

类 Dinic 算法

我们可以在 Dinic 算法的基础上进行改进，把 BFS 求分层图改为用 SPFA（由于有负权边，所以不能直接用 Dijkstra 来求一条单位费用之和最小的路径，也就是把 $w(u,v)$ 当做边权然后在残量网络上求最短路，当然在 DFS 中也要略作修改。这样就可以求得网络流图的最小费用最大流了。

如何建反向边？对于一条边 (u,v,w,c) （其中 w 和 c 分别为容量和费用），我们建立正向边 (u,v,w,c) 和反向边 $(v,u,0,-c)$ （其中 $-c$ 是使得从反向边经过时退回原来的费用）。

优化：如果你是“关于 SPFA 它死了”言论的追随者，那么你可以使用 Primal-Dual 原始对偶算法将 SPFA 改成 Dijkstra

时间复杂度：可以证明上界为 $O(nmf)$ 其中 f 表示流量。

参考代码：

```
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <queue>

const int N = 5e3 + 5, M = 1e5 + 5;
const int INF = 0x3f3f3f3f;
int n, m, tot = 1, lnk[N], cur[N], ter[M], nxt[M], cap[M], cost[M], dis[N],
ret;
bool vis[N];

void add(int u, int v, int w, int c) {
    ter[++tot] = v, nxt[tot] = lnk[u], lnk[u] = tot, cap[tot] = w, cost[tot]
= c;
}
void addedge(int u, int v, int w, int c) { add(u, v, w, c), add(v, u, 0, -
c); }
bool spfa(int s, int t) {
    memset(dis, 0x3f, sizeof(dis));
    memcpy(cur, lnk, sizeof(lnk));
    std::queue<int> q;
    q.push(s), dis[s] = 0, vis[s] = 1;
    while (!q.empty()) {
        int u = q.front();
        q.pop(), vis[u] = 0;
        for (int i = lnk[u]; i; i = nxt[i]) {
            int v = ter[i];
            if (cap[i] && dis[v] > dis[u] + cost[i]) {
                dis[v] = dis[u] + cost[i];
                if (!vis[v]) q.push(v), vis[v] = 1;
            }
        }
    }
}
```

```
    return dis[t] != INF;
}
int dfs(int u, int t, int flow) {
    if (u == t) return flow;
    vis[u] = 1;
    int ans = 0;
    for (int &i = cur[u]; i && ans < flow; i = nxt[i]) {
        int v = ter[i];
        if (!vis[v] && cap[i] && dis[v] == dis[u] + cost[i]) {
            int x = dfs(v, t, std::min(cap[i], flow - ans));
            if (x) ret += x * cost[i], cap[i] -= x, cap[i ^ 1] += x, ans += x;
        }
    }
    vis[u] = 0;
    return ans;
}
int mcmf(int s, int t) {
    int ans = 0;
    while (spfa(s, t)) {
        int x;
        while ((x = dfs(s, t, INF))) ans += x;
    }
    return ans;
}
int main() {
    int s, t;
    scanf("%d%d%d", &n, &m, &s, &t);
    while (m--) {
        int u, v, w, c;
        scanf("%d%d%d", &u, &v, &w, &c);
        addedge(u, v, w, c);
    }
    int ans = mcmf(s, t);
    printf("%d %d\n", ans, ret);
    return 0;
}
```

习题

[\[Luogu 3381\]\[模板\] 最小费用最大流](#) [\[Luogu 4452\]航班安排](#) [\[SDOI 2009\]晨跑](#) [\[SCOI 2007\]修车](#)
[\[HAOI 2010\]订货](#) [\[NOI 2012\]美食节](#)

参考链接

[OI Wiki](#)

From:

<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:

https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:lgwza:%E7%B1%BB_dinic_%E7%AE%97%E6%B3%95&rev=1597367449 

Last update: **2020/08/14 09:10**