Отчет по расчетной работе по дисциплине ПиОИвИС

Цель:

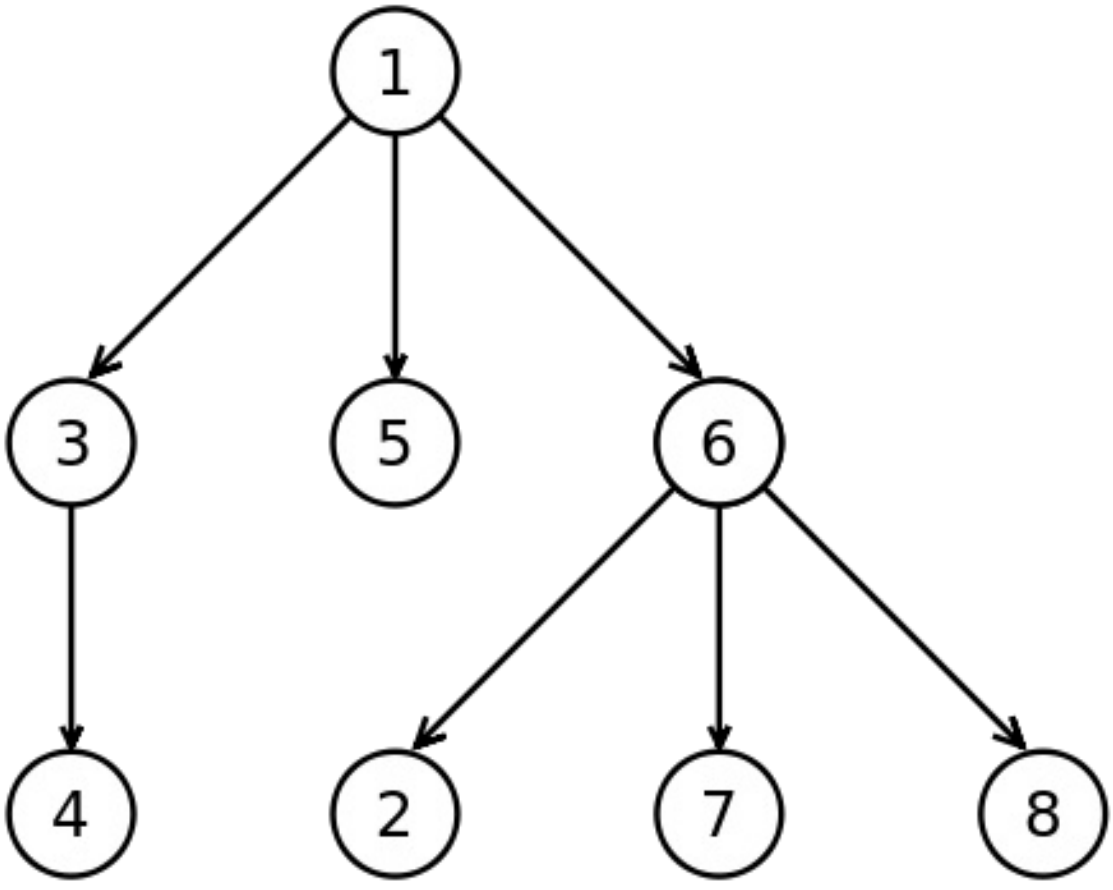
1.1. Ознакомиться с основами теории графов, способами представления графов, базовыми алгоритмами для работы с разными видами графов.

Условие задания

Определить вид графа: дерево
Граф задается списком смежности.
Вывод работы программы выводиться в консоль.

Базовые понятия

- **Граф** - совокупность двух множеств множества самих объектов, называемого множеством вершин, и множества их парных связей, называемого множеством рёбер.
- **Дерево** - связный ациклический граф. Связность означает наличие маршрута между любой парой вершин, ацикличность — отсутствие циклов. Отсюда, в частности, следует, что число рёбер в дереве на единицу меньше числа вершин, а между любыми парами вершин имеется один и только один путь.

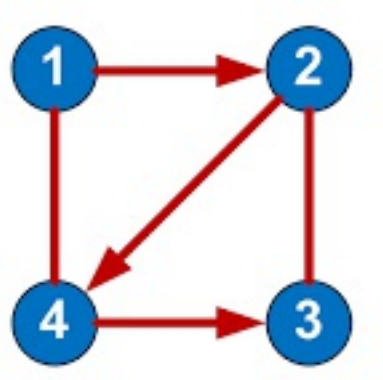


- **Списком смежности** — один из способов представления графа в виде коллекции списков вершин. Каждой вершине графа соответствует список, состоящий из

«соседей» этой вершины.

-

1	2, 4
2	3, 4
3	2
4	1, 3



Код C++:

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

using namespace std;

bool isConnected(const vector<vector<int>>& graph) {
    int n = graph.size();
    if (n == 0) return false;

    vector<bool> visited(n, false);
    queue<int> q;
    q.push(0);
    visited[0] = true;

    int visitedCount = 0;

    while (!q.empty()) {
        int node = q.front();
        q.pop();
        ++visitedCount;

        for (int neighbor : graph[node]) {
            if (!visited[neighbor]) {
                visited[neighbor] = true;
                q.push(neighbor);
            }
        }
    }

    return visitedCount == n;
}

bool isTree(const vector<vector<int>>& graph) {
    int n = graph.size();
    int edgeCount = 0;
```

```
    for (const auto& neighbors : graph) {
        edgeCount += neighbors.size();
    }
    edgeCount /= 2;

    return isConnected(graph) && edgeCount == n - 1;
}

int main() {
    vector<vector<int>> graph_1{
        {1},
        { 0, 2, 3 },
        { 1 },
        { 1 }
    };

    vector<vector<int>> graph_2{
        {1, 2},
        { 0, 2 },
        { 0, 1, 3 },
        { 2 }
    };

    if (isTree(graph_1)) {
        cout << "Graph_1 is a tree." << endl;
    }
    else {
        cout << "Graph_1 is not a tree." << endl;
    }

    if (isTree(graph_2)) {
        cout << "Graph_2 is a tree." << endl;
    }
    else {
        cout << "Graph_2 is not a tree." << endl;
    }

    return 0;
}
```

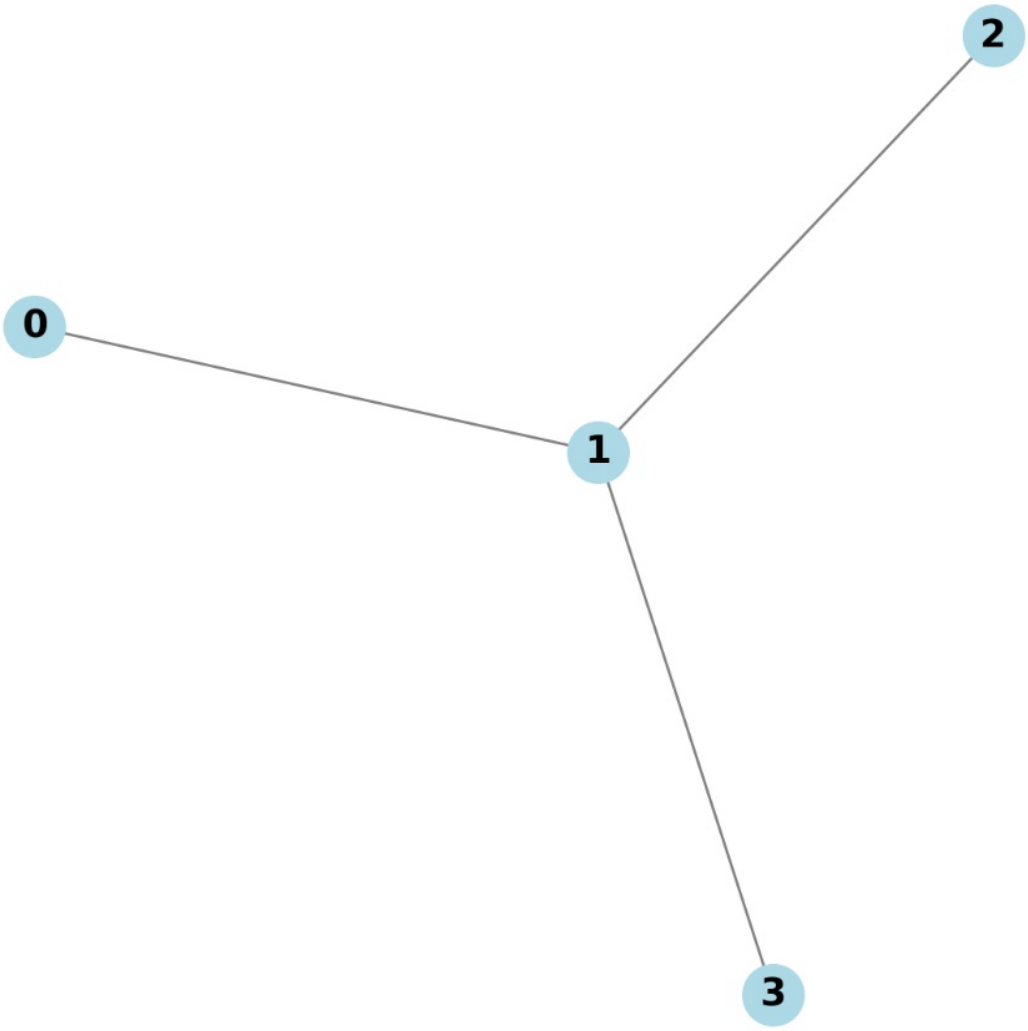
Описание алгоритма

- Задаем два графа списком смежности
- Затем с помощью дополнительных функции проверяем вид графа
    - Функция проверки на дерево:
        - 1)Функция isConnected обходит граф, начиная с вершины 0, и отмечает посещённые вершины.
        - 2)Все соседние связи подсчитываются для каждой вершины, а затем делятся на 2, чтобы избежать двойного счёта.
        - 3)Если граф связный и количество рёбер равно n−1, то граф

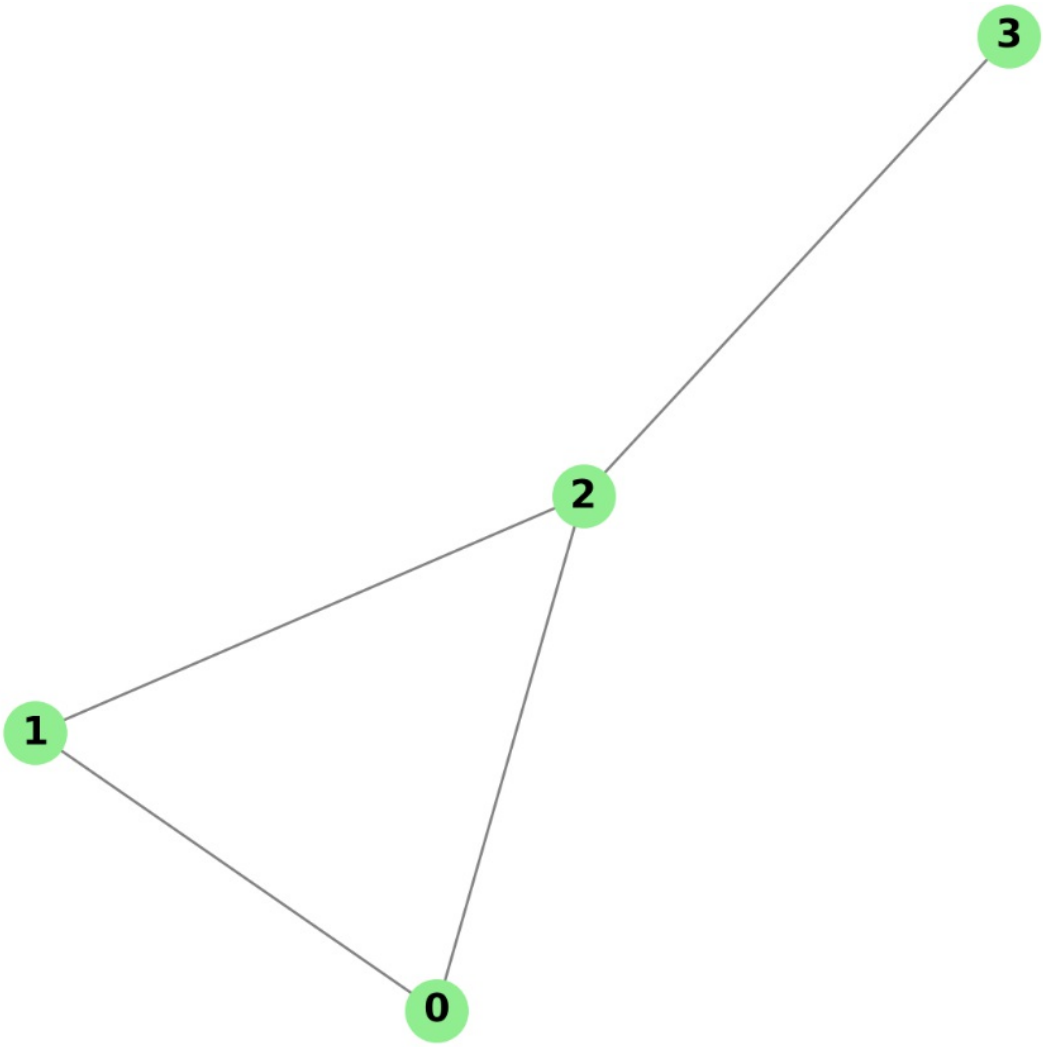
является деревом.

Описание алгоритма

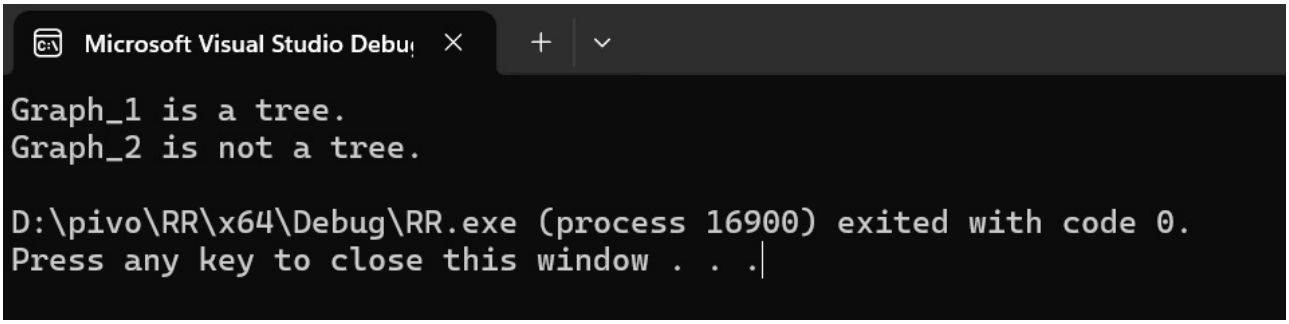
- Граф 1:



- Граф 2:



Пример работы



Вывод

В результате выполнения данной работы были получены следующие практические навыки:

- изучены основы теории графов
- изучены способы представления графов
- изучены базовые алгоритмы для работы с графами