Федеральное государственное бюджетное образовательное учреждение высшего образования «Байкальский государственный университет»

Прикладная информатика

Объектно-ориентированное программирование

Лабораторная работа №2.1

Вариант: 3

Выполнил студент

Вайкус Яков Михайлович

Группа № ИС-23-1

Преподаватель: Федотов Никита Андреевич

г. Иркутск

2025

Оглавление

[**Задание** 3](#_Toc207876137)

[**Код программы** 3](#_Toc207876138)

[Вывод программы: 4](#_Toc207876139)

# **Задание**

# Лабораторная работа 5: Построение минимального остовного дерева с проверкой связности Реализуйте алгоритм Краскала для построения минимального остовного дерева (МОД) графа, заданного списком рёбер (кортежи (v1, v2, weight)). Используйте циклы for для сортировки рёбер по весу, while для сборки дерева (пока количество рёбер < n-1). Условные операторы проверяют циклы (if find(v1) == find(v2): skip) с логическими условиями (if weight > 0 and not cycle: union). Арифметические операторы суммируют веса. Try-except обрабатывает KeyError при отсутствии вершины. Тестируйте на edge cases: несвязный граф (остов не построен), граф с отрицательными весами (raise ValueError). Выведите список рёбер МОД и общий вес.

# **Код программы**

class Kruskal:

    def \_\_init\_\_(self):

        self.parent = {}

    def find(self, v):

        try:

            if self.parent[v] != v:

                self.parent[v] = self.find(self.parent[v])

            return self.parent[v]

        except KeyError:

            raise KeyError(f"Вершина {v} отсутствует в множестве")

    def union(self, v1, v2):

        root1 = self.find(v1)

        root2 = self.find(v2)

        if root1 != root2:

            self.parent[root2] = root1

    def add(self, v):

        if v not in self.parent:

            self.parent[v] = v

def kruskal(edges):

    # Проверка на отрицательные веса

    for (v1, v2, w) in edges:

        if w < 0:

            raise ValueError("Граф содержит отрицательные веса!")

    ds = Kruskal()

    vertices = set()

    for v1, v2, \_ in edges:

        vertices.add(v1)

        vertices.add(v2)

        ds.add(v1)

        ds.add(v2)

    # Сортировка рёбер по весу (через for)

    sorted\_edges = []

    for e in edges:

        inserted = False

        for i in range(len(sorted\_edges)):

            if e[2] < sorted\_edges[i][2]:

                sorted\_edges.insert(i, e)

                inserted = True

                break

        if not inserted:

            sorted\_edges.append(e)

    mst = []

    total\_weight = 0

    i = 0

    # Пока рёбер в остове меньше n-1

    while len(mst) < len(vertices) - 1 and i < len(sorted\_edges):

        v1, v2, w = sorted\_edges[i]

        i += 1

        # Проверка на цикл

        if ds.find(v1) != ds.find(v2):

            ds.union(v1, v2)

            mst.append((v1, v2, w))

            total\_weight += w

        # иначе пропускаем ребро

    # Проверка связности

    if len(mst) != len(vertices) - 1:

        print("Граф несвязный — остов не построен.")

        return None, None

    return mst, total\_weight

# ================== Тесты ==================

import networkx as nx

G = nx.Graph()

G.add\_weighted\_edges\_from([

    ('A', 'B', 4),

    ('A', 'C', 2),

    ('B', 'C', 1),

    ('B', 'D', 5),

    ('C', 'D', 8),

    ('C', 'E', 10),

    ('D', 'E', 2),

])

mst\_nx = nx.minimum\_spanning\_tree(G, algorithm="kruskal")

print("Библиотека:", list(mst\_nx.edges(data=True)))

print("Общий вес:", sum(d["weight"] for \_,\_,d in mst\_nx.edges(data=True)))

# 1. Связный граф

edges1 = [

    ('A', 'B', 4),

    ('A', 'C', 2),

    ('B', 'C', 1),

    ('B', 'D', 5),

    ('C', 'D', 8),

    ('C', 'E', 10),

    ('D', 'E', 2),

]

mst, weight = kruskal(edges1)

print("Граф 1:", mst)

print("Общий вес:", weight)

# 2. Несвязный граф

edges2 = [

    ('A', 'B', 1),

    ('C', 'D', 2),

]

mst, weight = kruskal(edges2)

# 3. Граф с отрицательным весом

edges3 = [

    ('A', 'B', -5),

    ('B', 'C', 2)

]

try:

    mst, weight = kruskal(edges3)

except ValueError as e:

    print("Ошибка:", e)

# **Вывод программы:**

Библиотека: [('A', 'C', {'weight': 2}), ('B', 'C', {'weight': 1}), ('B', 'D', {'weight': 5}), ('D', 'E', {'weight': 2})]

Общий вес: 10

Граф 1: [('B', 'C', 1), ('A', 'C', 2), ('D', 'E', 2), ('B', 'D', 5)]

Общий вес: 10

Граф несвязный — остов не построен.

Ошибка: Граф содержит отрицательные веса!