Dinic

Back

Description

Dinic 算法是在网络流计算最大流的强多项式复杂度的算法。

类似于复杂度为 O(|V||E|^2)的 Edmonds–Karp 算法，Dinic 算法的复杂度为 O(|V|^2|E|)

但是在大多数网络建模下，Dinic 的速度远优于其最坏复杂度。

请尝试让 Dinic 达到其理论最坏复杂度 (使代码中的计数器超过 1e6)。

Code

```
// Dinic's algorithm

#include <assert.h>
#include <iostream>

#include <list>
#include <queue>
#include <vector>

using namespace std;

const int MAXN = 100;
const int MAXM = 5000;
const int MAXW = 1e9;

struct edge {
    int to, cap;
    edge *rev;
};
list<edge> E[MAXN + 1];
long long ops = 0;

int dfs(int u, int t, int limit, vector<int> &dis) {
    if (u == t || !limit)
        return limit;
    int res = limit;
    for (edge &e : E[u]) {
        int v = e.to, w = e.cap;
        ops++;
        if (w && dis[v] == dis[u] + 1) {
            int flow = dfs(v, t, min(res, w), dis);
            e.cap -= flow;
```

```cpp
                e.rev->cap += flow;
                res -= flow;
            }
        }
        if (res == limit)
            dis[u] = MAXW + 1;
        return limit - res;
    }

    void dinic(int n, int m, int s, int t) {
        int flow = 0;
        while (true) {
            vector<int> dis(n + 1, MAXW + 1);
            queue<int> q;

            dis[s] = 0;
            q.push(s);
            while (!q.empty()) {
                int u = q.front();
                q.pop();
                for (edge e : E[u]) {
                    int v = e.to, w = e.cap;
                    ops++;
                    if (w && dis[v] > dis[u] + 1) {
                        dis[v] = dis[u] + 1;
                        q.push(v);
                    }
                }
            }
            if (dis[t] == MAXW + 1)
                break;
            flow += dfs(s, t, MAXW + 1, dis);
        }

        cout << flow << endl;
    }

    int main(int argc, char *argv[]) {
        int n, m, s, t;
        cin >> n >> m >> s >> t;
        assert(1 <= n && n <= MAXN);
        assert(1 <= m && m <= MAXM);
        assert(1 <= s && s <= n);
        assert(1 <= t && t <= n);

        long long sum = 0;
        for (int i = 0; i < m; i++) {
            int u, v, c;
            cin >> u >> v >> c;
            sum += c;
            assert(1 <= u && u <= n);
            assert(1 <= v && v <= n);
            assert(1 <= c && c <= MAXW);
        assert(1 <= sum && sum <= MAXW);
        assert(u != v);

        edge e1 = {v, c, nullptr};
        edge e2 = {u, 0, nullptr};
        E[u].push_back(e1);
        E[v].push_back(e2);
        E[u].back().rev = &E[v].back();
        E[v].back().rev = &E[u].back();
    }

    dinic(n, m, s, t);

    cerr << ops << endl;
}
```

Hack it!

Choose File No file chosen

submit