```cpp
#include "../bits/stdc++.h"
// 最小費用流
// O(FElogV)
// verified: http://judge.u-aizu.ac.jp/onlinejudge/review.jsp?rid=3382347
class minCostFlow
{
    using type = int;
    using pii = std::pair<int, int>;
    const int INF = 1e9;
    struct Edge
    {
        type to, cap, cost, rev;
        Edge(type to_, type cap_, type cost_, type rev_)
            : to(to_), cap(cap_), cost(cost_), rev(rev_) {}
    };
    int V;
    std::vector<std::vector<Edge>> G;
    // ポテンシャル
    std::vector<int> h;
    // 最短距離
    std::vector<int> dist;
    // 直前の頂点, 辺
    std::vector<int> prevv, preve;

  public:
    minCostFlow(int _V) : V(_V), G(_V), h(_V), dist(_V), prevv(_V), preve(_V) {}
    void add(int from, int to, int cap, int cost)
    {
        G[from].push_back(Edge(to, cap, cost, G[to].size()));
        G[to].push_back(Edge(from, 0, -cost, G[from].size() - 1));
    }
    int calc(int s, int t, int f)
    {
        int res = 0;
        fill(h.begin(), h.end(), 0);
        while (f > 0)
        {
            std::priority_queue<pii, std::vector<pii>, std::greater<pii>> que;
            fill(dist.begin(), dist.end(), INF);
            dist[s] = 0;
            que.push(pii(0, s));
            while (!que.empty())
            {
                pii p = que.top();
                que.pop();
                int v = p.second;
                if (dist[v] < p.first)
                    continue;
                for (size_t i = 0; i < G[v].size(); i++)
                {
                    Edge &e = G[v][i];
                    if (e.cap > 0 && dist[e.to] > dist[v] + e.cost + h[v] - h[e.to])
                    {
                        dist[e.to] = dist[v] + e.cost + h[v] - h[e.to];
                        prevv[e.to] = v;
                        preve[e.to] = i;
                        que.push(pii(dist[e.to], e.to));
                    }
                }
            }
            if (dist[t] == INF)
                return -1;
            for (int v = 0; v < V; v++)
                h[v] += dist[v];
            int d = f;
            for (int v = t; v != s; v = prevv[v])
            {
                d = std::min(d, G[prevv[v]][preve[v]].cap);
            }
            f -= d;
            res += d * h[t];
            for (int v = t; v != s; v = prevv[v])
            {
                Edge &e = G[prevv[v]][preve[v]];
                e.cap -= d;
                G[v][e.rev].cap += d;
            }
        }
        return res;
    }
};
```