图论

```
/* Dijkstra */
class DJ{
  public: int u, v;
    bool operator < (const DJ &o) const{
        return v > o.v;
    }
} ;
DJ mp(int a, int b){
    DJ o; o.u = a, o.v = b;
    return o;
}
int d[CN]; bool vis[CN]; priority_queue<DJ> Q;
void SP(int s){
    memset(d, 0x3f, sizeof(d)), memset(vis, 0, sizeof(vis));
    Q.push(mp(s, d[s] = 0));
    while(!Q.empty()){
        int u = Q.top().u; Q.pop();
        if(vis[u]) continue; vis[u] = 1;
        for(int l = G[u].size(), i = 0; i < l; i++){
            int v = G[u][i], c = V[u][i]; if(vis[v]) continue;
            if(d[v] > d[u] + c){
                d[v] = d[u] + c, Q.push(mp(v, d[v]));
            }
        }
    }
}

/* Tarjan SCC */
int n, k, from[CN], to[CN], te, col[CN], tc, sz[CN]; bool equ[CN];
vector<int> G[CN]; vector<bool> typ[CN];
class TJ{
  public:
    int dfn[CN], low[CN], id, stk[CN], top;
    bool ins[CN];
    void dfs(int u){
        dfn[u] = low[u] = ++id, stk[++top] = u; ins[u] = 1;
        for(int i = 0; i < G[u].size(); i++){
            int v = G[u][i];
            if(!dfn[v]){
                dfs(v); low[u] = min(low[u], low[v]);
            }
            else if(ins[v]) low[u] = min(low[u], low[v]);
        }
        if(dfn[u] == low[u]){
            tc++;
            while(1){
                int p = stk[top--];
                ins[p] = 0, col[p] = tc, sz[tc]++;
                if(p == u) break;
            }
        }
    }
    void work(){
        memset(dfn, 0, sizeof(dfn)), memset(ins, 0, sizeof(ins));
        id = top = tc = 0;
        for(int i = 1; i <= n; i++) if(!dfn[i]) dfs(i);
    }
} T;

/* DINIC 最大流 */
namespace DINIC {
    class fs{
      public: int to, nxt, cap, fl;
        void init(int t, int n, int c, int f) {to = t, nxt = n, cap = c, fl =
f;}
    } E[CN * 10]; int hd[CN], cur[CN], ecnt;
    void clr() {memset(hd, 0, sizeof(hd)), ecnt = 1;}
    void adde(int x, int y, int z) {
        E[++ecnt].init(y, hd[x], z, 0), hd[x] = ecnt;
        E[++ecnt].init(x, hd[y], 0, 0), hd[y] = ecnt;
    }
    int dep[CN]; queue<int> Q;
    bool bd(int s, int t){
        memset(dep, 0, sizeof(dep)), dep[s] = 1, Q.push(s);
        while(!Q.empty()){
            int u = Q.front(); Q.pop();
            for(int k = hd[u]; k; k = E[k].nxt){
                fs e = E[k];
                if(e.cap > e.fl && !dep[e.to]) dep[e.to] = dep[u] + 1,
Q.push(e.to);
            }
        }
        return dep[t];
    }
    int aug(int u, int t, int rst){
        if(u == t) return rst;
        int usd = 0;
        for(int k = cur[u]; k; k = E[k].nxt){
            fs e = E[k]; cur[u] = k;
            if(e.cap > e.fl && dep[e.to] == dep[u] + 1){
                int add = aug(e.to, t, min(rst - usd, e.cap - e.fl));
                if(add){
                    E[k].fl += add, E[k ^ 1].fl -= add, usd += add;
                    if(usd == rst) return usd;
                }
            }
        }
        return usd;
    }
    int mf(int s, int t){
        int res = 0;
        while(bd(s, t)) memcpy(cur, hd, sizeof(hd)), res += aug(s, t, INF);
        return res;
    }
}
```

```cpp
/* ZKW费用流 */
namespace ZKW{
    class fs {
      public: int to, nxt, cap, fl, w;
        void init(int t, int n, int c, int f, int ww) {
            to = t, nxt = n, cap = c, fl = f, w = ww;
        }
    } E[CN << 2];
    int hd[CN], ecnt = 1;
    void adde(int x, int y, int z, int c){
        E[++ecnt].init(y, hd[x], z, 0, c), hd[x] = ecnt;
        E[++ecnt].init(x, hd[y], 0, 0, -c), hd[y] = ecnt;
    }
    int cst, mf, d[CN]; bool vis[CN];
    bool bd(int t){
        if(vis[t]) return 1;
        int del = INF;
        for(int i = 2; i <= ecnt; i++){
            int u = E[i ^ 1].to, v = E[i].to;
            if(vis[u] && !vis[v] && E[i].cap > E[i].fl)
                del = min(del, d[u] + E[i].w - d[v]);
        }
        if(del == INF) return 0;
        for(int i = 1; i <= n; i++) if(vis[i]) d[i] -= del;
        return 1;
    }
    int aug(int u, int t, int rst){
        vis[u] = 1;
        if(u == t) return rst;
        for(int k = hd[u]; k; k = E[k].nxt){
            fs e = E[k];
            if(e.cap > e.fl && !vis[e.to] && d[u] + e.w == d[e.to]){
                int add = aug(e.to, t, min(rst, e.cap - e.fl));
                if(add) return E[k].fl += add, E[k ^ 1].fl -= add, add;
            }
        }
        return 0;
    }
    void mcmf(int s, int t){
        memset(d, 0, sizeof(d)), cst = mf = 0;
        do{
            memset(vis, 0, sizeof(vis));
            int add = aug(s, t, INF);
            mf += add, cst -= add * d[s];
        }
        while(bd(t));
        printf("%d %d\n", mf, cst);
    }
}
```