0 3 0 0 3
4 0 0 4 0 0 4
```

输出

```
9
```

D题： Dandan’s Lunch （续）

题解

显然，最终的局面为：解题数目排行第i的人会获得第i大的面包（解题数目最多的人一定可以获得最大的面包，第二可以获得第二大的面包，依次类推…）。

故只需知道到Dandan的解题数目在所有人中的排名rk，然后输出第rk大的面包大小即可。

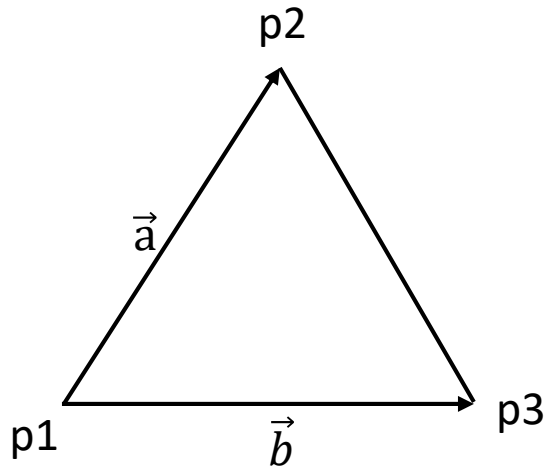
(Ⅰ)获取排名--排序：sort()。

(Ⅱ)已知三点坐标计算三角形的面积？

①底×高/2--可能会有精度误差(?)

②向量叉乘：

如图三角形的面积为$\frac{1}{2}|\vec{a}\times\vec{b}|$，故三角形的面积的两倍为$|\vec{a}\times\vec{b}|$。




```
Vector operator - (const Point &p1,const Point &p2){
    return Point(p1.x-p2.x,p1.y-p2.y);
}
ll Cross(Vector v1,Vector v2){
    return v1.x*v2.y-v1.y*v2.x;
}
ll area(Point p1,Point p2,Point p3){
    return abs(Cross(p2-p1,p3-p1));
}
```

E题： Easy Problem --by HNU-emofunc

题意

- 找一个数(<=1e18)，包含a1,a2,…,ak且是x的倍数。(0<=ai<=9,x<=1e8)

题解

- 令N=10234567890000000
- 1) N%x==0 , ans = N
- 2) N%x!=0, ans = N+(x-N%x)
- 那么显然对于x<=1e8, ans包含0到9且ans是x的倍数。

F题：FindtheAFeiNumbers

题意：给定一个自然数n(0<=n<=1e18)，求[0,n]里有多少个整数包含“520”(即所谓Afei数的数量)

解题思路：
数位DP入门题。

首先，设dp[i]表示长度为i的AFei数的数量。发现这样不行，因为按数位计算的话，每次只会考虑一位，而AFei数的特征是要有连续的三位，并且第一位是5，第二位是2，第三位是0。

这说明需要记录前两位，那么设dp[i][j]表示以j开头，后面还有i位的AFei数的数量。比如dp[4][25]就是表示形如25****的数字中，AFei数的数量。因为n只有1e18，所以i最大不超过19，而j是用来记录前两位的，所以j最大不超过99，所以声明dp数组的时候只需要long long dp[20][100]即可。

AFei数并没有规定“520”的数量有多少个，碰到这样的我习惯性反着算，也就是用dp[i][j]表示以j开头，后面还有i位的不是AFei数的数量。算到最后再减一下就行了。

状态转移方程：

$$dp[i][j] = \sum_{k=0}^{9} dp[i-1][(j\%10)*10+k] \text{ (当} j = 52 \text{的时候} k \neq 0)$$

举个例子：以f(****)表示形如****的不是afei数的数量，显然：

f[25****]=f[50***]+f[51***]+f[52***]+f[53***]+f[54***]+f[55***]+f[56***]+f[57***]+f[58***]+f[59***]

也就是dp[4][25] = $\sum_{k=0}^{9} dp[3]\ [(25\%10)*10+k]$

FindtheAFeiNumbers

解题思路(续):
解决了dp[i][j]的问题了，再来考虑如何让我们按位递推的时候的数字不超过n~~
假设当前算到第i位，接下来是i-1位，那么就有以下三个要求:
(1)如果前两位是52，那么这一位不能是0
(2)如果前面所有位和n的前几位一模一样，那么这一位的数字不能超过n的这一位
(3)如果前两位不是52并且前面的位和n的前几位不完全一样，那么这一位数字可以是0~9的任意一个数
按位递推的时候，只要满足了以上三个要求，就可以保证递推到最后一位的时候的数字是不超过n的非Afei数

```
ll solve(int pos, int pre, bool limit)
{// pos表示还剩几位没有算，pre记录前两位，limit记录前面递推过的位和n的前几位是否一样
     if(pos == -1) return 1;
     ll& d = dp[pos][pre];
     if(d != -1 && !limit) return d;

     ll ret = 0;
     int n = limit ? dig[pos] : 9; // 如果递推过的位和n的前几位一样，那么这一位最大只能到dig[pos]
     for(int i = 0; i <= n; ++ i)
     {
          if(pre == 52 && !i) // 如果去前两位是52，这一位不能是0
               continue;
          ret += solve(pos-1, (pre%10)*10 + i, limit && i==dig[pos]);
     }

     if(!limit) // 记忆化
     d = ret;
     return ret;
}
```

时间复杂度：$O(length(n)*100*10)$

G题：a+b+c+d=?

- 题解:这是一道很简单的题目，主要考察对于长整型数据范围的认知，直接加法是会出现错误的，需要在a,b,c,d都等于2^61时特判直接输出2^63,其他情况直接输出结果即可！

H题： Kuangyeye and hamburgers

- 题目大意:n个数字,k个方案,每个求从第a大数字到第b大之间所有数字的和,问所有方案中数字的和最大是多少

- 题解:先对所有的数字(汉堡的重量)w_i按由大到小排个序,求前缀和$s_i = \sum_{j=1}^{i} w_j$,每于方案i数字的和即为$s_{b_i} - s_{a_i-1}$,答案即为max{$s_{b_i} - s_{a_i-1}$}(for i in (1,k))

- 复杂度:max(k,nlogn)

I题： ii plays with gg

- NP态的定义：在双方都采用最优策略的前提下，先手必胜的点称为N态，先手必败的点称为P态。

1.结束状态为P状态。（已经到了结束状态，故先手必败）

2.所有下一状态中全为N状态，则当前状态为P状态。(不管你往哪走，你的对手都必胜，所以你必败）

3.所有下一状态中存在P状态，则当前状态为N状态。（只要你能走到让你对手必败的情况，你现在就是必胜）

ii plays with gg

败（P）	胜（N）	败（P）	胜（N）	败（P）	胜（N）
胜（N）	胜（N）	胜（N）	胜（N）	胜（N）	胜（N）
败（P）	胜（N）	败（P）	胜（N）	败（P）	胜（N）
胜（N）	胜（N）	胜（N）	胜（N）	胜（N）	胜（N）
败（P）	胜（N）	败（P）	胜（N）	败（P）	胜（N）

观察不难发现，必败点的坐标依次为(0,0)、(2,0)、(0,2)、(0,4)、(2,4)、(4,4)、(4,2)、(4,0);
所以结论为：当x%2==0&&y%2==0时，先手必败 否则先手必胜。

J题： Less Taolu

```
long long func(int x){
    if (x==1||x==0){
        return 1;
    }
    return (x*func(x-1)+(x-1)*func(x-2))%mod;
}
```

首先这么写肯定会T滴
拿func(4)举例，需要一次计算func(3)和一次func(2)
计算一次func(3)的时候，需要计算一次func(2)和一次func(1)
计算一次func(2)的时候，需要计算一次func(1)和一次func(0)
这样，算下去，func(10000)要计算很多很多次

那就加速一下吧

这么写

```
long long f[100010];

f[0]=f[1]=1;
for (int i=2;i<=100000;++i){
    f[i]=(i*f[i-1]+(i-1)*f[i-2])%mod;
}
```

开个1e5大小的数组记录答案
f[x]就是func(x)
这样，就不会出现f[x]需要被计算多次的现象了
时间复杂度和空间复杂度均为O(n)
恭喜你签到成功
（个人认为是历史上最简单的签到了）

K题：题目大意

- 考虑直角三角形三边的长度都是正整数的情形。
- 现在给你直角三角形斜边的长度，请回答这个斜边长度能不能是三边都是正整数的直角三角形的斜边长度。
- 比如说，当斜边长度为5时，另外两条直角边长度可以为正整数3和4。而当斜边长度为6时，不可能构成直角三角形，三边长度都是整数。
- 输入：首先输入一个正整数T（1 ≤ T≤ 103）表示有T组测试数据，随后T组测试数据，每组测试数据一行，包含一个正整数c（1 ≤ c≤ 45,000)，表示斜边长度
- 输出：对于每组测试数据，输出一行，输出”Yes”（没有双引号）表示可以构成这样的直角三角形，输出”No”（没有双引号）表示不能构成
- 本题采用暴力枚举，复杂度为O（c）。标程在1000组测试数据下的运行时间约为100毫秒。建议时间限制为1秒，空间限制为64M。

解题思路

- 这个题很容易想到的一个思路就是暴力枚举。是的，我们给的解题方法也是暴力枚举。但是，直接枚举的复杂度是$O(c^2)$，会超时(TLE)。所以我们需要将问题转化一下，使得枚举的复杂度是O(c)。
- 如果三角形三边满足如下关系，则是直角三角形。
- $a=m^2-n^2$
- $b=2mn$
- $c=m^2+n^2$

解题思路(续)

- 所以如果斜边长度能够表示成2个正整数的平方和，则能使得三边都是正整数。这样枚举的复杂度是O（c）。
- 另外，如果斜边长度是一个合数，其有一个因子能表示为2个正整数的平方和，那么也能使得三边都是正整数。比如c=15，有因子$5=1^2+2^2$，那么也是可以构成三边全是整数的直角三角形，每边长度乘以3即可。就是（9，12，15）。

标准答案

```
#include <stdio.h>
#define N 45001
int MK[N]={0},SQ[213];
//MK数组标记能否是整数三角形，为0表示不能，非0表示可以，SQ数组记录数的平方值
int main(){
 for(i=1;i<213;++i)SQ[i]=i*i;
  for(i=1;i<213;++i)
    for(j=i+1;j<213&&(k=SQ[i]+SQ[j])<N;++j)
      MK[k]=1;                    //所有能写成2整数平方和的被标记为能
  for(i=5;i<22501;++i)
    if(MK[i]==1)
      for(j=2;(k=j*i)<N;++j)
        MK[k]=j;                  //所有含2整数平方和的因子的正整数被标记为能
  scanf("%d",&t);
  while(t--){
    scanf("%d",&c);
    printf("%s\n",MK[c]?"Yes":"No");}
  return 0;}
```

L题：题目大意

- 求长度为n的数字串中，有多少个这样数字串，其数字之和是4的倍数（包括0）
- 输入：每组测试数据一行，包含一个正整数n（$1 \le c \le 10^9$)
- 输出：对于每组测试数据，输出一行，包含一个整数，表示有多少个这样数字串，其数字之和是4的倍数（包括0）。因为这个结果很大，所以将值模2019输出

- 本题快速幂，复杂度为O（logn）。标程在1000组测试数据下的运行时间约为60毫秒（第二标程运行时间约30毫秒）。建议时间限制为1秒，空间限制为64M。

解题思路

- 假设a_n 是长度为n时，数字和是4的倍数的数字串的个数
- 假设b_n 是长度为n时，数字和是4的倍数加1的数字串的个数
- 假设c_n 是长度为n时，数字和是4的倍数加2的数字串的个数
- 假设d_n 是长度为n时，数字和是4的倍数加3的数字串的个数
- 显然有
- $a_n = 3a_{n-1} + 2b_{n-1} + 2c_{n-1} + 3d_{n-1}$;
- $b_n = 3a_{n-1} + 3b_{n-1} + 2c_{n-1} + 2d_{n-1}$;
- $c_n = 2a_{n-1} + 3b_{n-1} + 3c_{n-1} + 2d_{n-1}$;
- $d_n = 2a_{n-1} + 2b_{n-1} + 3c_{n-1} + 3d_{n-1}$
- 写成矩阵形式，采用矩阵快速幂即可。

标准答案

```
#include <stdio.h>
#define M 2019
#define L 4
#define LP(i) for(i=0;i<L;++i)

void Product(int x[L][L],int y[L][L],int z[L][L]){
    int i,j,k,w[L][L]={0};
    LP(i)LP(j)LP(k)w[i][j]=(w[i][j]+x[i][k]*y[k][j])%M;
    LP(i)LP(j)z[i][j]=w[i][j];}

void FPow(int n,int r[L][L]){
    int i,j,A[L][L]={{3,2,2,3},{3,3,2,2},{2,3,3,2},{2,2,3,3}};
    LP(i)LP(j)r[i][j]=i==j?1:0;
    while(n){
      if(n&1)Product(r,A,r);
      Product(A,A,A);
      n>>=1;}}

int cal(int n){
    int r[L][L];
    FPow(n,r);
    return (3*r[0][0]+3*r[0][1]+2*r[0][2]+2*r[0][3])%M;}

int main(){
   int n;
   while(scanf("%d",&n)!=EOF&&n>0)
       printf("%d\n",cal(n-1));
   return 0;}
```

解题思路(续)

- 矩阵
- 3 2 2 3
- 3 3 2 2
- 2 3 3 2
- 2 2 3 3
- 有四个特征根，分别是0，10，1+i，1-i；
- 所以$a_n=x10^n+y\cos(n\pi/4)+z\sin(n\pi/4)$,根据a1=3,a2=25,a3=249,可得：
- $a_n=(10^n+2\mathrm{sqrt}(2)^n\cos(n\pi/4))/4$;
- 同样快速幂求得答案。

标准答案（2）

```
#include <stdio.h>
#define M 2019
int rt[2][32],et[8]={2,2,0,-4,-8,-8,0,16};
int FPow(int f,int n){
    int r=1,m=0;
    while(n){
        if(n&1)r=(r*rt[f][m])%M;
        m++;
        n>>=1;}
    return r;}

int cal(int n){
    int r1,r2;
    r1=FPow(0,n);
    r2=((FPow(1,n/8)*et[n%8])%M+M)%M;
    return ((r1+r2)*505)%M;
}
int main(){
    int n,i;
    rt[0][0]=10;rt[1][0]=16;
    for(i=1;i<32;++i){
        rt[0][i]=(rt[0][i-1]*rt[0][i-1])%M;
        rt[1][i]=(rt[1][i-1]*rt[1][i-1])%M;}
    while(scanf("%d",&n)!=EOF&&n>0)
        printf("%d\n",cal(n));
    return 0;}
```