2017-2018 ACM-ICPC, NEERC, Northern Subregional Contest

比赛情况

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

2020-06-24 13:00-18:00

提交记录

题解

A. Auxiliary Project

恰好用点亮 n 根数码管能构成的数字的和最大是多少。

dp 一下就可以了。

```
#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 per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 1e6+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}

int dig[10] = {6,2,5,5,4,5,6,3,7,6};
ll f[maxn];
int main()
{
    fastio();
```

```
    freopen("auxiliary.in","r",stdin);
    freopen("auxiliary.out","w",stdout);
    int n;cin>>n;
    rep(i,1,n){
        rep(j,0,9){
            if(i>=dig[j]&&(f[i-dig[j]]||i==dig[j])) f[i] = max(f[i],f[i-
dig[j]]+j);
        }
    }
    cout<<f[n]<<endl;
    return 0;
}
```

B. Boolean Satisfiability

题目大意

给一个逻辑表达式，符号只有或，非，问其中的变量有多少种赋值方法可以使整个表达式为真。

数据范围

表达式长度≤ 1000

题解

我们知道，对于一个只有或的表达式，那么赋值方法应该为2的变量数目次方-1，另外，对于一个变量，如果只有变量本身或者只有变量取非那么效果是一样的，所以我们需要特殊判断的情况只有一个变量同时有其本身也有其取非的情况，我们知道这两者必有一个为真，那么这个时候我们的答案就是2的变量数目次方。

代码

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <algorithm>
using namespace std;
char s[1005];
int ts[300],fs[300];
int main()
{
    freopen("boolean.in","r",stdin);
    freopen("boolean.out","w",stdout);
```

```
    scanf("%s",s);
    for (int i = 0;s[i];i++)
    {
        if (s[i]=='~')
        {
            fs[s[i+1]]++;
            i++;
        }
        else if ((s[i] >= 'a' && s[i] <= 'z')||(s[i] >= 'A' && s[i] <=
'Z'))
            ts[s[i]]++;
    }
    long long ans = 1;
    int dec = 1;
    for (int i = 'a';i <= 'z';i++)
    {
        if (ts[i] && fs[i]) dec = 0;
        if (ts[i] || fs[i]) ans = ans << 1;
    }
    for (int i = 'A';i <= 'Z';i++)
    {
        if (ts[i] && fs[i]) dec = 0;
        if (ts[i] || fs[i]) ans = ans << 1;
    }
    printf("%lld\n",ans-dec);
    return 0;
}
```

C. Consonant Fencity

题目大意

题目重新定义元音字母□bushi□□现在有19个辅音字母个7个元音字母。定义一个字符串的幸福值，为相邻两个辅音字母的大小写不同的数目，现在给你一个全为小写字母的字符串，想通过把某些字母全部变为大写来达到最大的幸福值，问应该怎么修改才能幸福值最大。

数据范围

字符串长度$\leq 10^6$

题解

首先我们知道只有修改辅音字母的时候会对幸福值产生影响，而现在辅音字母只有19个，所以只需要二进制枚举每一个字母是否改为大写然后进行计算即可。

代码

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <algorithm>
using namespace std;
const int N = 1e6+5;
int myno[300];
int ps[25][25];
char s[N];
bool isbig[25];
int main()
{
    freopen("consonant.in","r",stdin);
    freopen("consonant.out","w",stdout);
    int cnt = 0;
    for (int i = 'a';i <= 'z';i++)
    {
        if (i == 'a' || i == 'e' || i == 'i' || i == 'o' || i == 'u' || i
== 'w' || i == 'y')
            continue;
        myno[i] = ++cnt;
    }
    scanf("%s",s);
    for (int i = 1;s[i];i++)
        if (myno[s[i]] && myno[s[i-1]]) {
            if (s[i] > s[i-1])ps[myno[s[i-1]]][myno[s[i]]]++;
            else ps[myno[s[i]]][myno[s[i-1]]]++;
        }
    int maxf = -1;
    int ans = 0;
    for (int i = 0;i < (1<<19);i++) {
        for (int j = 0;j < 19;j++)
        {
            if ((i >> j) & 1)
                isbig[j+1] = true;
            else
                isbig[j+1] = false;
        }
        int tmpf = 0;
        for (int j = 1;j<= 19;j++)
            for (int k = j+1;k<= 19;k++)
                if (isbig[j]^isbig[k])
                    tmpf += ps[j][k];
        if (tmpf > maxf) {
            maxf = tmpf;
            ans = i;
        }
```

```
    }
    for (int j = 0;j < 19;j++)
    {
        if ((ans >> j) & 1)
            isbig[j+1] = true;
        else
            isbig[j+1] = false;
    }
    for (int i = 0;s[i];i++)
    {
        if (isbig[myno[s[i]]])
            printf("%c",s[i]-'a'+'A');
        else printf("%c",s[i]);
    }
    return 0;
}
```

H. Hidden Supervisors

一个森林合并成一棵树，且求最大的匹配数，匹配是指取一个父亲和它的一个儿子。一个节点不能匹配多次。

先处理每课树，猜一个贪心的结论，从叶子开始往上，能和自己的父亲匹配就立刻匹配。 合并树的时候，每次合并最多有 1 的贡献，贪心地让每个没有被匹配过的树根去连接另一棵树没有匹配过的节点。但注意最后要形成一棵树，比赛的时候写了好多次才过(而且写得很复杂)。。。

```
#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 TIMES 10
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 1e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}

int n,p[maxn],has[maxn],ans,belong[maxn];
vector<int> g[maxn],no[maxn];
void dfs(int u,int f,int rt)
{
```

```
    belong[u] = rt;
    for(int v:g[u]){
        dfs(v,u,rt);
    }
    if(!has[u] && (f!=0&&!has[f])) has[u]=has[f]=1,ans++;
    if(!has[u]) no[rt].pb(u);
}
struct node
{
    int id;
    bool operator < (node e) const
    {
        return no[id].size()>no[e.id].size();
    }
};
#define LOCAL
int main()
{
    fastio();

    #ifdef LOCAL
    freopen("hidden.in","r",stdin);
    freopen("hidden.out","w",stdout);
    #endif

    cin>>n;
    vector<int> root;
    rep(i,2,n){
        cin>>p[i];
        if(!p[i]) root.pb(i);
        else g[p[i]].pb(i);
    }
    vector<int> tot,rest;
    dfs(1,0,1); for(int x:no[1]) tot.pb(x);
    int cnt = 0;
    for(int x:root){
        dfs(x,0,x);
        if(has[x]){
            for(int y:no[x]) tot.pb(y);
            p[x] = 1;
        }else rest.pb(x);
    }
    auto cmp = [&] (int x,int y) {return no[x].size()>no[y].size();};
    sort(rest.begin(),rest.end(),cmp);
    for(int x:rest){
        if(tot.size()){
            p[x] = tot.back(); tot.pop_back(); ans++;
        }else{
            p[x] = 1; tot.pb(x);
        }
```

```
        for(int y:no[x]){
            if(y!=x) tot.pb(y);
        }
    }
    cout<<ans<<endl;
    rep(i,2,n) cout<<p[i]<<" \n"[i==n];
    return 0;
}
/*
7
0 0 0 0 0 0
*/
```

I. Intelligence in Perpendicularia

题目大意

在坐标系里放一圈平行于x轴或y轴的栅栏，人分别站在$(0,inf)$,$(0,-inf)$,$inf,0$,$-inf,0$四个位置看栅栏，问有多长的栅栏不论站在这四个位置哪里都没法看到。

数据范围

n节栅栏$1\leq n\leq 1000$

栅栏坐标$-10^6 \leq x_i,y_i \leq 10^6$

题解

由于栅栏只有平行于x轴和y轴的，而且栅栏围成了一个封闭图形，所以必有一种平移方案把栅栏移动成一个矩形和剩余的一些栅栏，易知人能看到的就是这个矩形的长度，所以剩余的就是看不到的栅栏。我们只需要计算总的栅栏长度和这个矩形的周长接口。

代码

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <algorithm>
using namespace std;
typedef long long ll;
const int N = 1005;
int x[N],y[N];
int main()
{
```

```
    freopen("intel.in","r",stdin);
    freopen("intel.out","w",stdout);
    int n;
    scanf("%d",&n);
    int maxx = -1e7;
    int maxy = -1e7;
    int minx = 1e7;
    int miny = 1e7;
    for (int i = 1;i<= n;i++)
    {
        scanf("%d%d",&x[i],&y[i]);
        maxx = max(maxx,x[i]);
        minx = min(minx,x[i]);
        maxy = max(maxy,y[i]);
        miny = min(miny,y[i]);
    }
    int ans = 0;
    x[n+1] = x[1],y[n+1] = y[1];
    for (int i = 1;i <= n;i++)
        ans += (abs(x[i]-x[i+1])+abs(y[i]-y[i+1]));
    ans -= (maxx-minx + maxy-miny)*2;
    if (maxx == minx || maxy == miny) ans = 0;
    printf("%d\n",ans);
    return 0;
}
```

replay

比赛总结