Atcoder Rugular Contest 122

比赛链接

C - Calculator

题意

给定 x,y 初值均为 0，接下来给定 4 种操作：

1. $x\gets x+1$
2. $y\gets y+1$
3. $x\gets x+y$
4. $y\gets x+y$

要求在 130 步操作内将 x 变为 $N(N\le 10^{18})$

赛时解法

不妨先规定一个 y 的最终值，然后利用操作 $3,4$ 对 x,y 进行更相减损术，当 x,y 其中一个为 0 时再利用操作 $1,2$ 暴力处理。

最后逆序输出即可。操作次数等于更相减损术次数加上 $\text{gcd}(x,y)$ 不难发现斐波那契数列是最理想的情况，但 N 不一定是斐波那契数。

不妨强制认为 N 是斐波那契数，于是根据斐波那契数通项公式不妨猜想 y 的最终值在 $\frac {2N}{\sqrt 5+1}$ 附近。

将 $\frac {2N}{\sqrt 5+1}-20\le y\le \frac {2N}{\sqrt 5+1}+20$ 都代入尝试即可。

```
LL cal(LL n,LL p){
    int pos=3;
    LL ans=0;
    while(p>0){
        LL t=n;
        while(t>=p){
            t-=p;
            ans++;
        }
        if(pos==3)pos=4;
        else pos=3;
        n=p;
        p=t;
    }
    return ans+n;
}
```

```
int main()
{
    LL n=read_LL();
    LL v=n*2/(sqrt(5)+1);
    for(LL i=max(v-20,0LL);i<=min(v+20,n);i++){
        if(cal(n,i)<125){
            LL p=i;
            stack<int> s;
            int pos=3;
            while(p>0){
                LL t=n;
                while(t>=p){
                    t-=p;
                    s.push(pos);
                }
                if(pos==3)pos=4;
                else pos=3;
                n=p;
                p=t;
            }
            enter(s.size()+n);
            LL tn=0,tp=0;
            _for(i,0,n){
                if(pos==3){
                    tn++;
                }
                else
                tp++;
                enter(pos-2);
            }
            while(!s.empty()){
                enter(s.top());
                if(s.top()==3)
                tn=tn+tp;
                else
                tp=tn+tp;
                s.pop();
            }
            return 0;
        }
    }
    return 0;
}
```

正解

假定操作序列为 $4,3,4,3,4,3\cdots $共操作 S 次，且最后一次操作为 3。

接下来考虑在该操作序列中插入 $1,2$ 操作，定义 $F(0)=F(1)=1,F(n)=F(n-1)+F(n-2)$

不妨规定仅在操作 3 后面或者最开始插入操作 1，操作 4 后面插入操作 2。

不难发现在第 $i(0\le i\le S)$ 次操作后插入一个操作最后对 x 的贡献为 $F(S-i)$。

于是问题转化为将 N 分解为若干斐波那契数。取 $S=\max\{F(S)\le 10^{18}\}=86$。

最后贪心分解 N 即可。显然 $F(i),F(i-1)$ 不可能同时存在与 N 的分解，否则可以用 $F(i+1)$ 替代，于是最大操作数为 $\lceil\frac {(S+1)}2\rceil=44$。

于是总操作数不超过 130 次。

D - XOR Game

题意

给定 $2N$ 个数和两名玩家。

两名玩家轮流操作，每轮玩家 A 先选一个数并删除该数，玩家 B 再选一个数并删除该数，然后该轮得分为两个数的异或和。

游戏总分为 n 轮得到的最大值，玩家 A 希望最大化得分，玩家 B 希望最小化得分，输出最终得分。

题解

显然玩家 A 的操作是没有意义的，因为可以先假设玩家 B 操作两个人，得到最优方案。然后玩家 A 实际操作时进行配对即可。

于是问题转化为最小化 N 对数的异或和的最大值。从高位到低位考虑得到，假设当前考虑到第 i 位。

若第 i 位为 0 和 1 的数都有偶数个，那显然第 i 位为 0 的数之间相互配对，第 i 位为 1 的数之间相互配对，直接递归即可。

否则答案一定为从第 i 位为 0 的数中选一个和从第 i 位为 1 的数中选一个的异或和的最小值。时间复杂度 $O(n\log v)$。

```
const int MAXN=4e5+5,MAXL=30,MAXS=MAXN*MAXL,inf=1<<30;
int a[MAXN],ch[MAXS][2],node_cnt;
void Insert(int v){
    int pos=0;
    for(int i=MAXL-1;i>=0;i--){
        int d=(v>>i)&1;
        if(!ch[pos][d]){
            ch[pos][d]=++node_cnt;
            ch[node_cnt][0]=ch[node_cnt][1]=0;
        }
        pos=ch[pos][d];
    }
}
int query(int v){
```

```
    int pos=0,ans=0;
    for(int i=MAXL-1;i>=0;i--){
        int d=(v>>i)&1;
        if(!ch[pos][d]){
            d^=1;
            ans|=1<<i;
        }
        pos=ch[pos][d];
    }
    return ans;
}
int solve(int lef,int rig,int pos){
    if(lef>rig||pos<0)return 0;
    int mid=lef-1;
    while(mid+1<=rig&&!((a[mid+1]>>pos)&1))mid++;
    if((mid-lef+1)%2==0)
    return max(solve(lef,mid,pos-1),solve(mid+1,rig,pos-1));
    ch[0][0]=ch[0][1]=0;
    node_cnt=0;
    _rep(i,lef,mid)Insert(a[i]);
    int ans=inf;
    _rep(i,mid+1,rig)
    ans=min(ans,query(a[i]));
    return ans;
}
int main()
{
    int n=read_int()*2;
    _for(i,0,n)a[i]=read_int();
    sort(a,a+n);
    enter(solve(0,n-1,MAXL-1));
    return 0;
}
```

E - Increasing LCMs

题意

给定长度为 N 的序列 A 要求对序列进行重新排列，使得 $B_i=\text{LCM}_{j=1}^i A_j$ 严格单调递增。

题解

首先确定 A 重新排列后最后一个元素 A_i 一定要满足条件 $\text{GCD} \left(A_i,\text{LCM}_{j\neq i}A_j\right)\neq A_i$ 发现 $\text{LCM}_{j\neq i}A_j$ 太大了，不利于计算。

不难发现，上述条件等价于 $\text{LCM}_{j\neq i} \left(\text{GCD}(A_i,A_j)\right)\neq A_i$

如果满足条件的 A_i 不存在，显然答案不存在。否则将所有满足条件的 A_i 以任意顺序放到序列尾部，然后处理剩余部分即可。

时间复杂度 $O(n^3\log v)$

```
const int MAXN=105;
LL a[MAXN];
LL gcd(LL a,LL b){
    while(b){
        LL t=b;
        b=a%b;
        a=t;
    }
    return a;
}
LL lcm(LL a,LL b){
    if(a==0||b==0)
    return a|b;
    LL g=gcd(a,b);
    return (a/g)*(b/g)*g;
}
int main()
{
    int n=read_int();
    _for(i,0,n)a[i]=read_LL();
    for(int i=n-1;i>=0;i--){
        int pos=-1;
        _rep(j,0,i){
            LL s=0;
            _rep(k,0,i){
                if(k==j)continue;
                s=lcm(s,gcd(a[j],a[k]));
            }
            if(s!=a[j]){
                pos=j;
                break;
            }
        }
        if(pos==-1){
            puts("No");
            return 0;
        }
        swap(a[pos],a[i]);
    }
    puts("Yes");
    _for(i,0,n)
    space(a[i]);
    return 0;
```

```
}
```

F - Domination

题意

二维平面给定 n 个红点和 m 个蓝点，可以任意次移动某个蓝点，费用为曼哈顿距离。

问满足下述情况的最小费用：

对每个红点 (x_r,y_r) 至少有 k 个蓝点 (x_b,r_b) 满足 $x_b\ge x_r,y_b\ge y_r$。

题解

首先考虑 $k=1$ 且只有一个红点 (x_r,y_r) 和多个蓝点的情况。

重新建图，图中只有 X 轴和 Y 轴上的点，且 X 轴的 $(0,0)$ 和 Y 轴的 $(0,0)$ 不是同一个点。

$(x,0)\to (x+1,0)$ 费用为 1，$(x+1,0)\to (x,0)$ 费用为 0， $(0,y)\to (0,y+1)$ 费用为 0，$(0,y+1)\to (0,y)$ 费用为 1。

对每个蓝点，代表一条从 $(0,y_b)$ 连向 $(x_b,0)$ 的费用为 0 的边。

于是红点被该蓝点覆盖的费用恰好为路径 $(0,y_r)\to (0,y_b)\to (x_b,0)\to (x_r,0)$ 的费用。

所以该情况下的最小费用为 $(0,y_r)\to (x_r,0)$ 的最短路。

首先对每个红点 (x_1,y_1)，如果存在另一个红点 (x_2,y_2) 满足 $x_2\ge x_1,y_2\ge y_1$，那 (x_1,y_1) 显然可以删除。

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

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:contest:arc_122&rev=1625217146

Last update: **2021/07/02 17:12**