Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

Выполнил студент группы КС-36 Кошкарев Иван Михайлович

Ссылка на репозиторий: https://github.com/MUCTR-IKT-CPP/KoshkarevIM\_36\_ALG/tree/main/Lab1

Приняли: Крашенинников Роман Сергеевич

Дата сдачи: 17.03

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

Реализовать модель грфафа, а также добавить к ней методы выводов различном виде. Реализовать функции генерации графов, и функцию создания минимального остового дерева из графа. Сравнить время создания остовных деревьев для разного количества вершин и рёбер

# Описание метода/модели.

Под графом в математике понимается абстракция реальной системы объектов безотносительно их природы, обладающих парными связями.

Вершина графа – это некоторая точка связанная с другими точками

Ребро графа – это линия, соединяющая две точки и олицетворяющая связь между ними

Граф – это множество вершин соединённых друг с другом произвольным образом множеством ребер

Существуют ориентированные, сложные и простые графы,

Специальные виды графов:

Полный граф – все вершины графа соединены друг с другому

Регулярный граф – степени всех вершин одинаковы

Биграф – все вершины разделяются на две группы и нет такого ребра который соединяет вершины из одной группы

Дерево – это такой граф, в котором не содержатся циклы, так же, для каждой вершины такого графа существует только один маршрут позволяющий добраться до этой вершины.

Остовными деревьями в графе называются такие подмножества ребер графа, которые создают дерево, содержащее все вершины графа.

Для взвешенного графа существует минимальное остовное дерево, это такое дерево пути в котором являются минимальными.

Метод Краскала для построение минимального остовного древа:

Алгоритм Крускала является альтернативой алгоритму Прима, и в свою очередь начинает построение дерева не с одной определенной точки, а со всех точек одновременно, считая их частями дерева связи между которыми нужно востановить.

Островное-Минмиальное-Дерево-Крускала (Граф)

Помещаем все ребра в очередь с приоритетом, упорядочивая по весу.

Счетчик = 0

До тех пор пока Счетчик не достиг размера графа – 1

Рассматриваем ребро соединяющее две вершины (v, w)

Проверяем, не существует ли маршрут между v и w

Добавляем ребро в итоговое дерево

Объединяем поддеревья v и w

Создав предварительно очередь в которую поместив все ребра в порядке возрастания их веса, мы последовательно проходим по всем ребрам.

На каждом шаге, мы проверяем не является ли ребро, таким которое соединяет уже соединенные вершины в итоговом графе, если ребро является таковым то мы его отбрасываем, иначе добавляем в итоговое дерево.

Доказательство правильности алгоритма строится от обратного, при этом так же как в алгоритме Прима предполагается, что существует такой момент когда алгоритм ошибается, но в этом случае появляется цикл, так как до того как появилось такое ребро уже было обработано ребро которое лежит в пути между двумя вершинами соединёнными неправильным ребром с большим весом.

Асимптотическая сложность алгоритма O(n\*m), но если постараться, можно получить O(m\*lg(m)). Алгоритм показывает лучшую чем алгоритм Прима производительность на разряженных графах.

# Выполнение задачи.

Для реализации модели и тестов я использовал язык Python из-за его простой архитектуры.

РЕАЛИЗАЦИЯ КЛАССА И МЕТОДОВ ВЫВОДА

# Класс графа  
class Graph:  
 def \_\_init\_\_(self, num\_vert, edges):  
 self.num\_vert = num\_vert  
 self.edges = edges  
  
 # Матрица смежности  
 def adjacency\_matrix(self):  
 matrix = [[0] \* self.num\_vert for \_ in range(self.num\_vert)]  
 for u, v, weight in self.edges:  
 matrix[u][v] = weight  
 matrix[v][u] = weight  
 return matrix  
  
 # Список смежности  
 def adjacency\_list(self):  
 adj\_list = {v: [] for v in range(self.num\_vert)}  
 for u, v, weight in self.edges:  
 adj\_list[u].append((v, weight))  
 adj\_list[v].append((u, weight))  
 return adj\_list  
  
 # Список ребер  
 def edge\_list(self):  
 return self.edges

РЕАЛИЗАЦИЯ МЕТОДА КРУСКАЛА

# Функция для поиска минимального остовного дерева (алгоритм Краскала)  
def kruskal(self):  
 # Вспомогательная функция для нахождения корня множества  
 def find(parent, i):  
 if parent[i] == i:  
 return i  
 return find(parent, parent[i])  
  
 # Вспомогательная функция для объединения двух множеств  
 def union(parent, rank, x, y):  
 xroot = find(parent, x)  
 yroot = find(parent, y)  
  
 # Присоединяем меньшее дерево к большему  
 if rank[xroot] < rank[yroot]:  
 parent[xroot] = yroot  
 elif rank[xroot] > rank[yroot]:  
 parent[yroot] = xroot  
 else:  
 parent[yroot] = xroot  
 rank[xroot] += 1  
  
 # Результирующее минимальное остовное дерево  
 result = []  
  
 # Индекс для сортированных рёбер  
 i = 0  
 # Индекс для результата  
 e = 0  
  
 # Шаг 1: Сортируем все рёбра по весу  
 self.edges = sorted(self.edges, key=lambda item: item[2])  
  
 parent = []  
 rank = []  
  
 # Создаём подмножества для каждой вершины  
 for node in range(self.num\_vert):  
 parent.append(node)  
 rank.append(0)  
  
 # Шаг 2: Проходим по всем рёбрам и добавляем их в дерево, если они не образуют цикл  
 while e < self.num\_vert - 1:  
 u, v, w = self.edges[i]  
 i += 1  
 x = find(parent, u)  
 y = find(parent, v)  
  
 # Если ребро не образует цикл, добавляем его в результат  
 if x != y:  
 e += 1  
 result.append((u, v, w))  
 union(parent, rank, x, y)  
  
 return result

РЕАЛИЗАЦИЯ ГЕНЕРАЦИИ СЛУЧАЙНЫХ ГРАФОВ

# Функция для создания связного взвешенного ненаправленного графа  
def create\_graph(num\_vertices, min\_edges, max\_weight=20):  
 edges = [] # Список рёбер  
  
 # Создаём граф - цепь  
 for i in range(1, num\_vertices):  
 weight = random.randint(1, max\_weight)  
 edges.append((i - 1, i, weight))  
  
 # Добавляем к каждой вершине оставшиеся случайные рёбра  
 for i in range(num\_vertices):  
 # Уже есть одно ребро в цепочке (кроме первой и последней вершины)  
 remaining\_edges = min\_edges - 2 if i not in (0, num\_vertices - 1) else min\_edges - 1  
  
 # Добавляем случайные рёбра  
 while remaining\_edges > 0:  
 j = random.randint(0, num\_vertices - 1)  
 if j != i and (i, j) not in edges and (j, i) not in edges: # Исключаем петли и дубликаты  
 weight = random.randint(1, max\_weight)  
 edges.append((i, j, weight))  
 remaining\_edges -= 1  
  
 return Graph(num\_vertices, edges)

РЕАЛИЗАЦИЯ ТЕСТОВ

# Параметры для создания графов  
num\_vertices\_list = [10, 20, 50, 100] # Количество вершин  
min\_edges\_list = [3, 4, 10, 20] # Минимальное количество рёбер на вершину  
max\_weight = 20  
num\_tests = 10 # Количество тестов для каждого графа  
  
graph = create\_graph(10, 5, max\_weight)  
matrix = graph.adjacency\_matrix()  
for mat in matrix:  
 print(mat)  
  
# Выполнение тестов  
for min\_edges in min\_edges\_list:  
 # Списки для хранения результатов  
 average\_execution\_times = []  
 sizes = []  
 print(f"\nТестирование графа с минимум {min\_edges} рёбрами:")  
 for num\_vertices in num\_vertices\_list:  
 print(f"\nТестирование графа с {num\_vertices} вершинами:")  
 execution\_times = []  
  
 for test in range(num\_tests):  
 # Создаём граф  
 graph = create\_graph(num\_vertices, min\_edges, max\_weight)  
  
 # Замер времени выполнения алгоритма Краскала  
 start\_time = time.time()  
 mst = graph.kruskal()  
 end\_time = time.time()  
  
 execution\_time = end\_time - start\_time  
 execution\_times.append(execution\_time)  
  
 print(f"Тест {test + 1}: Время выполнения = {execution\_time:.6f} секунд")  
  
 # Усредняем результаты  
 average\_time = sum(execution\_times) / num\_tests  
 average\_execution\_times.append(average\_time)  
 sizes.append(num\_vertices)  
  
 print(f"Среднее время выполнения для {num\_vertices} вершин: {average\_time:.6f} секунд")  
  
 # Построение графика  
 plt.figure(figsize=(10, 6))  
 plt.plot(sizes, average\_execution\_times, marker='o', linestyle='-', color='b')  
 plt.title(f"Зависимость времени выполнения алгоритма Краскала \nот количества вершин c минимум {min\_edges} рёбер у каждой вершины")  
 plt.xlabel("Количество вершин (N)")  
 plt.ylabel("Среднее время выполнения (секунды)")  
 plt.grid(True)  
 plt.show()

***Результаты:***

Матрица смежности:

[0, 9, 0, 0, 13, 0, 0, 17, 20, 7]

[9, 0, 9, 9, 0, 10, 0, 3, 0, 5]

[0, 9, 0, 9, 0, 4, 18, 17, 0, 2]

[0, 9, 9, 0, 6, 20, 0, 0, 0, 19]

[13, 0, 0, 6, 0, 8, 0, 0, 15, 14]

[0, 10, 4, 20, 8, 0, 18, 0, 17, 0]

[0, 0, 18, 0, 0, 18, 0, 10, 0, 10]

[17, 3, 17, 0, 0, 0, 10, 0, 6, 12]

[20, 0, 0, 0, 15, 17, 0, 6, 0, 6]

[7, 5, 2, 19, 14, 0, 10, 12, 6, 0]

Тестирование графа с минимум 3 рёбрами:

Тестирование графа с 10 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.000000 секунд

Среднее время выполнения для 10 вершин: 0.000000 секунд

Тестирование графа с 20 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.000000 секунд

Среднее время выполнения для 20 вершин: 0.000000 секунд

Тестирование графа с 50 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.000000 секунд

Среднее время выполнения для 50 вершин: 0.000000 секунд

Тестирование графа с 100 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.001000 секунд

Среднее время выполнения для 100 вершин: 0.000100 секунд

Тестирование графа с минимум 4 рёбрами:

Тестирование графа с 10 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.000000 секунд

Среднее время выполнения для 10 вершин: 0.000000 секунд

Тестирование графа с 20 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.000000 секунд

Среднее время выполнения для 20 вершин: 0.000000 секунд

Тестирование графа с 50 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.000000 секунд

Среднее время выполнения для 50 вершин: 0.000000 секунд

Тестирование графа с 100 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.001001 секунд

Тест 10: Время выполнения = 0.000000 секунд

Среднее время выполнения для 100 вершин: 0.000100 секунд

Тестирование графа с минимум 10 рёбрами:

Тестирование графа с 10 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.000000 секунд

Среднее время выполнения для 10 вершин: 0.000000 секунд

Тестирование графа с 20 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.000000 секунд

Среднее время выполнения для 20 вершин: 0.000000 секунд

Тестирование графа с 50 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000998 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.000000 секунд

Среднее время выполнения для 50 вершин: 0.000100 секунд

Тестирование графа с 100 вершинами:

Тест 1: Время выполнения = 0.001000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.001001 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000998 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.000999 секунд

Среднее время выполнения для 100 вершин: 0.000400 секунд

Тестирование графа с минимум 20 рёбрами:

Тестирование графа с 10 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.000000 секунд

Среднее время выполнения для 10 вершин: 0.000000 секунд

Тестирование графа с 20 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.000000 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.000000 секунд

Среднее время выполнения для 20 вершин: 0.000000 секунд

Тестирование графа с 50 вершинами:

Тест 1: Время выполнения = 0.000000 секунд

Тест 2: Время выполнения = 0.000000 секунд

Тест 3: Время выполнения = 0.000999 секунд

Тест 4: Время выполнения = 0.000000 секунд

Тест 5: Время выполнения = 0.000000 секунд

Тест 6: Время выполнения = 0.001003 секунд

Тест 7: Время выполнения = 0.000000 секунд

Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.000000 секунд

Среднее время выполнения для 50 вершин: 0.000200 секунд

Тестирование графа с 100 вершинами:

Тест 1: Время выполнения = 0.001001 секунд

Тест 2: Время выполнения = 0.001003 секунд

Тест 3: Время выполнения = 0.000000 секунд

Тест 4: Время выполнения = 0.000999 секунд

Тест 5: Время выполнения = 0.001000 секунд

Тест 6: Время выполнения = 0.000000 секунд

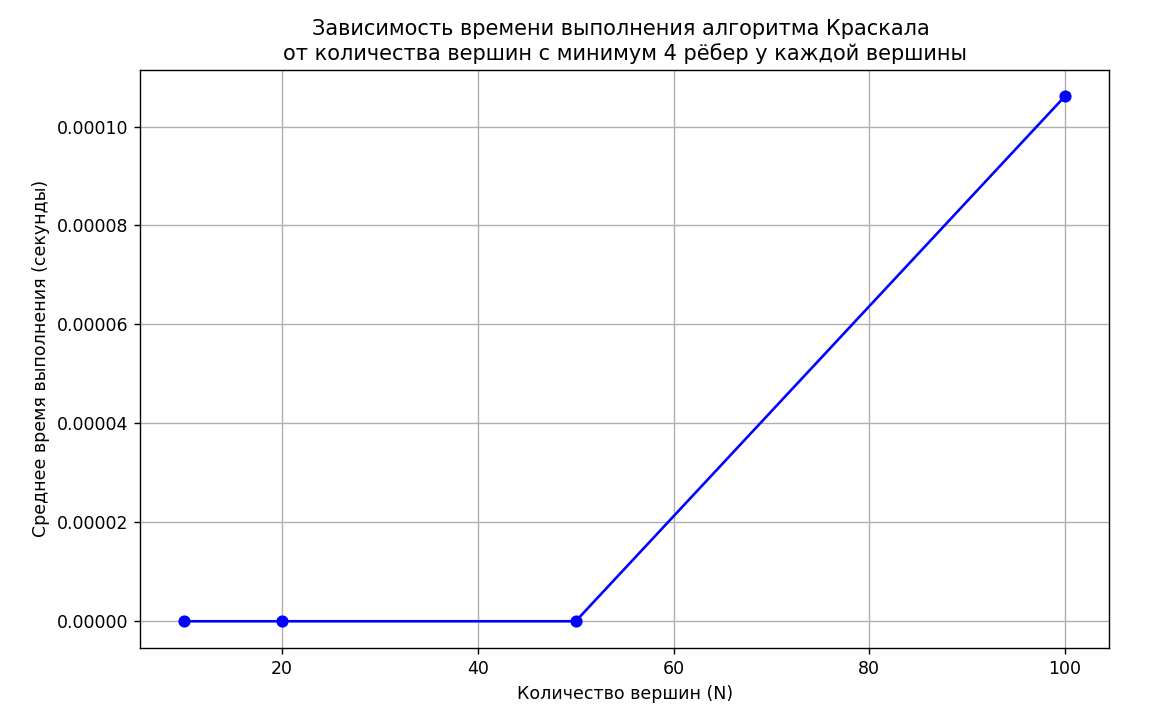
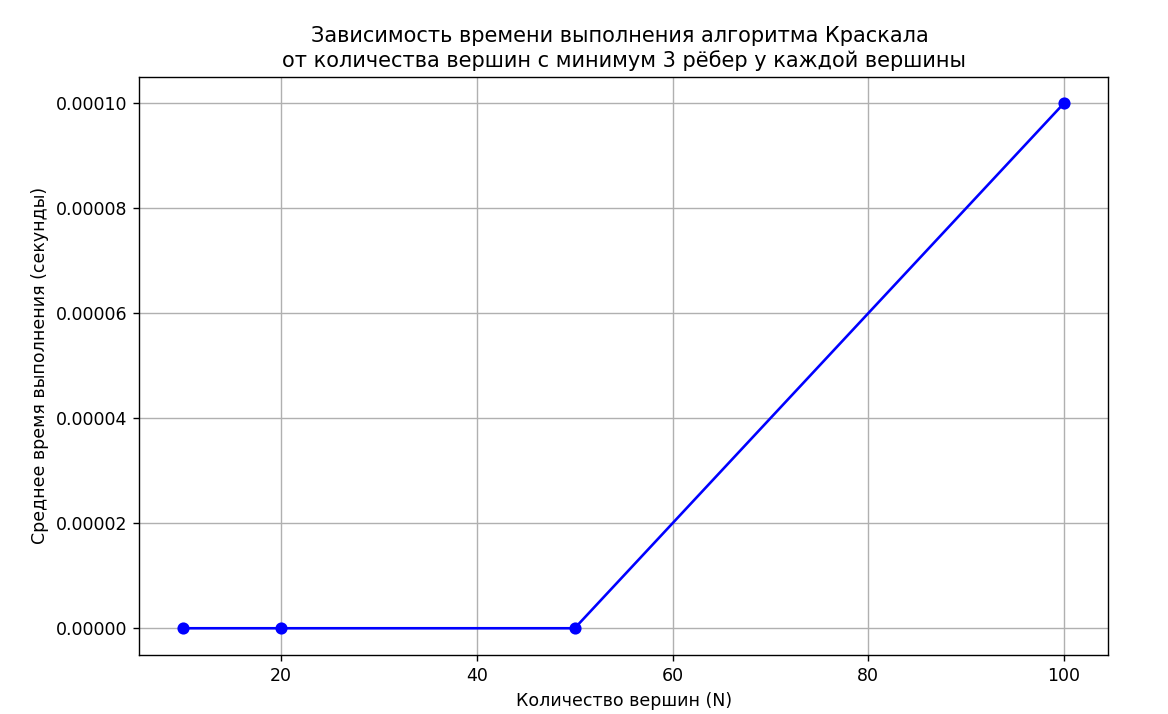
Тест 7: Время выполнения = 0.001004 секунд

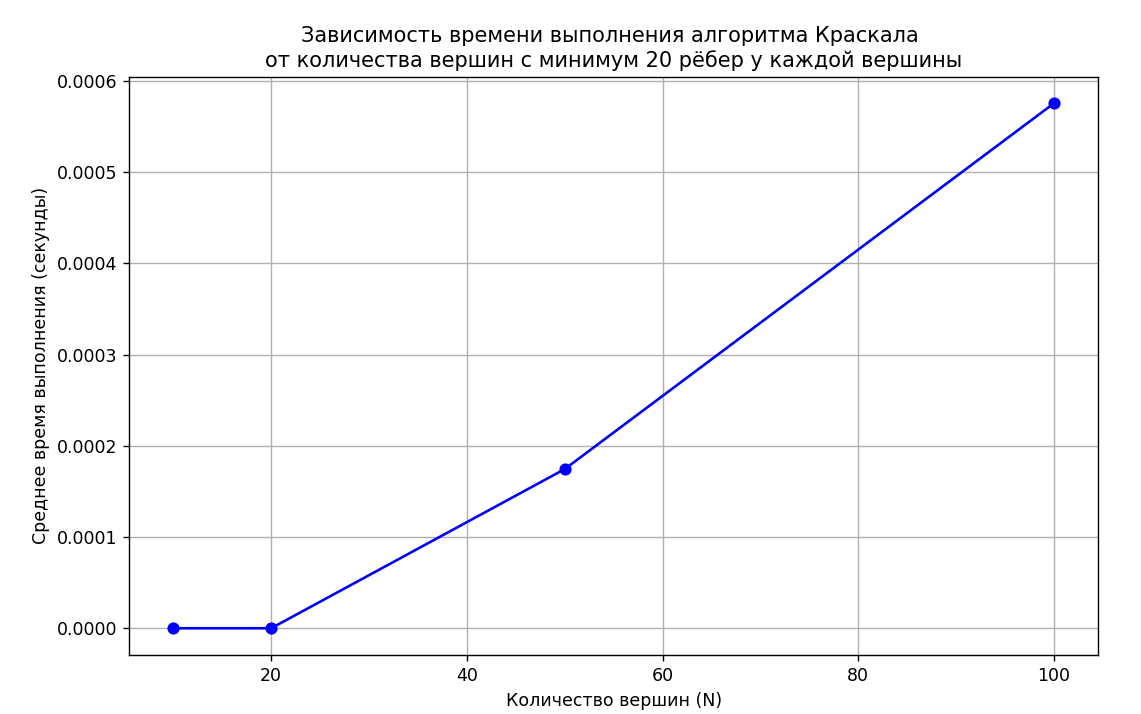
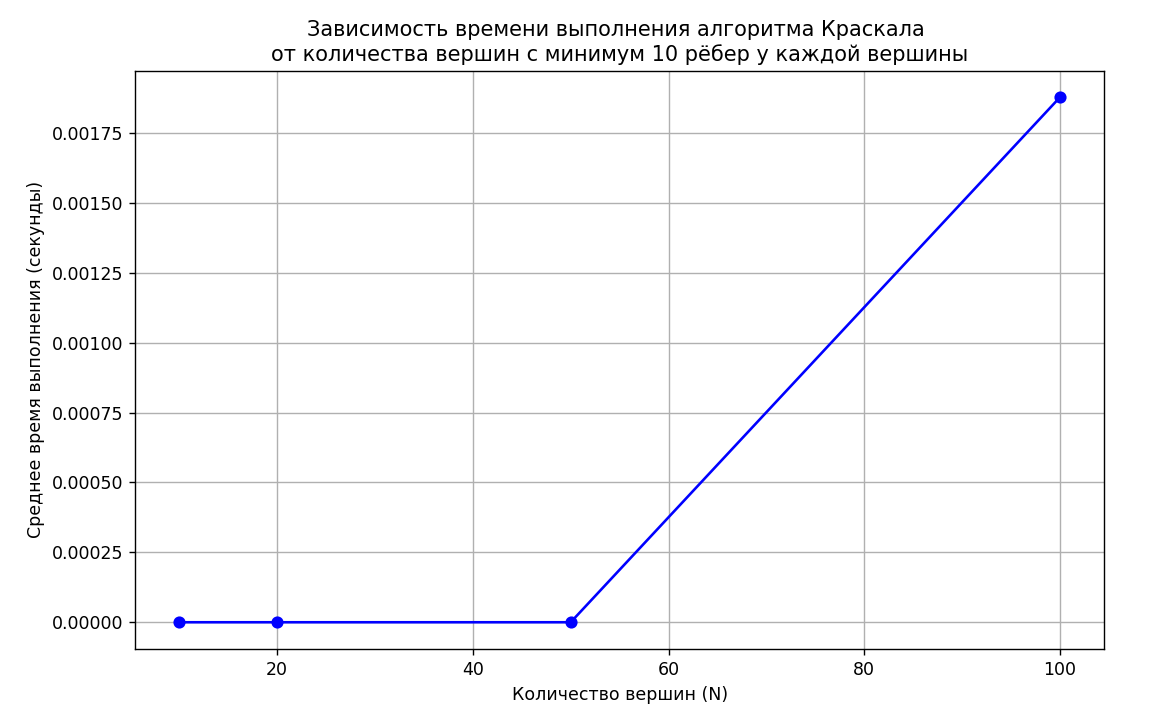
Тест 8: Время выполнения = 0.000000 секунд

Тест 9: Время выполнения = 0.000000 секунд

Тест 10: Время выполнения = 0.001001 секунд

Среднее время выполнения для 100 вершин: 0.000601 секунд





# Заключение.

Метод Краскала работает очень быстро, Но замедляется при большом количестве рёбер