同余最短路

算法简介

用于计算 $k_1a_1+k_2a_2+\cdots +k_na_n$ 在 $[0,m]$ 范围内能表示的数的算法。

算法实现

考虑建点 $0,1\cdots a_1-1$，然后对每个点 i 连边 $i\to (i+a_j)\bmod a_1(w=a_j)$，再跑最短路。

于是可以 $O(n\times a\log a)$ 计算出最小的可以表示成 $ka_1+r(0\le r\lt a_1)$ 的数，于是每个 r 对答案的贡献为 $\lfloor \frac {m-\text{dis}(r)}{a_1}\rfloor$。

不难发现可以重新排序选最小的 a 作为 a_1，另外每个点的相邻点可以在跑最短路算法时动态计算，这些都有利于卡常和节省空间。

算法例题

例题一

洛谷p2371

板子题。

```
const int MAXN=15,MAXV=5e5+5;
const LL inf=1e18;
int a[MAXN];
LL dis[MAXV];
bool vis[MAXV];
void dj(int n){
    priority_queue<pair<LL,int> >q;
    q.push(make_pair(0,0));
    _for(i,1,a[0])
    dis[i]=inf;
    while(!q.empty()){
        int u=q.top().second;q.pop();
        if(vis[u])
        continue;
        vis[u]=true;
        _for(i,1,n){
            int v=(u+a[i])%a[0],w=a[i];
            if(dis[v]>dis[u]+w){
                dis[v]=dis[u]+w;
                q.push(make_pair(-dis[v],v));
```

```
            }
        }
    }
}
LL calc(LL val){
    LL ans=0;
    _for(i,0,a[0]){
        if(dis[i]<=val)
        ans+=(val-dis[i])/a[0]+1;
    }
    return ans;
}
int main(){
    int n=read_int();
    LL ql=read_LL(),qr=read_LL();
    _for(i,0,n)
    a[i]=read_int();
    sort(a,a+n);
    dj(n);
    enter(calc(qr)-calc(ql-1));
    return 0;
}
```

例题二

HDU6071

题意

给定一个四元环，环上每条边有一个边权，且重复经过则计算多次贡献。

人物从二号点出发，最终回到二号点，问权值不小于 k 的路径的最小权值。

题解

任取一条与二号点相邻的边，设权值为 w□设 $\text{dis}(i,j)$ 表示从二号点出发到达 i 号点且距离模 $2w$ 等于 j 的最小距离。

那么从二号点出发能得到的路径的权值集合为 $\{\text{dis}(2,r)+2kw|0\le r\lt 2w,k\ge 0\}$□枚举 r 即可计算答案。

关于为什么选取 $2w$□可以解释为回到 2 号点后可以在任意一条边上来回移动，正确性不难证明。

```
const int MAXN=5,MAXV=6e4+5;
const LL inf=2e18;
int d[MAXN];
```

```
vector<pair<int,int> > g[MAXN];
LL dis[MAXN][MAXV];
bool vis[MAXN][MAXV];
void dj(int n){
    priority_queue<pair<LL,int> > q;
    q.push(make_pair(0,1));
    _for(i,0,4)_for(j,0,n)
    dis[i][j]=inf,vis[i][j]=false;
    dis[1][0]=0;
    while(!q.empty()){
        int u1=q.top().second;
        int u2=(-q.top().first)%n;
        q.pop();
        if(vis[u1][u2])
        continue;
        vis[u1][u2]=true;
        for(pair<int,int> p:g[u1]){
            int v1=p.first,w=p.second,v2=(u2+w)%n;
            if(dis[v1][v2]>dis[u1][u2]+w){
                dis[v1][v2]=dis[u1][u2]+w;
                q.push(make_pair(-dis[v1][v2],v1));
            }
        }
    }
}
LL calc(int val,LL k){
    LL ans=inf;
    _for(i,0,val){
        if(dis[1][i]>k)
        ans=min(ans,dis[1][i]);
        else
        ans=min(ans,dis[1][i]+(k-dis[1][i]+val-1)/val*val);
    }
    return ans;
}
void solve(){
    LL k=read_LL();
    _for(i,0,4)
    g[i].clear();
    _for(i,0,4){
        d[i]=read_int();
        g[i].push_back(make_pair((i+1)%4,d[i]));
        g[(i+1)%4].push_back(make_pair(i,d[i]));
    }
    dj(2*d[0]);
    enter(calc(2*d[0],k));
}
int main(){
    int T=read_int();
    while(T--)
    solve();
```

```
    return 0;
}
```

例题三

CF986F

题意

T 组询问，每组询问给定 n,k，求 n 是否能用若干个 k 的不为 1 的因子表示。

数据保证 k 的取值不超过 50 个。

题解

如果 n 能用若干个 k 的不为 1 的因子表示等价于 n 能用 k 的素因子表示。

预处理 $O(\sqrt k)$ 范围的数后对每个 k 进行因式分解，时间复杂度 $O\left(\frac {\sqrt k}{\ln k}\right)$。

如果 $k=1$ 显然无解。如果 k 是质数，只需要判定 $k\mid n$ 是否成立。

如果 k 有两种素因子，记为 a,b 于是等价于求解 $ax+by=n$ 的非负整数解。

找到最小的 y 满足 $y\equiv nb^{-1}\bmod x$ 然后验证此时 by 是否不超过 n 即可。

如果 k 有至少三种素因子，显然 k 的最小素因子不超过 $O\left(k^{\frac 13}\right)$ 跑同余最短路算法即可，时间复杂度 $O\left(k^{\frac 13}\log k\right)$。

总时间复杂度 $\left(50\left(\frac{\sqrt k}{\ln k}+k^{\frac 13}\log k\right)\right)$。

```
const int MAXV=4e7,MAXN=1e5+5,MAXQ=1e4+5;
const LL inf=1e18+5;
int prime[MAXV],pcnt;
bool vis[MAXV];
void Init(){
    _for(i,2,MAXV){
        if(!vis[i])prime[pcnt++]=i;
        for(int j=0;j<pcnt&&i*prime[j]<MAXV;j++){
            vis[i*prime[j]]=true;
            if(i%prime[j]==0)
            break;
        }
    }
}
vector<LL> g;
```

```
void get_frac(LL n){
    g.clear();
    _for(i,0,pcnt){
        if(n%prime[i]==0){
            g.push_back(prime[i]);
            while(n%prime[i]==0)n/=prime[i];
        }
    }
    if(n!=1)
    g.push_back(n);
}
int quick_pow(int n,int k,int mod){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*n%mod;
        n=1LL*n*n%mod;
        k>>=1;
    }
    return ans;
}
LL dis[MAXN];
void dj(int n){
    _for(i,0,n){
        dis[i]=inf;
        vis[i]=false;
    }
    dis[0]=0;
    priority_queue<LL> q;
    q.push(0);
    while(!q.empty()){
        int u=(-q.top())%n;q.pop();
        if(vis[u])
        continue;
        vis[u]=true;
        _for(i,1,g.size()){
            int v=(u+g[i])%n;
            if(dis[v]>dis[u]+g[i]){
                dis[v]=dis[u]+g[i];
                q.push(-dis[v]);
            }
        }
    }
}
bool query(LL n){
    if(g.size()==0)
    return false;
    if(g.size()==1)
    return n%g[0]==0;
    else if(g.size()==2){
        LL x=g[0],y=g[1];
        int b=(n%x)*quick_pow(y%x,x-2,x)%x;
```

```
        return y*b<=n;
    }
    else{
        int m=g[0];
        return dis[n%m]<=n;
    }
}
map<LL,vector<pair<int,LL> > >mp;
bool ans[MAXQ];
void solve(LL k,vector<pair<int,LL> > a){
    get_frac(k);
    if(g.size()>2)
    dj(g[0]);
    for(pair<int,LL> p:a)
    ans[p.first]=query(p.second);
}
int main(){
    Init();
    int q=read_int();
    _for(i,0,q){
        LL n=read_LL(),k=read_LL();
        mp[k].push_back(make_pair(i,n));
    }
    for(map<LL,vector<pair<int,LL> > >::iterator
iter=mp.begin();iter!=mp.end();iter++)
    solve(iter->first,iter->second);
    _for(i,0,q){
        if(ans[i])
        puts("YES");
        else
        puts("NO");
    }
    return 0;
}
```