Problem A 红绿灯

考察知识

模拟，思维

解题思路

可以发现由于红绿灯的周期，我们只需要关心$t\ mod\ B$的值，令这个值为t。

之后我们记录当前时刻，初始为0。每次将其加上t并对B取模。若当前时刻处于红灯范围，即$A...B-1$。我们需要将当前时刻置为0，表示等红灯的过程。

经过模拟k个路口之后输出当前的时刻即为答案。

示例代码

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;

int A,B,t,k;

int main()
{
    int T;
    scanf("%d",&T);
    while (T--)
    {
        scanf("%d%d%d%d",&A,&B,&t,&k);
        int cur=0;
        t%=B;
        for (int i=1;i<=k;i++)
        {
            cur=(cur+t)%B;
            if (i==k) break;
            if (cur>=A) cur=0;
        }
        printf("%d\n",cur);
    }
    return 0;
}
```

Problem B 露营

考察知识

枚举，暴力，数学

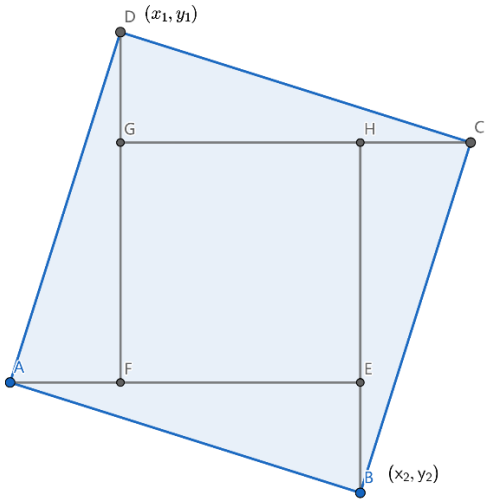
解题思路

在排列成$n \times n$的格点中找到最大的格点正方形，满足四个格点中至少3个点为空地，且不能出现石头。可以通过枚举所有格点正方形的方式判断条件是否满足并更新答案。

1. 枚举对角线

   假设对角线两端的坐标分别为$(x_1, y_1), (x_2, y_2)$，我们只考虑$x_1 \leq x_2$，$y_1 \leq y_2$的情况。（显然这样可以不遗漏地枚举出所有格点正方形）。

   不妨设$\Delta x = x_2 - x_1$，$\Delta y = y_2 - y_1$，当$\Delta x > \Delta y$时，如下图：



   由几何关系可得：

   $$\Delta x = DF + EB$$

   $$\Delta y = EF = AE - AF = DF - EB$$

   联立方程

   $$\begin{cases} \Delta x = DF + EB \\ \Delta y = DF - EB \end{cases}$$

   解得：

   $$\begin{cases} DF = \frac{\Delta x + \Delta y}{2} \\ EB = \frac{\Delta x - \Delta y}{2} \end{cases} (1)$$

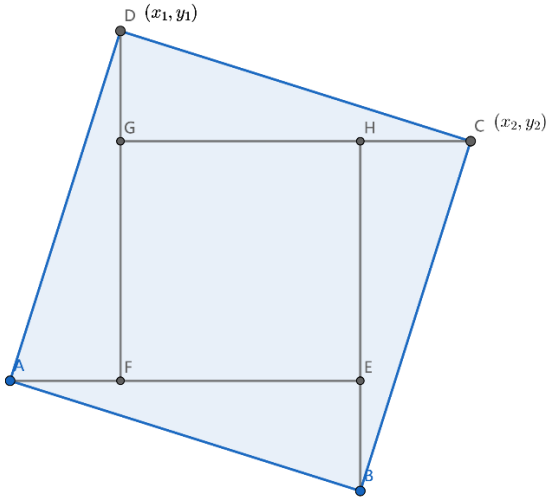
   由方程(1)可知，只有当$\Delta x, \Delta y$奇偶性相同时，正方形的另外两个顶点坐标为整数。再结合上图得到的几何关系可以计算出另外两点的坐标。通过四点的坐标判断可行性并更新答案。

   根据菱形面积公式可得到此时正方形面积：

   $$S = \frac{\Delta x^2 + \Delta y^2}{2}$$

   当$\Delta x < \Delta y$，$\Delta x = \Delta y$时，推导方法同上。

2. 枚举边



枚举边时对于另外两点坐标的推导方法与枚举对角线时基本相同。注意，虽然每一条边同时属于两个正方形内，在枚举时统一只考虑其中一种情况也可以得出正确答案。

上述两种方法时间复杂度均为$O(n^4)$

示例代码

```
#include <cmath>
#include <cstdio>
#include <cstring>
#include <iostream>
using namespace std;
string s[120];
int n,ans=0;
int f(int x,int y)
{
    if(x<0||y<0||x>=n||y>=n||s[x][y]=='*') return -10;
    return s[x][y]=='I'?1:0;
}
void chk(int x,int y)
{
    for(int i=n-1;i>=x;i--)
        for(int j=n-1;j>=y;j--)
        {
            if(i==x && j==y) continue;
            if(s[i][j]=='*') continue;
            int ret=f(x,y)+f(i,j);
            int dx=i-x;
            int dy=j-y;
            if(dx*dx+dy*dy<=ans) continue;
            if((dx+dy)%2==1) continue;
            int mn = abs(dy-dx)/2, mx = (dy+dx)/2;
            if(dx<=dy)
            {
                ret+=f(x+mx,y+mn)+f(x-mn,y+mx);
            }
            else
            {
```

```
                ret+=f(x+mx,y-mn)+f(x+mn,y+mx);
            }
            if(ret>=3) {ans=max(ans,dx*dx+dy*dy); }
        }
}
int main()
{
    cin>>n;
    for(int i=0;i<n;i++) cin>>s[i];
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
            if(s[i][j]!='*')
                chk(i,j);
    printf("%d\n",ans/2);
    return 0;
}
```

Problem C 西人保级路

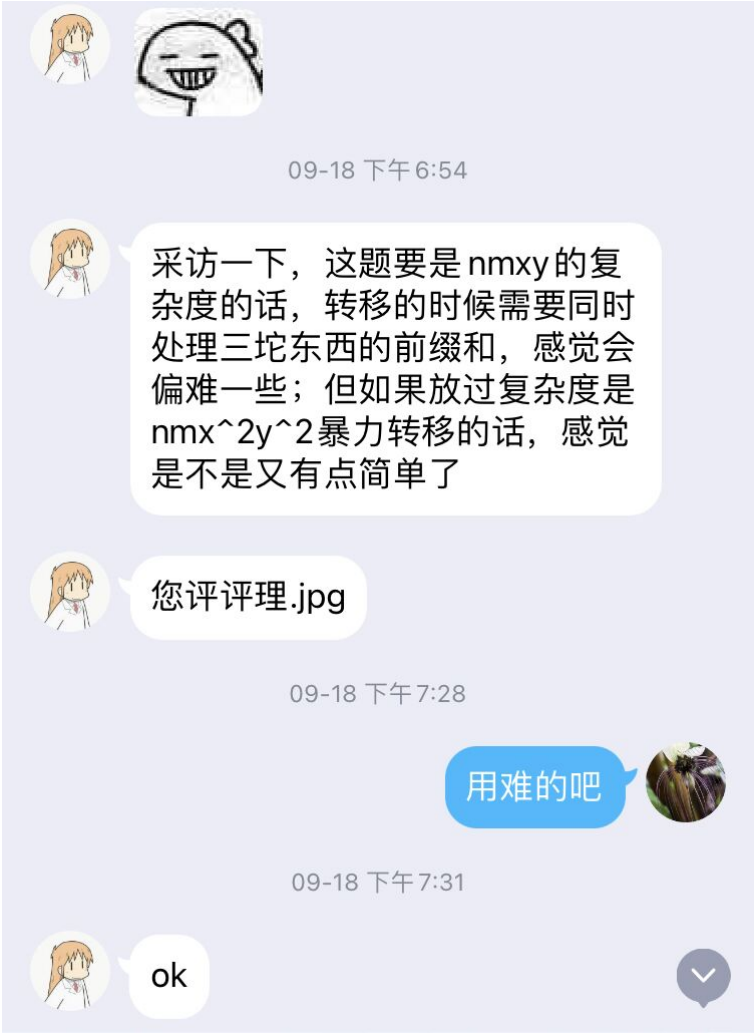
考察知识

动态规划、预处理

解题思路

因为题解过长且有很多的公式和图片，所以我直接用博客的形式来写题解了，请点击下面的网址查看：https://www.cnblogs.com/fried-chicken/p/13736254.html 。

幕后花絮



从结果来看，很多同学在这道题上给出了正确的DP定义，甚至给出了正确的状态转移方程，但没有想到那个优化导致了TLE，出题人表示很抱歉。

事实上，出题人在出这道题的时候自己也没有想到这一优化，出题人也是写完了题面和程序之后进行自测的时候超时了，才发现这样直接DP的时间复杂度过大，于是重新写了现在这个加有优化的程序。

另外，本题有一支队伍在比赛时通过排列组合的纯数学方法求出了答案，但因为比较复杂（指出题人看不懂）就不过多介绍了。

Problem D 虚假的树状数组

考察知识

建图，dfs，排序

解题思路

首先建立有向树，我们发现空间复杂度和时间复杂度都允许邻接矩阵的建图方式。也可以用二维数组或c++的vector存储每个点都能够指向哪些点，再将这些点按照编号从小到大进行排序。

之后要找到有向树的根结点，没有边指向的点即为根结点。

dfs的过程中，按照排好的顺序进行dfs，用二维数组或c++的vector存储每个点的生成数组，每次将dfs的那个儿子结点的生成数组拼接到父亲结点的生成数组后面。如果有排序标记就直接排序。

示例代码

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int maxn = 1010;

int n;
int a[maxn],flag[maxn];
vector<int> v[maxn],g[maxn];
int du[maxn];
int rt;

void dfs(int x)
{
    v[x].push_back(a[x]);
    for (int i=0;i<g[x].size();i++)
    {
        int q=g[x][i];
        dfs(q);
        for (int j=0;j<v[q].size();j++) v[x].push_back(v[q][j]);
    }
    if (flag[x]) sort(v[x].begin(),v[x].end());
}

int main()
{
    scanf("%d",&n);
    for (int i=1;i<=n;i++) scanf("%d",&a[i]);
    for (int i=1;i<=n;i++) scanf("%d",&flag[i]);
    for (int i=1;i<n;i++)
    {
        int x,y;
        scanf("%d%d",&x,&y);
        g[x].push_back(y);
        du[y]++;
    }
    for (int i=1;i<=n;i++)
    {
        if (du[i]==0) rt=i;
    }
    for (int i=1;i<=n;i++)
    {
        sort(g[i].begin(),g[i].end());
    }
    dfs(rt);
    printf("[");
    printf("%d",v[rt][0]);
    for (int i=1;i<v[rt].size();i++) printf(",%d",v[rt][i]);
    printf("]\n");
    return 0;
}
```