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;
}
```

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

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E5%90%8C%E4%BD%99%E6%9C%80%E7%9F%AD%E8%B7%AF&rev=1630755798

Last update: **2021/09/04 19:43**