2020牛客暑期多校训练营（第五场）

比赛情况

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

O 在比赛中通过 Ø 赛后通过 ! 尝试了但是失败了 - 没有尝试

比赛时间

2020-07-25 12:00-17:00

题解

B - Graph

不好意思我人傻了，这题我三年前做过。

可以发现两点间路径的异或和是不变的，所以可以用 a_u 表示从根节点到这里的异或和 u,v 间的边权即 $a_u\oplus a_v$ 求一个最小生成树。其实不太知道boruvka是什么，但当时yy的一个做法是从高到低位考虑，为 0 的和为 1 的自然而然分两个连通块递归地处理，而两个连通块间连一条边用trie树找最小的。

```
#include<bits/stdc++.h>
#define pa pair<int,int>
#define inf 0x3f3f3f3f
#define mp make_pair
using namespace std;
const int maxn=2e5;
int n,tot,a[maxn],s[maxn],t[maxn];
long long sum;
struct Trie
{
    int cnt,nxt[2];
}tr[maxn*30];
inline int read()
{
    char ch=getchar();int x=0;
    while(!(ch>='0'&&ch<='9'))ch=getchar();
    while(ch>='0'&&ch<='9')x=x*10+ch-'0',ch=getchar();
    return x;
}
inline void init()
{
    for(int i=0;i<=tot;i++)tr[i].nxt[0]=tr[i].nxt[1]=tr[i].cnt=0;
    tot=0;
```

```
}
inline void insert(int x)
{
    int p=0;
    for(int i=30,y;i>=0;i--)
    {
        y=(x>>i)&1;
        if(!tr[p].nxt[y])tr[p].nxt[y]=++tot;
        p=tr[p].nxt[y];
    }
    tr[p].cnt++;
}
inline pa find(int x)
{
    int p=0,ans=0;
    for(int i=30,y;i>=0;i--)
    {
        y=(x>>i)&1;
        if(tr[p].nxt[y])p=tr[p].nxt[y],ans|=y<<i;
        else p=tr[p].nxt[y^1],ans|=(y^1)<<i;
    }
    return mp(ans^x,tr[p].cnt);
}
inline void solve(int l,int r,int dep)
{
    if(l>=r)return;
    if(dep<0)return;
    int cnt1=0,cnt2=0;
    for(int i=l;i<=r;i++)
    if((a[i]>>dep)&1)s[cnt1++]=a[i];else t[cnt2++]=a[i];
    for(int i=0;i<cnt1;i++)a[l+i]=s[i];
    for(int i=0;i<cnt2;i++)a[l+cnt1+i]=t[i];
    init();
    pa tmp;int ans=inf,cnt=0;
    for(int i=0;i<cnt2;i++)insert(t[i]);
    for(int i=0;i<cnt1;i++)
    {
        tmp=find(s[i]);
        if(tmp.first<ans)ans=tmp.first,cnt=tmp.second;
        else if(tmp.first==ans)cnt+=tmp.second;
    }
    if(ans!=inf&&cnt)sum+=ans;
    solve(l,l+cnt1-1,dep-1);solve(l+cnt1,r,dep-1);
}
int head[maxn],cnt;
struct Node{int nxt,to,w;}edges[maxn*2];
void addedge(int u,int v,int
w){edges[++cnt]=Node{head[u],v,w},head[u]=cnt;}
void dfs(int u,int f)
{
```

```
    for(int i=head[u];~i;i=edges[i].nxt)
    {
        int v=edges[i].to;
        if(f!=v)a[v]=a[u]^edges[i].w,dfs(v,u);
    }
}
int main(){
    memset(head,-1,sizeof(head));
    n=read();
    for(int i=1;i<n;i++)
    {
        int u=read(),v=read(),w=read();
        addedge(u,v,w),addedge(v,u,w);
    }
    dfs(0,0);
    solve(1,n,30);
    printf("%lld\n",sum);
    return 0;
}
```

D - Drop Voicing

定义两种移动，分别是

Drop-2：把当前倒数第二个数字拿到最前面来。

Invert：把当前最前面的数字放到最后去

我们定义若干个Drop-2移动是一次操作，问最少要几次操作才能使得序列有序。

猜了个结论，把序列二倍之后以每一个位置为递签，求长度为n的序列中的最长不下降子序列，保留最大答案，最后的答案就是n减去这个答案。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1005;
int f[N],n,a[N];
int main()
{
    scanf("%d",&n);
    for (int i = 1;i<= n;i++)
    {
        scanf("%d",&a[i]);
        a[i+n] = a[i];
    }
    int tans = n;
    for (int i = 1;i<= n;i++)
```

```
    {
        int ans = 0;
        for (int j = 1;j<= n;j++)f[j] = 1;
        for (int j = i;j<=i+n-1;j++)
        {
            for (int k = i;k<j;k++) {
                if (a[j] > a[k])
                    f[j-i+1] = max(f[j-i+1],f[k-i+1]+1);

            }
            ans = max(ans,f[j-i+1]);
        }
        tans = min(tans,n-ans);
    }
    printf("%d\n",tans);
    return 0;
}
```

E - Bogo Sort

给一种置换 P问有多少种排列可以通过多次这个置换变成单位置换。

相当于求置换 P^k 有多少种不同的值，显然答案就是 P 的每个循环节的 lcm然后用 py 水一下高精度就可以了。

```
def gcd(a,b):
    if(b==0):
        return a
    else:
        return gcd(b,a%b)

n = int(input())
p = [0] + [int(x) for x in input().split()]
vis = [0 for i in range(n+1)]

ans = 1

for i in range(1,n+1):
    if(vis[i] == 0):
        u = p[i]
        l = 1
        vis[i] = 1
        while(vis[u] == 0):
            l+=1
            vis[u] = 1
            u = p[u]
```

```
        ans = ans*l//gcd(ans,l)

ans = str(ans)
if len(ans) > n:
    ans = ans[-n:-1]
print(ans)
```

F - DPS

签到题，记录最大值，输出一个50乘以当前值除以最大值长度的方块，最大值对应的方块要在里面画个星星，模拟即可。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 105;
int d[N];
int maxd = 0;
void output(int x,int id) {
    printf("+");
    for (int i = 1;i<= x;i++)
        printf("-");
    printf("+\n");
    printf("|");
    for (int i = 1;i<= x;i++)
    {
        if (i==x && d[id]==maxd) printf("*");
        else printf(" ");
    }
    printf("|%d\n",d[id]);
    printf("+");
    for (int i = 1;i<= x;i++)
        printf("-");
    printf("+\n");
}
int main()
{
    int n;
    scanf("%d",&n);
    for (int i = 1;i<= n;i++)
    {
        scanf("%d",&d[i]);
        maxd = max(maxd,d[i]);
    }
    for (int i = 1;i<= n;i++) {
        double tmp = (double)d[i]*50;
        tmp = tmp/maxd;
```

```
        tmp = ceil(tmp);
        output(tmp,i);
    }
    return 0;
}
```

I - Hard Math Problem

每个H四周要有一个E和一个G，方格填字母，问 n,m 趋于无穷大时，H最多方案能使得其密度为多少。

$\frac23$ 可以使得每个E被 4 个H占有，G同， $1:1:4$ 。

比赛总结与反思