牛客多校第二场

这场彻底跪了。下文贴了很多莫名WA掉的代码们。如果读者能找到到底是哪里WA掉了，麻烦您联系我一下。

D

唯一一道过了的水题。

```
#include<stdio.h>
#include<math.h>

int h1,h2,m1,m2,s1,s2;
int ans=0;

void solve()
{
    scanf("%d:%d:%d",&h1,&m1,&s1);
    scanf("%d:%d:%d",&h2,&m2,&s2);
    ans=abs((h1-h2)*3600+(m1-m2)*60+s1-s2);
}

int main()
{
    solve();
    printf("%d",ans);
    return 0;
}
```

F

以下是补题。

这个题TLE掉了。原因是单调队列不熟练，还需要一个记忆化的小技巧。

记忆化技巧代码加单调队列：

```
#include<stdio.h>

short Gcd[5010][5010];
short head,tail,q[5010];

int A[5010][5010],b[5010][5010];

int main()
```

```
{
    int n,m,k;
    scanf("%d%d%d",&n,&m,&k);
    long long res=0;
    int i;
    for(i=1;i<=n;i++)
    {
        head=1,tail=0;
        int j;
        for(j=1;j<=m;j++)
        {
            if(!Gcd[i][j])
            {
                int h;
                for(h=1;h*i<=n&&h*j<=m;h++)
                {
                    Gcd[h*i][h*j]=h;
                    A[h*i][h*j]=i*j*h;
                }
            }
            while(tail>=head&&j-q[head]>=k)
            {
                head++;
            }
            while(tail>=head&&A[i][j]>A[i][q[tail]])
            {
                tail--;
            }
            q[++tail]=j;
            if(j>=k)
            {
                b[i][j-k+1]=A[i][q[head]];
            }
        }
    }
    int j;
    for(j=1;j<=m-k+1;j++)
    {
        head=1,tail=0;
        for(i=1;i<=n;i++)
        {
            while(tail>=head&&i-q[head]>=k)
            {
                head++;
            }
            while(tail>=head&&b[i][j]>b[q[tail]][j])
            {
                tail--;
            }
            q[++tail]=i;
```

```
            if(i>=k)
            {
                res+=b[q[head]][j];
            }
        }
    }
    printf("%lld\n",res);
    return 0;
}
```

注意，样例在5000范围，5000乘5000的gcd数组不用short的话会爆内存。

C

我觉得从奇顶点开始DFS一定没错。但是评测机不这么认为……

```
#include<stdio.h>
#include<vector>

using namespace std;

int degree[200010];

vector<int> Adj[200010];
bool vis[200010] = {false};

int flag;

void DFS(int u)
{
    vis[u]=true;
    if(degree[u]%2==1)
    {
        printf("%d",u);
        (flag==0)?putchar(' '):putchar('\n');
        flag^=1;
    }
    int i;
    for(i=0;i<Adj[u].size();i++)
    {
        int v=Adj[u][i];
        if(vis[v]==false)
        {
            DFS(v);
        }
    }
}
```

```
int main()
{
    int n;
    scanf("%d",&n);
    if(n==1)
    {
        printf("1\n1 1\n");
        return 0;
    }
    int m=n-1;
    int x,y;
    int sum=0;
    while(m--)
    {
        scanf("%d%d",&x,&y);
        degree[x]++;
        degree[y]++;
        Adj[x].push_back(y);
        Adj[y].push_back(x);
        if(degree[x]%2==1&&degree[y]%2==1)
        {
            sum++;
        }
        else if(degree[x]%2==0&&degree[y]%2==0)
        {
            sum--;
        }
    }
    printf("%d\n",sum);
    flag=0;
    int u;
    for(u=0;u<n;u++)
    {
        if(vis[u]==false&&degree[u]%2==1)
        {
            DFS(u);
            break;
        }
    }
}
```

后记：我知道错到哪里了！我一直以为是不重复的覆盖……

其实重复也是可以的，也就是说我硬生生拔高了原题的难度……

唉，怪不得。那么代码就更简单了□low的不谈

上面这个代码是不重复覆盖的优秀代码。

下面来源于隔壁队伍的通关代码。我不知道为什么里面DFS还用到队列等等一大堆的东西。

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

#include<queue>

using namespace std;

int head[200005],tot;
int du[200005];

struct E
{
    int nxt,to;
};

struct E e[200005<<1];

void add(int x,int y)
{
    e[++tot].nxt = head[x];
    head[x] = tot;
    e[tot].to = y;
    e[++tot].nxt = head[y];
    head[y] = tot;
    e[tot].to = x;
}

queue<int> sons[200005];

int anss[200005][2],anscnt;

void dfs(int x,int fa)
{
    char isson = 1;
    int tomatch = 0;
    int i;
    for(i=head[x];i;i=e[i].nxt)
    {
        if (e[i].to!=fa)
        {
            dfs(e[i].to,x);
            isson = 0;
            tomatch += sons[e[i].to].size();
        }
    }
    if(isson)
    {
```

```
        sons[x].push(x);
    }
    queue<int> son2,son1;
    for(i=head[x];i;i=e[i].nxt)
    {
        if(e[i].to!=fa)
        {
            if(sons[e[i].to].size()==2)
            {
                son2.push(e[i].to);
            }
            else if(sons[e[i].to].size()==1)
            {
                son1.push(e[i].to);
            }
        }
    }
    if (son2.size()%2==1 && son1.size() == 0 && son2.size()!=1)
    {
        int x1=son2.front();
        son2.pop();
        int x2=son2.front();
        son2.pop();
        int x3=son2.front();
        son2.pop();
        anss[anscnt+1][0]=sons[x1].front();
        sons[x1].pop();
        anss[anscnt+1][1]=sons[x2].front();
        sons[x2].pop();
        anss[anscnt+2][0]=sons[x1].front();
        sons[x1].pop();
        anss[anscnt+2][1]=sons[x3].front();
        sons[x3].pop();
        son1.push(x2);
        son1.push(x3);
        anscnt+=2;
    }
    while(son2.size()>=2)
    {
        int x1 = son2.front();
        son2.pop();
        int x2 = son2.front();
        son2.pop();
        anss[anscnt+1][0] = sons[x1].front();
        sons[x1].pop();
        anss[anscnt+1][1] = sons[x2].front();
        sons[x2].pop();
        son1.push(x1);
        son1.push(x2);
        anscnt++;
```

```
    }
    while(son2.size()*2 + son1.size() > 2)
    {
        if (son2.size())
        {
            int x1 = son2.front();
            son2.pop();
            int x2 = son1.front();
            son1.pop();
            anss[anscnt+1][0] = sons[x1].front();
            sons[x1].pop();
            anss[anscnt+1][1] = sons[x2].front();
            sons[x2].pop();
            son1.push(x1);
            anscnt++;
        }
        else
        {
            int x1 = son1.front();
            son1.pop();
            int x2 = son1.front();
            son1.pop();
            anss[anscnt+1][0] = sons[x1].front();
            sons[x1].pop();
            anss[anscnt+1][1] = sons[x2].front();
            sons[x2].pop();
            anscnt++;
        }
    }
    for(i = head[x];i;i=e[i].nxt)
    {
        if (e[i].to!=fa)
        {
            while (sons[e[i].to].size())
            {
                int tmp = sons[e[i].to].front();
                sons[x].push(tmp);
                sons[e[i].to].pop();
            }
        }
    }
}

int main()
{
    int n;
    scanf("%d",&n);
    int x,y;
    int i;
    for(i=1;i<n;i++)
    {
```

```
        scanf("%d%d",&x,&y);
        add(x,y);
        du[x]++;
        du[y]++;
    }
    if(n==1)
    {
        printf("1\n1 1\n");
        return 0;
    }
    if(n==2)
    {
        printf("1\n1 2\n");
        return 0;
    }
    for(i=1;i<=n;i++)
    {
        if(du[i]!=1)
        {
            dfs(i,0);
            if(sons[i].size()==2)
            {
                anscnt++;
                anss[anscnt][0]=sons[i].front();
                sons[i].pop();
                anss[anscnt][1]=sons[i].front();
            }
            else
            {
                anscnt++;
                anss[anscnt][0]=sons[i].front();
                anss[anscnt][1]=i;
            }
            break;
        }
    }
    printf("%d\n",anscnt);
    for(i=1;i<=anscnt;i++)
    {
        printf("%d %d\n",anss[i][0],anss[i][1]);
    }
    return 0;
}
```

B

储存斜率用了两种做法。开longlong那个TLE掉了□double那个WA了，实在无语……

WA的double已经改正了。那么错的就不留了，只留下错的longlong存斜率版本。

```
#include<iostream>
#include<vector>
#include<map>

using namespace std;

vector<pair<long long,long long> > points;

long long gcd(long long a,long long b)
{
    return (b == 0) ? a : gcd(b, a % b);
}

int maxPoints()
{
    int res = 0;
    for (int i = 0; i < points.size(); ++i)
    {
        map<pair<int, int>, int> m;
        int duplicate = 1;
        for (int j = i + 1; j < points.size(); ++j)
        {
            if (points[i].first== points[j].first&& points[i].second==
points[j].second)
            {
                ++duplicate;
                continue;
            }
            long long x1=points[i].first;
            long long x2=points[j].first;
            long long y1=points[i].second;
            long long y2=points[j].second;
            if(x1*y2==y1*x2)
            {
                continue;
            }
            long long dx =x2*(x1*x1+y1*y1)-x1*(x2*x2+y2*y2);
            long long dy =y2*(x1*x1+y1*y1)-y1*(x2*x2+y2*y2);
            long long d = gcd(dx, dy);
            ++m[{dx / d, dy / d}];
        }
        res = max(res, duplicate);
        map<pair<int,int>,int>::iterator it;
        for(it = m.begin(); it != m.end(); ++it)
        {
            res = max(res, it->second + duplicate);
        }
    }
```

```
    return res;
}

int main()
{
    int n;
    scanf("%d",&n);
    long long x,y;
    while(n--)
    {
        cin >> x >> y;
        points.push_back(make_pair(x,y));
    }
    cout << maxPoints() << endl;
    return 0 ;
}
```

后记：这种采用了反演的写法要考虑eps——

事实证明，没有通过就是eps的问题。唉，太悲伤了……

以下是修改后的通关代码：

```
#include<stdio.h>

#include<vector>
#include<map>

using namespace std;

#define eps 1e-12

struct compare
{
    bool operator()(const double &key1,const double &key2)
    {
        if(key1-key2<(-eps)||key1-key2>eps)
        {
            return key1<key2;
        }
        else
        {
            return 0;
        }
    }
};

vector<pair<double,double> > points;
```

```
int maxPoints()
{
    int res=1;
    int i;
    for(i=0;i<points.size();++i)
    {
        map<double,int,compare> m;
        int j;
        for(j=i+1;j<points.size();++j)
        {
if(points[j].first*points[i].second==points[i].first*points[j].second)
            {
                continue;
            }
            double dx=points[j].first-points[i].first;
            double dy=points[j].second-points[i].second;
            ++m[dx/dy];
        }
        map<double,int>::iterator it;
        for(it=m.begin();it!=m.end();++it)
        {
            res=max(res,it->second+1);
        }
    }
    return res;
}

int main()
{
    int n;
    scanf("%d",&n);
    double x,y;
    while(n--)
    {
        scanf("%lf%lf",&x,&y);
        double temp=x*x+y*y;
        points.push_back(make_pair(x/temp,y/temp));
    }
    printf("%d\n",maxPoints());
    return 0;
}
```

我们需要学习**在map中引入自定义运算符**，从而实现map中的eps控制。

K

我一开始想手算积分，结果发现这个多元积分上下限有绝对值判定，换句话说积不出来。那么就只能使用积分算法了。

```
#include<stdio.h>
#include<math.h>

const double pi=acos(-1.0);

double r1,r2,r3;

double sqr(double x)
{
    return x*x;
}

double f(double j)
{
    double EG=sqrt(sqr(r1)+sqr(r2)-2*cos(j)*r1*r2);
    double AGE=acos((sqr(r1)+sqr(EG)-sqr(r2))/2/r1/EG);
    double alp=asin(r1*sin(AGE)/r3);
    return r3*(alp*sin(alp)+cos(alp))*EG/pi;
}

double swap(double *p1,double *p2)
{
    double temp;
    temp=(*p1);
    (*p1)=(*p2);
    (*p2)=temp;
}

int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        scanf("%lf%lf%lf",&r1,&r2,&r3);
        if(r1>r2)
        {
            swap(&r1,&r2);
        }
        if(r2>r3)
        {
            swap(&r2,&r3);
        }
        if(r1>r2)
        {
            swap(&r1,&r2);
        }
        double res=0,j=pi/10000.0;
        int i;
```

```
        for(i=0;i<5000;i++,j+=pi/5000.0)
        {
            res+=f(j);
        }
        printf("%.1lf\n",res/5000);
    }
    return 0;
}
```

积分算法原来也不难，模拟就好了。反正要求精度不高……

E

集合中可重复取数，求异或的最大值，集合大小很大。

有一个与FFT□NTT非常相似的东西叫FWT□比前两者简单，用于解决数组异或（不进位加法）卷积的问题。

FWT和它的逆，将数组异或（不进位加法）的卷积变为了乘法。

```
#include<stdio.h>

int n,nans[1<<18],one[1<<18],ans[200005];

void FWT(int *src,int t)
{
    int sz;
    for(sz = 2;sz <= 1<<18;sz<<=1)
    {
        int step = sz >> 1;
        int i;
        for(i = 0;i< 1<<18;i+=sz)
        {
            int j;
            for(j = i;j < i+step;j++)
            {
                int a = src[j];
                int b = src[j+step];
                src[j] = a+b;
                src[j+step] = a-b;
                if(t==-1)
                {
                    src[j]>>=1;
                    src[j+step]>>=1;
                }
            }
        }
    }
}
```

```
int main()
{
    scanf("%d",&n);
    int x;
    int i;
    for(i = 1;i<= n;i++)
    {
        scanf("%d",&x);
        one[x]=1;
        nans[x]=1;
        if (x > ans[1])
        {
            ans[1] = x;
        }
    }
    FWT(one,1);
    for(i = 2;i<= 20;i++)
    {
        FWT(nans,1);
        int j;
        for(j = 0;j < 1<<18;j++)
        {
            nans[j] = nans[j]*one[j];
        }
        FWT(nans,-1);
        for(j = 0;j < 1<<18;j++)
        {
            if (nans[j])
            {
                nans[j] = 1;
            }
        }
        for(j=(1<<18)-1;j>=0;j--)
        {
            if (nans[j])
            {
                ans[i] = j;
                break;
            }
        }
    }
    for(i = 21;i<= n;i++)
    {
        ans[i] = ans[i-2];
    }
    for(i = 1;i<= n;i++)
    {
        printf("%d ",ans[i]);
    }
```

```
}
```

J

对给定置换A开k次方□k是个大素数。

由群论知，对大素数k□k次方运算是一一映射，所以实质是个求数论倒数的运算。

由于模不是素数，用费马欧拉定理不方便，只能改用一次不定方程的扩展来求逆。

```
#include<stdio.h>

#include<vector>

using namespace std;

long long extgcd(long long a,long long b,long long& u,long long& v)
{
    long long d;
    if(b==0)
    {
        d = a;
        u=1,v=0;
    }
    else
    {
        d = extgcd(b,a%b,v,u);
        v -= a/b*u;
    }
    return d;
}

int n,k,vis[100005];
int P[100005],ans[100005],a[100005],tmp[100005],ttt[100005];

int main()
{
    scanf("%d%d",&n,&k);
    int i;
    for(i=1;i<=n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(i=1;i<=n;i++)
    {
        if(!vis[i])
        {
            vis[i] = 1;
            vector<int> pos;
```

```
            int u = a[i],len = 1;pos.push_back(i);
            while(u!=i)
            {
                len++;
                vis[u] = 1;pos.push_back(u);
                u = a[u];
            }
            long long x,y;
            extgcd(k,len,x,y);
            x = (x%len+len)%len;
            int j;
            for(j=1;j<=len;j++)
            {
                ans[pos[j-1]] = pos[(j-1+x)%len];
            }
        }
    }
    for(i=1;i<=n;i++)
    {
        printf("%d ",ans[i]);
    }
    return 0;
}
```