

codeforces round 652

A FashionableLee

题意：给 t 个正 n 边形，问这 t 个正多边形是否满足至少存在一条边水平并且存在一条边竖直。

题解：画几个就知道 n 为 4 的倍数时满足条件

B AccurateLee

题意：给一个01字符串，可以对字符串中相连的“10”串做操作，每次可以将其中的“1”或者“0”删去，问操作到最后的子串中字典序最小的时什么？

题解：首先可以明确，一个字符串如果以0打头，则这些最前面的0都消不去，如果以1结尾，则这些最后面的1都消不去，若存在中间部分，中间的10组合一定存在一种消除方法使得最后只剩下一个0，则答案便是将头，中，尾进行拼接的结果。

C RationalLee

题意：一共有 n 个数，要分给 k 个人，第 i 个人会拿到 w_i 个数，问如何分配，才能使每个人拿到的数中最大和最小数之和的求和最大？

题解：首先分析题意，每个人的拿到的最大和最小值都会对答案做出贡献，先考虑最大值的贡献，每个数只能分给一个人，那么最大值的最大贡献一定是排名后 k 位的人拿到数的和，之后考虑最小值，如果那个人就拿一个那么最小值的贡献一定是等于最小值的，由此出发，发现拿的数越少，可能得到的最小数越小，于是将人按照 w 从小到大排序，将 a 数组按照从大到小排序，从第 $k+1$ 个数开始分配数即可，这样必定能保证最小值做的贡献最大。

D TediousLee

题意：难以表述，放链接<https://codeforces.com/contest/1369/problem/D>

题解：令第 n 种的答案为 $dp[n]$ 画几种情况会发现，第 n 种情况的树的根的左右分支的形状都是为 $n-2$ 的情况，而中间分支为 $n-1$ 的情况，所以大概可以看出 $dp[n]$ 与 $dp[n-1]$ 和 $dp[n-2]$ 有关，进一步观察，发现，只有在 $3n$ 的情况下，根这个节点才会被占用，而 $3n+1, 3n+2$ 这两种情况下，根节点不会被占用，则不难发现 $3n+3$ 的情况是由一个 $3n+2$ 与两个 $3n+1$ 组成的，而这三个未被占用的根又与新根组成了一个满足条件的情况所以是 $2dp[3n+1]+dp[3n+2]+1$ ，而其他的情况则不存在这个问题，那么 dp 转移方程便是 $dp[n]=dp[n-1]+2dp[n-2]+(n \bmod 3 == 0)$

E DeadLee

题意：有 n 种食物和 k 个人，规定第 i 种食物的容量为 w_i 而第 i 个人喜欢吃第 a_i 种食物和 b_i 种食物，人按照一定的顺序来吃食物，如果他喜欢吃的食物还有剩余，他都会吃一个，有两个还有

剩余就分别吃一个，否则哪个剩下了吃哪个，如果没有剩下就要吃我，我好怕，问有没有一种吃的顺序保障不会出现没得吃的情况。

题解：这是一个贪心的问题，最不会贪心子，首先记录下，每种食物有几个人爱吃，如果第 i 种食物爱吃的人小于总容量，则爱吃这种食物的人一定是后被选择的，因为一定能让这些吃饱，考虑将这些人爱吃的另一种食物扩大（假设他们不会吃），这样就会有新的满足人数小于容量的食物出现，再次消除，直到小不点下去位置，看答案存的人数是否大于 k 即可。需要再较小复杂度内进行删除插入工作所以我们开一个 $\text{set}, \text{set}[k]$ 中存喜欢吃 k 的人。

```
#include <bits/stdc++.h>
using namespace std;
const int maxn=2e5+13;
struct node{ int x,y;
}a[maxn]; //存每个人喜欢哪两种食物
set<int> b[maxn]; //存每种有几个人喜欢吃 set log 删除和插入
int w[maxn]; //存每种食物的总量
queue<int> q;
int ans[maxn];
int flag[maxn];
int cnt;
int main()
{
    int n,m,u,v;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&w[i]);
    }
    for(int i=1;i<=m;i++)
    {
        scanf("%d%d",&u,&v);
        b[u].insert(i);
        b[v].insert(i);
        a[i].x=u,a[i].y=v;
    }
    for(int i=1;i<=n;i++)
    {
        if(b[i].size()<=w[i])
        {
            q.push(i);
            flag[i]=1;
        }
    }
    while(!q.empty())
    {
        int now=q.front();
        q.pop();
        while(!b[now].empty())
        {
```

大

```

int t=*b[now].begin();
b[a[t].x].erase(t);
b[a[t].y].erase(t);
int k=(a[t].x==now?a[t].y:a[t].x);
if(!flag[k]&&b[k].size()<=w[k])//抹消这个人的存在, 并将另一个容量扩
{
    flag[k]=1;
    q.push(k);
}
ans[++cnt]=t;
}
}
if(cnt<m) return printf("DEAD"),0;
printf("ALIVE\n");
for(int i=cnt;i;i--)
{
    printf("%d ",ans[i]);
}
}

```

F BareLee

题意：博弈问题，\$a\$和\$b\$玩游戏，共进行\$n\$轮游戏，每轮游戏的先手为上场比赛的输者，比赛内容如下，两个数\$a\$和\$b\$满足\$a < b\$，每次可以将\$a\$变成\$a+1\$或者是\$2a\$，第一个超过的\$b\$的玩家输。第一轮现手为主人公，问主人公是否有策略使自己\$n\$轮下来必输或者必赢。

题解：恶心的博弈论，记每轮的数为\$s\$和\$e\$，可以先分别计算必胜或者必输的可能，假设第\$i\$论必胜可能为\$win[i]\$，必输的可能为\$lose[i]\$，首先考虑\$win\$

- 如果\$e\$为奇数，则先手没有胜利的可能，因为对方总可以保证轮到现手时为奇数，而先手会最先将其变成抄过\$e\$的偶数。(归纳法也可以证)
- 如果\$e\$为偶数；
- 如果有\$\frac{e}{2} < s\$，则若\$s\$为奇数，先手必胜(只能一个一个的加)
- 如果有\$\frac{e}{2} \geq s > \frac{e}{4}\$，则先手必胜，(先手乘2转化为上一种情况)
- 如果有\$s < \frac{e}{4}\$，则问题直接转化为\$win(s, \frac{e}{4})\$

再考虑必输的可能性

- 既然想快点输那么肯定拼命乘2，如果一开始有\$\frac{e}{2} < s\$，则先手可以必输，否则，问题转化为谁先让局面达到\$\frac{e}{2} < s\$这种情况，则对方可以必输，即\$win(s, \frac{e}{2})\$

考虑完一局中必胜与必输，注意这含义是可以凭借自己的意愿在先手时必胜或必输。如果在上一句满足必胜必输都可以，则比赛完全掌控在先手的手里，之后先手可以随意获胜或输掉来获得最终的结果。如果上一句既没有办法获胜有没有办法一定输，那么比赛掌握在对方手里，对方可以操控比赛，上一句可以必胜且没办法输，则对方先手，结果要取反，上一句不能保证赢但是必输，则自己先手。这样最后可以得到答案。(双方都会尽量让自己掌握比赛的主导权)

```

#include <bits/stdc++.h>
using namespace std;

```

```
const int maxn=1e5+13;
typedef long long ll;
int win[maxn]; //记录每一局是否肯定赢
int lose[maxn]; //记录每一局是否肯定输
ll s[maxn], e[maxn];
bool getwin(ll s, ll e) //能否根据自己的意愿让先手赢
{
    if(e&1)
    {
        if(s%2==0)
        {
            return 1;
        }
        else return 0;
    }
    else
    {
        if(2*s>e)
        {
            if(s&1) return 1;
            else return 0;
        }
        else if(4*s>e) return 1;
        else return getwin(s, e/4);
    }
}

bool getlose(ll s, ll e) //能否根据自己的意愿让先手输
{
    if(2*s>e) return 1;
    else return getwin(s, e/2);
}

int main()
{
    int t;
    scanf("%d", &t);
    lose[0]=1;
    for(int i=1; i<=t; i++) scanf("%lld%lld", &s[i], &e[i]);
    for(int i=1; i<=t; i++)
    {
        if(lose[i-1]&&win[i-1])
        {
            return printf("1 1"), 0;
        }
        else if(!lose[i-1]&&!win[i-1])
        {
            return printf("0 0"), 0;
        }
        else if(lose[i-1])
        {
            win[i]=getwin(s[i], e[i]);
        }
    }
}
```

```
        lose[i]=getlose(s[i],e[i]);
    }
    else
    {
        win[i]=!getwin(s[i],e[i]);
        lose[i]=!getlose(s[i],e[i]);
    }
}
printf("%d %d",win[t],lose[t]);
}
```

From:

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

Permanent link:

https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:manespace:codeforces_round_652_div2 

Last update: **2020/06/27 23:16**