Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

Лабораторная работа № 6
"Задачи 1160, 1162, 1080"

по дисциплине Алгоритмы и структуры данных

Выполнила: студентка гр. **R3238**
поток **2.1**

Нечаева А. А.

Преподаватель: *Тропченко Андрей Александрович*

Санкт-Петербург, 2024

1 Цель

Разработать и реализовать алгоритмы для решения задач 1160, 1162 и 1080.

2 Задача 1160

1160. Network

Ограничение времени: 1.0 секунды
Ограничение памяти: 64 МБ

Andrew is working as system administrator and is planning to establish a new network in his company. There will be N hubs in the company, they can be connected to each other using cables. Since each worker of the company must have access to the whole network, each hub must be accessible by cables from any other hub (with possibly some intermediate hubs).

Since cables of different types are available and shorter ones are cheaper, it is necessary to make such a plan of hub connection, that the maximum length of a single cable is minimal. There is another problem - not each hub can be connected to any other one because of compatibility problems and building geometry limitations. Of course, Andrew will provide you all necessary information about possible hub connections.

You are to help Andrew to find the way to connect hubs so that all above conditions are satisfied.

Исходные данные

The first line contains two integer: N - the number of hubs in the network ($2 \le N \le 1000$) and M — the number of possible hub connections ($1 \le M \le 15000$). All hubs are numbered from 1 to N. The following M lines contain information about possible connections - the numbers of two hubs, which can be connected and the cable length required to connect them. Length is a positive integer number that does not exceed 10^6. There will be no more than one way to connect two hubs. A hub cannot be connected to itself. There will always be at least one way to connect all hubs.

Результат

Output first the maximum length of a single cable in your hub connection plan (the value you should minimize). Then output your plan: first output P - the number of cables used, then output P pairs of integer numbers - numbers of hubs connected by the corresponding cable. Separate numbers by spaces and/or line breaks.

Пример

исходные данные	результат
4 6 1 2 1 1 3 1 1 4 2 2 3 1 3 4 1 2 4 1	1 4 1 2 1 3 2 3 3 4

Автор задачи: Andrew Stankevich
Источник задачи: ACM ICPC 2001. Northeastern European Region, Northern Subregion

Рис. 1. Условие задачи 1160.

2.1 Краткое описание алгоритма

В основе реализации лежит алгоритм **Краскала** – алгоритм поиска минимального остовного дерева (англ. *minimum spanning tree, MST*) во взвешенном неориентированном связном графе.

Идея алгоритма Краскала: последовательное построение подграфа F графа G, стремясь на каждой итерации достроить F до MST. Включим в F все вершины G. Перейдем к обходу множества ребер графа G в порядке неубывания весов ребер. Если какое-то ребро соединяет вершины одной компоненты связности F, то оно не может быть включено в F, так как при его добавлении возникнет цикл. Иначе добавляем это ребро в F. На последней итерации ребро объединит две компоненты связности, полученный подграф будет минимальным остовным деревом графа G.

1. Входные данные: в первой строке содержится два целых числа: N − число хабов в сети ($2 \leq N \leq 1000$) и M − число возможных соединений хабов ($1 \leq M \leq 15000$). Все хабы имеют номера от 1 до N. Следующие M строк содержат информацию о возможных соединениях − номера двух хабов, которые могут быть соединены кабелем, и длину соответствующего кабеля.
2. Зададим структуру для соединения (начальный хаб, конечный и длина кабеля). При считывании данных будем записывать все соединения в структуру данных *std::vector*. Отсортируем полученный вектор по неубыванию.
3. Теперь добавляем соединения (ребра в граф), вершины обозначаем такими номерами, которые соответствуют номерам подграфов, не соединенных с другими подграфами.
4. Если мы объединяем два подграфа, то присваиваем объединению наименьший из исходных номеров компонент. В конце все вершины должны оказаться соединенными.
5. Выходные данные: вывести максимальное значение длины кабеля в полученном плане соединений (значение, которое нужно минимизировать). Далее вывести количество использованных кабелей и вывести пары, соединенных хабов.

2.2 Листинг

Листинг 1. Исходный код для 1160

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


// special structure to store information about connections
struct _connection {
    int v_1;
    int v_2;
    int l;
};

// special comparator to sort possible connections
bool _compare(_connection c_1, _connection c_2) {
    return c_1.l < c_2.l;
}



int main(){

    int N, M, v_1, v_2, l;
    std::cin >> N >> M;
    // vector of possible connections
    std::vector<_connection> _pos_con;

    for (int i = 0; i < M; ++i) {

        std::cin >> v_1 >> v_2 >> l;
        _pos_con.push_back((_connection){v_1 - 1, v_2 - 1, l})
            ;
    }
    // here starts the Kruskal's algorithm
    std::sort(_pos_con.begin(), _pos_con.end(), _compare);

    int _visited_v[N];
    int _visited_e[M];

    for (int i = 0; i < N; ++i) {
```

```
39          _visited_v[i] = -1;
40      }
41
42      for (int i = 0; i < M; ++i) {
43          _visited_e[i] = 0;
44      }
45
46      int _subgraph_number = 0;
47      int _max = 0;
48      int _counter = 0;
49
50      for (int i = 0; i < M; ++i) {
51          if (_visited_v[_pos_con[i].v_1] != -1 && _visited_v[
                _pos_con[i].v_1] == _visited_v[_pos_con[i].v_2])
                continue;
52
53          else if (_visited_v[_pos_con[i].v_1] == -1 &&
                _visited_v[_pos_con[i].v_2] == -1) {
54              _visited_v[_pos_con[i].v_1] = ++_subgraph_number;
55              _visited_v[_pos_con[i].v_2] = _subgraph_number;
56
57          } else if ((_visited_v[_pos_con[i].v_1] != -1 &&
                _visited_v[_pos_con[i].v_2] == -1) ||
58                      (_visited_v[_pos_con[i].v_1] == -1 &&
                            _visited_v[_pos_con[i].v_2] != -1)) {
59
60              int _loc_subgraph = _visited_v[_pos_con[i].v_1] +
                    _visited_v[_pos_con[i].v_2] + 1;
61              _visited_v[_pos_con[i].v_1] = _loc_subgraph;
62              _visited_v[_pos_con[i].v_2] = _loc_subgraph;
63          } else {
64
65              int min_subgraph;
66              int max_subgraph;
67
68              if (_visited_v[_pos_con[i].v_1] < _visited_v[
                    _pos_con[i].v_2]) {
69                  min_subgraph = _visited_v[_pos_con[i].v_1];
70                  max_subgraph = _visited_v[_pos_con[i].v_2];
71              } else {
72                  max_subgraph = _visited_v[_pos_con[i].v_1];
73                  min_subgraph = _visited_v[_pos_con[i].v_2];
74              }
```

```
75
76              for (int j = 0; j < N; ++j) {
77                  if (_visited_v[j] == max_subgraph) {
78                      _visited_v[j] = min_subgraph;
79                  }
80              }
81          }
82          _visited_e[i] = 1;
83          _max = _pos_con[i].l;
84          ++_counter;
85
86      }
87      std::cout << _max << std::endl << _counter << std::endl;
88
89      for (int j = 0; j < M; ++j) {
90          if (_visited_e[j]) {
91              std::cout << _pos_con[j].v_1 + 1 << " " <<
                    _pos_con[j].v_2 + 1 << std::endl;
92          }
93      }
94
95      return 0;
96
97  }
```

2.3 Результат

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10636982	04:44:46 6 май 2024	mistDragon	1160. Network	Clang++ 17 x64	Accepted		0.078	616 КБ

Рис. 2. Результат отправки задачи 1160.

3 Задача 1162

1162. Currency Exchange

Ограничение времени: 1.0 секунды
Ограничение памяти: 64 МБ

Several currency exchange points are working in our city. Let us suppose that each point specializes in two particular currencies and performs exchange operations only with these currencies. There can be several points specializing in the same pair of currencies. Each point has its own exchange rates, exchange rate of A to B is the quantity of B you get for 1A. Also each exchange point has some commission, the sum you have to pay for your exchange operation. Commission is always collected in source currency.

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.

You surely know that there are N different currencies you can deal with in our city. Let us assign unique integer number from 1 to N to each currency. Then each exchange point can be described with 6 numbers: integer A and B - numbers of currencies it exchanges, and real RAB, CAB, RBA and CBA - exchange rates and commissions when exchanging A to B and B to A respectively.

Nick has some money in currency S and wonders if he can somehow, after some exchange operations, increase his capital. Of course, he wants to have his money in currency S in the end. Help him to answer this difficult question. Nick must always have non-negative sum of money while making his operations.

Исходные данные

The first line contains four numbers: N - the number of currencies, M - the number of exchange points, S - the number of currency Nick has and V - the quantity of currency units he has. The following M lines contain 6 numbers each - the description of the corresponding exchange point - in specified above order. Numbers are separated by one or more spaces. $1 \le S \le N \le 100$, $1 \le M \le 100$, V is real number, $0 \le V \le 10^3$.

For each point exchange rates and commissions are real, given with at most two digits after the decimal point, $10^{-2} \le \text{rate} \le 10^2$, $0 \le \text{commission} \le 10^2$.

Let us call some sequence of the exchange operations simple if no exchange point is used more than once in this sequence. You may assume that ratio of the numeric values of the sums at the end and at the beginning of any simple sequence of the exchange operations will be less than 10^4.

Результат

If Nick can increase his wealth, output YES, in other case output NO.

Примеры

исходные данные	результат
3 2 1 10.0 1 2 1.0 1.0 1.0 1.0 2 3 1.1 1.0 1.1 1.0	NO
3 2 1 20.0 1 2 1.0 1.0 1.0 1.0 2 3 1.1 1.0 1.1 1.0	YES

Автор задачи: Nick Durov
Источник задачи: ACM ICPC 2001. Northeastern European Region, Northern Subregion

Рис. 3. Условие задачи 1162.

3.1 Краткое описание алгоритма

В основе решения лежит применение алгоритма **Форда-Беллмана** – алгоритм нахождения кратчайшего пути из заданной вершины s до всех остальных вершин взвешенного графа $G = (V, E)$. Если в графе G есть циклю с отрицательным суммарным весом, притом достижимые из s, тогда кратчайших путей не существует.

1. Входные данные: первая строка содержит 4 целых числа: N – количество валют, M – количество обменных пунктов, S – номер валюты, имеющейся у Ника, и V – количество единиц валюты, имеющейся у него.

Следующие M строк содержат по 6 чисел каждая – описание соответствующего пункта обмена (номера валют (a, b), которые можно обменять в этом пункте, обменные курсы (rate) и комиссии (commission): rab, cab, rba, cba). Числа разделены одним или несколькими пробелами. $1 \leq S \leq N \leq 100$, $1 \leq M \leq 100$, V – вещественное число, $0 \leq V \leq 10^3$.

Для каждого пункта обмена курсы и комиссии являются вещественными числами с не более чем двумя цифрами после запятой.

2. зададим специальную структуру, в которой будем хранить информацию о каждой возможной операции с двумя валютами: номера валют, курс и комиссия, сохраним считанные характеристики обменных пунктов в структуре данных *vector*;

3. на каждом шаге будем рассматривать все возможные пути из каждой посещенной вершины, выбирать максимальный из возможных;

4. в результате у нас будет вектор, в котором будут записаны максимальные значения от начальной вершины;

5.Выходные данные: если Ник может увеличить свое состяние, вывести *"YES"*, иначе *"NO"*.

3.2 Листинг

Листинг 2. Исходный код для 1162

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


// special structure to store exchange points parameters
struct _exchange {
    int a;
    int b;
    double rate;
    double commission;
};

std::vector<_exchange> _exchanges;

double nodes[101];


int main() {

    int n, m, s;
    double v;

    std::cin >> n >> m >> s >> v;
    nodes[s] = v;

    for (int i = 0; i < m; ++i) {

        int a, b;
        double rab, cab, rba, cba;

        std::cin >> a >> b >> rab >> cab >> rba >> cba;

        _exchanges.push_back({a, b, rab, cab});
        _exchanges.push_back({b, a, rba, cba});
    }

    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < _exchanges.size(); ++j) {
```

```
40              nodes[_exchanges[j].b] = std::max(_exchanges[j].
                    rate *
41                      (nodes[_exchanges[j].a] - _exchanges[j].
                            commission),nodes[_exchanges[j].b]);
42          }
43      }
44
45      for (int i = 0; i < _exchanges.size(); ++i) {
46          if ( nodes[_exchanges[i].b] < _exchanges[i].rate * (
                nodes[_exchanges[i].a] - _exchanges[i].commission)
                ) {
47              std::cout << "YES";
48              exit(0);
49          }
50      }
51      std::cout << "NO";
52  }
```

3.3 Результат

Результаты проверки решений

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10637308	15:51:57 6 май 2024	mistDragon	1162. Currency Exchange	Clang++ 17 x64	Accepted		0.015	412 КБ

Рис. 4. Результат отправки задачи 1162.

4 Задача 1080

1080. Раскраска карты

Ограничение времени: 1.0 секунды
Ограничение памяти: 64 МБ

Рассмотрим географическую карту с N странами, занумерованными от 1 до N ($0 < N < 99$). Для каждой страны известны номера соседних стран, т.е. имеющих общую границу с данной. Из каждой страны можно попасть в любую другую, перейдя некоторое количество границ. Напишите программу, которая определит, возможно ли покрасить карту только в два цвета — красный и синий — так, что если две страны имеют общую границу, их цвета различаются. Цвет первой страны — красный. Ваша программа должна вывести одну возможную раскраску для остальных стран или сообщить, что такая раскраска невозможна.

Исходные данные

В первой строке записано число N. Из следующих N строк i-я строка содержит номера стран, с которыми i-я страна имеет границу. Каждое целое число в i-й строке больше, чем i, кроме последнего, которое равно 0 и обозначает конец списка соседей i-й страны. Если строка содержит 0, это значит, что i-я страна не соединена ни с одной страной с бо́льшим номером.

Результат

Вывод содержит ровно одну строку. Если раскраска возможна, эта строка должна содержать список нулей и единиц без разделителей между ними. i-я цифра в этой последовательности обозначает цвет i-й страны. 0 соответствует красному цвету, единица — синему. Если раскраска невозможна, выведите целое число –1.

Пример

исходные данные	результат
3 2 0 3 0 0	010

Автор задачи: Emil Kelevedzhiev
Источник задачи: Winter Mathematical Festival Varna '2001 Informatics Tournament

Рис. 5. Условие задачи 1080.

4.1 Краткое описание алгоритма

В основе решения задачи находится алгоритм **обхода графа в ширину** (англ. *BFS, Breadth-first search*).

Пусть $G = (V, E)$ – невзвешенный ориентированный граф (в нашем алгоритме будет частный случай – неориентированный граф), в котором выделена исходная вершина s. необходимо найти длину кратчайшего пути, если он существует, от одной заданной вершины до другой.

Для реализации алгоритма используется структура данных **очередь**, также используется множество посещенных вершин, которое вначале состоит из одной вершины s. На каждой итерации алгоритма берем из начала очереди вершину v и складываем все непосещенные смежные с ней вершины в конец очереди. Так продолжает пока очередь не пуста.

1. Входные данные: в первой строке записано число N. Из следующих N строк i-я строка содержит номера стран, с которыми i-я страна имеет границу. Каждое целое число в i-й строке больше, чем i, кроме последнего, которое равно 0 и обозначает конец списка соседей i-й страны. Если строка содержит 0, это значит, что i-я страна не соединены ни с одной страной с большим номером.

2. Покрасим начальную вершину в красный цвет, всех ее соседей соответственно в синий, так пройдем все вершины;

3. если какая-то вершина оказалась уже покрашеннойи возник конфликт раскраски, значит, данный граф невозможно раскрасить в два цета (граф не является двудольным);

4. если после завершения обхода остались нераскрашенными какие-то вершины, значит данный граф представлен более чем одной компонентой связности, проведем раскраску для оставшихся компонент;

5. Выходные данные: вывод содержит ровно одну строку. Если раскраска возможна, эта строка должна содержать список нулей и единиц без разделителей между ними. i-я цифра в этой последовательности обозначает цвет i-й страны. 0 соответствует красному цвету, единица – синему. Если раскраска невозможна, выведите целое число -1.

4.2 Листинг

Листинг 3. Исходный код для 1080

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


std::vector<int> color(100, -1);
```

```
7  std::vector<int> edge[100];
8
9
10 // bfs algorithm
11 void _my_bfs(int _start) {
12
13     std::queue<int> _queue;
14     _queue.push(_start);
15     color[_start] = 0;
16
17     while (!_queue.empty()) {
18         int v = _queue.front();
19         _queue.pop();
20
21         for (int i = 0; i < edge[v].size(); ++i) {
22             int _cur_edge = edge[v][i];
23
24             if (color[v] == color[_cur_edge]) {
25                 std::cout << -1;
26                 exit(0);
27             }
28
29             if (color[_cur_edge] == -1) {
30                 color[_cur_edge] = !color[v];
31                 _queue.push(_cur_edge);
32             }
33         }
34     }
35 }
36
37
38 int main() {
39     int n;
40
41     std::cin >> n;
42
43     for (int i = 0; i < n; ++i) {
44         int _token = -1;
45
46         while (_token != 0) {
47             std::cin >> _token;
48
49             if (_token != 0) {
```

```
50                  edge[i].push_back(--_token);
51                  edge[_token++].push_back(i);
52              }
53          }
54      }
55
56      _my_bfs(0);
57
58      // check that all vertexes are colored
59      for (int i = 0; i < n; ++i) {
60          if (color[i] == -1) {
61              _my_bfs(i);
62          }
63      }
64      for (int i = 0; i < n; ++i) {
65          std::cout << color[i];
66      }
67      return 0;
68  }
```

4.3 Результат

10637357	17:26:26 6 май 2024	mistDragon	1080. Раскраска карты	Clang++ 17 x64	Accepted		0.015	432 КБ

Рис. 6. Результат отправки задачи 1080.

5 Вывод по работе

В ходе выполнения данной лабораторной работы были реализованы алгоритмы для решения задач 1160, 1162 и 1080.

Решение задачи 1160 основывается на применении алгоритма Краскала, задачи 1162 – алгоритм Форда-Беллмана, задачи 1080 – алгоритм обхода графа в ширину.