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, 1650"

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 1650.

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 Вывод по работе

В ходе выполнения данной лабораторной работы были реализованы алгоритмы для решения задач 1160, 1162 и 1650.

Решение задачи 1160 основывается на применении алгоритма Краскала.