Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №4

по «Алгоритмам и структурам данных»

Timus

Выполнил:

Студент группы P3233

Перевозчиков И. С.

Преподаватели:

Косяков М. С.

Тараканов Д. С.

Санкт-Петербург

2022

Задача 1080. Раскраска карты

```
#include <iostream>
#include <vector>
#include <map>
#include <set>
#include <string>

using namespace std;

enum {
	RED,
	BLUE,
	COLORLESS
};

class Country {
public:
	int color = COLORLESS;
	vector <int> neighbors;
};

void dfs(int curr, int prev, vector <Country>& countries) {
	if (countries[prev].color == BLUE) {
		countries[curr].color = RED;
	}
	else {
		countries[curr].color = BLUE;
	}

	for (int i = 0; i < countries[curr].neighbors.size(); i++) {
		if (countries[countries[curr].neighbors[i]].color == COLORLESS) {
			dfs(countries[curr].neighbors[i], curr, countries);
		}
	}
}

string isPossible(vector <Country>& countries) {
	string ans = "";
	for (int i = 0; i < countries.size(); i++) {
		ans += to_string(countries[i].color);
		for (int j = 0; j < countries[i].neighbors.size(); j++) {
			if (countries[i].color ==
countries[countries[i].neighbors[j]].color) {
				return "-1";
			}
		}
	}
	return ans;
}


int main()
{
	int n;
	cin >> n;
	vector <Country> countries(n);
	for (int i = 0; i < n; i++) {
		int a;
		cin >> a;
		while (a != 0) {
			a--;
			countries[i].neighbors.push_back(a);
			countries[a].neighbors.push_back(i);
			cin >> a;
```

```
            }
        }

        countries[0].color = BLUE;
        dfs(0, 0, countries);
        cout << isPossible(countries) << '\n';
}
```

С помощью DFS раскрашивается карта. Далее, если граф двудольный, то карту можно покрасить. Иначе нельзя.

Алгоритмическая сложность: $O(n+m), n -$ количество вершин, $m -$ количество ребер.

Задача 1162. Currency exchange

```
#include<iostream>
#include<algorithm>
#include<vector>

using namespace std;


class Edge {
public:
	Edge(int a, int b, double rab, double cab) {
		this->a = a;
		this->b = b;
		this->rab = rab;
		this->cab = cab;
	}
	Edge() {

	}
	int a;
	int b;
	double rab;
	double cab;
};


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

	vector <Edge> edges(2 * m);
	vector <double> dist(n, INT32_MIN);

	dist[--s] = v;

	for (int i = 0; i < m; i++) {
		int a, b;
		double rab, cab, rba, cba;
		cin >> a >> b >> rab >> cab >> rba >> cba;
		a--;
		b--;
		edges[2 * i] = Edge(a, b, rab, cab);
		edges[2 * i + 1] = Edge(b, a, rba, cba);
	}

	for (int i = 0; i < n - 1; i++) {
		for (int j = 0; j < edges.size(); j++) {
			if (dist[edges[j].a] != INT32_MIN) {
				double newDist = (dist[edges[j].a] - edges[j].cab) *
edges[j].rab;
				if (newDist > 0 && dist[edges[j].b] < newDist) {
					dist[edges[j].b] = newDist;
				}
			}
		}
	}

	bool isIncreasable = false;

	for (int j = 0; j < edges.size(); j++) {
		if (dist[edges[j].a] != INT32_MIN) {
```

```
                        double newDist = (dist[edges[j].a] - edges[j].cab) *
edges[j].rab;
                        if (newDist > 0 && dist[edges[j].b] < newDist) {
                                dist[edges[j].b] = newDist;
                                isIncreasable = true;
                        }
                }
        }

        if (isIncreasable) {
                cout << "YES";
        }
        else {
                cout << "NO";
        }

        return 0;
}
```

Алгоритм Форда-Беллмана. Только ищутся не отрицательные циклы, а положительные. Если есть положительный цикл, то можно увеличить состояние, иначе нельзя.

Алгоритмическая сложность: $O(n * m)$, где $n -$ количество вершин, $m -$ количество ребер.