网络流__7 最大流的判定

> 对于n个点m条边的无向图，求最小的最大长度L，使得图中可以找出k条从S到T的没有重复路径，其中L为所有路径中的最大值，请你最小化这个最大值

- 思路

本题考虑二分出最大边数限制，然后使用最大流。一般来说，最大流的题目都会有一定的限制条件，比如次数限制，数值限制等。还可能会出现改变边权，或者从起点到终点多次的要求。

- 做法

根据源点和汇点建立流网络，对无向图的每一条边建立正向和反向两种边，容量都为 1 ，其中由于一开始每条边的流量不确定（待二分），因此只记录当前边的边长；

二分出最长边，根据将大于最长边的所有边删除后求S到T的最大流是否大于 k 来判断此时的二分答案是否合法。（S到T的最大流即是S到T无重边的路经数）

- 疑惑点

对于无向图在建边的时候，可以正向反向都建，而且分别对于其残留网络可以不用再多建了。就相当于，假如两条边都有流量的时候，即正向边与反向边都不经过即可。

```
const int N = 210 , M = (N + 40010) * 2;
const int INF = 1e8;
int n,m,K,S,T;
int e[M],ne[M],h[N],f[M],idx=0;
int cur[M],q[M],d[N],w[M];

void add(int a,int b,int c){
    e[idx] = b , w[idx] = c , ne[idx] = h[a] , h[a] = idx ++;
    e[idx] = a , w[idx] = c , ne[idx] = h[b] , h[b] = idx ++;
}

int find(int u,int limit){
    if(u == T) return limit;
    int flow = 0;

    for(int i=cur[u];~i && flow < limit; i=ne[i]){
        cur[u] = i;
        int ver = e[i];

        if(d[ver]  == d[u] + 1 && f[i]){
            int t = find(ver,min(f[i],limit - flow));
            if(!t) d[ver] = -1;
            f[i] -= t , f[i^1] += t , flow += t;
        }
    }
    return flow;
}

bool bfs(){
    memset(d,-1,sizeof d);
    int hh = 0 , tt = 0;
    q[0] = S , cur[S] = h[S] , d[S] = 0;
```

```
    while(hh <= tt){
        int u = q[hh ++];
        for(int i=h[u];~i;i=ne[i]){
            int ver = e[i];
            if(d[ver] == -1 && f[i]){
                d[ver] = d[u] + 1;
                cur[ver] = h[ver];
                if(ver == T) return true;
                q[++ tt] = ver;
            }
        }
    }
    return false;
}

int dinic(){
    int ans = 0 , flow = 0;
    while(bfs()) while(flow = find(S,INF)) ans += flow;
    return ans;
}

bool check(int mid){
    for(int i=0;i<idx;i++){
        if(w[i] > mid) f[i] = 0;
        else f[i] = 1;
    }
    return dinic() >= K;
}
//================================
int main(){
    memset(h,-1,sizeof h);

    n = read() , m = read() , K = read();
    S = 1 , T = n;
    rep(i,1,m){
        int a = read() , b = read() , c = read();
        add(a,b,c);
    }
    int l = 1 , r = 1e6;
    while(l < r){
        int mid = l + r >> 1;
        if(check(mid)) r = mid;
        else l = mid + 1;
    }
    print(l);
    return 0;
}
```

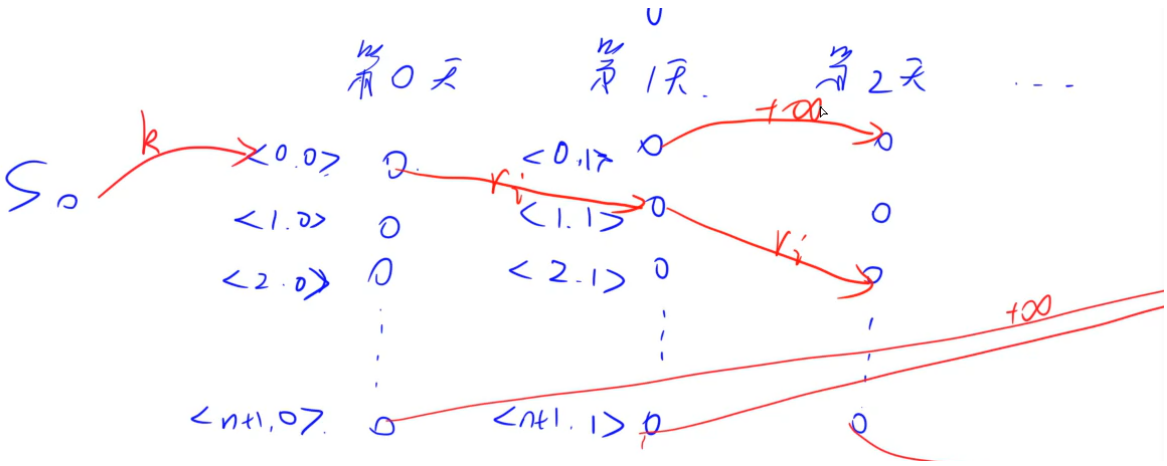
星际转移问题

> 从地球到月球由n个太空站，有m个飞船在做某些太空站中的循环飞行，每艘飞船都有其容量。问地球上有k个人时最短需要多长时间能把人全部从地球运往月球

- 思路

由于要求天数的最小值，此外本题的属性还包括了每一座空间站，因此我们借助分层图这一技巧来进行建图，分层图，即将流量与距离产生联系。建图如下：

流网络：`<i，j>`表示第i个工作站第j天所代表的点



图上有四种边：

[1]从源点S向`<0,0>`连一条容量为k的边，表示有k个人

[2]对于每一个点`<i,j>`向`<i,j+1>`连一条容量为正无穷的边表示每个人都可以在当前工作站停留一天

[3]对于第j列到第j+1列中，根据所有公交车对用站中的转换连接一条容量为公交车容量的边，表示可以通过公交车转换到下一站

[4]从所有天的第n+1个点向汇点T连一条容量为正无穷的边，表示已经到达了月球

`day`可以用二分来枚举，但是由于`day`随着人数增多而增多而且删除比增加麻烦，因此直接枚举即可。

```
const int N = 1201 * 25 + 10, M = 2*((N + 1100 + 18 * 1101) + 10), INF = 1e8;
int n,m,S,T,k,p;
int e[M],ne[M],h[N],f[M],idx=0;
int q[N],cur[N],d[N],fa[N];
struct Ship{
    int h,r,s[35];
}ship[35];
void init() {rep(i,0,n+1) fa[i] = i;}
int find(int x){ return x == fa[x] ? fa[x] : fa[x] = find(fa[x]);}
int get(int i,int d) {return (n+2)*d+i;}
void add(int a,int b,int c){
    e[idx] = b, f[idx] = c, ne[idx] = h[a], h[a] = idx ++;
    e[idx] = a, f[idx] = 0, ne[idx] = h[b], h[b] = idx ++;
}

int find(int u,int limit){
    if(u == T) return limit;
    int flow = 0;

    for(int i=cur[u];~i && flow < limit;i=ne[i]){
        cur[u] = i;
        int ver = e[i];
        if(d[ver] == d[u] + 1 && f[i]){
            int t = find(ver,min(f[i],limit - flow));
            if(!t) d[ver] = -1;
            f[i] -= t , f[i^1] += t ,flow += t;
        }
    }
```

```
    return flow;
}

bool bfs(){
    memset(d,-1,sizeof d);
    int hh = 0 , tt = 0;
    q[0] = S , d[S] = 0, cur[S] = h[S];

    while(hh <= tt){
        int u = q[hh ++];
        for(int i=h[u];~i;i=ne[i]){
            int ver= e[i];
            if(d[ver] == -1 && f[i]){
                d[ver] = d[u] + 1;
                cur[ver] = h[ver];
                if(ver == T) return true;
                q[++ tt] = ver;
            }
        }
    }
    return false;
}

int dinic(){
    int ans = 0, flow = 0;
    while(bfs()) while(flow = find(S,INF)) ans += flow;
    return ans;
}
//================================
int main(){
    memset(h,-1,sizeof h);
    n = read(), m = read(), k = read();
    S = N - 2 , T = N - 3;

    init();
    rep(i,0,m-1){
        ship[i].h = read(), ship[i].r = read();
        rep(j,0,ship[i].r-1){
            ship[i].s[j] = read();
            if(ship[i].s[j] == -1) ship[i].s[j] = n + 1;
            if(j){
                int pa = find(ship[i].s[j-1]) , pb = find(ship[i].s[j]);
                if(pa!=pb) fa[pa] = pb;
            }
        }
    }
    if(find(0) != find(n+1) ) { puts("0"); exit(0); }

    add(S,get(0,0),k); add(get(n+1,0),T,INF);
    int day = 1, res = 0;
    while(true){
        rep(i,0,n+1) add(get(i,day-1),get(i,day),INF); //[2]
        add(get(n+1,day),T,INF);//[3]
        rep(j,0,m-1){
            int r = ship[j].r;
            add(get(ship[j].s[(day-1)%r],day-
1),get(ship[j].s[day%r],day),ship[j].h);
        }
```

```
        res += dinic();  //当前流网络已经能够运送[最大流]人数
        if(res>=k) break;
        day++;
    }
    printf("%d\n",day);
    return 0;
}
```