

同余最短路

算法简介

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

算法实现

考虑建点 $0,1,\dots,a_1-1$ 然后对每个点 i 连边 $i \rightarrow (i+a_j) \bmod a_1 (w=a_j)$ 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 $ka_1+r(0 \leq r < a_1)$ 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

[洛谷p2371](#)

```

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;
}
```

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=1630674873

Last update: 2021/09/03 21:14