|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра параллельных вычислительных технологий | | |
| Практическое задание № 8 | | |
| по дисциплине «Методы построения и анализа алгоритмов» | | |
| **Алгоритм Прима** | | |
|  | | |
|  |  | АРИФУЛОВ ТИГРАН |
| Группа ПМ-01 | ДЫЧКО АРСЕНИЙ |
|  | ЖИЖЧЕНКО ЛЕОНИД |
|  | САМСОНОВ СЕМЕН |
|  | ЯКОВЛЕВА ЕЛЕНА |
|  |  |
| Преподаватель | домников пётр александрович |
|  |  |
| Новосибирск, 2021 | | |

# Задание:

Написать подпрограмму построения минимального остовного дерева в взвешенном графе с использованием алгоритма Прима. Граф задается в файле в следующем формате. Первое число – количество вершин в графе. Второе число – количество ребер в графе. Далее перечислены ребра графа, заданные тройками чисел: первое число – начальная вершина ребра, второе число – конечная вершина ребра, третье число – вес ребра (натуральное число). На выходе файл с весом минимального дерева и ребрами дерева.

# Текст программы:

#include <iostream>

#include <vector>

#include <queue>

#include <fstream>

#define pb push\_back

#define mp make\_pair

#define F first

#define S second

#define sz(x) (int)x.size()

using pii = std::pair<int, int>;

using namespace std;

int main()

{

ifstream in("input.txt");

int n, m;

in >> n >> m;

vector < vector < pii > > g(n + 1); // начальный граф (0 элемент не используется) (pii используется вида конецРебра - весРебра)

for (int i = 1; i <= m; ++i)

{

int u, v, w;

in >> u >> v >> w;

g[u].pb(mp(v, w));

g[v].pb(mp(u, w));

}

in.close();

priority\_queue < pii, vector < pii >, greater < pii > > q; // Очередь вес-вершина

vector < int > key(n + 1, INT\_MAX); // Храним вес минимального ребра, исходящего из выбранной вершины

vector < int > p(n + 1, -1); // Храним вершину, куда указывает минимальное ребро из выбранной вершины

vector < bool > was(n + 1, false); // Для отметок, прошли ли мы вершину

int start = 1; // Начальная вершина обхода

int min\_w = 0; // Вес минимального графа

q.push(mp(0, start));

while (!q.empty())

{

int len = q.top().F; // Длина ребра

int v = q.top().S; // Вершина, куда ребро привело

q.pop();

if (was[v]) // Если уже были в этой вершине, то пропускаем ребро

continue;

was[v] = true; // Помечаем вершину как пройденную, считаем вес ребра, который вёл до неё

min\_w += len;

for (auto& cur : g[v]) // Находим все рёбра этой вершины

{

int to = cur.F; // Вершина, куда текущее ребро ведёт

int w = cur.S; // Вес текущего ребра

if (!was[to] && key[to] > w) // Если мы ещё не были в вершине и вес минимального ребра этой вершины больше, чем вес текущего ребра, то

{

key[to] = w; // Запоминаем для этой вершины вес минимального ребра

q.push(mp(w, to)); // Пушим вершину с весом до неё в очередь

p[to] = v; // Отмечаем, что из этой вершины можно попасть в вершину [v]

}

}

}

ofstream out("output.txt");

out << min\_w << endl;

for (int i = 2; i <= n; ++i)

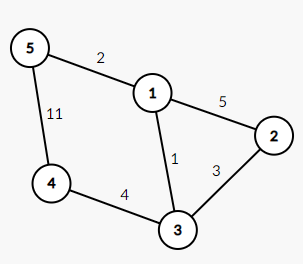
out << p[i] << " - " << i << endl;

out.close();

return 0;

}

# Результат работы программы:

1. Тест №1
   1. Входные данные:

5 6

1 2 5

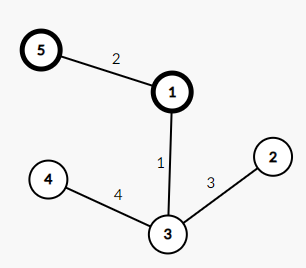
1 3 1

1 5 2

2 3 3

3 4 4

4 5 11

* 1. Выходные данные:

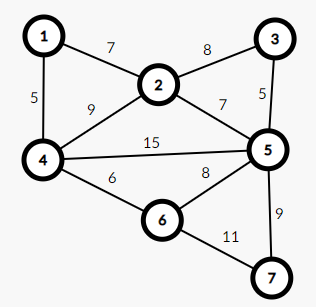
10

3 - 2

1 - 3

3 - 4

1 - 5

1. Тест №2
   1. Входные данные:

7 11

1 2 7

2 3 8

1 4 5

2 4 9

2 5 7

3 5 5

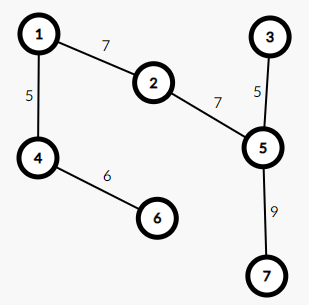
4 5 15

5 6 8

4 6 6

6 7 11

5 7 9

* 1. Выходные данные:

39

1 - 2

5 - 3

1 - 4

2 - 5

4 - 6

5 - 7