Лабораторная работа № 9

Варианты

ФИО	Дата защиты	№ Заданий
Азатжонова Мехринисо Азизжон Кизи	10.05	1, 2, 3
Аплеев Дмитрий Артурович	10.05	2, 7, 8
Белисов Глеб Андреевич	10.05	2, 3, 4
Бородин Максим Андреевич	10.05	7, 3, 8
Букинов Даниил Дмитриевич	10.05	1, 4, 5
Вабищевич Андрей Владимирович	10.05	4, 6, 8
Вечеренко Анастасия Дмитриевна	10.05	8, 4, 7
Галимзянов Иван Фанилевич	10.05	3, 1, 16
Данилов Назар Олегович	10.05	8, 12, 1
Дашкевич Олеся Сергеевна	10.05	12, 8, 5
Дмитриев Никита Антонович	10.05	2, 13, 14
Забродский Александр Владимирович	10.05	12, 1, 13
Захватова Лада Евгеньевна	24.05	6, 9, 12
Корчагин Роман Павлович	24.05	9, 13, 2
Крестьянова Елизавета Федоровна	24.05	3, 8, 9
Лазуренко Александр Витальевич	24.05	5, 9, 13
Лактионова Елизавета Артемовна	24.05	2, 9, 10
Маркозубова Анастасия Кирилловна	24.05	14, 10, 7
Озеров Антон Александрович	24.05	9, 5, 10
Рожин Данил Александрович	24.05	5, 7, 9
Савальский Матвей Иванович	24.05	11, 7, 12

Селезнев Иван Александрович	24.05	15, 11, 8
Титов Георгий Константинович	24.05	1, 12, 13
Усольцева Дарья Дмитриевна	24.05	10, 14, 3
Черняховский Илья Сергеевич	24.05	2, 5, 8
Арройо Ариас Николас Рафаэль	17.05	4, 10, 12
Борисевич Анна Валериевна	17.05	1, 4, 7
Буцкий Даниил Павлович	17.05	2, 11, 12
Голованов Дмитрий Игоревич	17.05	15, 4, 1
Дармороз Максим Дмитриевич	17.05	7, 10, 13
Зенин Данил Дмитриевич	17.05	3, 10, 11
Зубов Алексей Андреевичч	17.05	10, 14, 3
Иванов Семён Алексеевич	17.05	1, 6, 7
Игнатьев Алексей Юрьевич	17.05	9, 13, 2
Катунина Полина Михайловна	17.05	5, 11, 13
Килебе Нтангу Дьевина Фришели	17.05	14, 3, 15
Клещенко Ольга Дмитриевна	17.05	8, 12, 1
Лапшина Юлия Сергеевна	17.05	10, 6, 11
Маатук Айхам	17.05	3, 6, 7
Можаев Илья Сергеевич	17.05	7, 11, 15
Неустроев Сергей Алексеевич	17.05	5, 15, 1
Новиков Николай Викторович	17.05	6, 10, 14
Олойеде Абдуллах Олувасеун	17.05	1, 10, 11
Олойеде Джибрил Болуватифе	17.05	4, 8, 12
Опалев Владимир Константинович	17.05	11, 15, 4
Орлова Алёна Александровна	31.05	3, 7, 11

Панин Дмитрий Владимирович	31.05	13, 9, 6
Панина Анна Сергеевна	31.05	1, 14, 15
Родионов Анатолий Александрович	31.05	1, 5, 9
Румянцев Михаил Владимирович	31.05	6, 2, 5
Сергеева Агата Сергеевна	31.05	3, 4, 5
Смирнов Георгий Валерьевич	31.05	5, 8, 11
Сомов Дмитрий Антонович	31.05	6, 10, 14
Стафеев Иван Алексеевич	31.05	3, 6, 9
Тимаков Егор Павлович	31.05	2, 15, 16
Умралиева Надежда Рафаэльевна	31.05	9, 12, 15
Фалькович Максим Сергеевич	31.05	2, 6, 10
Федореева Софья Игоревна	31.05	3, 12, 13
Феофанов Никита Романович	31.05	8, 11, 14
Филиппов Леонид Викторович	31.05	3, 14, 15
Ходакова Мария Александровна	31.05	13, 2, 14
Цырульников Артём Алексеевич	31.05	1, 8, 9
Чепля Владислав Павлович	31.05	4, 14, 2
Шарыпов Егор Антонович	31.05	4, 7, 10
Шмайлова Ольга Вадимовна	31.05	2, 5, 6
Яковлев Игорь Сергеевич	31.05	7, 11, 15

Задание 1

Напишите программу, которая реализует сортировку массива целых чисел методом поразрядной сортировки (Radix Sort). Для этого требуется создать функции для выделения цифры из числа, создания массива подсчета и собственно реализации сортировки. Программа должна быть способна сортировать массив чисел любой длины и содержащих произвольное количество разрядов.

Также требуется обосновать выбор данного метода сортировки для конкретной задачи и провести анализ сложности алгоритма в лучшем, худшем и среднем случае.

Подготовьте набор тестовых данных, на которых будет демонстрироваться работа программы, включая тесты на случай пустого массива, массива с одним числом, отсортированного и неотсортированного массивов.

Задание 2

Напишите программу на языке программирования, которая реализует сортировку массива строк методом сортировки подсчетом (Counting Sort). Ваша программа должна учитывать регистр символов при сортировке.

Задание также включает в себя реализацию юнит-тестов для проверки корректности работы алгоритма. Напишите несколько тестовых случаев, включающих в себя сценарии с массивом строк разных длин, строками на разных языках, а также пустым массивом.

Дополнительным требованием к программе является возможность выбора направления сортировки (по возрастанию или убыванию) через входной параметр.

Для успешного выполнения задания необходимо также провести анализ временной сложности алгоритма сортировки подсчетом в зависимости от количества элементов в массиве и объема алфавита.

Задание 3

Напишите программу на языке программирования, которая реализует гибридный алгоритм сортировки, объединяющий два различных метода сортировки: сортировку вставками (Insertion Sort) и сортировку слиянием (Merge Sort). Алгоритм должен автоматически выбирать метод сортировки в зависимости от размера подаваемого массива: для маленьких массивов использовать сортировку вставками, для больших - сортировку слиянием.

Для усложнения задания, добавьте возможность сравнивать производительность вашего гибридного алгоритма с другими известными алгоритмами сортировки, такими как быстрая сортировка (Quick Sort) и пирамидальная сортировка (Heap Sort). Реализуйте замер времени выполнения сортировки на различных входных данных и сделайте анализ эффективности каждого алгоритма в зависимости от размера массива и степени его отсортированности.

Также необходимо написать дополнительные тесты для проверки корректности работы алгоритма в различных случаях, включая случаи с большими входными данными и случаи, когда массив уже отсортирован или отсортирован в обратном порядке.

Задание 4

Представьте, что у вас есть компания, которая занимается доставкой товаров клиентам. У вас есть список клиентов, каждый из которых заказал определенное количество товаров. Ваша задача - организовать доставку товаров максимального числа клиентов, используя жадный алгоритм.

Дано:

- Список клиентов с указанием количества товаров, которые они заказали.

- Вместимость вашего транспортного средства, т.е. максимальное количество товаров, которое вы можете доставить за один рейс.

Ваша задача:

Напишите программу, которая будет принимать на вход список клиентов с их заказами и вместимость транспортного средства. Затем программа должна использовать жадный алгоритм для определения оптимальной последовательности доставки клиентов, чтобы максимизировать число клиентов, которые могут быть обслужены за один рейс.

Для усложнения задания, добавьте возможность работы с различными вариантами вместимости транспортного средства и различными способами определения оптимальной последовательности доставки (например, учитывая как количество товаров, так и расстояние до клиента).

Кроме того, требуется провести анализ эффективности жадного алгоритма в сравнении с другими алгоритмами, такими как динамическое программирование или поиск в глубину, на различных входных данных и с разными параметрами.

Задание 5

Представим, что у вас есть список задач, каждая из которых имеет стоимость выполнения и дедлайн завершения. Ваша цель - выполнить максимальное количество задач, получив максимальную общую стоимость, при условии, что вы можете выполнять только одну задачу за раз. Напишите программу, которая будет принимать на вход список задач с их стоимостями и дедлайнами и определять оптимальный порядок выполнения задач с использованием жадного алгоритма.

Задание:

- Создайте структуру данных для хранения информации о задачах (стоимость выполнения, дедлайн).

- Напишите функцию для сортировки задач по их стоимости (от самой дорогой к самой дешевой).

- Реализуйте жадный алгоритм, который будет выбирать задачи для выполнения в порядке максимизации общей стоимости, учитывая их дедлайны.

- Выведите итоговый порядок выполнения задач и общую стоимость выполненных задач.

Дополнительное усложнение задания:

Добавьте возможность работать с различными ограничениями на количество задач, которые могут быть выполнены за определенный период времени. Попробуйте оценить эффективность жадного алгоритма на различных наборах задач.

Задание 6

Представим, что у вас есть граф, представленный в виде списка ребер, где каждое ребро имеет вес. Ваша задача - найти минимальное остовное дерево этого графа с использованием жадного алгоритма.

Дано:

- Граф в виде списка ребер с указанием веса каждого ребра.

- Граф связный и неориентированный.

Задание:

- Напишите функцию, которая будет принимать на вход граф в виде списка ребер и находить минимальное остовное дерево с использованием алгоритма Прима или Крускала (выберите один из них).

- Реализуйте выбранный жадный алгоритм для поиска минимального остовного дерева.

- Выведите на экран список ребер, входящих в минимальное остовное дерево, и их общий вес.

Дополнительное усложнение задания:

Добавьте возможность работы с ориентированными графами и обработку возможных циклов при построении остовного дерева. Попробуйте оценить алгоритм на различных сценариях графов с разными структурами и весами ребер.

Задание 7

Представим, что у вас есть большой набор данных, представленных в виде несортированного массива целых чисел. Ваша задача - разработать программу, которая будет эффективно находить "тройку" чисел в этом массиве, сумма которых равна заданному числу target. Требуется найти все уникальные тройки чисел, чья сумма равна target.

Задание:

- Напишите функцию, которая будет принимать на вход массив целых чисел и целевое число target, затем находить и выводить список уникальных троек чисел, сумма которых равна target.

- Реализуйте алгоритм, который найдет все возможные комбинации троек чисел из массива, сумма которых равна target. Обратите внимание, что тройки должны быть уникальными, то есть не должны повторяться.

- Учтите эффективность вашего алгоритма, особенно при работе с большими массивами данных.

Дополнительное усложнение задания:

Добавьте возможность обрабатывать массив данных с дубликатами и дать студентам задание на оптимизацию алгоритма для работы с такими сценариями.

Задание 8

Представим, что у вас есть текстовый файл размером в несколько гигабайт, содержащий огромное количество слов. Ваша задача - разработать программу,

которая эффективно найдет и выведет наиболее часто встречающиеся слова в этом файле, учитывая регистр символов и знаки препинания.

Задание:

- Напишите программу, которая будет обрабатывать текстовый файл по словам, считая количество вхождений каждого слова.

- Реализуйте алгоритм, который находит наиболее часто встречающиеся слова в файле.

- Обработка файла должна быть эффективной и не занимать слишком много памяти.

- Учтите специфику языка программирования, который будете использовать, для обработки больших объемов данных.

Дополнительное усложнение задания:

Добавьте возможность игнорировать стоп-слова (например, артикли, предлоги и т. д.) при подсчете частоты слов.

Задание 9

Представим, что у вас есть набор данных, представленный в виде графа с вершинами и ребрами. Каждая вершина имеет уникальный идентификатор и вес (число). Ваша задача - разработать программу, которая найдет кратчайший путь между двумя заданными вершинами в графе, учитывая веса ребер.

Задание:

- Напишите алгоритм для поиска кратчайшего пути между двумя вершинами в графе с учетом весов ребер.

- Учтите возможные циклы в графе и обработайте такие случаи в алгоритме.

- Реализуйте любой метод обхода графа (например, метод Дейкстры) подходящий для данной задачи.

- Протестируйте работу программы на различных наборах данных, чтобы убедиться в её корректности и эффективности.

Дополнительное усложнение задания:

Добавьте возможность работы с ориентированным графом и учитывайте направленность ребер при поиске кратчайшего пути.

Такое задание позволит студентам углубленно изучить работу алгоритмов поиска кратчайшего пути в графах, а также применить полученные знания в практической задаче нахождения оптимального маршрута в сложной сети вершин и ребер.

Задание 10

Представим, что у вас есть набор данных, представленный в виде массива целых чисел. Ваша задача - разработать программу, которая эффективно найдет подмассив (непрерывную последовательность элементов), сумма элементов которого минимально больше заданного порога.

Задание:

- Напишите алгоритм, который найдет такой подмассив с минимальной суммой больше заданного значения порога.

- Реализуйте алгоритм с временной сложностью O(n), где n - количество элементов в массиве.

- Учтите случаи, когда существует несколько подмассивов с одинаковой минимальной суммой больше порога.

- Протестируйте программу на различных входных данных, включая случаи с разными значениями порога и разной длиной массива.

Сложность задачи заключается в том, чтобы эффективно найти и вернуть подмассив с минимальной суммой элементов, которая превышает заданный порог, при этом обеспечивая оптимальное время выполнения алгоритма и эффективное использование памяти.

Задание 11

Представим, что у вас есть набор данных, представленный в виде неориентированного графа с вершинами и взвешенными ребрами. Некоторые ребра в графе являются дополнительными путями, которые можно использовать только определенное количество раз. Ваша задача - разработать программу, которая найдет самый короткий путь между двумя заданными вершинами, учитывая веса ребер и ограничения на количество использований дополнительных путей.

Задание:

- Напишите алгоритм, который найдет самый короткий путь между двумя вершинами в графе, учитывая веса ребер и ограничения на количество использований дополнительных путей.

- Учтите, что при прохождении дополнительного пути он может использоваться только определенное количество раз.

- Реализуйте алгоритм таким образом, чтобы он обрабатывал случаи, когда необходимо использовать дополнительные пути для нахождения самого короткого пути.

- Протестируйте программу на различных графах с разными весами ребер и разными ограничениями на использование дополнительных путей.

Это задание потребует от студентов глубокого понимания алгоритмов поиска кратчайшего пути в графе, а также возможности учитывать дополнительную логику обработки ограничений на пути. Такое упражнение поможет студентам развить навыки работы со сложными структурами данных и алгоритмами на практике.

Задание 12

Представим, что у вас есть набор данных, представленный в виде строк, каждая из которых представляет собой набор пар ключ-значение. Ваша задача - разработать программу, которая эффективно обработает эти строки и выполнит операции добавления, удаления и поиска элементов по ключу.

Задание:

- Напишите структуру данных для хранения пар ключ-значение и соответствующие методы для добавления элемента, удаления элемента по ключу и поиска элемента по ключу.

- Реализуйте алгоритм для обработки строк, содержащих операции добавления, удаления и поиска элементов. Для добавления элемента используйте оператор "+", для удаления - оператор "-", для поиска - оператор "?".

- Протестируйте программу на различных наборах данных, включая случаи с повторяющимися ключами и разными ключами.

Это задание позволит студентам применить знания о структурах данных и алгоритмах на практике, чтобы разработать эффективный способ обработки данных в виде пар ключ-значение. Такое упражнение поможет студентам улучшить навыки работы с коллекциями данных и операциями добавления, удаления и поиска элементов.

Задание 13

У вас есть набор данных, представленный в виде массива целых чисел. Ваша задача - разработать программу, которая найдет все подмассивы этого массива, сумма элементов в которых является простым числом.

Задание:

- Напишите функцию, которая принимает на вход массив целых чисел и находит все подмассивы, сумма элементов в которых является простым числом.

- Реализуйте алгоритм, который эффективно находит все такие подмассивы и выводит их на экран.

- Протестируйте программу на различных входных данных, включая массивы с разным количеством элементов и значениями.

Это задание позволит студентам применить знания о работе с массивами, алгоритмах и проверке чисел на простоту. Они смогут разработать эффективный подход к поиску подмассивов с суммой, являющейся простым числом, и научатся применять различные структуры данных для решения задачи.

Задание 14

Задание:

Представьте, что у вас есть набор данных о графе, представленном в виде списка смежности. Каждая вершина графа имеет уникальный идентификатор. Ваша задача - написать программу, которая найдет все пути из одной вершины графа в другую.

Требования:

1. Реализуйте функцию, которая принимает список смежности графа и два уникальных идентификатора вершин - начальную и конечную. Функция должна вернуть все возможные пути из начальной вершины в конечную.

2. Используйте алгоритм обхода графа в ширину (BFS) для нахождения путей. Обеспечьте корректное завершение работы алгоритма даже в случае наличия циклов в графе.

3. Для каждого найденного пути выведите его на экран, представив вершины пути по их уникальным идентификаторам.

4. Протестируйте программу на различных наборах данных, включая графы с разным количеством вершин и ребер.

Это задание позволит студентам углубить свои знания о работе с графами, алгоритмах обхода и структурах данных для хранения графов. Они смогут оценить и применить эффективность алгоритма BFS при поиске путей в графе.

Задание 15

Представьте, что у вас есть набор точек на плоскости, заданных своими координатами (x, y). Ваша задача - написать программу, которая найдет пару точек с минимальным расстоянием между ними.

Требования:

1. Реализуйте функцию, которая принимает список точек на плоскости и находит пару точек с минимальным расстоянием между ними.

2. Для поиска пары ближайших точек используйте алгоритм сканирующей прямой (Sweep Line Algorithm), который позволяет эффективно находить ближайшие точки.

3. Выведите на экран найденную пару точек и их расстояние друг от друга.

4. Протестируйте программу на различных наборах точек, включая случаи с разным количеством и распределением точек на плоскости.

Это задание поможет студентам углубить свои знания о работе с точками на плоскости, алгоритмах для поиска ближайших пар и оптимизации алгоритмов для работы с большими объемами данных. Они смогут применить свои навыки в области алгоритмов и структур данных для решения сложных задач на практике.

Задание 16

Вам нужно реализовать структуру данных "Куча" (Heap) на основе бинарного дерева. Куча должна поддерживать две операции: добавление элемента и извлечение минимального элемента. Ваша задача - написать программу, которая реализует кучу и провести анализ ее временной сложности.

Требования:

1. Реализуйте класс Heap, который будет представлять структуру данных "Куча" на основе бинарного дерева. Куча должна быть реализована как мин-куча, где значение каждого узла меньше или равно значений его потомков.

2. В классе Heap должны быть реализованы методы для добавления элемента в кучу и извлечения минимального элемента.

3. Проведите анализ временной сложности операций добавления и извлечения минимального элемента в куче. Объясните, какие структуры данных (например, бинарное дерево, массив) и алгоритмы (например, "просеивание вниз", "просеивание наверх") используются для эффективной работы с кучей.

4. Продемонстрируйте работу вашей реализации кучи на наборе данных, включающем несколько элементов, и проверьте правильность работы операций добавления и извлечения элементов.