Codeforces Round #651 (Div. 2)

A. Maximum GCD

1到n中任选两个数字a,b能够得到的最大$\gcd(a,b)$

题解$\gcd(\lfloor\frac n 2 \rfloor,2·\lfloor\frac n 2 \rfloor)=\lfloor\frac n 2 \rfloor$

```
#include<cstdio>
#include<cstring>
#include<iostream>
using namespace std;
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        int n;
        scanf("%d",&n);
        printf("%d\n",n/2);
    }
    return 0;
}
```

B. GCD Compression

数组a大小为$2n$先任选两个数字丢掉，再将其余数字两两一组变成二者的和化为大小$n-1$的数组b要求使得数组b的最大公因数不为1。

题解：两奇数相加得偶数，两偶数相加还是偶数。只要使得奇、偶数数目为偶数个即可。故如果初始奇、偶数为偶数个，扔掉两奇数或两偶数；初始奇、偶数为奇数个，扔掉一奇一偶。最大公因数2

```
#include<cstdio>
#include<cstring>
#include<iostream>
#include<stack>
using namespace std;
int a[5005];
stack<int>e,o;
int main()
{
    int t;
```

```
    scanf("%d",&t);
    while(t--)
    {
        int n;
        scanf("%d",&n);
        for(int i=1;i<=2*n;i++)
        {
            scanf("%d",&a[i]);
            if(a[i]&1)o.push(i);
            else e.push(i);
        }
        if(o.size()&1)o.pop(),e.pop();
        else if(o.size()>=2)o.pop(),o.pop();
        else e.pop(),e.pop();
        while(!o.empty())
        {
            printf("%d ",o.top());
            o.pop();
            printf("%d\n",o.top());
            o.pop();
        }
        while(!e.empty())
        {
            printf("%d ",e.top());
            e.pop();
            printf("%d\n",e.top());
            e.pop();
        }
    }
    return 0;
}
```

C. Number Game

初始给出正整数n□两玩家轮流操作，每次可以选择在数字大于一的情况下减一或除以大于一的奇因子。无法操作者败

题解：分类讨论。

- n为奇数。除以它本身先手必胜（1是特例，必败）。

- n为偶数且没有大于一的奇因子。先手只能减一之后对方得到上一种局面必胜，故必败（2是特例，必胜）。

- n为偶数有大于一的奇因子。先手将奇因子除掉将第二种局面交给对方，故如果因子2个数不为一，先手必胜。而当因子2个数为一时，若奇因子个数为一□$2·p$减一变奇数必败，除以奇因子必败，总之

必败；若奇因子个数不为一，可以除以部分奇因子将$2\cdot p$的局面交给对方以获得必胜。

```
#include<cstdio>
#include<cstring>
#include<iostream>
using namespace std;
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        int n;
        scanf("%d",&n);
        if(n==1){puts("FastestFinger");continue;}
        if(n==2){puts("Ashishgup");continue;}
        if(n&1){puts("Ashishgup");continue;}
        int cnt1=0,cnt2=0;
        while(n%2==0)n/=2,cnt2++;
        for(int i=3;i*i<=n;i+=2)
        {
            while(n%i==0)n/=i,cnt1++;
        }
        if(n!=1)cnt1++;
        if((cnt2>1&&cnt1>0)||cnt1>1)puts("Ashishgup");
        else puts("FastestFinger");
    }
    return 0;
}
```

D. Odd-Even Subsequence

给定一个长度为n的数组a，可以选择长度为k的子序列（重新标号，从1开始），请最小化$\min(\max(s_1, s_3, s_5, \ldots), \max(s_2, s_4, s_6, \ldots))$。

题解：这个长得还挺二分的哦，check某个mid能否成立的时候，分成使得奇数位满足和使得偶数位满足两种，贪心地看能否凑够子序列长度。

```
#include<cstdio>
#include<cstring>
#include<iostream>
using namespace std;
int n,k,a[200005];
bool check(int x)
{
```

```
    int j=1;
    for(int i=1;i<=n;i++)
    {
        if((j&1)&&a[i]<=x)j++;
        else if((j&1)==0)j++;
        if(j>k)return true;
    }
    j=1;
    for(int i=1;i<=n;i++)
    {
        if((j&1)==0&&a[i]<=x)j++;
        else if(j&1)j++;
        if(j>k)return true;
    }
    return false;
}
int main()
{
    scanf("%d%d",&n,&k);
    int l=1000000000,r=1,ans;
    for(int i=1;i<=n;i++)scanf("%d",&a[i]),l=min(l,a[i]),r=max(r,a[i]);
    while(l<=r)
    {
        int mid=(l+r)>>1;
        if(check(mid))r=mid-1,ans=mid;
        else l=mid+1;
    }
    printf("%d\n",ans);
    return 0;
}
```

E. Binary Subsequence Rotation

给出长度为n的01字符串s,t每次操作可以选一个子序列顺时针旋转，问最少几次从s变到t

题解：能够变换成功当仅当两串为anagram显然满足$s_i=t_i$的位置是不需要被考虑的。而观察发现剩余位置里每次旋转形如$\mathsf{01010101101010}$的子序列是最优的，可以贪心地扫一遍详见代码。

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e6+10;
int n,cnt1,cnt2;
char s[N],t[N];
int main()
{
```

```
    scanf("%d%s%s",&n,s,t);
    for(int i=0;s[i];i++)if(s[i]=='1')cnt1++;
    for(int i=0;t[i];i++)if(t[i]=='1')cnt2++;
    if(cnt1!=cnt2)puts("-1");
    else
    {
        int w0=0,w1=0;    //waiting for 0\waiting for 1
        for(int i=0;s[i];i++)
        if(s[i]!=t[i])
        {
            if(s[i]=='0')
            {
                if(w0)w0--;
                w1++;
            }
            else
            {
                if(w1)w1--;
                w0++;
            }
        }
        printf("%d\n",w0+w1);
    }
    return 0;
}
```

F. The Hidden Pair

交互。给出一棵$n(2 \le n \le 1000)$个节点的树，对方选定两个节点s,f给你猜。你可以做出若干询问，询问给出一个点集a，对方会提供点集中到s,f距离之和最小的点和这个距离。

easy version最多14问，hard version 11问。

题解：我还挺喜欢这个题的，赛时wa了E看这个还挺有思路的，刚刚好写完，结果莫名其妙的wa了就结束了。后来发现是打错了某个地方的变量名，，，

首先可以想到，询问全部点，距离之和最小的只能是在s到f路径上，且距离之和是$d_0=dist(s,f)$，于是我们得到一个sf链上的点x_0，问题数量差不多是$\log n$级别的，于是想到二分。二分得到的d_0，每次dfs得到距离x_0为mid的点集询问，找出最远的使得$d=d_0$的距离及点x_1，则x_1是距离x_0较远一端的端点（得到答案之一）。之后再询问距离x_1为d_0的点集得到另一个答案x_2。

其间，二分区间如果是$[1,d_0]$可能会在二分过程中消耗10个问题，总问题数12过不了hard version，可以想到距离链上一点较远的点距离肯定大于链长之半，所以二分$[\frac {d_0} 2,d0]$可以恰控制在11次询问内。

```
#include<cstdio>
```

```
#include<cstring>
#include<iostream>
#include<vector>
using namespace std;
int n,head[1005],cnt;
char s[100];
vector<int>v;
struct Node
{
    int nxt,to;
}Edges[5005];
void addedge(int u,int v)
{
    Edges[++cnt].nxt=head[u];
    head[u]=cnt;
    Edges[cnt].to=v;
}
void dfs(int u,int f,int d)
{
    if(d==0){v.push_back(u);return;}
    for(int i=head[u];~i;i=Edges[i].nxt)
    {
        int v=Edges[i].to;
        if(v!=f)dfs(v,u,d-1);
    }
}
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        scanf("%d",&n);
        memset(head,-1,sizeof(head));
        cnt=0;
        for(int i=1;i<n;i++)
        {
            int u,v;
            scanf("%d%d",&u,&v);
            addedge(u,v),addedge(v,u);
        }
        printf("? %d ",n);
        for(int i=1;i<=n;i++)printf("%d ",i);
        puts(""),fflush(stdout);
        int x0,d0,x,d;
        scanf("%d%d",&x0,&d0);
        int l=d0/2,r=d0,ans1=x0;
        while(l<=r)
        {
            v.clear();
```

```
            int mid=(l+r)>>1;
            dfs(x0,0,mid);
            if(v.size()==0){r=mid-1;continue;}
            printf("? %d ",v.size());
            for(int i=0;i<v.size();i++)printf("%d ",v[i]);
            puts(""),fflush(stdout);
            scanf("%d%d",&x,&d);
            if(d==d0)ans1=x,l=mid+1;
            else r=mid-1;
        }
        v.clear();
        dfs(ans1,0,d0);
        printf("? %d ",v.size());
        for(int i=0;i<v.size();i++)printf("%d ",v[i]);
        puts(""),fflush(stdout);
        scanf("%d%d",&x,&d);
        printf("! %d %d\n",ans1,x);
        cin>>s;
    }
    return 0;
}
```