**Лабораторная работа.**

**Поиск кратчайших путей в графе. Построение остовного дерева графа.**

Для созданного Вами взвешенного ориентированного графа, состоящего как минимум из 15 вершин, реализовать по вариантам:

1. Алгоритм поиска кратчайшего пути.
2. Сделав тот же самый граф неориентированным, построить его остовное дерево минимальной стоимости.

Матрицу смежности ( значения весов каждого ребра) лучше определить в начале программы.

Должны быть представлены промежуточные результаты.

По каждому кратчайшему пути указать предшествующие вершины.

Граф и полученное остовное дерево должны быть изображены на рисунках в отчете.

Варианты заданий:

|  |  |  |
| --- | --- | --- |
| № вар. | Алгоритм, реализуемый в п.1 задания | Алгоритм, п.2 задания |
| 1 | Дейкстры | Прима |
| 2 | Беллмана-Форда | Крускала |
| 3 | Флойда-Уоршелла | Прима |
| 4 | Дейкстры | Крускала |
| 5 | Беллмана-Форда | Прима |
| 6 | Флойда-Уоршелла | Крускала |
| 7 | Дейкстры | Крускала |
| 8 | Беллмана-Форда | Прима |
| 9 | Флойда-Уоршелла | Крускала |
| 10 | Дейкстры | Прима |
| 11 | Флойда-Уоршелла | Крускала |
| 12 | Беллмана-Форда | Прима |
| 13 | Дейкстры | Крускала |
| 14 | Флойда-Уоршелла | Прима |
| 15 | Дейкстры | Крускала |

**Лабораторная работа «Бинарные деревья поиска»**

**Задание**

1. Реализовать функции вставки, поиска, удаления узла, обхода дерева, вывода дерева на экран, нахождения высоты дерева и количества узлов.
2. Реализовать заданную функцию в соответствии с вариантом: *T* – тип ключей, *D* – диапазон изменения значений ключей.

|  |  |  |  |
| --- | --- | --- | --- |
| № | *T* | *D* | Функция |
| 1 | **int** | [100; 200] | Подсчет суммы длин путей от корня до каждого из узлов, содержащих четные числа |
| 2 | **char** | [а..я, А..Я] | Подсчет количества гласных в листьях |
| 3 | **int** | [0; 100] | Подсчет количества нечетных чисел в узлах, имеющих ровно два поддерева |
| 4 | **char** | [а..я, А..Я] | Подсчет количества согласных на 2 и 3 уровнях |
| 5 | **int** | [-50; 50] | Подсчет суммы четных отрицательных чисел в узлах, поддеревья которых содержат не более 4 узлов |
| 6 | **char** | [а..я, А..Я] | Подсчет количества гласных на четных уровнях |
| 7 | **int** | [0; 100] | Определить сумму четных чисел |
| 8 | **char** | [a..z, A..Z] | Подсчет количества согласных в узлах, высота поддеревьев которых одинакова |
| 9 | **char** | [a..z, A..Z] | Определить, каких букв в дереве больше - гласных или согласных |
| 10 | **char** | [а..я, А..Я] | Определить число узлов в левом и правом поддеревьях |
| 11 | **int** | [10; 90] | Определить два минимальных элемента |
| 12 | **char** | [а..я, А..Я] | Определить количество элементов дерева на каждом уровне |
| 13 | **int** | [-100; 100] | Определить, каких чисел больше - положительных или отрицательных. |
| 14 | **char** | [а..я, А..Я] | Все гласные буквы заменить насимвол «\*» |
| 15 | **int** | [-50; 50] | Определить сумму элементов, кратных 5 |