Министерство образования Республики Беларусь
Учреждение образования
“Брестский Государственный технический университет”
Кафедра ИИТ

Лабораторная работа №5

По дисциплине “Математические основы интеллектуальных систем”

Тема: “Нахождение минимального остового дерева связанного неориентированного графа”

Выполнил:
Студент 2 курса
Группы ИИ-21
Кирилович А. А.
Проверил:
Козинский А. А.

Брест 2022

Цель: научиться находить минимальное остовное дерево во взвешенном связном графе с помощью алгоритмов Прима и Краскала.

Ход работы:

Задание 1

Найти минимальное остовое дерево для заданного графа алгоритмом Прима.
(1,2),(1,3),(1,4),(2,3),(2,5),(3,4),(3,5),(4,5)
[5,3,6,3,4,2,5,1]

```
#include "../graph_LIB.hh"

int main() {
	convert c;
	std::string file_path = "connections.txt";
	std::vector<int> nodes = c.reading_file(file_path);
	file_path = "weights.txt";
	std::vector<int> weights = c.reading_file(file_path);

	int max_node = c.count_of_nodes(nodes);
	std::vector<std::vector<int>> adjacencyMatrix = c.adjancy(nodes, weights, max_node);

	alg search;
	std::vector<std::vector<int>> tree;
	tree = search.Prim(adjacencyMatrix);
	std::vector<int> couple = c.couple_from_adjancy(tree);

	c.print(couple);
}
```

Реализация функции Prim из файла Graph_LIB:

```
std::vector<std::vector<int>> Prim(std::vector<std::vector<int>> &adjacencyMatrix) {
	int max_node = adjacencyMatrix.size();
	std::vector<std::vector<int>> tree(max_node, std::vector<int>(max_node));
	std::vector<int> key(max_node);
	std::vector<int> parent(max_node);
	std::vector<int> visited(max_node);
	for (int i = 0; i < max_node; i++) {
		key[i] = INT_MAX;
		visited[i] = 0;
	}
	key[0] = 0;
	parent[0] = -1;
	for (int i = 0; i < max_node - 1; i++) {
		int min = INT_MAX;
		int min_index = 0;
		for (int j = 0; j < max_node; j++) {
			if (visited[j] == 0 && key[j] < min) {
				min = key[j];
				min_index = j;
			}
		}
		visited[min_index] = 1;
		for (int j = 0; j < max_node; j++) {
			if (visited[j] == 0
			&& adjacencyMatrix[min_index][j]
			&& adjacencyMatrix[min_index][j] < key[j]) {
				parent[j] = min_index;
				key[j] = adjacencyMatrix[min_index][j];
			}
		}
	}
	for (int i = 1; i < max_node; i++) {
		tree[i][parent[i]] = adjacencyMatrix[i][parent[i]];
		tree[parent[i]][i] = adjacencyMatrix[i][parent[i]];
	}
	return tree;
}
```

```
(2,4),(3,4),(4,5),(5,6)
```

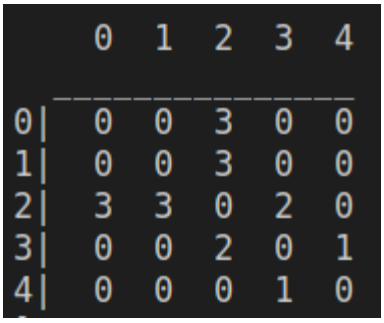
Задание 2

Найти минимальное остовое дерево для заданного графа алгоритмом Крускаля.

```
#include "../graph_LIB.hh"

int main() {
	convert c;
	std::string file_path = "connections.txt";
	std::vector<int> nodes = c.reading_file(file_path);
	file_path = "weights.txt";
	std::vector<int> weights = c.reading_file(file_path);

	int max_node = c.count_of_nodes(nodes);
	std::vector<std::vector<int>> adjacencyMatrix = c.adjancy(nodes, weights, max_node);

	alg search;
	std::vector<std::vector<int>> tree;
	tree = search.Kruskal(adjacencyMatrix);

	c.print(tree);
}
```

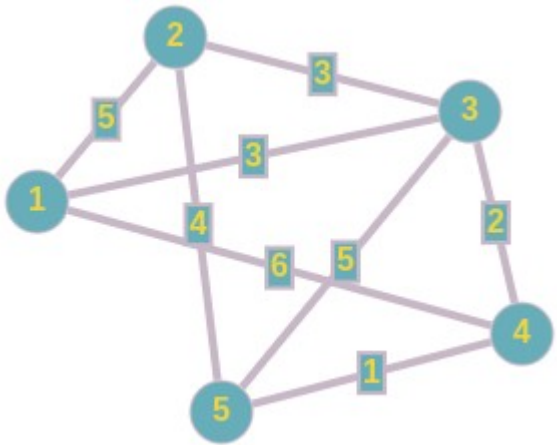
Реализация функции Kruskal из файла Graph_LIB:

```
std::vector<std::vector<int>> Kruskal(std::vector<std::vector<int>> &adjacencyMatrix) {
	int max_node = adjacencyMatrix.size();
	std::vector<std::vector<int>> tree(max_node, std::vector<int>(max_node));
	std::set<std::set<int>> nodes;
	std::vector<std::vector<int>> edges;
	for (int i = 0; i < max_node; i++) {
		for (int j = i + 1; j < max_node; j++) {
			if (adjacencyMatrix[i][j]) {
				edges.push_back({ adjacencyMatrix[i][j], i, j });
			}
		}
	}
	std::sort(edges.begin(), edges.end());

	for (int i = 0; i < max_node; i++) {
		nodes.insert(std::set<int>{i});
	}

	for (int i = 0; i < edges.size(); i++) {
		std::set<int> node1;
		std::set<int> node2;
		for (auto it = nodes.begin(); it != nodes.end(); it++) {
			if (it->find(edges[i][1]) != it->end()) {
				node1 = *it;
			}
			if (it->find(edges[i][2]) != it->end()) {
				node2 = *it;
			}
		}
		if (node1 != node2) {
			nodes.erase(node1);
			nodes.erase(node2);
			node1.insert(node2.begin(), node2.end());
			nodes.insert(node1);
			tree[edges[i][1]][edges[i][2]] = edges[i][0];
			tree[edges[i][2]][edges[i][1]] = edges[i][0];
		}
	}
	return tree;
}
```

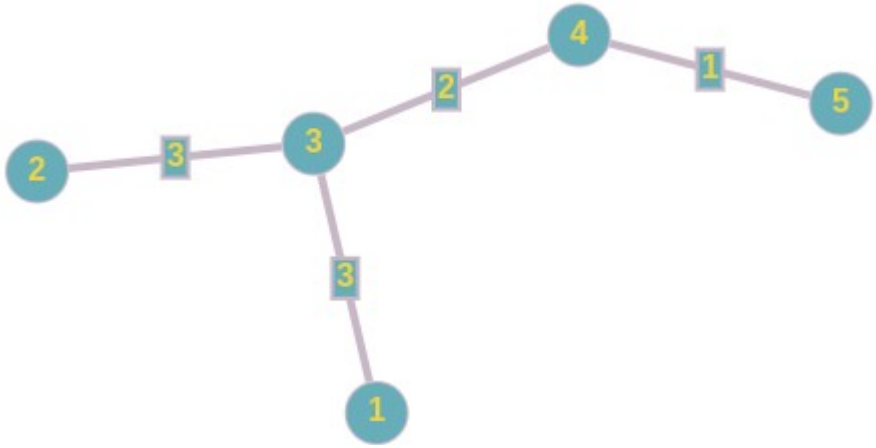


Вывод: в ходе лабораторной работы я научился находить минимальное остовное дерево во взвешенном связном графе с помощью алгоритмов Прима и Краскала.

Граф:



Минимальное остовное дерево:



Вопросы.

1. Для какого графа определяет число остовых деревьев формула Кэли.
   Ответ: для полного.
2. Какое остовое дерево находится алгоритмом Дейкстры?
   Ответ: минимальное.
3. Может ли быть несколько минимальных остовых деревьев?
   Ответ: да.