ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

УТВЕРЖДЕНА

Решением Ученого совета
АНО ВО «Центральный университет»
«24» июня 2025 г.
Протокол №2

Рабочая программа дисциплины (модуля)
«Алгоритмы»

Направление подготовки: *02.03.01 Математика и компьютерные науки*

Направленность (профиль) подготовки: *Разработка*

Квалификация (степень) выпускника: *бакалавр*

Форма обучения: *очная*

Срок освоения программы: *4 года*

Год набора: *2025*

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 0215B0C000D1B3FEB44A37EE631E0C27D4
Владелец: Ивашкевич Евгений Васильевич
Действителен: с 13.01.2026 до 13.04.2027

Электронный документ

Москва
2025

Содержание

1. **Краткая характеристика дисциплины (модуля)** 3
2. **Перечень планируемых результатов обучения** 5
3. **Тематический план** 7
4. **Содержание дисциплины (модуля)** 7
5. **Учебно-методическое обеспечение** 8
6. **Материально-техническое обеспечение** 8
7. **Методические и оценочные материалы** 10

1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Алгоритмы» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – бакалавриат по направлению 02.03.01 «Математика и компьютерные науки», профиль «Разработка», утвержденный приказом Министерства науки и высшего образования Российской Федерации № 807 от 23.08.2017 года.

Изучение дисциплины (модуля) «Алгоритмы» формирует у студентов умение разрабатывать и создавать оптимизированные решения, что критически важно для производительности программного обеспечения. Кроме того, эти знания являются основой для дальнейшего изучения более сложных тем в области компьютерных наук и программирования.

Место дисциплины (модуля) в структуре образовательной программы

Настоящая дисциплина (модуль) включена в учебный план по программе подготовки бакалавриата по направлению 02.03.01 «Математика и компьютерные науки», профиль «Разработка» и входит в Блок Факультативные дисциплины.

Дисциплина (модуль) является выборной и доступна для изучения на 1 или 2 курсе в 1, 2, 3 или 4 семестре на выбор.

Цель изучения дисциплины (модуля): заключается в формировании у студентов навыков проектирования и анализа эффективных алгоритмов и структур данных для решения различных задач программирования.

Задачи изучения дисциплины (модуля):

— изучить основные структуры данных и классические алгоритмы сортировки, поиска и обхода графов для понимания их принципов работы и применения;

— освоить методы анализа сложности алгоритмов по времени и памяти, включая асимптотическую оценку, для оценки их эффективности и выбора оптимальных решений;

— применить алгоритмические подходы к решению практических задач, таких как оптимизация, криптография и обработка больших данных, с использованием языков программирования и инструментов разработки.

В результате освоения дисциплины (модуля) обучающийся должен:

знать:

— понятие и принципы анализа алгоритмов: временная и пространственная сложность, основные типы сложности (\(O(1)\), \(O(n)\), \(O(\log n)\), \(O(n^2)\)), их примеры и подходы к оценке (лучший, худший случай, амортизационный анализ);

— базовые алгоритмы поиска и сортировки: бинарный поиск, сортировки вставками, слиянием и кучей, их особенности, требования и сложность;

— рекурсия: её компоненты, ограничения, дерево вызовов, применение в задачах;

— принципы и преимущества динамического программирования: разбиение задач на подзадачи, мемоизация, итеративный подход, примеры на задачах НОП и НВП;

— основы жадных алгоритмов: локальная оптимизация, их применение и ограничения;

— алгоритмы строкового поиска: расстояние Левенштейна, префикс-функция, алгоритм КМП, Рабина-Карпа, их применения и особенности;

— основы хеширования: хеш-функции, хеш-таблицы, методы разрешения коллизий, применение хеширования в задачах;

— теория графов: понятие вершин, рёбер, взвешенных графов, представление графов (матрица и список смежности), основы BFS и DFS, слабая и сильная связность, топологическая сортировка, компоненты сильной связности и конденсация графа;

— алгоритмы поиска кратчайшего пути: Дейкстры и Флойда-Уоршелла, их применение, особенности и оптимизация;

— основы работы с деревьями: структура бинарных деревьев поиска (BST), свойства, операции вставки, удаления, поиска, обходов (pre-order, in-order, post-order), сложность операций;

уметь:

— оценивать временную и пространственную сложность алгоритмов, анализировать их эффективность;

— реализовывать и модифицировать алгоритмы поиска и сортировки (бинарный поиск, сортировки вставками, слиянием, кучей);

— создавать рекурсивные алгоритмы, выявлять избыточные вычисления и оптимизировать их;

— реализовывать алгоритмы динамического программирования (мемоизация, итеративный подход) для задач, связанных с оптимизацией, например, НОП и НВП;

— решать задачи оптимизации с использованием жадных алгоритмов, анализировать их эффективность;

— разрабатывать алгоритмы строкового поиска (КМП, Рабина-Карпа), восстанавливать пути преобразований (например, расстояние Левенштейна);

— использовать хеш-функции для создания хеш-таблиц и решения задач на основе хеширования;

— реализовывать алгоритмы обхода графов (BFS, DFS), находить компоненты связности, топологическую сортировку, компоненты сильной связности (алгоритм Тарьяна);

— использовать алгоритмы поиска кратчайшего пути (Дейкстра, Флойд-Уоршелл) для анализа сетей, маршрутизации, планирования;

— работать с бинарными деревьями поиска (BST): вставка, удаление, поиск, обходы, балансировка;

владеть:

— навыком анализа алгоритмов с точки зрения эффективности и выбора подходящих подходов для оптимизации времени и памяти;

— навыком решения сложных прикладных задач, включая оптимизацию, маршрутизацию, обработку строк и графов;

— навыком оптимизации алгоритмов и структур данных для работы с большими объёмами данных;

— навыком разработки алгоритмов с использованием динамического программирования, жадных методов и методов строкового поиска для прикладных задач;

— навыком адаптации алгоритмов и выбора оптимальных структур данных (графы, деревья, хеш-таблицы) для специфических задач;

— навыком анализа графовых структур и выявления их особенностей, таких как связность, наличие циклов, порядок выполнения задач;

— навыком проектирования и разработки приложений, использующих структуры данных, такие как деревья, графы, хеш-таблицы, для эффективной организации и обработки данных.

2. Перечень планируемых результатов обучения

Компетенции, формируемые в результате освоения дисциплины (модуля) при проведении учебных занятий в форме контактной работы обучающихся с педагогическими работниками Университета и в форме самостоятельной работы обучающихся:

Компетенция	Содержание компетенции	Индикатор компетенции	Перечень планируемых результатов обучения по дисциплине (модулю)
УК-1.	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.1.	Знает методы поиска и анализа информации в области разработки, основные принципы критической оценки источников информации и их релевантности.
		УК-1.2.	Умеет критически оценивать источники информации и синтезировать данные из различных источников для решения задач, применять системный подход к анализу и решению комплексных проблем
		УК-1.3.	Имеет практический опыт работы с современными инструментами и технологиями для обработки информации, формулировании и структурировании задач на основе полученной информации
УК-2.	Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений	УК-2.1.	Знает действующие правовые нормы, регулирующие деятельность в области решения задач, основные методы и подходы к определению круга задач
		УК-2.2.	Умеет определять круг задач в рамках поставленной цели, выбирать оптимальные способы решения задач, учитывая имеющиеся ресурсы и ограничения
		УК-2.3.	Имеет практический опыт применения знаний о правовых нормах и ресурсах в реальных ситуациях, разработки и реализации решений в соответствии с установленными ограничениями
ОПК-1.	Способен находить, формулировать и решать актуальные и значимые проблемы прикладной и компьютерной математики	ОПК-1.1.	Знает основные методы и подходы к решению задач прикладной и компьютерной математики, включая алгоритмы, математическое моделирование и теорию оптимизации, а также современные инструменты и технологии, используемые в этой области
		ОПК-1.2.	Умеет анализировать и формулировать математические задачи, применять

			соответствующие методы и алгоритмы для их решения, а также интерпретировать и представлять результаты в понятной и доступной форме
		ОПК-1.3.	Имеет практический опыт работы над проектами или исследованиями в области прикладной и компьютерной математики, включая участие в конкурсах, олимпиадах или научных публикациях, где были решены актуальные и значимые задачи
ПК-1.	Способен формулировать задачи с математической точностью, обосновывать утверждения строго и анализировать полученные результаты в области математики и компьютерных наук	ПК-1.1.	Знает методы и подходы к формулированию задач, а также основные принципы математического доказательства и анализа результатов.
		ПК-1.2.	Умеет корректно ставить и формулировать математические задачи, применять строгие методы доказательства и анализировать полученные результаты.
		ПК-1.3.	Имеет опыт работы с задачами в области математики и компьютерных наук, включая применение математических методов для решения практических задач
ПК-2.	Способен решать типовые задачи профессиональной деятельности в области разработки, опираясь на информационную и библиографическую культуру, используя информационно-коммуникационные технологии и учитывая основные требования информационной безопасности	ПК-2.1.	Знает основы информационной и библиографической культуры, а также принципы информационной безопасности и применения информационно-коммуникационных технологий в профессиональной деятельности
		ПК-2.2.	Умеет эффективно использовать информационно-коммуникационные технологии для решения стандартных задач профессиональной деятельности, учитывая требования информационной безопасности
		ПК-2.3.	Имеет опыт работы с информационными ресурсами и технологиями в области разработки, включая соблюдение норм информационной безопасности

3. Тематический план

№п/п	Наименование раздела дисциплины (модуля)	Трудоемкость, академические часы				ТКУ (текущий контроль успеваемости)
		Очная форма				
		Аудиторная работа		Контроль	Самостоятельная работа	
		Лекции	Семинары			
1	Сложность, итеративные сортировки и бинарный поиск	2	2		3	Домашнее задание Квиз
2	Рекурсия, рекурсивные сортировки и куча	2	2		3	Домашнее задание
3	Динамическое программирование и задачи о подпоследовательности	2	2		3	Домашнее задание Квиз
4	Жадные алгоритмы и расстояние Левенштейна	2	2		3	Домашнее задание Квиз
5	Поиск строки в тексте	2	2		3	Домашнее задание Квиз
6	Хеширование	2	2		3	Домашнее задание Контесты
7	Графы	2	2		3	Проект
8	Обход графа в глубину. Связность и сортировка	2	2		3	Домашнее задание Контесты
9	Кратчайший путь в графе	2	2		4	Домашнее задание Контесты
10	Деревья	2	2		4	Домашнее задание Контесты
	Зачет			4		Проект
	Итого:	***20***	***20***	***4***	***32***	
Объем дисциплины (модуля) (в ак. ч.)		***76***				
Объем дисциплины (модуля) (в зач. ед.)		***2***				

4. Содержание дисциплины (модуля)

№ п/п	Наименование раздела дисциплины (модуля)	Содержание дисциплины (модуля) по темам
1	Сложность, итеративные сортировки и бинарный поиск	Асимптотическая сложность алгоритмов. Бинарный поиск. Сортировка бинарными вставками
2	Рекурсия, рекурсивные сортировки и куча	Рекурсия. Сортировка слиянием. Куча и сортировка кучей
3	Динамическое программирование и задачи о подпоследовательности	Динамическое программирование. Задачи о подпоследовательностях
4	Жадные алгоритмы и расстояние Левенштейна	Жадные алгоритмы. Расстояние Левенштейна
5	Поиск строки в тексте	Префикс-функция. Алгоритм Кнута-Морриса-Пратта
6	Хеширование	Хеширование. Хеш-таблицы. Алгоритм Рабина-Карпа
7	Графы	Графы. Представление графа. Обход графа в ширину
8	Обход графа в глубину. Связность и сортировка	Обход в глубину. Топологическая сортировка. Связность графов
9	Кратчайший путь в графе	Алгоритм Дейкстры. Алгоритм Флойда-Уоршелла
10	Деревья	Деревья. Операции на BST. Обходы дерева

5. Учебно-методическое обеспечение

Университет располагает полным набором лицензионного и свободно распространяемого программного обеспечения, включая продукты отечественного производства.

Каждый студент в течение всего периода обучения получает индивидуальный неограниченный доступ к электронно-библиотечной системе и электронной информационно-образовательной среде университета. Эти системы предоставляют возможность доступа к ресурсам из любой точки, где есть подключение к сети Интернет, как на территории университета, так и за его пределами.

Студентам обеспечен удаленный доступ к современным профессиональным базам данных и информационным справочным системам.

Основная литература:

1. Меджедович, Д. Алгоритмы и структуры для массивных наборов данных : практическое руководство / Д. Меджедович, Э. Тахирович ; пер. с англ. А. В. Логунова. – Москва : ДМК Пресс, 2024. - 342 с. – ISBN 978-5-93700-250-1. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2205044.

2. Ахмад, И. 40 алгоритмов, которые должен знать каждый программист на Python : практическое руководство / И. Ахмад. - Санкт-Петербург : Питер, 2023. - 368 с. - (Библиотека программиста). - ISBN 978-5-4461-1908-0. - Текст : электронный. - URL: https://znanium.com/catalog/product/2122957.

3. Бхаргава, А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих : пособие / А. Бхаргава. - Санкт-Петербург : Питер, 2021. - 288 с. - (Серия «Библиотека программиста»). - ISBN 978-5-4461-0923-4. - Текст : электронный. - URL: https://znanium.com/catalog/product/1739631.

4. Рафгарден, Т. Совершенный алгоритм. Жадные алгоритмы и динамическое программирование : практическое руководство / Т. Рафгарден. - Санкт-Петербург : Питер, 2020. - 256 с. - (Серия «Библиотека программиста»). - ISBN 978-5-4461-1445-0. - Текст : электронный. - URL: https://znanium.com/catalog/product/1756121.

5. Протодьяконов, А. В. Алгоритмы Data Science и их практическая реализация на Python : учебное пособие / А. В. Протодьяконов, П. А. Пылов, В. Е. Садовников. - Москва ; Вологда : Инфра-Инженерия, 2022. - 392 с. - ISBN 978-5-9729-1006-9. - Текст : электронный. - URL: https://znanium.com/catalog/product/1902689.

Дополнительная литература:

1. Колдаев, В. Д. Структуры и алгоритмы обработки данных : учебное пособие / В.Д. Колдаев. — Москва : РИОР : ИНФРА-М, 2021. — 296 с. — (Высшее образование: Бакалавриат). — www.dx.doi.org/10.12737/2833. - ISBN 978-5-369-01264-2. - Текст : электронный. - URL: https://znanium.ru/catalog/product/1230215.

2. Палий, И. А. Линейное программирование : учебник для вузов / И. А. Палий. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2025. — 175 с. — (Высшее образование). — ISBN 978-5-534-04716-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/563472.

6. Материально-техническое обеспечение

Университет располагает материально-технической базой, соответствующей действующим противопожарным правилам и нормам и обеспечивающей проведение всех видов дисциплинарной и междисциплинарной подготовки, практической и научно-исследовательской работ обучающихся, предусмотренных учебным планом.

Помещения, которые представляют собой учебные аудитории для проведения занятий лекционного типа, занятий семинарского (практического) типа, групповых и

индивидуальных консультаций, текущего контроля и промежуточной аттестации, а также помещения для самостоятельной работы и помещения для хранения и профилактического обслуживания учебного оборудования. Помещения укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Изучение дисциплины (модуля) обеспечивается в учебных аудиториях, оснащенных:

— столами и стульями;
— компьютерной техникой;
— механическими калькуляторами;
— специализированным оборудованием, включая демонстрационное оборудование.

Помещения для самостоятельной работы обучающихся, в том числе приспособленные для использования инвалидами и лицами с ограниченными возможностями здоровья, оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду Университета.

Обучающимся предоставляется доступ (в том числе удаленный) к ресурсам информационно-телекоммуникационной сети «Интернет», электронным ресурсам (в том числе электронным библиотечным системам, современным профессиональным базам данных и информационным справочным системам):

№	Наименование портала (издания, курса, документа)	Ссылка
1.	Научная электронная библиотека elibrary.ru библиотека	https://elibrary.ru/defaultx.asp
2.	База данных для IT-специалистов	https://habr.com
3.	База данных ScienceDirect	https://www.sciencedirect.com
4.	Официальный сайт Министерства науки и высшего образования Российской Федерации	https://minobrnauki.gov.ru/
5.	Федеральный портал «Российское образование"	https://www.edu.ru/
6.	Информационная система "Единое окно доступа к образовательным ресурсам"	http://window.edu.ru/
7.	Единая коллекция цифровых образовательных ресурсов	http://school-collection.edu.ru/
8.	Федеральный центр информационно - образовательных ресурсов	http://fcior.edu.ru/

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), в том числе комплект лицензионного программного обеспечения, современные профессиональные базы данных и информационные справочные системы:

Наименование ПО	Производство	Лицензионное / свободно распространяемое
Операционные системы:		
Microsoft Imagine (Windows Client, Server)	зарубежное	лицензионное
Браузеры:		
Яндекс.Браузер	отечественное	свободно распространяемое
Google Chrome	зарубежное	свободно распространяемое
Офисные приложения:		
Microsoft Imagine (Visio, OneNote)	зарубежное	лицензионное
TeXstudio	зарубежное	свободно распространяемое
Adobe Acrobat Reader	зарубежное	свободно распространяемое
Программное обеспечение для планирования и учета времени:		

Toggle app	зарубежное	свободно распространяемое
Системы управления проектами:		
Microsoft Imagine (Project)	зарубежное	лицензионное
Системы управления базами данных:		
Microsoft Imagine (SQL Server)	зарубежное	лицензионное
Системы резервного копирования (backup):		
Acronis Backup Advanced for HyperV	зарубежное	лицензионное
Справочно-правовые системы:		
КонсультантПлюс: справочно-правовая система	отечественное	лицензионное
Средства антивирусной защиты:		
Kaspersky Endpoint Security для бизнеса Стандартный Russian Edition	отечественное	лицензионное
Среды разработки:		
Visual Studio Code	зарубежное	свободно распространяемое
Bash (Unix shell)	зарубежное	свободно распространяемое
Anaconda	зарубежное	свободно распространяемое
Robotic Operating System	зарубежное	свободно распространяемое
CopelliaSim	зарубежное	свободно распространяемое
Google Colaboratory	зарубежное	свободно распространяемое
Пакеты программных средств и библиотек:		
AutoPsy	зарубежное	свободно распространяемое
Interactive Disassembler (IDA)	зарубежное	свободно распространяемое
Системы управления библиографической информацией:		
Zotero	зарубежное	свободно распространяемое
Сервисы и службы:		
Bind	зарубежное	свободно распространяемое
Docker	зарубежное	свободно распространяемое

7. Методические и оценочные материалы

Методические указания для обучающихся по освоению дисциплины (модуля)

В процессе изучения дисциплины (модуля) «Алгоритмы» в рамках текущего контроля успеваемости используются такие виды учебной работы, как лекция, семинары и домашние задания, контесты, квизы, проекты, а также различные виды самостоятельной работы обучающихся по заданию преподавателя, направленные на развитие навыков профессиональной лексики, закрепление практических профессиональных компетенций, поощрение инициатив.

Лекция – систематическое, последовательное, монологическое изложение преподавателем учебного материала, как правило, теоретического характера.

В процессе лекций рекомендуется вести конспект лекций: кратко и схематично фиксировать основные идеи, выводы и обобщения лекции; выделять важные мысли, ключевые слова и термины. Необходимо отметить вопросы или материалы, которые вызывают затруднения, и попытаться найти ответы в рекомендованной литературе. Если разобраться в материале не удается, следует сформулировать вопрос и задать его преподавателю на консультации или во время семинарского (практического) занятия.

Семинар — это форма учебной деятельности, проводимая в учебном заведении под руководством преподавателя, где студенты активно участвуют в обсуждениях, практических заданиях и других формах взаимодействия.

Для успешной подготовки к семинару рекомендуется заранее ознакомиться с темой занятия и основными материалами, чтобы иметь возможность активно участвовать в

обсуждении. Также полезно подготовить вопросы и идеи для обсуждения, что поможет глубже понять материал и продемонстрировать заинтересованность.

Квиз – это интерактивное тестирование, направленное на проверку знаний и понимания изучаемого материала.

Для успешной подготовки к квизу рекомендуется внимательно изучить основные понятия и методы, изучаемые на курсе. Полезно решать практические задачи и примеры, чтобы закрепить теоретические знания. Также стоит ознакомиться с типичными вопросами и форматами заданий, чтобы лучше подготовиться к тестированию.

Контест – интерактивная платформа с заданиями разного уровня сложности и автоматической проверкой результатов.

Контест позволяет оперативно оценивать усвоение материала и выявлять пробелы в знаниях через тесты и практические задачи. Такой формат способствует регулярной самопроверке и повышает мотивацию к изучению дисциплины.

Проект – исследовательская работа по курсу и презентация результатов.

Для успешной подготовки к проекту рекомендуется: четко определить цели и задачи проекта; составить план работы, разбив проект на этапы с указанием сроков выполнения каждого из них; использовать разнообразные источники информации и инструменты для исследования темы; регулярно проверять прогресс и вносить коррективы в план, если это необходимо.

Домашнее задание – набор задач по темам недели.

При работе над домашними заданиями важно внимательно ознакомиться с требованиями и сроками выполнения. Рекомендуется разбивать задания на этапы, чтобы избежать перегрузки и лучше усвоить материал. Использовать различные источники информации, включая учебники и онлайн-ресурсы, для более глубокого понимания темы.

Самостоятельная работа – работа студентов, направленная на углубленное изучение отдельных тем и вопросов учебной дисциплины (модуля).

В процессе самостоятельной работы студенты взаимодействуют с рекомендованными материалами при минимальном участии преподавателя. Задачи студента включают работу с конспектами лекций (обработка текста), повторное изучение учебных материалов планов и тезисов ответов, изучение дополнительных тем, выполнение учебно-исследовательских заданий и другое.

Система оценивания результатов обучения по дисциплине (модулю)

Критерии получения уровня и оценивания сформированности компетенций по дисциплине (модулю) «Алгоритмы»

Оценивание уровня учебных достижений, обучающихся по дисциплине (модулю), осуществляется в виде текущего контроля успеваемости и промежуточной аттестации.

Промежуточная аттестация по дисциплине (модулю) осуществляется в форме ***зачета***, при этом проводится оценка компетенций, сформированных по дисциплине.

Для оценивания текущего контроля успеваемости и промежуточной аттестации используется десятибалльная шкала оценивания, которая соотносится с традиционной пятибалльной шкалой следующим образом:

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
10	Отлично	Зачтено	Студент полностью владеет знаниями,

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
9	Отлично	Зачтено	изложенными в рабочей программе, и глубоко осмысляет дисциплину. Он самостоятельно и логически последовательно отвечает на все вопросы, акцентируя внимание на наиболее важном. Умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя ключевые моменты и устанавливая причинно-следственные связи. Четко формулирует ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
8	Отлично	Зачтено	
7	Хорошо	Зачтено	Студент обладает знаниями предмета почти в полном объеме рабочей программы и самостоятельно, логически последовательно и всесторонне отвечает на все вопросы, акцентируя внимание на наиболее значимых моментах. Он умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя его ключевые аспекты и устанавливая причинно-следственные связи. Формулирует свои ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные ситуационные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты предмета с практическими задачами.
6	Хорошо	Зачтено	
5	Удовлетворительно	Зачтено	Студент обладает базовыми знаниями по дисциплине (модулю), но испытывает трудности при самостоятельных ответах и использует неточные формулировки. В ходе ответов он допускает ошибки, касающиеся сути вопросов. Студент
4	Удовлетворительно	Зачтено	

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
			способен решать только самые простые задачи и владеет лишь минимальным набором методов исследования.
3	Не сдан	Не зачтено	Студент не овладел обязательным минимумом знаний по предмету и не может ответить на вопросы, даже если преподаватель задает дополнительные наводящие вопросы.
2	Не сдан	Не зачтено	
1	Не сдан	Не зачтено	

Дисциплина (модуль) «Алгоритмы» оценивается следующим образом:

Активность	Вес	Описание
Домашние задания	40%	Набор задач по темам недели
Проекты	40%	Исследовательская работа по курсу и презентация результатов
Контесты	15%	Интерактивная платформа с заданиями разного уровня сложности и автоматической проверкой результатов
Квизы	5%	Интерактивное тестирование, направленное на проверку знаний и понимания изучаемого материала

Формула расчёта итоговой оценки по дисциплине (модулю) «Алгоритмы»: «0,4 х Домашние задания + 0,4 х среднее за проекты +0,15 х среднее за контесты + 0,05 х квизы».

Текущий контроль успеваемости обучающихся по дисциплине (модулю)

Примерные задания для контеста

Контест 1: Обход графа в глубину, связность, топологическая сортировка и кратчайшие пути в графе

Легкий уровень (4 задания)

1. **Тестовое задание:** Что означает "обход в глубину" (DFS) в графе?
   A) Посещение всех вершин в порядке возрастания номеров
   B) Исследование как можно глубже вдоль каждой ветви перед возвратом
   C) Посещение вершин в ширину
   D) Сортировка ребер по весам
   Правильный ответ: B) Исследование как можно глубже вдоль каждой ветви перед возвратом
   Объяснение: DFS использует стек для погружения в глубину, что позволяет исследовать ветви последовательно.
2. **Открытое задание:** Опишите, как работает обход в глубину (DFS) на простом примере графа с 3 вершинами и 2 ребрами. Укажите порядок посещения вершин.
   Правильный ответ: Предположим граф: вершины A, B, C; ребра A-B, B-C. Начинаем с A: посещаем A, затем B (глубже), затем C. Порядок: A → B → C.
   Объяснение: DFS следует по ребрам как можно глубже, используя рекурсию или стек.
3. **Тестовое задание:** Какова временная сложность DFS для графа с n вершинами и m ребрами?
   A) O(1)
   B) O(n)

C) $O(n + m)$
D) $O(n^2)$
Правильный ответ: C) $O(n + m)$
Объяснение: DFS посещает каждую вершину и ребро ровно один раз, что дает линейную сложность.

4. **Открытое задание:** Что такое связный граф? Приведите пример несвязного графа и объясните, как его сделать связным.
Правильный ответ: Связный граф — это граф, где есть путь между любой парой вершин. Пример несвязного: две отдельные вершины без ребер. Чтобы сделать связным, добавить ребро между ними.
Объяснение: В связном графе все вершины достижимы; несвязный имеет компоненты.

Средний уровень (4 задания)

5. **Тестовое задание:** Для каких графов подходит алгоритм Дейкстры?
A) С отрицательными весами ребер
B) С неотрицательными весами ребер
C) Только с циклами
D) Только с одной вершиной
Правильный ответ: B) С неотрицательными весами ребер
Объяснение: Дейкстра требует неотрицательных весов, чтобы жадный выбор работал корректно.
6. **Открытое задание:** Объясните, что произойдет в DFS для графа с циклом, если не отслеживать посещенные вершины. Приведите пример.
Правильный ответ: Возникнет бесконечный цикл. Пример: граф с вершинами A-B-C-A (цикл). Без маркировки DFS будет повторно заходить в цикл.
Объяснение: Маркировка посещенных предотвращает повторные посещения и циклы.
7. **Тестовое задание:** Топологическая сортировка применяется к:
A) Несвязным графам
B) Направленным ациклическим графам (DAG)
C) Полным графам
D) Графам с циклами
Правильный ответ: B) Направленным ациклическим графам (DAG)
Объяснение: Топологическая сортировка упорядочивает вершины без циклов, сохраняя зависимости.
8. **Открытое задание:** Как определить число компонент связности в неориентированном графе? Опишите алгоритм на примере графа с 4 вершинами: A-B, C-D.
Правильный ответ: Запустить DFS или BFS от каждой непосещенной вершины и подсчитать запуски. В примере: 2 компоненты (A-B и C-D).
Объяснение: Каждый запуск соответствует компоненте; алгоритм маркирует все достижимые вершины.

Сложный уровень (4 задания)

9. **Тестовое задание:** Что такое "релаксация" в алгоритме Дейкстры?
A) Удаление ребра
B) Проверка и обновление расстояния до вершины через соседа
C) Сортировка вершин
D) Построение кучи
Правильный ответ: B) Проверка и обновление расстояния до вершины через соседа

Объяснение: Релаксация обновляет dist[v] = min(dist[v], dist[u] + w(u,v)), если путь короче.

10. **Открытое задание:** Почему алгоритм Дейкстры не работает с отрицательными весами? Приведите контрпример.
    Правильный ответ: Жадный выбор может выбрать неправильный путь, пропустив лучшие с отрицательными ребрами. Контрпример: вершины A-B-C, ребра A-B (1), B-C (1), A-C (-2). Дейкстра выберет A-B-C (2), но правильный A-C (-2).
    Объяснение: Отрицательные веса нарушают инвариант, что кратчайший путь к выбранной вершине найден.
11. **Тестовое задание:** В топологической сортировке с помощью DFS, порядок добавления вершин в стек — это:
    A) Порядок открытия
    B) Порядок закрытия (после обработки потомков)
    C) Случайный порядок
    D) Порядок по степеням
    Правильный ответ: B) Порядок закрытия (после обработки потомков)
    Объяснение: Стек заполняется при возврате, и его разворот дает топологический порядок.
12. **Открытое задание:** Алгоритм Флойда-Уоршелла находит кратчайшие пути между всеми парами вершин. Опишите его основную идею и сложность.
    Правильный ответ: Использует динамическое программирование: для каждой промежуточной вершины k обновляет пути i-j через k. Сложность $O(n^3)$.
    Объяснение: Матрица расстояний обновляется итеративно, учитывая все возможные пути.

Контест 2: Деревья (деревья, операции на BST, обходы дерева)

Легкий уровень (4 задания)

1. **Тестовое задание:** Что такое дерево в теории графов?
   A) Граф с циклами
   B) Связный ациклический граф
   C) Граф с одинаковым числом вершин и ребер
   D) Граф только с направленными ребрами
   Правильный ответ: B) Связный ациклический граф
   Объяснение: Дерево — это ациклический связный граф.
2. **Открытое задание:** Опишите, что такое корень дерева и лист. Приведите простой пример дерева.
   Правильный ответ: Корень — вершина без родителей. Лист — вершина без детей. Пример: дерево с корнем A, детьми B и C, где B и C — листья.
   Объяснение: Корень — точка входа; листья — конечные вершины.
3. **Тестовое задание:** Какой обход дерева посещает корень первым?
   A) In-order
   B) Post-order
   C) Pre-order
   D) Level-order
   Правильный ответ: C) Pre-order
   Объяснение: Pre-order: корень, левое, правое.
4. **Открытое задание:** Что такое высота дерева? Как она рассчитывается для дерева с корнем и двумя детьми?
   Правильный ответ: Высота — максимальное расстояние от корня до листа. Для дерева с корнем и двумя детьми-листьями высота 1.
   Объяснение: Расстояние считается по ребрам; листья на уровне 1.

Средний уровень (4 задания)

5. **Тестовое задание:** В BST (бинарном дереве поиска), где находится элемент, меньший корня?
   A) В правом поддереве
   B) В левом поддереве
   C) В корне
   D) В любом поддереве
   Правильный ответ: B) В левом поддереве
   Объяснение: Левое поддерево содержит меньшие, правое — большие значения.
6. **Открытое задание:** Опишите операцию вставки в BST. Приведите пример вставки числа 5 в дерево с корнем 3.
   Правильный ответ: Сравниваем с корнем: если меньше, идем влево; если больше, вправо. Для 5 > 3: вставляем вправо.
   Объяснение: BST поддерживает порядок для быстрого поиска.
7. **Тестовое задание:** Какой обход дерева дает отсортированный порядок в BST?
   A) Pre-order
   B) Post-order
   C) In-order
   D) Level-order
   Правильный ответ: C) In-order
   Объяснение: In-order в BST посещает в порядке возрастания.
8. **Открытое задание:** Что такое балансировка дерева? Почему она важна? Приведите пример несбалансированного дерева.
   Правильный ответ: Балансировка минимизирует высоту для равномерного распределения. Важна для O(log n) операций. Пример: цепочка из 3 вершин.
   Объяснение: Несбалансированное дерево может деградировать до списка, увеличивая время.

Сложный уровень (4 задания)

9. **Тестовое задание:** Какова временная сложность поиска в худшем случае для несбалансированного BST?
   A) O(1)
   B) O(log n)
   C) O(n)
   D) $O(n^2)$
   Правильный ответ: C) O(n)
   Объяснение: В худшем случае (список) поиск линейный.
10. **Открытое задание:** Опишите алгоритм удаления узла в BST. Приведите пример удаления листа.
    Правильный ответ: Если лист — просто удалить. Если с одним ребенком — заменить на ребенка. Если с двумя — заменить на преемника (минимум правого поддерева). Пример: удалить лист — убрать узел.
    Объяснение: Удаление сохраняет свойства BST.
11. **Тестовое задание:** Что такое AVL-дерево?
    A) Дерево с произвольной высотой
    B) Самобалансирующееся бинарное дерево поиска
    C) Дерево только для чтения
    D) Дерево без корня
    Правильный ответ: B) Самобалансирующееся бинарное дерево поиска
    Объяснение: AVL поддерживает баланс факторов высоты для O(log n).
12. **Открытое задание:** Сравните pre-order и post-order обходы дерева. Приведите пример порядка для дерева с корнем A, левым B, правым C.

Правильный ответ: Pre-order: A, B, C. Post-order: B, C, A. Pre-order посещает корень первым, post-order — последним.
Объяснение: Pre-order: корень перед поддеревьями; post-order — после.

Примерные задания для квизов

Сложность, итеративные сортировки и бинарный поиск (Асимптотическая сложность алгоритмов. Бинарный поиск. Сортировка бинарными вставками)

1. **Вопрос:** Какова асимптотическая сложность бинарного поиска в худшем случае?
   A) O(1)
   B) O(log n)
   C) O(n)
   D) O(n log n)
   Правильный ответ: B) O(log n)
   Объяснение: Бинарный поиск делит массив пополам на каждой итерации, что дает логарифмическую сложность.

2. **Вопрос:** Что является необходимым условием для применения бинарного поиска?
   A) Массив должен быть пустым
   B) Массив должен быть отсортированным
   C) Массив должен содержать только уникальные элементы
   D) Массив должен быть динамическим
   Правильный ответ: B) Массив должен быть отсортированным
   Объяснение: Бинарный поиск требует упорядоченного массива для корректной работы.

3. **Вопрос:** Какова временная сложность сортировки бинарными вставками в худшем случае?
   A) O(n)
   B) O(n log n)
   C) $O(n^2)$
   D) O(log n)
   Правильный ответ: C) $O(n^2)$
   Объяснение: Хотя вставка использует бинарный поиск, общая сложность остается квадратичной из-за сдвигов элементов.

4. **Вопрос:** Что означает нотация O(n log n)?
   A) Линейная сложность
   B) Логарифмическая сложность
   C) Квадратичная сложность
   D) Линейно-логарифмическая сложность
   Правильный ответ: D) Линейно-логарифмическая сложность
   Объяснение: O(n log n) описывает алгоритмы, где время растет пропорционально n умноженному на log n.

5. **Вопрос:** Сколько сравнений может потребоваться в бинарном поиске для массива из 64 элементов в худшем случае?
   A) 6
   B) 32
   C) 64
   D) 128

Правильный ответ: A) 6
Объяснение: 2^6 = 64, так что максимум 6 шагов ($\log_2(64) = 6$).

6. **Вопрос:** В сортировке бинарными вставками бинарный поиск используется для:
   A) Сортировки всего массива
   B) Поиска позиции для вставки элемента
   C) Обмена элементов
   D) Подсчета количества элементов
   Правильный ответ: B) Поиска позиции для вставки элемента
   Объяснение: Бинарный поиск помогает быстро найти место вставки в отсортированной части массива.

7. **Вопрос:** Какова пространственная сложность бинарного поиска?
   A) O(1)
   B) O(log n)
   C) O(n)
   D) $O(n^2)$
   Правильный ответ: A) O(1)
   Объяснение: Бинарный поиск использует постоянное дополнительное пространство.

8. **Вопрос:** Что такое "асимптотическая сложность"?
   A) Точное время выполнения
   B) Поведение алгоритма при больших n
   C) Количество строк кода
   D) Скорость процессора
   Правильный ответ: B) Поведение алгоритма при больших n
   Объяснение: Асимптотика описывает рост времени/памяти с увеличением размера входа.

9. **Вопрос:** Бинарный поиск может быть применен к несортированному массиву?
   A) Да, всегда
   B) Нет, никогда
   C) Только если массив содержит числа
   D) Только для строк
   Правильный ответ: B) Нет, никогда
   Объяснение: Без сортировки бинарный поиск не гарантирует корректный результат.

10. **Вопрос:** В сортировке бинарными вставками, если массив уже отсортирован, сложность будет:
    A) $O(n^2)$
    B) O(n log n)
    C) O(n)
    D) O(log n)
    Правильный ответ: C) O(n)
    Объяснение: В лучшем случае вставка не требует сдвигов, только проверок.

Рекурсия, рекурсивные сортировки и куча (Рекурсия. Сортировка слиянием. Куча и сортировка кучей)

1. **Вопрос:** Что такое рекурсия в программировании?
   A) Цикл с фиксированным числом итераций

B) Функция, вызывающая сама себя
C) Использование внешних библиотек
D) Обработка массивов без циклов
Правильный ответ: B) Функция, вызывающая сама себя
Объяснение: Рекурсия — это техника, где функция решает задачу, разбивая ее на подзадачи того же типа.

2. **Вопрос:** Какова временная сложность сортировки слиянием?
A) O(n)
B) O(n log n)
C) $O(n^2)$
D) O(log n)
Правильный ответ: B) O(n log n)
Объяснение: Сортировка слиянием использует divide-and-conquer, что дает такую сложность.

3. **Вопрос:** Что такое куча (heap) в контексте алгоритмов?
A) Стек вызовов
B) Дерево, где каждый узел <= или >= потомков
C) Линейный массив
D) Хеш-таблица
Правильный ответ: B) Дерево, где каждый узел <= или >= потомков
Объяснение: Куча — это бинарное дерево с heap-свойством (min-heap или max-heap).

4. **Вопрос:** Какой алгоритм сортировки использует кучу?
A) QuickSort
B) MergeSort
C) HeapSort
D) BubbleSort
Правильный ответ: C) HeapSort
Объяснение: HeapSort строит кучу и извлекает элементы по порядку.

5. **Вопрос:** Что происходит при переполнении стека в рекурсивной функции?
A) Программа завершается успешно
B) Возникает ошибка stack overflow
C) Увеличивается скорость выполнения
D) Уменьшается память
Правильный ответ: B) Возникает ошибка stack overflow
Объяснение: Глубокая рекурсия может исчерпать стек вызовов.

6. **Вопрос:** В сортировке слиянием массив делится на:
A) Две равные части
B) Три части
C) На элементы
D) Случайно
Правильный ответ: A) Две равные части
Объяснение: Рекурсивно делится пополам до базового случая.

7. **Вопрос:** Какова пространственная сложность сортировки слиянием?
A) O(1)
B) O(log n)
C) O(n)

D) $O(n^2)$
Правильный ответ: C) O(n)
Объяснение: Требуется дополнительный массив для слияния.

8. **Вопрос:** Что такое базовый случай в рекурсии?
   A) Самый сложный шаг
   B) Условие остановки рекурсии
   C) Вызов другой функции
   D) Цикл внутри функции
   Правильный ответ: B) Условие остановки рекурсии
   Объяснение: Без базового случая рекурсия будет бесконечной.

9. **Вопрос:** В куче корень всегда:
   A) Минимальный элемент (для min-heap)
   B) Максимальный элемент (для max-heap)
   C) Случайный элемент
   D) Последний элемент
   Правильный ответ: A) Минимальный элемент (для min-heap) или B) Максимальный элемент (для max-heap), но в зависимости от типа; стандартно для min-heap — A.
   Объяснение: В min-heap корень минимален, в max-heap — максимален.

10. **Вопрос:** HeapSort является стабильной сортировкой?
    A) Да
    B) Нет
    C) Только для чисел
    D) Только для строк
    Правильный ответ: B) Нет
    Объяснение: HeapSort не сохраняет порядок равных элементов.

Поиск строки в тексте (Префикс-функция. Алгоритм Кнута-Морриса-Пратта)

1. **Вопрос:** Что такое префикс-функция в алгоритме KMP?
   A) Длина текста
   B) Массив длин наибольших префиксов, совпадающих с суффиксами
   C) Количество символов в паттерне
   D) Индекс начала совпадения
   Правильный ответ: B) Массив длин наибольших префиксов, совпадающих с суффиксами
   Объяснение: Префикс-функция помогает пропускать ненужные сравнения.

2. **Вопрос:** Какова временная сложность алгоритма Кнута-Морриса-Пратта?
   A) O(n)
   B) O(m + n)
   C) $O(n^2)$
   D) O(m * n)
   Правильный ответ: B) O(m + n)
   Объяснение: Где m — длина паттерна, n — текста; линейное время.

3. **Вопрос:** Зачем нужна префикс-функция в KMP?
   A) Для сортировки текста
   B) Для определения сдвига при несовпадении
   C) Для хеширования

D) Для сжатия данных
Правильный ответ: B) Для определения сдвига при несовпадении
Объяснение: Она указывает, сколько символов можно пропустить.

4. **Вопрос:** В KMP, если паттерн "ABA", префикс-функция для позиции 2 (B) равна:
   A) 0
   B) 1
   C) 2
   D) 3
   Правильный ответ: A) 0
   Объяснение: Для "AB" префикс "A" не совпадает с суффиксом "B".

5. **Вопрос:** Алгоритм KMP эффективен для:
   A) Поиска в маленьких текстах
   B) Поиска подстроки в больших текстах
   C) Сортировки массивов
   D) Вычисления хешей
   Правильный ответ: B) Поиска подстроки в больших текстах
   Объяснение: Он минимизирует сравнения, работая за O(m + n).

6. **Вопрос:** Что возвращает алгоритм KMP?
   A) Количество совпадений
   B) Индексы всех вхождений паттерна
   C) Хеш текста
   D) Длину текста
   Правильный ответ: B) Индексы всех вхождений паттерна
   Объяснение: KMP находит все позиции, где паттерн встречается в тексте.

7. **Вопрос:** Префикс-функция вычисляется для:
   A) Текста
   B) Паттерна
   C) Обоих
   D) Ни для одного
   Правильный ответ: B) Паттерна
   Объяснение: Префикс-функция строится для паттерна заранее.

8. **Вопрос:** В KMP сдвиг при несовпадении равен:
   A) 1 всегда
   B) Длине паттерна
   C) Значению префикс-функции
   D) Случайному числу
   Правильный ответ: C) Значению префикс-функции
   Объяснение: Сдвиг основан на префикс-функции для избежания лишних сравнений.

9. **Вопрос:** KMP работает хуже наивного поиска при:
   A) Повторяющихся паттернах
   B) Уникальных паттернах
   C) Маленьких текстах
   D) Больших текстах
   Правильный ответ: B) Уникальных паттернах
   Объяснение: На уникальных паттернах преимущество KMP меньше, но все равно эффективен.

10. **Вопрос:** Префикс-функция для паттерна "AAAA" на позиции 3 равна:
    A) 0
    B) 1
    C) 2
    D) 3
    Правильный ответ: D) 3
    Объяснение: Префикс "AAA" совпадает с суффиксом "AAA".

Примерные домашние задания

Домашнее задание: Динамическое программирование и задачи о подпоследовательностях

1. **Тестовое задание:** Что такое динамическое программирование?
   A) Метод решения задач путем полного перебора всех вариантов
   B) Метод разбиения задачи на подзадачи с пересечением решений и их запоминанием
   C) Алгоритм для сортировки массивов
   D) Способ поиска в дереве
   Правильный ответ: B) Метод разбиения задачи на подзадачи с пересечением решений и их запоминанием
   Объяснение: ДП избегает повторных вычислений, используя memoization или табличный подход.

2. **Открытое задание:** Найдите длину наибольшей общей подпоследовательности (LCS) для строк "ABCBDAB" и "BDCABA". Опишите шаги решения с помощью ДП-таблицы.
   Правильный ответ: Длина LCS = 4 (например, "BCBA" или "BDAB"). Создайте таблицу 8x7: dp[i][j] = dp[i-1][j-1] + 1, если символы равны, иначе max(dp[i-1][j], dp[i][j-1]).
   Объяснение: ДП строит таблицу, сравнивая символы и выбирая максимумы.

3. **Тестовое задание:** Какова временная сложность решения задачи LCS для строк длины m и n?
   A) O(m + n)
   B) O(m * n)
   C) O(2^(m+n))
   D) O(min(m, n))
   Правильный ответ: B) O(m * n)
   Объяснение: Таблица ДП имеет размер m x n, и каждая ячейка вычисляется за O(1).

4. **Открытое задание:** Найдите длину наибольшей возрастающей подпоследовательности (LIS) в массиве [10, 22, 9, 33, 21, 50, 41, 60]. Опишите алгоритм ДП.
   Правильный ответ: LIS = 5 (например, 10, 22, 33, 50, 60). Используйте массив dp[i] = max(dp[j] + 1) для j < i, где arr[j] < arr[i].
   Объяснение: ДП вычисляет LIS для каждого элемента, сравнивая с предыдущими.

5. **Открытое задание:** Вычислите редакционное расстояние (расстояние Левенштейна) между строками "kitten" и "sitting". Покажите ДП-таблицу и объясните операции вставки/удаления/замены.
   Правильный ответ: Расстояние = 3 (kitten → sitten → sittin → sitting: 1 замена, 1

вставка, 1 замена). Таблица 7x8: dp[i][j] = min(вставка, удаление, замена).
Объяснение: ДП минимизирует операции для преобразования одной строки в другую.

Домашнее задание: Жадные алгоритмы и расстояние Левенштейна

1. **Тестовое задание:** Что такое жадный алгоритм?
   A) Алгоритм, который всегда выбирает глобально оптимальное решение
   B) Алгоритм, который делает локально оптимальный выбор на каждом шаге, надеясь на глобальный оптимум
   C) Алгоритм для полного перебора
   D) Алгоритм только для сортировки
   Правильный ответ: B) Алгоритм, который делает локально оптимальный выбор на каждом шаге, надеясь на глобальный оптимум
   Объяснение: Жадные алгоритмы не всегда дают оптимальное решение, но часто работают для специфических задач.

2. **Открытое задание:** Решите задачу о размене монет жадным алгоритмом: разменяйте 93 копейки монетами 25, 10, 5, 1 коп. Укажите количество каждой монеты.
   Правильный ответ: 3 монеты по 25 (75), 1 по 10 (18), 3 по 1 (3) = 7 монет. Жадно берем максимальную монету.
   Объяснение: Для канонических систем (как эта) жадный алгоритм оптимален.

3. **Тестовое задание:** Для каких задач жадный алгоритм дает оптимальное решение?
   A) Только для задач с отрицательными весами
   B) Для задач с неотрицательными весами и свойствами жадности (например, задача о рюкзаке 0/1)
   C) Для всех задач
   D) Только для сортировки
   Правильный ответ: B) Для задач с неотрицательными весами и свойствами жадности (например, задача о рюкзаке 0/1) — подождите, исправление: для дробного рюкзака жадный оптимален, для 0/1 — нет. Правильный: для задач вроде планирования интервалов или дробного рюкзака.
   Объяснение: Жадные работают, когда локальный выбор не портит глобальный (например, сортировка по плотности в дробном рюкзаке).

4. **Открытое задание:** Вычислите расстояние Левенштейна между "abc" и "adc" с помощью ДП. Объясните, почему это ДП, а не жадный алгоритм.
   Правильный ответ: Расстояние = 1 (замена b на d). Таблица 4x4: dp[3][3] = 1. Это ДП, потому что учитывает все пути, а не жадный выбор.
   Объяснение: Левенштейн — ДП, так как проверяет все комбинации операций; жадный мог бы ошибиться в сложных случаях.

5. **Открытое задание:** Примените жадный алгоритм к задаче выбора интервалов: даны интервалы [(1,3), (2,4), (3,5), (4,6)]. Выберите максимальное количество непересекающихся.
   Правильный ответ: Выберите (1,3), затем (4,6) — 2 интервала. Сортируйте по концу и жадно выбирайте.
   Объяснение: Жадный выбор по раннему окончанию дает оптимальное решение.

Домашнее задание: Поиск строки в тексте и Хеширование

1. **Тестовое задание:** Что такое префикс-функция в алгоритме КМП?
   A) Массив длин наибольших префиксов, совпадающих с суффиксами
   B) Хеш-значение строки
   C) Количество вхождений подстроки
   D) Сортированный массив символов
   Правильный ответ: A) Массив длин наибольших префиксов, совпадающих с суффиксами
   Объяснение: Префикс-функция помогает пропускать символы при несовпадении.

2. **Открытое задание:** Постройте префикс-функцию для строки "ABABAB". Укажите значения для каждого префикса.
   Правильный ответ: π = [0, 0, 1, 2, 3, 4]. Для "ABA" π[2]=1 (A), "ABAB" π[3]=2 (AB), и т.д.
   Объяснение: Вычисляется итеративно: если символы равны, увеличиваем, иначе сдвигаем по предыдущему значению.

3. **Тестовое задание:** Какова сложность алгоритма Рабина-Карпа для поиска подстроки длины m в тексте длины n?
   A) O(n)
   B) O(m + n)
   C) O(m * n)
   D) O(n log n)
   Правильный ответ: B) O(m + n) — в среднем, с хешированием; в худшем O(m*n) при коллизиях.
   Объяснение: Хеширование позволяет быстрый сдвиг окна, но коллизии могут потребовать проверки.

4. **Открытое задание:** Используйте алгоритм КМП для поиска "AB" в "AABAB". Покажите процесс и позиции вхождений.
   Правильный ответ: Вхождения на позициях 1 (AA) и 3 (AB). Префикс-функция для "AB": [0,0]. Сдвиг по π при несовпадении.
   Объяснение: КМП строит автомат для эффективного поиска без повторных сравнений.

5. **Открытое задание:** Реализуйте простую хеш-таблицу для хранения строк ["apple", "banana", "cherry"] с разрешением коллизий методом цепочек. Вставьте "date" и найдите "banana".
   Правильный ответ: Хеш-функция: например, сумма кодов ASCII. Цепочки: вставка в список по индексу. "banana" найдена в цепочке.
   Объяснение: Хеш-таблицы обеспечивают O(1) доступ в среднем, цепочки разрешают коллизии.

Примерное описание для проектов

Проект по теме "Динамическое программирование и задачи о подпоследовательностях"

Название проекта: "Динамические подпоследовательности: от теории к практике"
Описание проекта:

В рамках этого проекта студенты разработают программное решение (например, на Python или C++), реализующее ключевые алгоритмы динамического программирования (ДП) для задач о подпоследовательностях. Проект включает анализ и реализацию

наибольшей общей подпоследовательности (LCS), наибольшей возрастающей подпоследовательности (LIS) и редакционного расстояния (расстояние Левенштейна). Студенты должны не только написать код, но и объяснить принципы ДП, включая табличный подход и memoization, на основе примеров из домашних заданий. Проект может быть выполнен индивидуально или в паре, с акцентом на практическое применение теории для решения реальных задач (например, сравнение текстов или анализ последовательностей данных).

Цель проекта:

Освоить основы динамического программирования на примерах задач о подпоследовательностях, таких как LCS, LIS и редакционное расстояние, путем реализации алгоритмов, их тестирования и анализа эффективности. Проект поможет студентам понять, как ДП разбивает задачи на подзадачи с пересечением решений, избегая повторных вычислений, и применить это в программировании.

Задачи проекта:

1. Изучить теоретические основы ДП: объяснить, что такое ДП (метод разбиения на подзадачи с запоминанием решений), и сравнить с другими подходами (например, полный перебор).
2. Реализовать алгоритмы ДП для трех задач: LCS (с построением ДП-таблицы), LIS (с массивом dp) и редакционное расстояние (с таблицей операций вставки/удаления/замены).
3. Протестировать реализации на примерах из домашних заданий (например, LCS для "ABCBDAB" и "BDCABA", LIS для [10, 22, 9, 33, 21, 50, 41, 60], расстояние Левенштейна для "kitten" и "sitting").
4. Проанализировать временную и пространственную сложность (например, O(m*n) для LCS) и оптимизировать код (например, использовать одномерный массив для LIS).
5. Создать отчет с примерами работы, диаграммами ДП-таблиц и выводами о преимуществах ДП над наивными методами.

Этапы выполнения проекта:

1. **Подготовительный этап (1-2 недели):** Изучите теорию ДП из лекций и домашних заданий. Решите тестовые и открытые задания вручную, чтобы понять алгоритмы. Выберите язык программирования и инструменты (например, Jupyter Notebook для Python).
2. **Реализационный этап (2-3 недели):** Напишите код для каждого алгоритма. Начните с LCS (с таблицей dp), затем LIS (с dp-массивом), и закончите редакционным расстоянием (с min-операциями). Добавьте функции для вывода таблиц и результатов.
3. **Тестирование и отладка (1 неделя):** Протестируйте на примерах, проверьте корректность (например, LCS должна дать длину 4). Измерьте время выполнения для больших входов (например, строки длиной 100 символов) и исправьте ошибки.
4. **Анализ и документация (1 неделя):** Напишите отчет: опишите алгоритмы, покажите таблицы, объясните сложность и преимущества ДП. Подготовьте презентацию (5-10 минут) с демонстрацией кода.
5. **Защита проекта:** Представьте проект преподавателю, ответьте на вопросы о ДП и коде.

Критерии защиты и оценки:

Проект оценивается по шкале 0-10 баллов.

• **Корректность реализации (3 балла):** Код работает правильно на всех примерах (LCS=4, LIS=5, расстояние=3); нет ошибок в логике ДП.

• **Эффективность и оптимизация (2 балла):** Правильная сложность (O(m*n)), использование memoization/таблиц; код оптимизирован (например, не использует лишнюю память).

• **Анализ и документация (2 балла):** Отчет включает объяснения, таблицы, сравнение с наивными методами; код прокомментирован.

• **Тестирование и демонстрация (1,5 балла):** Полные тесты, измерение производительности; успешная защита с ответами на вопросы.

• **Креативность и глубина (1,5 балла):** Добавлены расширения (например, восстановление подпоследовательности или сравнение с другими алгоритмами); проект идет дальше базовых требований.

Максимальный балл за полный, аккуратный проект с презентацией. Минимальный проходной — 6 баллов.

Задания для промежуточной аттестации по дисциплине (модулю)

№ п/п	Задание	Ответ	Компетенция
1	Укажите обозначение для асимптотической сложности алгоритма в худшем случае.	O (или O-большое, O-нотация)	УК-1
2	Укажите термин для функции, которая вызывает сама себя в алгоритмах.	Рекурсия	УК-1
3	Укажите название метода решения задач путем разбиения на подзадачи с хранением промежуточных результатов.	Динамическое программирование	УК-1
4	Укажите стратегию выбора локально оптимального решения на каждом шаге в алгоритмах.	Жадный алгоритм (или жадная стратегия)	УК-1
5	Укажите алгоритм сортировки, подходящий для больших массивов при ограниченной памяти.	Сортировка слиянием	УК-2
6	Укажите метод обхода графа, минимизирующий использование памяти.	Обход в ширину (или BFS)	УК-2
7	Укажите подход к разрешению коллизий в хеш-таблицах.	Открытое хеширование (или цепочки)	УК-2
8	Укажите структуру данных для быстрого поиска в отсортированных элементах.	Бинарное дерево поиска (или BST)	УК-2
9	Укажите асимптотическую сложность сортировки кучей.	O(n log n) (или O(n*log n))	ОПК-1
10	Укажите метод вычисления редакционного расстояния между строками.	Динамическое программирование	ОПК-1
11	Укажите структуру данных, используемую в алгоритме Кнута-Морриса-Пратта для поиска подстроки.	Префикс-функция	ОПК-1
12	Укажите принцип работы алгоритма Дейкстры для нахождения кратчайшего пути.	Жадный алгоритм	ОПК-1
13	Укажите аббревиатуру задачи о наибольшей общей подпоследовательности.	LCS (или Longest Common Subsequence)	ПК-1
14	Укажите тип графа, допускающий топологическую сортировку.	DAG (или Directed Acyclic Graph)	ПК-1

15	Укажите метод нахождения кратчайших путей между всеми парами вершин в графе.	Алгоритм Флойда-Уоршелла (или динамическое программирование)	ПК-1
16	Укажите операцию добавления элемента в бинарное дерево поиска.	Вставка	ПК-1
17	Укажите способ хранения графа с использованием массивов смежных вершин.	Список смежности	ПК-2
18	Укажите алгоритм поиска подстроки в тексте с использованием хеш-функций.	Рабина-Карпа	ПК-2
19	Укажите метод обхода графа для обнаружения циклов.	Обход в глубину (или DFS)	ПК-2
20	Укажите тип обхода дерева, приводящий к сортированному порядку элементов.	In-order (или прямой, обратный)	ПК-2