

2016 Multi-University Training Contest 3

比赛情况

题号	A	B	C	D	E	F	G	H	I	J	K
状态	0	0	0	0	-	-	-	-	-	0	0

0 在比赛中通过 0 赛后通过! 尝试了但是失败了- 没有尝试

比赛时间

2020-05-20 13:00-18:00

提交记录

D: Accepted 2020-05-20 17:39:39

D: Runtime Error(Access_Violation) 2020-05-20 17:19:16

D: Time Limit Exceeded 2020-05-20 17:05:43

J: Accepted 2020-05-20 16:48:33

C: Accepted 2020-05-20 16:41:33

C: Wrong Answer 2020-05-20 16:35:54

C: Wrong Answer 2020-05-20 16:30:29

J: Wrong Answer 2020-05-20 16:26:31

J: Wrong Answer 2020-05-20 16:22:36

K: Accepted 2020-05-20 13:57:23

B: Accepted 2020-05-20 13:36:47

A: Accepted 2020-05-20 13:14:42

题解

A-Sqrt Bo

有一个函数 $f(x) = \left\lfloor \sqrt{x} \right\rfloor$ 问对于给定的 n $f^k(n) = 1$ 的最小 k 如果 $k > 5$ 输出 "TAT"

n 的取值到 10^{100} , 很容易想到在一定范围外 k 就大于 5 了, 试一下发现 1^{18} 就大于 5, 所以判断一下长度小于 18 的时候就暴力看多少次, 否则就输出 "TAT"

```
#include<bits/stdc++.h>
#define ll long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<" "; cout<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 2e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}
int main()
{
    fastio();
    string s;
    while(cin>>s){
        int len = s.size();
        if(len>=18) cout<<"TAT"<<endl;
        else{
            ll t = 0;
            rep(i,0,len-1) t=t*10+(s[i]-'0');
            //show1(t);
            if(t==1) {cout<<0<<endl;continue;}
            int flag = 1;
            rep(i,1,5){
                t = sqrt(t);
                if(t==1) {cout<<i<<endl;flag=0;break;}
            }
            if(flag) cout<<"TAT"<<endl;
        }
    }
    return 0;
}
```

B-Permutation Bo

题目大意

两个长度为 n 的序列 h,c 给出 h 是长度为 n 的任意排列，求公式

$$\sum_{i=1}^n c_i [h_{i-1} \leq h_i; \text{and}; h_{i+1} \leq h_i] \text{ 另 } h_0 = 0, h_{n+1} = 0$$

数据范围

$$n \leq 10^3, 0 \leq c_i \leq 10^3$$

解题思路

看到h是排列，觉得能造成括号中条件的排列数是固定的，遂打表，发现在位置1,n产生这种情况的概率是 $\frac{1}{2}$ ，其他位置产生这种情况的概率是 $\frac{1}{3}$ ，然后按照这个依次乘c就行。

代码



C-Life Winner Bo

给一个 $n \times m$ 的棋盘，问 $\{\text{king, rook, knight, queen}\}$ 的先手胜负情况

基本上都是靠先SG函数打表再总结规律，但是 $\{\text{queen}\}$ 实在总结不出来，最后优化到 $O(NM)$ 的打表就过了（代码写得贼丑）

```
#include<bits/stdc++.h>
#define ll long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<" "; cout<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 2e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}
int mp[1005][1005],row[1005],col[1005],dia[3005];
int t,n,m;
void printMap()
{
    rep(i,1,n){rep(j,1,m)cout<<mp[i][j]<<" ";cout<<endl;}
}
int main()
{
    fastio();
```

```
int _;
n=m=1000;
mp[n][m] = 0;
rep(i,1,n) row[i]=2;
rep(i,1,m) col[i]=2;
rep(i,1,m-1) mp[n][i]=1,dia[i-n+1001]=1;
rep(i,1,n-1) mp[i][m]=1,dia[m-i+1001]=1;
for(int i=1;i;i++){
    if(n-i<0 || m-i<0) break;
    mp[n-i][m-i] = 1;
}
for(int i=n-1;i;i--) for(int j=m-1;j;j--){
    if(j-i == m-n) continue;
    int flag = 1;
    //rep(k,i+1,n) if(!mp[k][j]) flag=0;
    if(row[i]<m-j) flag = 0;
    //rep(k,j+1,m) if(!mp[i][k]) flag=0;
    if(col[j]<n-i) flag = 0;
    if(dia[j-i+1001]<min(n-i,m-j)) flag = 0;
    //for(int k=1;;k++){
    //    if(k+i>n || k+j>m) break;
    //    if(!mp[k+i][k+j]) flag=0;
    //}
    if(flag) mp[i][j]=0;
    else {mp[i][j] = 1; row[i]++, col[j]++, dia[j-i+1001]++;}
}
//printMap();
for(cin>>_;;_--){
    cin>>t>>n>>m;
    if(t==1){
        if((n&1) && (m&1)) cout<<"G"<<endl;
        else cout<<"B"<<endl;
    }else if(t==2){
        if(n==m) cout<<"G"<<endl;
        else cout<<"B"<<endl;
    }else if(t==3){
        int f = -1;
        for(int k=0;k<=1000;k++){
            if(n-3*k==1 && m-3*k==1){f=0;break;}
            if((n-3*k+1==1&&m-3*k+2==1) || (n-3*k+2==1&&m-3*k+1==1))
{f=1;break;}
        }
        if(f==1) cout<<"B"<<endl;
        else if(f==0)cout<<"G"<<endl;
        else cout<<"D"<<endl;
    }else{
        if(mp[1000-n+1][1000-m+1]) cout<<"B"<<endl;
        else cout<<"G"<<endl;
    }
}
```

```

    return 0;
}

```

D-Gambler Bo

solved by Zars19

给出元素为 $\{0,1,2\}$ 的 $N \times M$ 的矩阵，对某个格子进行操作时对该位置的值加 2 ，四周分别加 1 ，结果都是模 3 意义下的。给出的矩阵最终必能在 $2NM$ 次操作内化为全 0 ，输出一种操作。

题解：模 3 意义下的高斯消元，对于每一个位置 $a_{i,j} + 2 \times \text{opt}_{i,j} + \sum \text{opt}_{i+dx,j+dy} \equiv 0 \pmod 3$ 格子至多 30×30 个，看起来 $O(N^3M^3)$ 好像不太行，但事实上每一个方程所关联到的变量不多，矩阵较为稀疏，注意`if(!a[mxline][i])continue;`即可。

code:

```

#include<iostream>
#include<cstdio>
#include<cstring>
#include<cassert>
#include<cstdlib>
#include<cmath>
#include<vector>
#include<algorithm>
#define LL long long
using namespace std;
int read()
{
    int x=0,f=1;char c=getchar();
    while(c<'0' || c>'9'){if(c=='-')f=-1;c=getchar();}
    while(c>='0' && c<='9'){x=x*10+c-'0';c=getchar();}
    return x*f;
}
int n,m,chess[33][33],a[1001][1001],f[1001],res;
int num(int x,int y){return (x-1)*m+y;}
int dx[4]={-1,1,0,0};
int dy[4]={0,0,-1,1};
bool in(int x,int y){return x>=1&&x<=n*m&&y>=1&&y<=m;}
void exgcd(LL a,LL b,LL &d,LL &x,LL &y)
{
    if(!b)d=a,x=1,y=0;
    else exgcd(b,a%b,d,y,x),y-=x*(a/b);
}
LL inv(LL a,LL p)
{
    LL d,x,y;exgcd(a,p,d,x,y);
    return (x+p)%p==0?p:(x+p)%p;
}
void Gauss()

```

```
{
    for(int i=1;i<=n*m;i++)
    {
        int mxline=i;
        for(int j=i;j<=n*m;j++)
            if(a[j][i]>a[mxline][i])mxline=j;
        if(!a[mxline][i])continue;
        if(mxline!=i)
            for(int j=i;j<=n*m+1;j++)swap(a[mxline][j],a[i][j]);
        for(int j=i+1;j<=n*m;j++)
        {
            if(!a[j][i])continue;
            int t=a[j][i]*inv(a[i][i],3)%3;
            for(int k=i;k<=n*m+1;k++)
                a[j][k]=(a[j][k]-(t*a[i][k]%3)+3)%3;
        }
    }
    for(int i=n*m;i;i--)
    {
        for(int j=n*m;j>i;j--)
            a[i][n*m+1]=(a[i][n*m+1]-f[j]*a[i][j]%3)+3)%3;
        f[i]=a[i][n*m+1]*inv(a[i][i],3)%3;
        if(f[i])res+=f[i];
    }
}
int main()
{
    int T=read();
    while(T--)
    {
        memset(a,0,sizeof(a));
        n=read(),m=read(),res=0;
        for(int i=1;i<=n;i++)
            for(int j=1;j<=m;j++)
                chess[i][j]=read();
        for(int i=1;i<=n;i++)
            for(int j=1;j<=m;j++)
            {
                a[num(i,j)][n*m+1]=(3-chess[i][j])%3;
                a[num(i,j)][num(i,j)]=2;
                for(int k=0;k<4;k++)
                {
                    int tx=i+dx[k],ty=j+dy[k];
                    if(!in(tx,ty))continue;
                    a[num(tx,ty)][num(i,j)]=1;
                }
            }
        Gauss();
        printf("%d\n",res);
        for(int i=1;i<=n;i++)
```

```

    for(int j=1;j<=m;j++)
    for(int k=1;k<=f[num(i,j)];k++)
    printf("%d %d\n",i,j);
}
return 0;
}

```

J-Rower Bo

题目大意

有一条在坐标轴上的河，一条船目前在\$(0,a)\$，水流速度\$v_2\$沿x轴，船的速度\$v_1\$始终指向原点，问船需要多久到达原点。

数据范围

$$0 \leq a, v_1, v_2 \leq 100$$

解题思路

如图

设r为距原点距离, x为x轴上距原点距离.

$$\frac{dr}{dt} = v_1 - v_2 \cos \theta$$

$$\frac{dx}{dt} = v_1 \cos \theta - v_2$$

设T为到达原点所用时间

$$dr = \int_0^T (v_1 - v_2 \cos \theta) dt \Rightarrow v_1 T - v_2 \int_0^T \cos \theta dt = a$$

$$dx = \int_0^T (v_1 \cos \theta - v_2) dt \Rightarrow v_1 \int_0^T \cos \theta dt - v_2 T = 0$$

$$\Rightarrow v_2 T = v_1 \int_0^T \cos \theta dt, v_1 T - a = v_2 \int_0^T \cos \theta dt$$

$$\Rightarrow \frac{v_2 T}{v_1 T - a} = \frac{v_1}{v_2}$$

$$\Rightarrow T = \frac{av_1}{v_1^2 - v_2^2}$$

代码

```
#include <stdio.h>
#include <math.h>
int main()
{
    int a,v1,v2;
    while (scanf("%d%d%d",&a,&v1,&v2)!=EOF) {
        if (a == 0) {printf("0\n");}
        else if (v1 <= v2) {printf("Infinity\n");}
        else { printf ("%lf\n", (double)a*v1/(v1*v1 - v2*v2)); }
    }
    return 0;
}
```

K-Teacher Bo

题目大意

给出 N 个点，问是否存在四元组 (A,B,C,D) 其中 $A < B, C < D, A \neq B$ or $C \neq D$ 使得 A 和 B 之间的曼哈顿距离等于 C 和 D 之间的曼哈顿距离

数据范围

$T \leq 50$

$N \leq 10^5$

$0 \leq \text{坐标范围} \leq 10^5$

解题思路

发现给定的坐标范围比较小，因而两点间曼哈顿距离的种类最多有 $2 \times \text{坐标范围}$ 种，根据鸽巢原理，答案必可以在 $2 \times \text{坐标范围} + 1$ 次找到，所以即使看似是 $O(n^2)$ 的遍历求解实际复杂度也是 $O(\text{坐标范围})$ 的，因而直接暴力即可。

代码

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <algorithm>
using namespace std;
const int N = 1e5+5;
```

```
bool vis[N<<1];
int x[N],y[N];
int main()
{
    int cas;
    scanf("%d",&cas);
    while (cas--) {
        int n,m;
        scanf("%d%d",&n,&m);
        for (int i = 0;i <= 2*m;i++) vis[i] = false;
        for (int i = 1;i <= n;i++) {
            scanf("%d%d",&x[i],&y[i]);
        }
        bool flag = false;
        for (int i = 1;i <= n && !flag;i++) {
            for (int j = i+1;j <= n && !flag;j++)
            {
                int dis = abs(x[i]-x[j])+abs(y[i]-y[j]);
                if (vis[dis]) flag = true;
            }
            for (int j = i+1;j <= n && !flag;j++)
            {
                int dis = abs(x[i]-x[j])+abs(y[i]-y[j]);
                vis[dis] = true;
            }
        }
        if (!flag) printf("NO\n");
        else printf("YES\n");
    }
    return 0;
}
```

replay

13:00-13:30

_wzx27开到A发现可打表找范围做，提交，正确。

Zars19发现J是可做物理题，和_wzx27讨论，好像要积分，然后不会。

发现很多人过B□读题□infinity37提出概率大概是固定的。

13:30-14:00

infinity37找到1/3和1/2概率规律，提交B□正确。

三人讨论K题，过得人太多了，感到迷惑。

infinity37提出曼哈顿距离种数依赖于坐标范围，种数不多，鸽巢原理暴力即可 Zars19附和。

infinity37提交K题，正确。

14:00-14:30

_wzx27开到D题发现可做，提出爆搜DP说。

Zars19加入D题资源开发，提出贪心说（后证实为假）。

14:30-15:00

infinity37阐述了C题概况，手玩得到king解和castle猜想

_wzx27和Zars19加入进行一些讨论

15:00-15:30

讨论了knight平局问题 king解打表得到证实 castle猜想可以直接归纳证明。

进行了很多对queue的讨论 Zars19打了一些表（虽然最后不太有用）

15:30-16:00

_wzx27提出C题皇后的O(NM)做法。去写C

16:00-16:30

_wzx27提交C题，wa两次后正确。

infinity37去积分了，得到J题柿子 Zars19和_wzx27以为然。

infinity37提交J题，wa了2次，其间更正了下无解条件。

16:30-17:00

infinity37使用G++提交J题，正确，之前疑似是选用C++编译器导致的精度问题。

Zars19提出D题高斯消元说，开始写。

17:00-17:30

Zars19提交D题，由于没有continue掉多解消元时主对角线上元素为0的情况TLE一次。然后出现了令人疑惑的RE

17:30-18:00

种种推理下没有道理RE，扩大数组范围再交，数据范围可能假了，D正确。

总结

Zars19

大家这次状态比上次好，不过可能是因为题比上次的可做，区分度没有把我们区分出去，不过我不太行好几个题反应很迟钝orz，rank1是8题，比我们题多的一般会做出来相对更可做的G，我错了这好像是我的树DP和不太可做我们没开到的E好像是主席树）或者I字符串DP，罚时有点多，6题里面我们很靠后了，对一些套路更熟悉的话有些题应该可以更快想到，另外交题的时候可以冷静一点。

infinity37

这次比赛签到题过的比较快，但是中部用了很长时间解题，说明我们的解题速度还有待加强，同时认识到了我们在博弈论上有一些缺陷，那么就这么愉快的决定了，这周就学博弈论（）

_wzx27

这次做得比上次好一点，可能是因为简单一点（不过为什么会有物理题啊）。不过感觉自己手速和思维都好慢啊，有待提高。以及知识点不够全，有些题不怎么有思路，还是得拓宽知识面才行。

From:

<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:

https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:wangzai_milk:20200520%E6%AF%94%E8%B5%9B%E8%AE%B0%E5%BD%95&rev=1590423980

Last update: 2020/05/26 00:26