SPFA

Back

Description

最短路径快速算法 (Shortest Path Faster Algorithm, SPFA)，一般也被称为带有队列优化的 Bellman-Ford 算法。

相较于 Bellman-Ford 算法，SPFA 的最坏复杂度和其一致为O(|V||E|)

但是在实际使用中，在很多情况下，SPFA 的速度远优于其最坏复杂度。

请尝试让 SPFA 达到其理论最坏复杂度 (使代码中的计数器超过 2e6)。

Code

```cpp
// SPFA algorithm

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

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

using namespace std;

const int MAXN = 2000;
const int MAXM = 8000;
const int MAXW = 1e9;

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

void spfa(int n, int m, int start, int end) {
    vector<int> dist(n + 1, MAXW + 1);
    vector<bool> vis(n + 1, false);
    queue<int> q;

    dist[start] = 0;
    vis[start] = true;
    q.push(start);
    while (!q.empty()) {
```

```
            int u = q.front();
            q.pop();
            vis[u] = false;
            for (edge e : E[u]) {
                int v = e.to, w = e.cost;
                ops++;
                if (dist[v] > dist[u] + w) {
                    dist[v] = dist[u] + w;
                    if (!vis[v]) {
                        vis[v] = true;
                        q.push(v);
                    }
                }
            }
        }
        cout << dist[end] << 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);

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

            E[u].push_back({v, w});
            E[v].push_back({u, w});
        }

        spfa(n, m, s, t);

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

Hack it!

Choose File No file chosen submit