МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение инклюзивного высшего образования
«Российский государственный университет социальных технологий»
(ФГБОУ ИВО «РГУ СоцТех»)

УТВЕРЖДАЮ

Проректор по образовательной деятельности

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

Б1.В.03 Теория алгоритмов
наименование дисциплины

01.03.02 «Прикладная математика и информатика»
шифр и наименование направления подготовки

Вычислительная математика и информационные технологии
направленность (профиль)

Москва 2024

Содержание

1. ПАСПОРТ ФОНДА ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине «Теория алгоритмов»

Оценочные средства составляются в соответствии с рабочей программой дисциплины и представляют собой совокупность контрольно-измерительных материалов (типовые задачи (задания), контрольные работы, тесты и др.), предназначенных для измерения уровня достижения обучающимися установленных результатов обучения.

Оценочные средства используются при проведении текущего контроля успеваемости и промежуточной аттестации.

Таблица 1 - Перечень компетенций, формируемых в процессе освоения дисциплины

Код компетенции	**Наименование результата обучения**
ПК-7	Способен к разработке и применению алгоритмических и программных решений в области системного и прикладного программного обеспечения
	ПК-7.1. Знает теоретические основы разработки программных и алгоритмических решений в области системного и прикладного программного обеспечения; математические методы решения задач, процедурный и объектно-ориентированный подходы к разработке информационных систем; актуальные проблемы в области программирования; методы и технологии программирования; языки программирования, основы технологии модульного программирования на языках высокого уровня. ПК-7.2. Умеет применить математический метод для решения задачи; подобрать рациональную технологию программирования для решения профессиональной задачи; создавать программные продукты и алгоритмические решения в области системного и прикладного программного обеспечения. ПК-7.3. Владеет навыками применения математических методов для решения задач и применения стандартных алгоритмов; навыками разработки и создания алгоритмических и программных решений в области системного и прикладного программного обеспечения; навыками разработки программных приложений с использованием современных языков программирования.

Конечными результатами освоения дисциплины являются сформированные когнитивные дескрипторы «знать», «уметь», «владеть», расписанные по отдельным компетенциям. Формирование дескрипторов происходит в течение всего семестра по этапам в рамках контактной работы, включающей различные виды занятий и самостоятельной работы, с применением различных форм и методов обучения (табл.2).

Таблица 2 - Формирование компетенций в процессе изучения дисциплины:

Код компетенции	Уровень освоения компетенций	Индикаторы достижения компетенций	Вид учебных занятий[1], работы, формы и методы обучения, способствующие формированию и развитию компетенций[2]	Контролируемые разделы и темы дисциплины[3]	Оценочные средства, используемые для оценки уровня сформированности компетенции[4]
ПК-7		*Знает*			
	Недостаточный уровень	ПК-7. Студент не знает теоретические основы разработки программных и алгоритмических решений в области системного и прикладного программного обеспечения; математические методы решения задач, процедурный и объектно-ориентированный подходы к разработке информационных систем; актуальные проблемы в области программирования. Студент не знает основные принципы построения поисковых, сортирующих и вычислительных алгоритмов, основы стратегии разработки алгоритмов. Студент не умеет определять наихудшее и среднее время работы алгоритмов; выбирать и применять различные методы для разработки алгоритмов; проводить сравнительный. Не владеет технологиями	Лекционные и практические занятия, самостоятельная работа обучающихся, подготовка и сдача промежуточной аттестации.	Раздел 1. Роль алгоритмов в вычислениях. Раздел 2. Основы разработки и анализа алгоритмов. Раздел 3. Рост функций. Раздел 4. Разделяй и властвуй. Раздел 5. Вероятностный анализ и рандомизированные алгоритмы. Раздел 6. Алгоритмы сортировки.	Текущий контроль – устный опрос, контрольная работа.

[1] Лекционные занятия, практические занятия, лабораторные занятия, самостоятельная работа…

[2] Необходимо указать активные и интерактивные методы обучения (например, интерактивная лекция, работа в малых группах, методы мозгового штурма и т.д.), способствующие развитию у обучающихся навыков командной работы, межличностной коммуникации, принятия решений, лидерских качеств.

[3] Наименование темы (раздела) берется из рабочей программы дисциплины.

[4] Оценочное средство должно выбираться с учетом запланированных результатов освоения дисциплины, например:
«Знать» – собеседование, коллоквиум, тест…
«Уметь», «Владеть» – индивидуальный или групповой проект, кейс-задача, деловая (ролевая)
игра, портфолио…

		оценки эффективности алгоритмов; навыками доказательства корректности алгоритмов; навыками производить алгоритмизацию и анализ времени работы алгоритмов.			
	Базовый уровень	ПК-7.1. Студент имеет несистематизированные знания теоретических основ разработки программных и алгоритмических решений в области системного и прикладного программного обеспечения; математических методов решения задач, процедурного и объектно-ориентированного подхода к разработке информационных систем; актуальных проблем в области программирования Студент имеет несистематизированные знание основных принципов построения поисковых, сортирующих и вычислительных алгоритмов, основы стратегии разработки алгоритмов.	Лекционные и практические занятия, самостоятельная работа обучающихся, подготовка и сдача промежуточной аттестации.	Раздел 1. Роль алгоритмов в вычислениях. Раздел 2. Основы разработки и анализа алгоритмов. Раздел 3. Рост функций. Раздел 4. Разделяй и властвуй. Раздел 5. Вероятностный анализ и рандомизированные алгоритмы. Раздел 6. Алгоритмы сортировки.	Текущий контроль – устный опрос, контрольная работа.
	Средний уровень	ПК-7.1. Студент знает основное содержание материала дисциплины. Студент самостоятельно выделяет главные положения в изученном материале. Знает теоретические основы разработки программных и алгоритмических решений в области системного и прикладного программного обеспечения; математические методы решения задач, процедурный и объектно-ориентированный подходы к разработке информационных систем;	Лекционные и практические занятия, работа в малых группах, интерактивная лекция, дискуссия, самостоятельная работа обучающихся, подготовка и сдача промежуточной аттестации.	Раздел 1. Роль алгоритмов в вычислениях. Раздел 2. Основы разработки и анализа алгоритмов. Раздел 3. Рост функций. Раздел 4. Разделяй и властвуй. Раздел 5. Вероятностный анализ и рандомизированные	Текущий контроль – устный опрос, контрольная работа.

		актуальные проблемы в области программирования. Студент знает основные принципы построения поисковых, сортирующих и вычислительных алгоритмов, основы стратегии разработки алгоритмов, но допускает незначительные ошибки		алгоритмы. Раздел 6. Алгоритмы сортировки.	
	Высокий уровень	ПК-7.1. Студент самостоятельно выделяет главные положения в изученном материале. Знает теоретические основы разработки программных и алгоритмических решений в области системного и прикладного программного обеспечения; математические методы решения задач, процедурный и объектно-ориентированный подходы к разработке информационных систем; актуальные проблемы в области программирования. Студент знает принципы построения поисковых, сортирующих и вычислительных алгоритмов; основы стратегии разработки алгоритмов.	Лекционные и практические занятия, работа в малых группах, интерактивная лекция, дискуссия, самостоятельная работа обучающихся, подготовка и сдача промежуточной аттестации.	Раздел 1. Роль алгоритмов в вычислениях. Раздел 2. Основы разработки и анализа алгоритмов. Раздел 3. Рост функций. Раздел 4. Разделяй и властвуй. Раздел 5. Вероятностный анализ и рандомизированные алгоритмы. Раздел 6. Алгоритмы сортировки.	Текущий контроль – устный опрос, контрольная работа.
		Умеет			
	Базовый уровень	ПК-7.2. Студент испытывает затруднения при выборе стратегии разработки алгоритмов. Студент умеет на базовом уровне применить математический метод для решения задачи; подобрать рациональную технологию программирования для решения профессиональной задачи; создавать программные продукты и алгоритмические решения в области системного и	Лекционные и практические занятия, работа в малых группах, интерактивная лекция, дискуссия, самостоятельная работа обучающихся, подготовка и сдача промежуточной аттестации.	Раздел 1. Роль алгоритмов в вычислениях. Раздел 2. Основы разработки и анализа алгоритмов. Раздел 3. Рост функций. Раздел 4. Разделяй и властвуй. Раздел 5. Вероятностный анализ и	Текущий контроль – устный опрос, контрольная работа.

		прикладного программного обеспечения. Студент испытывает затруднения при определении наихудшего и среднего времени работы алгоритмов; выборе и применении различных методов для разработки алгоритмов; проведении сравнительного анализа времени работы алгоритмов.		рандомизированные алгоритмы. Раздел 6. Алгоритмы сортировки.	
	Средний уровень	ПК-7.2 Студент умеет применить математический метод для решения задачи; подобрать рациональную технологию программирования для решения профессиональной задачи; создавать программные продукты и алгоритмические решения в области системного и прикладного программного обеспечения. Студент умеет определять наихудшее и среднее время работы алгоритмов; выбирать и применять различные методы для разработки алгоритмов; проводить сравнительный анализ времени работы алгоритмов, но допускает незначительные ошибки.	Лекционные и практические занятия, работа в малых группах, интерактивная лекция, дискуссия, самостоятельная работа обучающихся, подготовка и сдача промежуточной аттестации.	Раздел 1. Роль алгоритмов в вычислениях. Раздел 2. Основы разработки и анализа алгоритмов. Раздел 3. Рост функций. Раздел 4. Разделяй и властвуй. Раздел 5. Вероятностный анализ и рандомизированные алгоритмы. Раздел 6. Алгоритмы сортировки.	Текущий контроль – устный опрос, контрольная работа.
	Высокий уровень	ПК-7.2. Студент умеет анализировать прикладную задачу, находить оптимальные методы ее решения. Студент свободно умеет применить математический метод для решения задачи; подобрать рациональную технологию программирования для решения профессиональной задачи; создавать программные продукты и алгоритмические решения в области системного и	Лекционные и практические занятия, работа в малых группах, интерактивная лекция, дискуссия, самостоятельная работа обучающихся, подготовка и сдача промежуточной аттестации.	Раздел 1. Роль алгоритмов в вычислениях. Раздел 2. Основы разработки и анализа алгоритмов. Раздел 3. Рост функций. Раздел 4. Разделяй и властвуй. Раздел 5. Вероятностный анализ и	Текущий контроль – устный опрос, контрольная работа.

		прикладного программного обеспечения. Студент умеет определять наихудшее и среднее время работы алгоритмов; выбирать и применять различные методы для разработки алгоритмов; проводить сравнительный анализ времени работы алгоритмов.		рандомизированные алгоритмы. Раздел 6. Алгоритмы сортировки.	
		Владеет			
	Базовый уровень	ПК-7.3. Студент на базовом уровне владеет навыками применения математических методов для решения задач и применения стандартных алгоритмов; навыками разработки и создания алгоритмических и программных решений в области системного и прикладного программного обеспечения; технологиями оценки эффективности алгоритмов; навыками доказательства корректности алгоритмов; навыками производить алгоритмизацию.	Лекционные и практические занятия, работа в малых группах, интерактивная лекция, дискуссия, самостоятельная работа обучающихся, подготовка и сдача промежуточной аттестации.	Раздел 1. Роль алгоритмов в вычислениях. Раздел 2. Основы разработки и анализа алгоритмов. Раздел 3. Рост функций. Раздел 4. Разделяй и властвуй. Раздел 5. Вероятностный анализ и рандомизированные алгоритмы. Раздел 6. Алгоритмы сортировки.	Текущий контроль – устный опрос, контрольная работа.
	Средний уровень	ПК-7.3. Студент владеет знаниями всего изученного материала. Владеет навыками применения математических методов для решения задач и применения стандартных алгоритмов; навыками разработки и создания алгоритмических и программных решений в области системного и прикладного программного обеспечения. Студент владеет технологиями оценки эффективности алгоритмов; навыками	Лекционные и практические занятия, работа в малых группах, интерактивная лекция, дискуссия, самостоятельная работа обучающихся, подготовка и сдача промежуточной аттестации.	Раздел 1. Роль алгоритмов в вычислениях. Раздел 2. Основы разработки и анализа алгоритмов. Раздел 3. Рост функций. Раздел 4. Разделяй и властвуй. Раздел 5. Вероятностный анализ и рандомизированные	Текущий контроль – устный опрос, контрольная работа.

		доказательства корректности алгоритмов; навыками производить алгоритмизацию, но допускает незначительные ошибки.		алгоритмы. Раздел 6. Алгоритмы сортировки.	
	Высокий уровень	ПК-7.3. Студент владеет концептуально-понятийным аппаратом, научным языком и терминологией. Свободно владеет навыками применения математических методов для решения задач и применения стандартных алгоритмов; навыками разработки и создания алгоритмических и программных решений в области системного и прикладного программного обеспечения. Студент владеет технологиями оценки эффективности алгоритмов; навыками доказательства корректности алгоритмов; навыками производить алгоритмизацию.	Лекционные и практические занятия, работа в малых группах, интерактивная лекция, дискуссия, самостоятельная работа обучающихся, подготовка и сдача промежуточной аттестации.	Раздел 1. Роль алгоритмов в вычислениях. Раздел 2. Основы разработки и анализа алгоритмов. Раздел 3. Рост функций. Раздел 4. Разделяй и властвуй. Раздел 5. Вероятностный анализ и рандомизированные алгоритмы. Раздел 6. Алгоритмы сортировки.	Текущий контроль – устный опрос, контрольная работа.

2. ПЕРЕЧЕНЬ ОЦЕНОЧНЫХ СРЕДСТВ[5]

Таблица 3

№	Наименование оценочного средства	Характеристика оценочного средства	Представление оценочного средства в ФОС
1.	Устный опрос	Средство контроля усвоения учебного материала темы, раздела или разделов дисциплины, организованное как учебное занятие в виде собеседования преподавателя с обучающимися.	Вопросы по темам/разделам дисциплины
2.	Контрольная работа	Средство проверки умений применять полученные знания для решения задач определенного типа по теме или разделу	Комплект контрольных заданий по вариантам

[5] Указываются оценочные средства, применяемые в ходе реализации рабочей программы данной дисциплины.

3. ОПИСАНИЕ ПОКАЗАТЕЛЕЙ И КРИТЕРИЕВ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ

Оценивание результатов обучения по дисциплине «Математика» осуществляется в соответствии с Положением о текущем контроле успеваемости и промежуточной аттестации обучающихся.

Предусмотрены следующие виды контроля: текущий контроль (осуществление контроля всех видов аудиторной и внеаудиторной деятельности обучающегося с целью получения первичной информации о ходе усвоения отдельных элементов содержания дисциплины) и промежуточная аттестация (оценивается уровень и качество подготовки по дисциплине в целом).

Показатели и критерии оценивания компетенций, формируемых в процессе освоения данной дисциплины, описаны в табл. 4.

Таблица 4.

Код компетенции	Уровень освоения компетенции	Индикаторы достижения компетенции	Критерии оценивания результатов обучения
ПК-7		Знает	
	Недостаточный уровень Оценка «незачтено», «неудовлетворительно».	*ПК-7.1.*	*Не знает значительной части материала курса, не способен самостоятельно выделять главные положения в изученном материале дисциплины.*
	Базовый уровень Оценка «зачтено», «удовлетворительно».	*ПК-7.1.*	*Знает не менее 50 % основного материала курса, однако испытывает затруднения в его применении.*
	Средний уровень Оценка «зачтено», «хорошо».	*ПК-7.1.*	*Знает основную часть материала курса, способен применить изученный материал на практике, испытывает незначительные затруднения в решении профессиональных задач.*
	Высокий уровень Оценка «зачтено», «отлично».	*ПК-7.1.*	*Показывает глубокое знание и понимание материала, способен применить изученный материал на практике.*
		Умеет	
	Базовый уровень	*ПК-7.2.*	*Умеет воспроизвести не менее 50 % основного материала курса, однако испытывает затруднения при решении практических задач.*
	Средний уровень	*ПК-7.2.*	*Умеет решать стандартные профессиональные задачи с применением полученных знаний, испытывает незначительные затруднения в решении задач.*
	Высокий уровень	*ПК-7.2.*	*Умеет решать стандартные профессиональные задачи с применением полученных знаний, показывает глубокое знание и понимание материала, способен решить задачу при изменении формулировки.*
		Владеет	
	Базовый уровень	*ПК-7.3.*	*Студент владеет основными навыками применения математических методов для решения задач и применения стандартных алгоритмов; навыками разработки и создания алгоритмических и программных решений в области системного и прикладного программного обеспечения; навыками разработки программных приложений с использованием современных языков программирования; технологиями оценки эффективности алгоритмов; навыками*

			доказательства корректности алгоритмов; навыками производить алгоритмизацию. Имеет несистематизированные знания основных разделов дисциплины.
	Средний уровень	*ПК-7.3.*	*Студент владеет знаниями всего изученного материала, владеет навыками применения математических методов для решения задач и применения стандартных алгоритмов; навыками разработки и создания алгоритмических и программных решений в области системного и прикладного программного обеспечения; навыками разработки программных приложений с использованием современных языков программирования; технологиями оценки эффективности алгоритмов; навыками доказательства корректности алгоритмов; навыками производить алгоритмизацию. Испытывает незначительные затруднения в решении практических задач.*
	Высокий уровень	*ПК-7.3.*	*Свободно владеет навыками применения математических методов для решения задач и применения стандартных алгоритмов; навыками разработки и создания алгоритмических и программных решений в области системного и прикладного программного обеспечения; навыками разработки программных приложений с использованием современных языков программирования; технологиями оценки эффективности алгоритмов; навыками доказательства корректности алгоритмов; навыками производить алгоритмизацию.* *Студент владеет концептуально-понятийным аппаратом, научным языком и терминологией профессиональной деятельности.*

4. Методические материалы, определяющие процедуры оценивания результатов обучения

Задания в форме устного опроса:

Устный опрос используется для текущего контроля успеваемости обучающихся по дисциплине в качестве проверки результатов освоения материала. Каждому студенту выдается свой собственный, узко сформулированный вопрос. Ответ должен быть четким и кратким, содержащим все основные характеристики описываемого понятия. В своем ответе студент должен показать умения прослеживать причинно-следственные связи и навыки рассуждений и доказательства.

Контрольная работа

Средство проверки умений применять полученные знания для решения задач определенного типа по теме или разделу

5. Материалы для проведения текущего контроля и промежуточной аттестации

Задания в форме устного опроса

Семестр 4

Что такое алгоритмы.

1) Приведите реальные примеры задач, в которых возникает потребность в сортировке или вычислении выпуклой оболочки.
2) Какими ещё параметрами, кроме скорости, можно характеризовать алгоритм на практике?
3) Выберите одну из встречавшихся вам ранее структур данных и опишите её преимущества и ограничения.
4) Что общего между задачей об определении кратчайшего пути и задачей о коммивояжере? Чем они различаются?
5) Сформулируйте задачу, в которой необходимо только наилучшее решение. Сформулируйте также задачу, в которой может быть приемлемым решение, достаточно близкое к наилучшему.

Алгоритмы как технология.

1) Приведите пример приложения, для которого необходимо алгоритмическое наполнение на уровне приложений, и обсудите функции этих алгоритмов.
2) Предположим, на одной и той же машине проводится сравнительный анализ реализаций двух алгоритмов сортировки, работающих вставкой и слиянием. Для сортировки n элементов вставкой необходимо $8n^2$ шагов, а для сортировки слиянием – $64n\,lg\,n$ шагов. При каком значении n время сортировки вставкой превысит время сортировки слиянием?
3) При каком минимальном значении n алгоритм, время работы которого определяется формулой $100n^2$, работает быстрее, чем алгоритм, время работы которого выражается как $2^{\square}$, если оба алгоритма выполняются на одной и той же машине?

Сортировка вставкой.

1) Используя рис.2.2 в качестве образца, проиллюстрируйте работу процедуры Insertion-Sort по сортировке массива A = <31,41,59,26,41,58>.

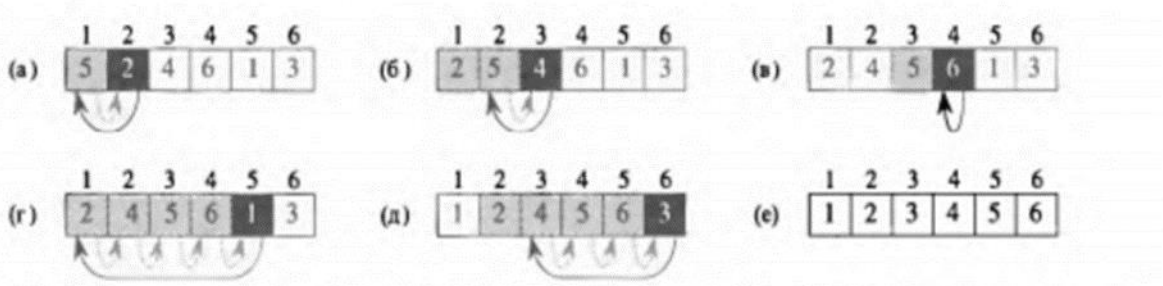

Рис. 2.2. Операции процедуры INSERTION-SORT над массивом $A = \langle 5, 2, 4, 6, 1, 3 \rangle$. Элементы массива обозначены квадратиками, над которыми находятся индексы, а внутри — значения соответствующих элементов. Части **(а)**–**(д)** этого рисунка соответствуют итерациям цикла **for** в строках 1–8 псевдокода. В каждой итерации черный квадратик содержит значение ключа из $A[j]$, которое сравнивается со значениями серых квадратиков, расположенных слева от него (строка псевдокода 5). Серыми стрелками указаны те значения массива, которые сдвигаются на одну позицию вправо (строка 6), а черной стрелкой — перемещение ключа (строка 8). В части **(е)** показано конечное состояние отсортированного массива.

2) Перепишите процедуру Insertion-Sort для сортировки в невозрастающем порядке вместо неубывающего.

3) Рассмотрим задачу поиска.
   Вход. Последовательность из *n* чисел $A = \langle a_1, a_2, \ldots, a_n \rangle$ и значение *v*.
   Выход. Индекс *i*, такой, что $v = A[i]$, или специальное значение NIL, если v в A отсутствует.
   Составьте псевдокод линейного поиска, при работе которого выполняется сканирование последовательности в поисках значения *v*. Докажите корректность алгоритма с помощью инвариантности цикла. Убедитесь, что выбранный инвариант цикла удовлетворяет трем необходимым условиям.

4) Рассмотрим задачу сложения двух *n*-битовых двоичных целых чисел, хранящихся в *n*-элементных массивах *A* и *B*. Сумму этих двух чисел необходимо занести в двоичной форме в *(n+1)*-элементный массив *C*. Приведите строгую формулировку задачи и составьте псевдокод для сложения этих двух чисел.

Анализ алгоритмов.

1) Выразите функцию $n^3/1000 - 100n^2 - 100n + 3$ Θ-обозначениях.

2) Рассмотрим сортировку элементов массива *A*, которая выполняется следующим образом. Сначала определяется наименьший элемент массива *A*, который становится на место элемента *A[1]*. Затем производится поиск второго наименьшего элемента массива *A*, который становится на место элемента *A[2]*. Этот процесс продолжается для первых *n-1* элементов массива *A*. Запишите псевдокод этого алгоритма, известного как сортировка выбором (selection sort). Какой инвариант цикла сохраняется для этого алгоритма? Почему его достаточно выполнить для первых *n-1* элементов, а не для всех n элементов? Определите время

работы алгоритма в наилучшем и наихудшем случаях и запишите его в Θ-обозначениях.

3) Вновь обратимся к алгоритму линейного поиска (см. сортировка вставкой 3). Для скольких элементов входной последовательности в среднем нужно произвести проверку, если предполагается, что все элементы массива с равной вероятностью могут иметь искомое значение? Что происходит в наихудшем случае? Чему равно время работы алгоритма линейного поиска в среднем и в наихудшем случаях в Θ-обозначениях? Обоснуйте свой ответ.

4) Каким образом можно модифицировать почти каждый алгоритм, чтобы получить оптимальное время работы в наилучшем случае?

Разработка алгоритмов.

1) Используя в качестве образца рис 2.4, проиллюстрируйте работу алгоритма сортировки слиянием для массива A = <3,41,52,26,38,57,9,49>.

Рис. 2.4. Процесс сортировки слиянием массива $A = \langle 5, 2, 4, 7, 1, 3, 2, 6 \rangle$. Длины подлежащих слиянию отсортированных последовательностей возрастают в ходе работы алгоритма.

2) Перепишите процедуру Merge так, чтобы в ней не использовались сигнальные значения. Сигналом к остановке должен служить тот факт, что все элементы массива L или массива R скопированы обратно в массив A, после чего в этот массив копируются элементы, оставшиеся в непустом массиве.

3) Воспользуйтесь методом математической индукции для доказательства того, что, когда n является точной степенью 2, решением рекуррентного соотношения

$$T(n) = \begin{cases} 2, & \text{если } n = 2 \\ 2T(n/2) + n, & \text{если } n = 2^k, \quad k > 1 \end{cases}$$

является $T(n) = n \lg n$.

4) Сортировку вставкой можно представить в виде рекурсивной последовательности. Чтобы отсортировать массив $A[1 \ldots n]$, сначала нужно рекурсивно отсортировать

массив $A[1..n-1]$, после чего в этот отсортированный массив помещается элемент $A[n]$. Запишите рекуррентное уравнение для времени работы этой рекурсивной версии сортировки вставкой.

5) Возвращаясь к задаче поиска (см. сортировка вставкой 3), нетрудно заметить, что если последовательность A отсортирована, то можно сравнить значение среднего элемента этой последовательности с искомым значением *v* и сразу исключить половину последовательности из дальнейшего рассмотрения. Бинарный поиск (binary search) – это алгоритм, в котором такая процедура повторяется неоднократно, что всякий раз приводит к уменьшению оставшейся части последовательности в два раза. Запишите псевдокод алгоритма бинарного поиска (либо итеративный, либо рекурсивный). Докажите, что время работы этого алгоритма в наихудшем случае составляет $\Theta(\lg n)$
6) Заметим, что в цикле while в строках 5-7 процедуры Insertion-Sort в разделе 2.1 для сканирования (в обратном порядке) отсортированного подмассива $A[1..j-1]$ используется линейный поиск. Можно ли использовать бинарный поиск (см. упр. 2.3.5) вместо линейного, чтобы время работы этого алгоритма в наихудшем случае улучшилось и стало равным $\Theta(n \lg n)$.
7) Разработайте алгоритм со временем работы $\Theta(n \lg n)$, который для заданного множества S из n целых чисел и другого целого числа x определяет, имеются ли во множестве S два элемента, сумма которых равна x.

Асимптотические обозначения.

1) Пусть $f(n)$ и $g(n)$ – асимптотически неотрицательные функции. Докажите с помощью базового определения Θ-обозначений, что $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.
2) Покажите, что для любых действительных констант a и b, где $b > 0$, выполняется соотношение $(n+a)^b = \Theta(n^b)$.
3) Поясните, почему утверждение «время работы алгоритма A равно как минимум $O(n^2)$» лишено смысла.
4) Справедливы ли соотношения $2^{n+1} = O(2^n)$ и $2^{2n} = O(2^n)$?
5) Докажите, что для любых двух функций $f(n)$ и $g(n)$ мы имеем $f(n) = \Theta(g(n))$ тогда и только тогда, когда $f(n) = O(g(n))$ и $f(n) = \Omega(g(n))$.
6) Докажите, что время работы алгоритма равно $\Theta(g(n))$ тогда и только тогда, когда его время работы в наихудшем случае равно $O(g(n))$, а в наилучшем - $\Omega(g(n))$.
7) Докажите, что множество $o(g(n)) \cap \omega(g(n))$ является пустым.

Стандартные обозначения и часто встречающиеся функции.

1) Покажите, что если функции $f(n)$ и $g(n)$ монотонно неубывающие, то таковыми же являются и функции $f(n) + g(n)$ и $f(g(n))$, а если вдобавок $f(n)$ и $g(n)$ неотрицательны, то монотонно неубывающей является и функция $f(n) \cdot g(n)$.
2) Докажите уравнение $a^{\log_b c} = c^{\log_b a}$.
3) Докажите уравнение $\lg(n!) = \Theta(n \lg n)$. Докажите также, что $n! = \omega(2^n)$ и $n! = o(n^n)$
4) Докажите, что из $k \ln k = \Theta(n)$ вытекает $k = \Theta\left(n/\ln n\right)$/

Семестр 5

Задача поиска максимального подмассива.

1) Что возвращает процедура Find-Maximum-Subarray, когда все элементы A отрицательны?
2) Напишите псевдокод для решения задачи поиска максимального подмассива методом грубой силы. Ваша процедура должна выполняться за время $\Theta(n^2)$.
3) Реализуйте и метод грубой силы, и рекурсивный алгоритм на своем компьютере. Каким оказывается размер задачи точки пересечения n_0, в которой рекурсивный алгоритм превосходит алгоритм грубой силы? Далее измените базовый случай рекуррентного алгоритма, применяя алгоритм грубой силы при размере задачи, не превосходящей n_0. Меняет ли это точку пересечения?
4) Предположим, что мы меняем определение задачи поиска максимального подмассива, позволяя конечному результату быть пустым массивом и полагая, что сумма значений пустого массива равна нулю. Как бы вы изменили любой алгоритм, не допускающий решения в виде пустого массива, чтобы такой результат в виде пустого подмассива стал возможным?
5) Воспользуйтесь приведенными далее идеями для разработки не рекурсивного алгоритма поиска максимального подмассива за линейное время. Начните с левого конца массива и двигайте вправо, отслеживая найденный к данному моменту максимальный подмассив. Зная максимальный подмассив массива $A[1..j]$, распространите ответ на поиск максимального подмассива, заканчивающегося индексом j + 1, воспользовавшись следующим наблюдением: максимальный подмассив массива $A[1..j+1]$ представляет собой либо максимальный подмассив массива $A[1..j]$, либо подмассив вида $A[i..j+1]$ для некоторого $1 \le i \le j+1$. Определите максимальный подмассив вида $A[i..j+1]$ за константное время, зная максимальный подмассив, заканчивающийся индексом j.

Алгоритм Штрассена для умножения матриц.

1) Воспользуйтесь алгоритмом Штрассена для вычисления произведения матриц
$$\begin{pmatrix}1 & 3\\ 7 & 5\end{pmatrix}\begin{pmatrix}6 & 8\\ 4 & 2\end{pmatrix}$$
Покажите, как вы это делаете.
2) Запишите псевдокод алгоритма Штрассена.
3) Как модифицировать алгоритм Штрассена для перемножения матриц размером $n \times n$, где n не является точной степенью *2*? Покажите, что получающийся в результате алгоритм выполняется за время $\Theta(n^{\lg 7})$.
4) Чему равно наибольшее k, такое, что если вы можете перемножить 3×3 –матрицы с помощью k умножений (не предполагая коммутативности умножения), то вы можете перемножить матрицы размером $n \times n$ за время $o(n^{\lg 7})$.? Каким должно быть время работы такого алгоритма?
5) В. Пан (V. Pan) открыл способ перемножения матриц размером 68×68 с использованием только 132 464 умножений, способ перемножения матриц размером 70×70 с использованием 143 640 умножений и способ перемножения матриц размером 72×72 с использованием 155 424 умножений. Какой из методов

дает нам лучшее асимптотическое время работы при его использовании в алгоритме «разделяй и властвуй» для перемножения матриц? Проведите сравнение с алгоритмом Штрассена.

6) Насколько быстро вы сумеете матрицу размером $kn \times n$ на матрицу размером □ $\times kn$, применяя алгоритм Штрассена в качестве подпрограммы? Ответьте на тот же вопрос для ситуации, когда мы меняем входные матрицы местами.
7) Покажите, как перемножить комплексные числа $a + bi$ и $c + di$ используя только три умножения действительных чисел. Алгоритм должен получать a, b, c и d в качестве входных данных и возвращать действительную $(ac - bd)$ и мнимую $(ad + bc)$ части произведения по отдельности.

Метод подстановки решения рекуррентных соотношений.

1) Покажите, что решением $T(n) = T(n-1) + n$ является $O(n^2)$.
2) Покажите, что решением $T(n) = T(\lceil n/2 \rceil) + 1$ является $O(\lg n)$.
3) Мы видели, что решением $T(n) = 2T(\lfloor n/2 \rfloor) + n$ является $O(n \lg n)$. Покажите, что решение этого рекуррентного соотношения также представляет собой $\Omega(n \lg n)$. Сделайте вывод, что решением рассматриваемого рекуррентного соотношения является $\Theta(n \lg n)$.
4)

$$T(n) = 2T(\lfloor n/2 \rfloor) + n \ , \qquad (4.19)$$

Покажите, что преодолеть трудность, связанную с граничным условием $T(1) = 1$ в рекуррентном соотношении (4.19) можно путем выбора другого предположения индукции, не меняя при этом граничных условий.

5)

$$T(n) = \begin{cases} \Theta(1) \ , & \text{если } n = 1 \ , \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) \ , & \text{если } n > 1 \ . \end{cases} \qquad (4.3)$$

Покажите, что $\Theta(n \lg n)$ является решением «точного» рекуррентного соотношения (4.3) для сортировки слиянием.

6) Покажите, что решением рекуррентности $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$ является $O(n \lg n)$.
7) Используя основной метод раздела 4.5, можно показать, что решением рекуррентного соотношения $T(n) = 4T(n/3) + n$ является $T(n) = \Theta(n^{\log_3 4})$. Покажите, что не получается выполнить доказательство методом подстановок с гипотезой индукции $T(n) \le cn^{\log_3 4}$. Затем покажите, как вычитание члена меньшего порядка делает доказательство возможным.
8) Используя основной метод раздела 4.5, можно показать, что решением рекуррентного соотношения $T(n) = 4T(n/2) + n$ является $T(n) = \Theta(n^2)$. Покажите, что не получается выполнить доказательство методом подстановок с гипотезой индукции $T(n) \le cn^2$. Затем покажите, как вычитание члена меньшего порядка делает доказательство возможным.

9) Решите рекуррентное соотношение $T(n) = 3T(\sqrt{n}) + \log n$ путем замены переменных. Ваше решение должно быть асимптотически точным. При решении не беспокойтесь о том, чтобы все значения являлись целыми числами.

Метод деревьев рекурсии.

1) Воспользуйтесь деревом рекурсии для определения точной асимптотической верхней границы рекуррентного соотношения $T(n) = 3T(\lfloor n/2 \rfloor) + n$. Для проверки своего ответа используйте метод подстановок.
2) Воспользуйтесь деревом рекурсии для определения точной асимптотической верхней границы рекуррентного соотношения $T(n) = T(n/2) + n^2$. Для проверки своего ответа используйте метод подстановок.
3) Воспользуйтесь деревом рекурсии для определения точной асимптотической верхней границы рекуррентного соотношения $T(n) = 4T(n/2 + 2) + n$. Для проверки своего ответа используйте метод подстановок.
4) Воспользуйтесь деревом рекурсии для определения точной асимптотической верхней границы рекуррентного соотношения $T(n) = 2T(n-1) + 1$. Для проверки своего ответа используйте метод подстановок.
5) Воспользуйтесь деревом рекурсии для определения точной асимптотической верхней границы рекуррентного соотношения $T(n) = T(n-1) + T(n/2) + n$. Для проверки своего ответа используйте метод подстановок.
6) Обратившись к соответствующему дереву рекурсии, докажите, что решением рекуррентного соотношения $T(n) = T(n/3) + T(2n/3) + cn$, где c представляет собой константу, является $\Omega(n \lg n)$.
7) Построй дерево рекурсии для рекуррентного соотношения $T(n) = 4T(\lfloor n/2 \rfloor) + cn$, где c – константа, и найдите точную асимптотическую границу его решения, проверьте её с помощью метода подстановок.
8) Найдите с помощью дерева рекурсии точную асимптотическую оценку решения рекуррентного соотношения $T(n) = T(n-a) + T(a) + cn$, где $a \geq 1$ и $c > 0$ являются константами.
9) Найдите с помощью дерева рекурсии точную асимптотическую оценку решения рекуррентного соотношения $T(n) = T(\alpha n) + T((1-\alpha)n) + cn$, где α - константа из диапазона $0 < \alpha < 1$; константой является и $c > 0$.

Основной метод.

1) С помощью основной теоремы найдите точные асимптотические границы следующих рекуррентных соотношений.
   а. $T(n) = 2T(n/4) + 1$
   б. $T(n) = 2T(n/4) + \sqrt{n}$
   в. $T(n) = 2T(n/4) + n$
   г. $T(n) = 2T(n/4) + n^2$
2) Профессор хочет разработать алгоритм матричного умножения, асимптотически более быстрый, чем алгоритм Штрассена. Его алгоритм будет использовать метод

«разделяй и властвуй», разбивая каждую матрицу на части размером $n/4 \times n/4$, причем шаги разделения и комбинирования выполняются за время $\Theta(n^2)$.Профессору требуется определить, сколько подзадач должен создавать его алгоритм, чтобы определить алгоритм Штрассена. Если алгоритм профессора создает a подзадач, то рекуррентное соотношение для времени работы $T(n)$ принимает вид $T(n) = aT(n/4) + \Theta(n^2)$. Каково наибольшее целочисленное значение a, для которого алгоритм профессора оказывается асимптотически быстрее алгоритма Штрассена?

3) Покажите с помощью основного метода, что $T(n) = \Theta(\lg n)$ является решением рекуррентного соотношения $T(n) = 2T(n/2) + \Theta(1)$, возникающего в ходе анализа алгоритма бинарного поиска. (Алгоритм бинарного поиска описан в упр. 2.3.5)
4) Можно ли применить основной метод к рекуррентному соотношению $T(n) = 4T(n/2) + n^2 \lg n$? Обоснуйте свой ответ. Найдите асимптотическую верхнюю границу решения этого рекуррентного соотношения.
5) Рассмотрим условие регулярности $af(n/b) \le cf(n)$ для некоторой константы $c < 1$, являющееся частью случая 3 основной теоремы. Приведите примеры констант $a \ge 1$ и $b > 1$ и функции $f(n)$, которые удовлетворяют всем условиям случая 3 основной теоремы, кроме условия регулярности.

Доказательство основной теоремы.

1)

$$n_j = \begin{cases} n\,, & \text{если } j = 0\,, \\ \lceil n_{j-1}/b \rceil\,, & \text{если } j > 0\,. \end{cases} \qquad (4.27)$$

Приведите простое и точное выражение для n_j в уравнении (4.27) для случая, когда b – положительное целое число (а не произвольное действительное).

2) Покажите, что если выполняется соотношение $f(n) = \Theta(n^{\log_b a} \lg^k n)$, где $k \ge 0$, то основное рекуррентное соотношение имеет решение $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$. Для простоты рассмотрите только случай точных степеней b.
3) Покажите, что в случае 3 основной теоремы одно из условий излишнее в том смысле, что из условия регулярности $af(n/b) \le cf(n)$ для некоторой константы $c < 1$ следует, что существует константа $\epsilon > 0$, такая, что $f(n) = \Omega(n^{\log_b a + \epsilon})$.

Контролируемые компетенции: ПК-7

Оценка компетенций осуществляется в соответствии с таблицей 4.

Контрольная работа

Семестр 4

Задача 1

Ниже приведена таблица, строки которой соответствуют различным функциям $f(n)$, а столбцы – значениям времени *t*. Заполните таблицу максимальными значениями *n*, для которых задача может быть решена за время *t*, если предполагается, что время работы алгоритма, необходимое для решения задачи, равно $f(n)$ микросекунд.

	Секунда	Минута	Час	День	Месяц	Год	Век
$\log n$							
$\sqrt{n}$							
n							
$n \log n$							
n^2							
n^3							
2^n							
$n!$							

Задача 2

Рассмотрите сортировку n элементов массива A, которая выполняется следующим образом. Сначала определяется наименьший элемент массива A, который ставится на место элемента A[1]. Затем производится поиск второго наименьшего элемента массива A, который ставится на место элемента A[2]. Этот процесс продолжится для первых n-1 элементов массива A. Запишите псевдокод этого алгоритма, известного как *сортировка выбором (selection sort)*. Какой инвариант цикла сохраняется для этого алгоритма? Почему его достаточно выполнить для первых n-1 элементов, а не для всех n элементов? Определите время работы алгоритма в наилучшем и в наихудшем случаях и запишите его в Θ обозначениях.

Задача 3

Рассмотрите сортировку n элементов массива A, которая называется *сортировка вставкой (insertion sort).* Она напоминает способ к которому прибегают игроки для сортировки имеющихся на руках карт. Сначала в левой руке нет ни одной карты и все они лежат на столе рубашкой вверх. Далее со стола берется по одной карте, каждая из которых помещается в нужное место среди карт, которые находятся в левой руке. Чтобы определить, куда поместить очередную карту, ее масть и достоинство сравниваются с мастью и достоинством карт в руке. После каждого шага карты в левой руке будут отсортированы. Пусть сравнение проводится в направлении слева направо. Запишите псевдокод алгоритма сортировки вставкой. Какой инвариант цикла сохраняется для этого алгоритма? Определите время работы алгоритма в наилучшем и в наихудшем случаях и запишите его в Θ обозначениях.

Задача 4

Сортировка вставкой малых массивов в процессе сортировки слиянием

Несмотря на то, что с увеличением количества сортируемых элементов время сортировки методом слияний в наихудшем случае растет как $\Theta(n \lg n)$, а время сортировки вставкой – как $\Theta(n^2)$, благодаря постоянным множителям на практике для малых размеров задач на большинстве машин сортировка вставкой выполняется быстрее. Таким образом, есть смысл использовать сортировку вставок в процессе сортировке методом слияний, когда подзадачи становятся достаточно маленькими. Рассмотрите модификацию алгоритма

сортировки слиянием, в котором n/k подмассивов длиной k сортируются вставкой, после чего они объединяются с помощью обычного механизма слияния. Величина k должна быть найдена в процессе решения задачи.

a. Покажите, что сортировка вставкой позволяет отсортировать n/k подпоследовательностей длиной k каждая за время $\Theta(nk)$ в худшем случае.
b. Покажите, как выполнить слияние этих подпоследовательностей за время $\Theta(n \lg(n/k))$ в наихудшем случае.
c. Если такой модифицированный алгоритм выполняется за время $\Theta(nk + n \lg(n/k))$ в наихудшем случае, то чему равно наибольшее значение k как функции от n, для которого модифицированный алгоритм в Θ-обозначениях имеет то же время работы, что и стандартная сортировка слиянием?
d. Как следует выбирать k на практике?

Задача 5

Корректность пузырьковой сортировки

Пузырьковая сортировка представляет собой популярный, но не эффективный алгоритм сортировки. В его основе лежит многократная перестановка соседних элементов, нарушающих порядок сортировки.

Bubblesort(A)

```
1   for i = 1 to A.length – 1
2           for j = A.length downto i + 1
3                   If A[j] < A[j - 1]
4                           Поменяем A[j] и A[j - 1] местами
```

a. Пусть $\overline{A}$ обозначает выход процедуры Bubblesort(A). Для доказательства корректности процедуры Bubblesort необходимо доказать, что она завершается и что

$$\overline{A}[1] \leq \overline{A}[2] \leq \cdots \leq \overline{A}[n]$$

где $n = A.length$. Что ещё необходимо доказать для того, чтобы показать, что процедура Bubblesort действительно выполняет сортировку?

В следующих двух частях доказываются неравенства (2.3).

b. Точно сформулируйте инвариант цикла for в строках 2-4 и докажите, что он выполняется. Доказательство должно иметь ту же структуру доказательства инварианта цикла, которая ранее использовалась в аналогичных доказательствах в данной главе.

c. С помощью условия завершения инварианта цикла, доказанного в части (b), сформулируйте инвариант цикла for в строках 1-4, который позволил бы доказать неравенства (2.3). Доказательство должно иметь ту же структуру доказательства инварианта цикла, которая использовалась ранее в аналогичных доказательствах в данной главе.

d. Определите время пузырьковой сортировки в наихудшем случае и сравните его со временем сортировки вставкой.

Задача 6

Корректность правила Горнера

Следующий фрагмент кода реализует правило Горнера для вычисления полинома

$$P(x) = \sum_{i=0}^{n} a_i x^i = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + xa_n) \ldots))$$

для заданных коэффициентов $a_0, a_1, a_2, \ldots, a_n$ и значения x.

```
1   y = 0
2   for I = n downto 0
3           y = a_i + x * y
```

a. Чему равно время работы этого фрагмента кода правила Горнера в \theta-обозначениях?

b. Напишите псевдокод, реализующий алгоритм обычного вычисления полинома, когда каждое слагаемое полинома вычисляется отдельно. Определите асимптотическое время работы этого алгоритма и сравните его со временем работы алгоритма, основанного на правиле Горнера.

c. Рассмотрим следующий инвариант цикла.

В начале каждой итерации цикла for в строках 2 и 3

$$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k$$

Рассмотрим сумму без членов как равную нулю. Следую структуре доказательства инварианта цикла, которая использовалась ранее в данной главе, воспользуйтесь указанным инвариантом цикла, чтобы показать, что по завершении работы $y = \sum_{k=0}^{n} a_k x^k$.

d. Сделайте заключение, что в приведенном фрагменте кода правильно вычисляется значение полинома, который задается коэффициентами $a_0, a_1, a_2, \ldots, a_n$.

Задача 7

Инверсии

Пусть $A[1 \ldots n]$ представляет собой массив из n различных чисел. Если $i < j$ и $A[i] > A[j]$, то пара (i, j) называется инверсией A.

a. Перечислите пять инверсий массива $< 2,3,8,6,1 >$.

b. Какой массив из элементов множества $\{1,2, \ldots, n\}$ содержит максимальное количество инверсий? Сколько инверсий в этом массиве?

c. Какая существует взаимосвязь между временем сортировки методом вставок и количеством инверсий во входном массиве? Обоснуйте свой ответ.

d. Разработайте алгоритм, определяющий количество инверсий, содержащихся в произвольной перестановке n элементов, время работы которого в наихудшем случае равно $\Theta(n \lg n)$. (Указание: модифицируйте алгоритм сортировки слиянием.)

Задача 8

Асимптотическое поведение полиномов

Пусть $p(n) = \sum_{i=0}^{d} a_i n^i$,

где $a_d > 0$, представляет собой полином степени d от n, и пусть k является константой. Используя определения асимптотических обозначений, докажите следующие свойства.

а. Если $k \geq d$, то $p(n) = O(n^k)$.

б. Если $k \leq d$, то $p(n) = \Omega(n^k)$.

в. Если $k = d$, то $p(n) = \Theta(n^k)$.

г. Если $k > d$, то $p(n) = o(n^k)$.

д. Если $k < d$, то $p(n) = \omega(n^k)$.

Задача 9

Относительный асимптотический рост

Для каждой пары приведенных в таблице выражений *(A, B)* укажите, каким отношением *A* связано с *B*: O, Ω, o, ω или Θ. Предполагается, что $k \geq 1$, $\epsilon > 0$ и $c > 1$ – константы. Ваш ответ должен выражаться таблицей, в каждой ячейке которой указано значение «Да» или «Нет».

A	B	O	o	Ω	ω	Θ
$(\lg n)^k$	n^ϵ					
n^k	c^n					
$\sqrt{n}$	$n^{\sin n}$					
2^n	$2^{n/2}$					
$n^{\lg c}$	$c^{\lg n}$					
$\lg(n!)$	$\lg(n^n)$					

Задача 10

Упорядочение по скорости асимптотического роста

а. Расположите приведенные ниже функции по скорости их асимптотического роста, т.е. постройте такую последовательность функций $g_1, g_2, \ldots, g_{30}$ что $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3), \ldots, g_{29} = \Omega(g_{30})$. Разбейте свой список на классы эквивалентности так,

чтобы функции $f(n)$ и $g(n)$ находились в одном и том же классе тогда и только тогда, когда $f(n) = \Theta(g(n))$.

$\lg(\lg^* n)$	$2^{\lg^* n}$	$(\sqrt{2})^{\lg n}$	n^2	$n!$	$(\lg n)!$
$\left(\frac{3}{2}\right)^n$	n^3	$\lg^2 n$	$\lg(n!)$	2^{2^n}	$n^{1/\lg n}$
$\ln \ln n$	$\lg^* n$	$n2^n$	$n^{\lg \lg n}$	$\ln n$	1
$2^{\lg n}$	$(\lg n)^{\lg n}$	e^n	$4^{\lg n}$	$(n+1)!$	$\sqrt{\lg n}$
$\lg^*(\lg n)$	$2^{\sqrt{2 \lg n}}$	n	2^n	$n \lg n$	$2^{2^{n+1}}$

б. Приведите пример неотрицательной функции $f(n)$, такой, что для всех функций $g_i(n)$ из части (а) $f(n)$ не принадлежит ни множеству $O(g_i(n))$, ни множеству $\Omega(g_i(n))$.

Семестр 4

Задача 1

Свойства асимптотических обозначений

Пусть $f(n)$ и $g(n)$ – асимптотически положительные функции. Докажите или опровергните справедливость каждого из приведенных ниже утверждений.

а. Из $f(n) = O(g(n))$ вытекает $g(n) = O(f(n))$.

б. $f(n) + g(n) = \Theta(\min(f(n), g(n)))$.

в. Из $f(n) = O(g(n))$ вытекает $\lg f(n) = O(\lg g(n))$, где $\lg g(n) \geq 1$ и $f(n) \geq 1$ для всех достаточно больших n.

г. Из $f(n) = O(g(n))$ вытекает $2^{f(n)} = O(2^{g(n)})$.

д. $f(n) = O((f(n))^2)$.

е. Из $f(n) = O(g(n))$ вытекает $g(n) = \Omega(f(n))$.

ж. $f(n) = \Theta(f(\frac{n}{2}))$.

з. $f(n) + o(f(n)) = \Theta(f(n))$.

Задача 2

Вариации определений O и Ω.

Некоторые авторы определяют Ω несколько иначе, чем это делаем мы; давайте для такого альтернативного определения будем использовать символ Ω_∞ (читается как «омега бесконечность»). Будем говорить, что $f(n) = \Omega_\infty(g(n))$, если существует положительная константа c, такая, что $f(n) \geq cg(n) \geq 0$ для бесконечного количества целых чисел n.

а. Покажите что для любых двух асимптотически неотрицательных функций $f(n)$ и $g(n)$ выполняется одно из соотношений $f(n) = O(g(n))$ и $f(n) = \Omega_\infty(g(n))$ (или они оба), в то время как при использовании Ω вместо Ω_∞ это утверждение ложно.

б. Опишите потенциальные преимущества и недостатки применения Ω_∞ вместо Ω для характеристики времени работы программ.

Некоторые авторы определяют несколько иначе и O; для такого альтернативного определения будем использовать символO'. Будем говорить, что $f(n) = O'(g(n))$ тогда и только тогда, когда $|f(n)| = O(g(n))$.

в. Что произойдет в теореме 3.1 с каждым из направлений «тогда и только тогда, когда», если заменить O на O', но оставить без изменений Ω?

Некоторые авторы определяют $\tilde{O}$ (читается как «о с тильдой») для обозначения O, в котором игнорируются логарифмические множители:

$$\tilde{O}(g(n)) = \left\{ \begin{matrix} f(n): \text{существуют положительные константы } c, k \text{ и } n_0, \\ \text{такие, что } 0 \le f(n) \le cg(n)\lg^k n \text{ для всех } n \ge n_0 \end{matrix} \right\}$$

Задача 3

Итерированные функции

Оператор итерации *, используемый в функции lg*, можно применить к любой монотонно возрастающей функции $f(n)$, определенной на множестве действительный чисел. Для заданной константы $c \in \mathbb{R}$ итерированная функция определяется следующим образом:

$$f_c^*(n) = \min\{i \ge 0 : f^{(i)}(n) \le c\}$$

Она может не быть вполне определенной во всех случаях. Другими словами, величина $f_c^*(n)$ представляет собой количество итерированных применений функции f, требующееся для того, чтобы уменьшить её аргумент до значения, не превышающего c.

Для каждой из приведенных далее функций $f(n)$ и констант с дайте максимально точную оценку функции $f_c^*(n)$.

$f(n)$	c	$f_c^*(n)$
$(n-1)$	0	
$\lg n$	1	
$n/2$	1	
$n/2$	2	
$\sqrt{n}$	2	
$\sqrt{n}$	1	
$n^{1/3}$	2	
$n/\lg n$	2	

Задача 4

Примеры рекуррентных соотношений

Определите верхнюю и нижнюю асимптотические границы функции $T(n)$ для каждого из представленных ниже рекуррентных соотношений. Считаем, что $T(n)$ при $n \geq 2$ является константой. Попытайтесь сделать эту оценку как можно более точной и обоснуйте свой ответ.

а. $T(n) = 2T\left({}^{n}/_{2}\right) + n^4$

б. $T(n) = T\left({}^{7n}/_{10}\right) + n$

в. $T(n) = 16T\left({}^{n}/_{4}\right) + n^2$

г. $T(n) = 7T\left({}^{n}/_{3}\right) + n^2$

д. $T(n) = 7T\left({}^{n}/_{2}\right) + n^2$

е. $T(n) = 2T\left({}^{n}/_{4}\right) + \sqrt{n}$

ж. $T(n) = T(n-2) + n^2$

Задача 5

Стоимости передачи параметров

Предполагается, что передача параметров при вызове процедуры занимает фиксированное время, даже если передается N-элементый массив. Для большинства вычислительных систем это предположение справедливо, поскольку передается не сам массив, а указатель на него. В данной задаче исследуются три стратегии передачи параметров.

1. Массив передается посредством указателя. Время $= \Theta(1)$.

2. Массив передается посредством копирования. Время $= \Theta(N)$, где N – размер массива.

3. Массив передается путем копирования только некоторого поддиапазона, к которому обращается вызываемая процедура. Время $= \Theta(q - p + 1)$ при передаче подмассива $A[p\,..\,q]$.

а. Рассмотрите рекурсивный алгоритм бинарного поиска, предназначенный для нахождения числа в отсортированном массива (см. упр. 2.3.5). Приведите рекуррентные соотношения, описывающие время бинарного поиска в наихудшем случае, если массивы передаются с помощью каждого из описанных выше методов, и дайте точные верхние границы решений этих рекуррентных соотношений. Пусть размер исходной задачи равен N, а размер подзадачи – n.

б. Выполните задание (а) для алгоритма Merge-Sort из раздела 2.3.1.

Задача 6

Другие примеры рекуррентных соотношений

Дайте верхнюю и нижнюю асимптотические оценки функции $T(n)$ в каждом из приведенных ниже рекуррентных соотношений. Предполагается, что $T(n)$ для достаточно

малых значений n является постоянной величиной. Постарайтесь, чтобы оценки были как можно более точными и обоснуйте ответы.

а. $T(n) = 4T(n/3) + n \lg n$

б. $T(n) = 3T(n/3) + n/\lg n$

в. $T(n) = 4T(n/2) + n^2\sqrt{n}$

г. $T(n) = 3T(n/3 - 2) + n/2$

д. $T(n) = 2T(n/2) + n/\lg n$

е. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$

ж. $T(n) = T(n-1) + 1/n$

з. $T(n) = T(n-1) + \lg n$

и. $T(n) = T(n-2) + 1/\lg n$

к. $T(n) = \sqrt{n}T(\sqrt{n}) + n$

Задача 7

Числа Фибоначчи

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_i &= F_{i-1} + F_{i-2} \qquad \text{для } i \geq 2 \end{aligned} \tag{3.22}$$

В этой задаче раскрываются свойства чисел Фибоначчи, определенных с помощью рекуррентного соотношения (3.22). Воспользуемся для решения рекуррентного соотношения Фибоначчи, методом производящих функций. Определим **производящую функцию** (или **формальный степенной ряд**) $\mathcal{F}$ как

$$\mathcal{F}(z) = \sum_{i=0}^{\infty} F_i z^i$$

$$= 0 + z + z^2 + 2z^3 + 3z^4 + 5z^5 + 8z^6 + 13z^7 + 21z^8 + \cdots,$$

где F_i представляет собой -е число Фибоначчи.

а. Покажите, что $\mathcal{F}(z) = z + z\mathcal{F}(z) + z^2\mathcal{F}(z)$.

б. Покажите, что

$$\mathcal{F}(z) = \frac{z}{1 - z - z^2}$$

$$= \frac{z}{(1-\phi z)(1-\hat{\phi} z)}$$

$$= \frac{1}{\sqrt{5}}\left(\frac{1}{1-\phi z} - \frac{1}{1-\hat{\phi} z}\right)$$

где $\phi = \frac{1+\sqrt{5}}{2} = 1{,}61803\ldots$ и $\hat{\phi} = \frac{1-\sqrt{5}}{2} = -0{,}61803\ldots$

в. Покажите, что $\mathcal{F}(z) = \sum_{i=0}^{\infty} \frac{1}{\sqrt{5}}\left(\phi^i - \widehat{\phi^i}\right) z^i$

г. Воспользуйтесь п. (в) для доказательства того, что $F_i = \phi^i/\sqrt{5}$ (округленное до ближайшего целого) для всех $i > 0$. (*Указание*: обратите внимание, что $|\hat{\phi}| < 1$.)

Задача 8

Тестирование микросхем

В распоряжении профессора есть n предположительно идентичных микросхем («чипов»), которые в принципе способны тестировать друг друга. В тестирующее приспособление за один раз можно поместить две микросхему. При этом каждая микросхема тестирует соседнюю и выдает отчет о результате тестирования. Исправная микросхема всегда выдает правильные результаты тестирования другой микросхемы, а результатам неисправной микросхемы доверять нельзя. Таким образом, возможны четыре варианта результатов тестирования, приведенные в таблице ниже.

Отчет *A*	Отчет *B*	Заключение
B исправна	*A* исправна	Либо обе исправны, либо обе неисправны
B исправна	*A* неисправна	Неисправна как минимум одна микросхема
B неисправна	*A* исправна	Неисправна как минимум одна микросхема
B неисправна	*A* неисправна	Неисправна как минимум одна микросхема

а. Покажите, что, если как минимум $n/2$ микросхем неисправны, профессор не сможет точно определить исправные микросхемы, какой бы стратегией попарных испытаний он не пользовался. (Предполагается, что неисправные микросхемы не договариваются между собой, чтобы обмануть профессора.)

б. Рассмотрим задачу о поиске одной исправной микросхемы среди n микросхем, если предполагается, что исправно более половины всех микросхем. Покажите, что $\lfloor {}^{n}/_{2} \rfloor$ попарных тестирований достаточно для сведения этой задачи к подзадаче, размер которой приблизительно в два раза меньше.

в. Покажите, что можно найти все исправные микросхемы с помощью $\Theta(n)$ попарных тестирований в предположении, что исправны более половины микросхем. Сформулируйте и решите рекуррентное соотношение, описывающее количество тестирований.

Задача 9

Массивы Монжа

Массив A размером $m \times n$, состоящий из действительных чисел, является **массивом Монжа** (Monge array), если для всех i, j, k и l, таких, что $1 \leq i < k \leq m$ и $1 \leq j < l \leq n$, мы имеем

$$A[i, j] + A[k, l] \leq A[i, l] + A[k, j]$$

Другими словами, при любом выборе двух строк и двух столбцов из массива Монжа (при этом элементы массива на пересечении выбранных строк и столбцов образуют прямоугольник) сумма элементов в левом верхнем и правом нижнем углах полученного прямоугольника не превышает сумму элементов в левом нижнем и правом верхнем его углах. Приведем пример массива Монжа.

10	17	13	28	23
17	22	16	29	23
24	28	22	34	24
11	13	6	17	7
45	44	32	37	23
36	33	19	21	6
75	66	51	53	34

а. Докажите, что массив является массивом Монжа тогда и только тогда, когда для всех $i = 1, 2, \ldots, m - 1$ и $j = 1, 2, \ldots, n - 1$ выполняется условие

$$A[i, j] + A[i + 1, j + 1] \leq A[i, j + 1] + A[i + 1, j]$$

(*Указание:* для прямого доказательства воспользуйтесь методом математической индукции отдельно для строк и для столбцов.)

б. Приведенный ниже массив не является массивом Монжа. Измените в нем один элемент таким образом, чтобы он стал массивом Монжа. (*Указание:* воспользуйтесь п. (а).)

37	23	22	32
21	6	7	10
53	34	30	31
32	13	9	6
43	21	15	8

в. Пусть $f(i)$ представляет собой индекс столбца, содержащего крайний слева минимальный элемент в строке i. Докажите, что $f(1) \leq f(2) \leq \cdots \leq f(m)$ для любого массива Монжа размером $m \times n$.

г. Далее приведено описание алгоритма «разделяй и властвуй», предназначенного для вычисления крайнего слева минимального элемента в каждой строке, входящей в состав массива Монжа A размером $m \times n$.

Постройте подматрицу A' матрицы A, состоящую из четных строк матрицы A. С помощью рекурсивной процедуры найдите в каждой строке матрицы A' крайний слева

минимальный элемент. Затем найдите крайний слева минимальный элемент среди нечетных строк матрицы A.

Объясните, как найти крайний слева минимальный элемент среди нечетных строк матрицы A (предполагается, что крайний слева минимальный элемент среди четных столбцов этой матрицы известен) за время $O(m+n)$.

д. Запишите рекуррентное соотношение, описывающее время работы алгоритма из п. (г). Покажите, что его решение имеет вид □$(m + n \lg m)$.

Контролируемые компетенции: ПК-7

Оценка компетенций осуществляется в соответствии с таблицей 4.

Вопросы к экзамену (5 семестр)

1. Что такое алгоритмы?
2. Алгоритмы как технология.
3. Алгоритм сортировки вставкой.
4. Анализ алгоритма сортировки вставкой.
5. Задача сортировки выбором.
6. Анализ алгоритма сортировки выбором.
7. Разработка алгоритма сортировки слиянием.
8. Анализ алгоритма сортировки слиянием.
9. Асимптотические обозначения.
10. Сравнение функций.
11. Задача поиска максимального подмассива.
12. Метод подстановки решения рекуррентных соотношений.
13. Анализ алгоритма поиска максимального подмассива.
14. Алгоритм Штрассена для умножения матриц.
15. Метод подстановки решения рекуррентных соотношений.
16. Как угадать решение и избежать ошибок.
17. Замена переменных.
18. Метод деревьев рекурсии.
19. Основной метод.
20. Основная теорема о рекуррентных соотношениях.
21. Использование основного метода.
22. Задача о найме.
23. Анализ наихудшего случая в задаче о найме.
24. Вероятностный анализ.
25. Рандомизированные алгоритмы.
26. Индикаторная случайная величина.
27. Лемма о математическом ожидании индикаторной случайной величины.
28. Лемма о математическом ожидании количества наймов.
29. Анализ задачи о найме с помощью индикаторных случайных величин.
30. Задачи о гардеробщике и инверсии массива.
31. Изменения, которые требуется внести в алгоритм найма для рандомизации. Код случайной перестановки.
32. Лемма о математическом ожидании стоимости найма с кодом случайной перестановки.
33. Массивы после случайной перестановки. Лемма о равномерном распределении.
34. Парадокс дней рождения.

35. Анализ с применением индикаторной случайной величины.
36. Случайное наполнение корзин пронумерованными шарами.
37. Последовательность выпадения орлов.
38. Задача о найме в оперативном режиме.
39. Вероятностный подсчет.
40. Поиск в неотсортированном массиве.

Контролируемые компетенции: ПК-7

Оценка компетенций осуществляется в соответствии с таблицей 4.