```
struct Edge {
    int from, to, cap, flow, cost;
};

struct MCMF {
    int n, m, s, t;
    vector<Edge> edges;
    vector<int> G[maxn];
    int inq[maxn];
    int d[maxn];
    int p[maxn];
    int a[maxn];

    void init(int n) {
        this->n = n;
        for(int i = 0; i <= n ;i++) G[i].clear();
        edges.clear();
    }

    void AddEdge(int from, int to, int cap, int cost) {
        edges.pb((Edge){from, to, cap, 0, cost});
        edges.pb((Edge){to, from, 0, 0, -cost});
        m = edges.size();
        G[from].pb(m-2);
        G[to].pb(m-1);
    }

    bool SPFA(int s, int t, int &flow, int& cost) {
        this->s = s, this->t = t;
        for(int i = 0; i <= n; i++) d[i] = INF;
        memset(inq, 0, sizeof(inq));
        d[s] = 0; inq[s] = 1; p[s] = 0; a[s] = INF;

        queue<int> Q;
        Q.push(s);
        while(!Q.empty()) {
            int u = Q.front(); Q.pop();
            inq[u] = 0;
            for(int i = 0; i < G[u].size(); i++) {
                Edge& e = edges[G[u][i]];
                if(e.cap > e.flow && d[e.to] > d[u]+ e.cost) {
                    d[e.to] = d[u]+e.cost;
                    p[e.to] = G[u][i];
                    a[e.to] = min(a[u], e.cap-e.flow);
                    if(!inq[e.to]) { Q.ps(e.to); inq[e.to] = 1; }
                }
```

```
            }
        }
        if(d[t] == INF) return false;
        flow += a[t];
        cost += d[t] * a[t];
        int u = t;
        while(u != s) {
            edges[p[u]].flow += a[t];
            edges[p[u]^1].flow -= a[t];
            u = edges[p[u]].from;
        }
        return true;
    }

    int MinCost(int s, int t) {
        this->s = s, this->t = t;
        int flow = 0, cost = 0;
        while(SPFA(s, t, flow, cost));
        return cost;
    }

};
```