

**Федеральное государственное автономное образовательное  
учреждение высшего образования  
«Московский физико-технический институт  
(национальный исследовательский университет)»**

**УТВЕРЖДЕНО**  
**Проректор по учебной работе**

**А.А. Воронов**

	<b>Рабочая программа дисциплины (модуля)</b>
<b>по дисциплине:</b>	Алгоритмы и структуры данных
<b>по направлению:</b>	Биотехнология
<b>профиль подготовки:</b>	Биотехнология
	Физтех-школа Биологической и Медицинской Физики кафедра информатики и вычислительной математики
<b>курс:</b>	2
<b>квалификация:</b>	бакалавр

Семестры, формы промежуточной аттестации:

3 (осенний) - Дифференцированный зачет

4 (весенний) - Дифференцированный зачет

Аудиторных часов: 180 всего, в том числе:

лекции: 60 час.

семинары: 0 час.

лабораторные занятия: 120 час.

Самостоятельная работа: 180 час.

Всего часов: 360, всего зач. ед.: 8

Количество контрольных работ, заданий: 4

Программу составили:

Т.Ф. Хирьянов, старший преподаватель

М.Н. Герцев, канд. физ.-мат. наук

Программа обсуждена на заседании кафедры информатики и вычислительной математики 06.02.2020

## Аннотация

Курс является базой для дальнейшего детального изучения языков и средств программирования. Студенты ознакомятся с классическими структурами данных: список, стек, очередь, очередь с приоритетом, ассоциативный массив. Приобретут практику реализации базовых алгоритмов: на графах, перебор с возвратом, жадные алгоритмы. Обучающиеся изучат понятие алгоритмической сложности. Студенты научатся измерять производительность программ, усвоят основные методы их оптимизации.

### 1. Цели и задачи

#### Цель дисциплины

Научить студентов программировать на языке Python 3 на уровне, достаточном для использования ИКТ в курсе вычислительной математики, в исследовательской научной и в последующей профессиональной деятельности.

#### Задачи дисциплины

- обеспечить чёткое понимание студентами основ информатики и ИКТ, включая некоторые области математики (системы счисления, логика, дискретная математика, теория графов);
- сформировать у обучающихся представление о архитектуре ЭВМ, операционной системе и прикладных вычислительных процессах;
- обучить студентов основным алгоритмам обработки числовой и текстовой информации;
- сформировать у обучающихся навык использования языка программирования Python 3 для решения конкретных прикладных задач;
- научить студентов писать программный код коллективно с использованием промышленного стиля программирования и утилит, необходимых при совместной работе над программным продуктом.

### 2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
ОПК-5 Способен участвовать в проведении фундаментальных и прикладных исследований и разработок, самостоятельно осваивать новые теоретические, в том числе, математические методы исследований	ОПК-5.2 Обладает способностью к освоению новых знаний на основе изучения литературы, научных статей и других источников

### 3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны знать:

- основы теории алгоритмов;
- свойства алгоритмов, проблемы алгоритмической сложности и алгоритмической неразрешимости;
- основы дискретной математики;
- основы алгоритмического языка программирования Python;
- общие характеристики интерпретируемых и компилируемых языков программирования;
- идеологию объектно-ориентированного подхода;
- общие понятия о структурах данных: стеки, очереди, списки, деревья, таблицы;
- основы архитектуры электронно-вычислительной машины (ЭВМ), представление информации в ЭВМ и архитектурные принципы повышения их производительности;
- основные принципы устройства и работы операционной системы;
- приёмы разработки программ;
- принципы программирования структур данных для современных программ, типовые решения, применяемые для создания программ;
- основные принципы построения и использования баз данных;
- основы работы с пакетами прикладных программ в области математики и физики.

уметь:

- выбирать оптимальные алгоритмы для современных программ;
- разрабатывать полные законченные программы на одном из языков высокого уровня; программы на одном или нескольких языках программирования, как индивидуально, так и в команде, с использованием современных средств написания и отладки программ;
- применять объектно-ориентированный подход для написания программ;
- использовать знания по информатике для приложений в инновационной, конструкторско-технологической и производственно-технологической сферах деятельности;
- работать как на уровне языка командного интерпретатора, так и с использованием графического пользовательского интерфейса;
- использовать сигналы и оконные сообщения для взаимодействия процессов между собой и с операционной системой;
- создавать безопасные программы, использовать современные средства для написания и отладки программ;
- работать с пакетами прикладных программ, включая использование развитых графических возможностей этих пакетов.

владеть:

- языком программирования Python и методами создания программ с использованием стандартных библиотек;
- средствами отладки программ на Python;
- навыками программирования с использованием средств операционной системы для решения исследовательских задач;
- основами работы с прикладными пакетами Python и принципами написания дополнительных модулей;
- навыками освоения современных архитектур ЭВМ.

#### **4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий**

##### **4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий**

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Основы архитектуры ПК	2		4	3
2	Переменные в Python	2		4	3
3	Однопроходные алгоритмы	2		4	3
4	Условный оператор и основы логики	2		4	3
5	Строки в Python	2		4	3

6	Списки и алгоритмы на списках	2		4	3
7	Цикл for и генераторы списков	2		4	3
8	Функции в языке Python	2		4	3
9	Бисекция и сортировка списка	2		4	5
10	Нерекурсивные сортировки	2		4	5
11	Рекурсия	2		4	5
12	Динамическое программирование	2		4	5
13	Рекурсивные сортировки	2		4	5
14	Множества и словари в Python	2		4	5
15	Обобщение пройденного материала	2		4	36
16	Классы и исключения в Python	2		4	3
17	Стек, дек и очередь	2		4	3
18	Хеш-таблицы	2		4	3
19	Введение в теорию графов	2		4	3
20	Поиск в глубину	2		4	3
21	Поиск в ширину	2		4	3
22	Гамильтонов и Эйлеров цикл	2		4	3
23	Задача о коммивояжёре	2		4	3
24	Орграфы	2		4	5
25	Двоичные деревья поиска	2		4	5
26	Сравнение строк	2		4	5
27	Поиск подстроки в строке	2		4	5
28	Особенности интерпретатора Python	2		4	5
29	Основы языка C++	2		4	5
30	Обобщение курса	2		4	36
Итого часов		60		120	180
Подготовка к экзамену		0 час.			
Общая трудоёмкость		360 час., 8 зач.ед.			

#### 4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

Семестр: 3 (Осенний)

##### 1. Основы архитектуры ПК

Основы архитектуры компьютера. Принципы фон Неймана.  
Операционная система. Место прикладных программ.  
Разделы жесткого диска. Файловая система.  
Виртуальные машины.  
Компиляция и интерпретация.  
Отличие интерпретируемых и компилируемых языков.  
Свободное программное обеспечение. 4 свободы свободного ПО.  
Свободные лицензии: GPL, MIT, BSD, Apache. Почему GPLv3 лучше

##### 2. Переменные в Python

Преимущества и недостатки языка Python 3  
Дзен Python. Antigravity  
Python2 и Python3  
Ресурсы для обучения Python: stepic.com, checkio.org, pythontutor.com  
Концепция присваивания в Python  
Переменные, значения и их типы. Понятие о динамической типизации.

Обмен двух переменных значениями.  
Кортежи и их использование.  
Кортежи переменных. Обмен значений.  
Арифметические операции. Возведение в степень, деление нацело.  
«Hello, World!» на Python

### 3. Однопроходные алгоритмы

Цикл while. Инструкции управления циклом.  
Позиционные системы счисления  
Литералы чисел в Python  
Разложение числа на цифры.  
Однопроходные алгоритмы: подсчёт, сумма, произведение.  
Среднее арифметическое.  
Среднеквадратическое отклонение: однопроходный алгоритм.

### 4. Условный оператор и основы логики

Оператор if. Каскадная условная конструкция elif.  
Логические операции в Python.  
Основы алгебры логики  
Однопроходные алгоритмы: поиск числа в потоке, максимум.  
Тест простоты числа.  
Разложение числа на множители.

### 5. Строки в Python

ASCII и Unicode.  
Тип str. Длина строки len(s). Неизменяемость строки.  
Срезы строк.  
Методы строк find, count, replace, startswith, endswith.  
Наивный поиск подстроки в строке.  
Приведение строки к числу с указанием системы счисления.

### 6. Списки и алгоритмы на списках

Тип list. Изменяемость списка.  
Ссылочная модель данных в Python. Операторы == и is. Копирование объектов.  
Алгоритм обращения массива.  
Алгоритм циклического сдвига в массиве.  
Срезы списков. Присваивание в срез. Методы списка.  
Стандартные функции len, max, min, sum.  
Список строк. Методы split и join для строки.  
Тип tuple как замороженный list.

### 7. Цикл for и генераторы списков

Функция range()  
Цикл for и его особенности в Python.  
List comprehensions: генерация списков.  
Двумерные массивы (списки списков). Вложенная генерация.  
Использование списка как стека. Метод pop()

### 8. Функции в языке Python

Подключение модулей инструкцией `import`

Модуль `math`

Модуль `random`

Запись арифметических выражений в выражения на Python.

Создание функции в Python.

Полиморфизм в Python. Duck typing.

Значения параметров по умолчанию.

Именованные параметры.

## 9. Бисекция и сортировка списка

Поиск корня функции методом бисекции.

Поиск значения в упорядоченном массиве методом бисекции.

Случайное перемешивание массива в Python.

Сортировка обезьяны.

Сортировка вставками.

## 10. Нерекursивные сортировки

Сортировка выбором.

Сортировка методом пузырька.

Сортировка дурака

Сортировка подсчётом.

Поразрядная сортировка.

Прагматическая сортировка `TimSort`.

Формы вызова `A.sort()` и `sorted(A)`.

## 11. Рекурсия

Рекурсия. Прямой и обратный ход рекурсии.

Факториал числа.

Вычисление чисел Фибоначчи.

Проблема алгоритмической сложности задачи.

Ханойские башни.

Генерация всех перестановок (рекурсивная)

Максимальная глубина рекурсии в Python

## 12. Динамическое программирование

Одномерное динамическое программирование.

Двумерное динамическое программирование

Наибольшая общая подпоследовательность.

Наибольшая возрастающая подпоследовательность

Рекурсия с кешированием на примере факториала.

## 13. Рекурсивные сортировки

Рекурсивные сортировки. Быстрая сортировка. Сортировка слиянием.

Модуль `heapq`

Пирамида (куча). Пирамидальная сортировка.

Устойчивость сортировок.

## 14. Множества и словари в Python

Тип `set`. Множества и работа с ними.

Тип dict. Словарь (ассоциативный массив) и операции с ним.

Dict comprehensions: генерация множеств и словарей.

Частотный анализ для строк.

Генераторы, yield.

## 15. Обобщение пройденного материала

Скрипты командной строки на Python.

Анализ аргументов командной строки в Python

Операции с файлами и директориями в Python

## Семестр: 4 (Весенний)

## 16. Классы и исключения в Python

Классы в Python. Перегрузка операторов.

Исключения в Python. Генерирование и перехват исключений.

Списки: односвязный, двусвязный, кольцо.

## 17. Стек, дек и очередь

Стек. Дек.

Очередь.

Очередь с приоритетами. Пирамида (куча).

Очередь событий графического приложения.

## 18. Хеш-таблицы

Хеш-функция. Хеширование.

Открытая хеш-таблица.

Закрытая хеш-таблица.

Проблема удаления из закрытой хеш-таблицы. Перехеширование.

## 19. Введение в теорию графов

Введение в теорию графов.

Взвешенный граф.

Пути в графах.

Расстояние между двумя вершинами.

Графы и способы их представления: список рёбер, матрица смежности, списки смежности

## 20. Поиск в глубину

Определение дерева.

Остовное дерево графа.

Поиск в глубину.

Связность неориентированных графов: выделение компонент связности.

## 21. Поиск в ширину

Поиск в ширину.

Алгоритм Дейкстры.

Восстановление кратчайшего пути.

Алгоритм Флойда-Уоршелла \*

Алгоритм Беллмана-Форда \*

## 22. Гамильтонов и Эйлеров цикл

Построение гамильтонова цикла.

Эйлеров цикл. Эйлеров путь.

Минимальное остовное дерево. Алгоритм Прима.

## 23. Задача о коммивояжёре

Задача о коммивояжере

NP-полные задачи: решение среди экспоненциального множества кандидатов.

Сложные и простые задачи: сравнение нескольких пар задач, которые формулируются похоже, но имеют разную сложность

Приближенные алгоритмы для NP-полных задач.

## 24. Орграфы

Орграфы.

Сильно связанные компоненты.

Поиск в глубину в ориентированных графах.

Ориентированные ациклические графы.

Топологическая сортировка.

## 25. Двоичные деревья поиска

Двоичное дерево поиска.

Декартово дерево («дуча»).

Балансировка деревьев. AVL-дерево. Красно-чёрное дерево.

## 26. Сравнение строк

Проверка равенства строк. Простой и вероятностный алгоритмы.

Вычисление расстояния Левенштейна.

Поиск подстроки в строке.

## 27. Поиск подстроки в строке

Алгоритм Рабина-Карпа

Конечный автомат для поиска подстроки «abcd», «ababc».

Алгоритм Кнута-Морриса-Пратта \*

Алгоритм Ахо-Корасика \*

## 28. Особенности интерпретатора Python

Погружение в Python

lambda

декораторы

Передача функции как объекта

Виртуальная машина Python

Потребление памяти в Python

Проблемы ссылочной модели и глубокого копирования объектов

## 29. Основы языка C++

Введение в язык C++



Плюсы и минусы интерпретируемых языков  
Типы целых чисел языка C++  
Оператор цикла for в C++  
Оператор ветвления if в C++  
Функции в C++  
Использование cpython

### 30. Обобщение курса

Продуктивность C++ против Python  
Производительность работы программ  
Разложение числа на множители  
Оценка производительности модулем statistics  
numpy написан на Си  
Готовая реализация функции на cpython

## 5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

Большая лекционная аудитория, подходящая для учебного потока (факультет, оснащённая мультимедиа проектором и экраном для чтения лекций.  
Учебные аудитории, учебный сетевой компьютерный класс с установленным необходимым программным обеспечением:  
Интерпретатор Python 3, Одна из сред разработки: IDLE, PyCharm Community Edition, IDE Spyder, Microsoft Visual Studio Code.

## 6. Перечень рекомендуемой литературы

### Основная литература

1. Python на практике [Текст], создание качественных программ с использованием параллелизма, библиотек и паттернов/М. Саммерфилд, -М, ДМК Пресс, 2014

### Дополнительная литература

1. Алгоритмы и программы на языках C и PYTHON. Сортировка. Поиск. Строки, Электронная версия печатной публикации / В. В. Прут. — Москва, МФТИ, 2020
2. Программирование на Python 3, подробное руководство/М. Саммерфилд,-СПб, Символ-Плюс, 2020

## 7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

1. <http://python.org>
2. <http://github.com>
3. <http://judge.mipt.ru>
4. <http://acm.mipt.ru>

## 8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)

На ПК в компьютерных классах должно быть установлено следующее ПО:

1. Операционная система GNU/Linux;
2. Интерпретатор Python версии не ниже 3.6;
3. Командный интерпретатор Ipython с отладчиком;
4. Среда разработки JetBrains Python Charm community edition;
5. Среда разработки IDLE;

6. Библиотеки ScyPy для Python 3;
7. Библиотеки PyGame;
8. Библиотеки WxPython и PyQt;
9. Компилятор g++ из коллекции gcc с поддержкой стандарта C++11;
10. Среда разработки JetBrains Clion с учебной лицензией;
11. СУБД SQLite и MySQL.

На лекциях используются мультимедийные технологии, включая демонстрацию презентаций.

Для контроля и коррекции знаний, обучающиеся могут использовать компьютерное тестирование, в том числе на сайте [judge.mipt.ru](http://judge.mipt.ru).

В процессе самостоятельной работы обучающихся возможно использование любые среды программирования.

## **9. Методические указания для обучающихся по освоению дисциплины (модуля)**

Изложение материала происходит преимущественно на лекциях, сопровождается мультимедиа-презентацией с примерами кода и блок-схемами алгоритмов. На лабораторных занятиях также происходит изложение нового материала: в начале каждой лабораторной работы, по мере необходимости, а также в личных беседах тьютора с рабочими группами. На контестах изложение нового материала исключено, преподаватель оказывает только консультативное содействие для успешного решения задач.

Учёт, контроль и оценка знаний студентов.

В течение лабораторной работы успеваемость отслеживается автоматически — по результатам констестов, а также тьютором по своевременности сдач лабораторных работ. Таким образом достигается раннее выявление отстающих студентов с передачей докладных в деканат.

Посещаемость лекций не отмечается, но каждый констест завязан на материал прошедшей лекции, что делает посещение лекций насущной необходимостью в течение семестра.

Дифференцированный зачёт принимает лектор в устной форме с учётом оценки по констестам и оценки по практике. Устный ответ практически исключает списывание, показывает владение базовой терминологией предмета, умение говорить на языке информатики, а также позволяет проверить знание сложных алгоритмов, которые долго программируются, но могут быть относительно легко устно объяснены.

Самостоятельная домашняя работа предполагается после каждой лабораторной работы.

Дополнительная литература:

1. Програмируем на Python. Майкл Доусон. Издательство: Питер ISBN 978-5-459-00314-7, 978-1435455009; 2012 г.
2. Вычислительная математика на смартфонах, коммуникаторах и ноутбуках с использованием программных сред Python. Игорь Соловьев, Александр Червяков, Андрей Репин. Издательство: Лань. ISBN 978-5-8114-1120-7; 2011 г.
3. Python. Создание приложений. Уэсли Дж. Чан. Издательство: Вильямс. ISBN 978-5-8459-1793-5 , 978-0-13-267820-9; 2015 г.
4. Python. Карманный справочник. Марк Лутц. Издательство: Вильямс ISBN 978-5-8459-1965-6; 2014 г.

**ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)**

<b>по направлению:</b>	Биотехнология
<b>профиль подготовки:</b>	Биотехнология Физтех-школа Биологической и Медицинской Физики кафедра информатики и вычислительной математики
<b>курс:</b>	2
<b>квалификация:</b>	бакалавр

Семестры, формы промежуточной аттестации:

- 3 (осенний) - Дифференцированный зачет
- 4 (весенний) - Дифференцированный зачет

**Разработчики:**

Т.Ф. Хирьянов, старший преподаватель  
М.Н. Герцев, канд. физ.-мат. наук

## 1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
ОПК-5 Способен участвовать в проведении фундаментальных и прикладных исследований и разработок, самостоятельно осваивать новые теоретические, в том числе, математические методы исследований	ОПК-5.2 Обладает способностью к освоению новых знаний на основе изучения литературы, научных статей и других источников

## 2. Показатели оценивания компетенций

В результате изучения дисциплины «Алгоритмы и структуры данных» обучающийся должен:

### знать:

- основы теории алгоритмов;
- свойства алгоритмов, проблемы алгоритмической сложности и алгоритмической неразрешимости;
- основы дискретной математики;
- основы алгоритмического языка программирования Python;
- общие характеристики интерпретируемых и компилируемых языков программирования;
- идеологию объектно-ориентированного подхода;
- общие понятия о структурах данных: стеки, очереди, списки, деревья, таблицы;
- основы архитектуры электронно-вычислительной машины (ЭВМ), представление информации в ЭВМ и архитектурные принципы повышения их производительности;
- основные принципы устройства и работы операционной системы;
- приёмы разработки программ;
- принципы программирования структур данных для современных программ, типовые решения, применяемые для создания программ;
- основные принципы построения и использования баз данных;
- основы работы с пакетами прикладных программ в области математики и физики.

### уметь:

- выбирать оптимальные алгоритмы для современных программ;
- разрабатывать полные законченные программы на одном из языков высокого уровня; программы на одном или нескольких языках программирования, как индивидуально, так и в команде, с использованием современных средств написания и отладки программ;
- применять объектно-ориентированный подход для написания программ;
- использовать знания по информатике для приложений в инновационной, конструкторско-технологической и производственно-технологической сферах деятельности;
- работать как на уровне языка командного интерпретатора, так и с использованием графического пользовательского интерфейса;
- использовать сигналы и оконные сообщения для взаимодействия процессов между собой и с операционной системой;
- создавать безопасные программы, использовать современные средства для написания и отладки программ;
- работать с пакетами прикладных программ, включая использование развитых графических возможностей этих пакетов.

### владеть:

- языком программирования Python и методами создания программ с использованием стандартных библиотек;
- средствами отладки программ на Python;
- навыками программирования с использованием средств операционной системы для решения исследовательских задач;
- основами работы с прикладными пакетами Python и принципами написания дополнительных модулей;
- навыками освоения современных архитектур ЭВМ.

### **3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю**

Контроль освоения обучающимися учебного материала осуществляется по средствам проведения учебных констестов и устный опрос в начале занятия по теме прошлого занятия.

### **3. Перечень типовых контрольных заданий, используемых для оценки знаний, умений, навыков**

Примерный перечень контрольных вопросов по теории за 3-й семестр:

1. Основы архитектуры компьютера. Принципы фон Неймана.
2. Отличие интерпретируемых и компилируемых языков.
3. Концепция присваивания в Python
4. Обмен двух переменных значениями.
5. Кортежи и их использование.
6. Цикл while. Инструкции управления циклом.
7. Позиционные системы счисления
8. Однопроходные алгоритмы: подсчёт, сумма, произведение.
9. Оператор if. Каскадная условная конструкция elif.
10. Логические операции в Python.
11. Основы алгебры логики
12. Однопроходные алгоритмы: поиск числа в потоке, максимум.
13. Тест простоты числа.
14. Разложение числа на множители.
15. Тип str. Длина строки len(s). Неизменяемость строки.
16. Срезы строк.
17. Методы строк find, count, replace, startswith, endswith.
18. Наивный поиск подстроки в строке.
19. Ссылочная модель данных в Python. Операторы == и is. Копирование объектов.
20. Алгоритм обращения массива.
21. Алгоритм циклического сдвига в массиве.
22. Срезы списков. Присваивание в срез. Методы списка.
23. Список строк. Методы split и join для строки.
24. Цикл for и его особенности в Python.
25. Listcomprehensions: генерация списков.
26. Двумерные массивы (списки списков). Вложенная генерация.
27. Полиморфизм в Python. Ducktyping.
28. Именованные параметры.
29. Поиск корня функции методом бисекции.
30. Поиск значения в упорядоченном массиве методом бисекции.
31. Сортировка обезьяны.
32. Сортировка вставками.
33. Сортировка выбором.
34. Сортировка методом пузырька.
35. Сортировка дурака
36. Сортировка подсчётом.
37. Поразрядная сортировка.
38. Прагматическая сортировка TimSort.
39. Рекурсия. Прямой и обратный ход рекурсии.
40. Проблема алгоритмической сложности задачи.
41. Ханойские башни.
42. Генерация всех перестановок (рекурсивная)
43. Одномерное динамическое программирование.
44. Двумерное динамическое программирование
45. Рекурсия с кэшированием на примере факториала.
46. Рекурсивные сортировки. Быстрая сортировка. Сортировка слиянием.
47. Пирамида (куча). Пирамидальная сортировка.
48. Устойчивость сортировок.
49. Проверка равенства строк. Простой и вероятностный алгоритмы.

50. Вычисление расстояния Левенштейна.
51. Поиск подстроки в строке.

Примерный перечень контрольных вопросов по теории за 4-й семестр:

1. Тип set. Множества и работа с ними.
2. Тип dict. Словарь (ассоциативный массив) и операции с ним.
3. Dictcomprehensions: генерация множеств и словарей.
4. Частотный анализ для строк.
5. Исключения в Python. Генерирование и перехват исключений.
6. Списки: односвязный, двусвязный, кольцо.
7. Стек. Дек.
8. Очередь.
9. Очередь с приоритетами. Пирамида (куча).
10. Очередь событий графического приложения.
11. Хеш-функция. Хеширование.
12. Открытая хеш-таблица.
13. Закрытая хеш-таблица.
14. Проблема удаления из закрытой хеш-таблицы. Перехеширование.
15. Взвешенный граф.
16. Расстояние между двумя вершинами.
17. Графы и способы их представления: список рёбер, матрица смежности, списки смежности
18. Определение дерева.
19. Поиск в глубину.
20. Связность неориентированных графов: выделение компонент связности.
21. Поиск в ширину.
22. Алгоритм Дейкстры.
23. Алгоритм Форда–Беллмана
24. Алгоритм Флойда–Уоршелла.
25. Восстановление кратчайшего пути.
26. Построение гамильтонова цикла.
27. Эйлеров цикл. Эйлеров путь.
28. Минимальное остовное дерево. Алгоритм Прима.
29. Алгоритм Краскала.
30. Задача о коммивояжере
31. Орграфы.
32. Топологическая сортировка.
33. Двоичное дерево поиска.
34. Декартово дерево («дуча»).
35. Балансировка деревьев. AVL-дерево. Красно-чёрное дерево.
36. Алгоритм Рабина-Карпа
37. Конечный автомат для поиска подстроки «abcd», «ababc».

На дифференцированном зачёте предлагается ответить на два-три вопроса по теории и решить одну короткую алгоритмическую задачу на бумаге без использования компьютера.

Пример задания на устном зачёте:

1. Сортировка обезьяны.
2. Рекурсия. Прямой и обратный ход рекурсии.
3. Тип set. Множества и работа с ними.
4. Задача: реализовать слияние двух отсортированных списков.

#### **4. Критерии оценивания**

Промежуточная аттестация по дисциплине «Алгоритмы и структуры данных» осуществляется в форме дифференцированного зачета.

Дифференцированный зачёт принимает лектор в устной форме с учётом оценки по контестам и оценки по лабораторному практикуму. Устный ответ практически исключает списывание, показывает владение базовой терминологией предмета, умение говорить на языке информатики, а также позволяет проверить знание сложных алгоритмов, которые долго программируются, но могут быть относительно легко устно объяснены.

Оценка по десятибалльной шкале за работу на лабораторном практикуме выставляется преподавателем практикума исходя из количества и качества выполненных практических работ за семестр. Оценка за выполнение контестов выставляется автоматически исходя из суммарного рейтинга обучающегося в системе Ejudgeи также нормируется к десятибалльной шкале.

Итоговая оценка за зачёт не может отличаться от среднего арифметического оценок по контестам и по практическим лабораторным работам более чем на три балла.

#### **5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности**

Время проведения зачёта составляет 30 минут на одного обучающегося.

Во время подготовки к ответу обучающиеся не могут пользоваться литературой, печатными материалами, рукописными записями, а также электронными средствами (сотовыми телефонами, планшетами, умными часами и т.п.).