Университет ИТМО
Кафедра ВТ

Задачи 4 (1 - 2, 4)
Алгоритмы и Структуры Данных

Выполнил: Федоров Сергей
Группа: P3212

Санкт-Петербург
2020 г.

- # Задача 1 - Раскраска Карты - 1080

Представим страны как граф. Вершины - страны, ребра - границы. Наша задача покрасить вершины граф. Так как у нас два цвета, то граф должен будет получится двудольным (или нет). Поэтому красим граф обход в ширину.

O(|V| + |E|)

Код:

```
//
// Created by Sergey Fedorov on 28/05/2020.
//

#include <iostream>
#include <vector>
#include <queue>

using namespace std;

const int START_COLOR = 0;

bool btf(int root, vector<vector<int>> &edge, vector<int> &color) {
    queue<int> to_visit;

    to_visit.push(root);
    color[root] = START_COLOR;

    bool finished = false;
    bool success = true;

    while (!to_visit.empty() && !finished) {
        int from = to_visit.front(); to_visit.pop();

        for (int i = 0; i < edge[from].size(); i++) {
            int to = edge[from][i];

            if (color[from] == color[to]) {
                success = false;
                finished = true;
            } else if (color[to] == -1) {
                color[to] = !color[from];
                to_visit.push(to);
            }
        }
    }

    return success;
}

int main(){
    int n; cin >> n;
    vector<int> colors(n, -1);
    vector<vector<int>> borders(n);

    for (int a = 0; a < n; a++) {
        int b;
        cin >> b;
        while (b != 0) {
                b--;
                borders[a].push_back(b);
```

```
                borders[b].push_back(a);
                cin >> b;
            }
        }

        bool success = true;
        for (int i = 0; i < n; i++) if (colors[i] == -1) {
            bool result = btf(i, borders, colors);
            success &= result;
            if (!result) break;
        }

        if (success) {
            for (int i = 0; i < n; i++) cout << colors[i];
        } else {
            cout << "-1";
        }
}
```

- **Задача 2 - Российские Газопроводы - 1450**

Здесь нам надо найти наибольший путь. Это обратно поиску наименьшего пути. У нас есть пару алгоритмов на выбор, но алгоритм Форда-Беллмана показался самым легким, поэтому возьмем его.

Есть пару способов находить наибольший путь вместо наименьшего. В данном случае делаем веса отрицательными (благо Алг Ф-Б умеет с ними работать).

(плюс есть вот такой вот пост, весьма оправдывающий мой выбор https://stackoverflow.com/questions/3447566/dijkstras-algorithm-in-c)

`O(|V| * |E|)`

Код:

```
//
// Created by Sergey Fedorov on 28/05/2020.
//

#include <iostream>
#include <vector>

using namespace std;

#define INF (INT_MAX/2)

struct pipe {
    int a, b, meta_dist;
};

int main() {
    int n, m;
    cin >> n >> m;

    vector<pipe> pipes(m);

    for (int i = 0; i < m; ++i) {
        int a, b, p;
        cin >> a >> b >> p;
        pipes[i] = {a - 1, b - 1,  -p};
    }
```

```
    int start, finish;
    cin >> start >> finish;
    start--;
    finish--;

    vector<int> sum_dist(n, INF);
    sum_dist[start] = 0;

    // Bellman-Ford algorithm (without check for cycles)
    for (int i = 0; i < n - 1; ++i) for (int j = 0; j < m; ++j) {
        if (sum_dist[pipes[j].a] != INF &&
            sum_dist[pipes[j].b] > sum_dist[pipes[j].a] + pipes[j].meta_dist) {
            sum_dist[pipes[j].b] = sum_dist[pipes[j].a] + pipes[j].meta_dist;
        }
    }

    if (sum_dist[finish] != INF) cout << -sum_dist[finish]; else cout << "No solution";
}
```

- ## Задача 3 - Network - 1160

TODO: Там надо будет построить MST.

Код:

```
TODO
```

- ## Задача 4 - Currency Exchange - 1162

Задача практически аналогична задаче 1160, однако тут по другому считаем расстояние, а так же в конце выбираем не какую-то конкретную вершину, а ищем хотя бы одну в которой получается прирост по финансам.

(
For example, if you want to exchange 100 US Dollars into Russian Rubles at the exchange point, where the exchange rate is 29.75, and the commission is 0.39 you will get (100 - 0.39) * 29.75 = 2963.3975RUR.
) ☹️

```
O(|V| * |E|)
```

Код:

```
//
// Created by Sergey Fedorov on 30/05/2020.
//

#include <iostream>
#include <vector>

#define EPSILON 1e-5

using namespace std;

struct exchange_point {
```

```
    int cur_a, cur_b;
    double rab, cab;
};

int main(){
    int n, m, s;
    double v;

    cin >> n >> m >> s >> v;

    vector<exchange_point> ex_ps(2 * m);
    vector<double> currencies(n + 1, 0);

    for (int i = 0; i < 2 * m; i += 2) {
        int a, b;
        double rab, cab, rba, cba;
        cin >> a >> b >> rab >> cab >> rba >> cba;

        ex_ps[i    ] = {a, b, rab, cab};
        ex_ps[i + 1] = {b, a, rba, cba};
    }

    currencies[s] = v;

    // Need to twist the algo, to find the biggest path
    // This time just negation didn't work :(
    for (int i = 0; i < n - 1; i++) for (auto &ex_p : ex_ps)
        currencies[ex_p.cur_b] = max(currencies[ex_p.cur_b], (currencies[ex_p.cur_a] -
ex_p.cab) * ex_p.rab);

    bool wealth_growth = false;
    for (auto &ex_p : ex_ps)
        wealth_growth |= (currencies[ex_p.cur_a] - ex_p.cab) * ex_p.rab >
currencies[ex_p.cur_b] + EPSILON;

    if (wealth_growth) cout << "YES"; else cout << "NO";
}
```

- **Задача 5 - Мобильные телеграфы - 1806**