БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

УТВЕРЖДАЮ
Проректор по учебной работе
и образовательным инновациям
_____О.Г. Прохоренко
«05» июля 2023 г.
Регистрационный № УД – 1301/ б.

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

Учебная программа учреждения высшего образования по учебной дисциплине для специальности:

6-05-0533-08 Компьютерная математика и системный анализ

2023 г.

Учебная программа составлена на основе образовательного стандарта специальности 6-05-0533-08 «Компьютерная математика и системный анализ» ОСВО 6-05-0533-08-2023, учебных планов БГУ № 6-5.4-56/01 от 15.05 2023, № 6-5.4-56/11ин. от 31.05.2023.

СОСТАВИТЕЛИ:
К. Г. Атрохов, старший преподаватель кафедры дифференциальных уравнений и системного анализа Белорусского государственного университета.

РЕЦЕНЗЕНТЫ:
С. А. Вельченко, старший преподаватель кафедры веб-технологий и компьютерного моделирования Белорусского государственного университета

В. А. Карпович, директор ООО «БелБиАйГрупп»

РЕКОМЕНДОВАНА К УТВЕРЖДЕНИЮ:
Кафедрой дифференциальных уравнений и системного анализа БГУ
(протокол № 13 от 14.06.2023)

Научно-методическим советом БГУ
(протокол № 9 от 29.06.2023)

Зав. кафедрой дифференциальных уравнений
и системного анализа Л. Л. Голубева

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Цели и задачи учебной дисциплины

Цель учебной дисциплины «Алгоритмы и структуры данных» – формирование знаний и практических навыков по написанию программ на языке Python (версии 3) с использованием основных структур данных и алгоритмических парадигм. При этом для освоения дисциплины предварительное знание языка Python не требуется, а обучение ведется через решение значительного количества прикладных задач, охватывающих основы программирования.

Задачи учебной дисциплины:

1. Изучение основ программирования на языке Python;
2. Изучение основных структур данных и их реализации в Python;
3. Приобретение навыков по использованию алгоритмов и алгоритмических парадигм для решения практических задач.

Место учебной дисциплины. В системе подготовки специалиста с высшим образованием учебная дисциплина относится **к модулю** «Компьютерная математика» государственного компонента.

Её преподавание тесно связано с дисциплинами «Компьютерная математика» и «Методы программирования».

Требования к компетенциям

Освоение учебной дисциплины «Алгоритмы и структуры данных» должно обеспечить формирование следующих компетенций:

универсальные компетенции:

УК – 2. Решать стандартные задачи профессиональной деятельности на основе применения информационно-коммуникационных технологий.

базовые профессиональные компетенции:

БПК-3. Применять математический аппарат в интеграции с компьютерными средами для создания и исследования моделей различных уровней абстракции.

В результате изучения учебной дисциплины студент должен:

знать:

- О-асимптотику и классы сложности алгоритмов;
- основные структуры данных (список, кортеж, стек, очередь, бинарная куча, словарь, множество, граф), их реализацию в Python и оценку сложности операций над ними;
- вычислительные алгоритмы (алгоритм Евклида, перевод числа в другую систему счисления, построение таблицы истинности, разложение числа на множители, приведение матрицы к ступенчатому виду, матричное произведение, метод Гаусса);
- приближенные вычислительные алгоритмы (метод Ньютона, разложение числа пи, численное интегрирование, метод Монте-

Карло);

- основные алгоритмические парадигмы (метод грубой силы, жадная стратегия, бинарный поиск, рекурсия, перебор с возвратом, «разделяй и властвуй», динамическое программирование);
- основные методы сортировки (пузырьком, вставками, выбором, слиянием, быстрая сортировка, сортировка кучей);
- основные алгоритмы на графах (обход в ширину, обход в глубину, топологическая сортировка, алгоритм Дейкстры);

уметь:

- составлять алгоритмы решения практических задач;
- писать программы и алгоритмы на языке Python (версии 3);
- анализировать эффективность алгоритмов и выбирать наиболее подходящий в конкретной ситуации;

владеть:

- навыками создания программ с использованием алгоритмов и алгоритмических парадигм для решения практических задач на языке Python;
- умением работать с различными инструментами для анализа и оптимизации алгоритмов.

Структура учебной дисциплины

Дисциплина изучается во втором семестре. Всего на изучение учебной дисциплины «Алгоритмы и структуры данных» отведено:

– в очной форме получения высшего образования: 90 часов, в том числе 50 аудиторных часов, из них: лекции – 16 часов, лабораторные занятия – 30 часов, управляемая самостоятельная работа – 4 часа.

Трудоемкость учебной дисциплины составляет 3 зачетные единицы.

Форма промежуточной аттестации по учебной дисциплине – зачет.

СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

Тема 1. Создание простейших программ

Язык программирования Python. Встроенные типы данных. Ветвления и циклы. Случайные числа. Алгоритм Евклида. Нахождение нулей функции. Метод Монте-Карло. Системы счисления. Разложение числа на множители.

Тема 2. Массивы и функции

Списки, кортежи и операции над ними. Вложенные циклы. Функции и модули. Вычисление средних. Биномиальные коэффициенты. Решето Эратосфена. Численное интегрирование. Матричное произведение. Метод Гаусса.

Тема 3. Интерактивный ввод, файлы и графика

Работа с файлами. Обработка данных. Простейшая графика и анимация. Линейный и бинарный поиск. Метод грубой силы. Частотность символов и слов.

Тема 4. Сложность алгоритмов, рекурсия, методы сортировки

Алгоритмы и их сложность. Асимптотический анализ. Рекурсивные алгоритмы. Принцип "Разделяй и властвуй". Генерация перестановок. Возведение в степень. Сортировки квадратичной сложности. Сортировка слиянием. Быстрая сортировка.

Тема 5. Стек, очередь, бинарная куча

Стек. Очередь. Очередь с приоритетом. Бинарная куча. Вычисление арифметических выражений. Сортировка кучей.

Тема 6. Словари, множества, графы

Хеширование. Словари. Множества. Представление графов. Поиск в ширину и глубину. Топологическая сортировка. Поиск кратчайших путей в графе. Алгоритм Дейкстры.

Тема 7. Жадные алгоритмы, перебор с возвратом, динамическое программирование

Жадные алгоритмы. Динамическое программирование. Построение минимального остовного дерева (алгоритм Прима). Перебор с возвратом. Размен монет. Поиск наибольшей общей подпоследовательности. Задача о N ферзях. Построение лабиринта и поиск пути в нем. Задача о рюкзаке 0-1.

УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА УЧЕБНОЙ ДИСЦИПЛИНЫ

Очная форма получения высшего образования с применением дистанционных образовательных технологий (ДОТ)

Номер раздела, темы	Название раздела, темы	Количество аудиторных часов					Количество часов УСР	Форма контроля знаний
		Лекции	Практические занятия	Семинарские	Лабораторные занятия	Иное		
1	2	3	4	5	6	7	8	9
	Алгоритмы и структуры данных	**16**			**30**		**4**	
1	Создание простейших программ	2			4			Отчет по лабораторной работе с устной защитой
2	Массивы и функции	2			4			Отчет по лабораторной работе с устной защитой
3	Интерактивный ввод, файлы и графика	2			4		2	Отчет по лабораторной работе с устной защитой, контрольная работа
4	Сложность алгоритмов, рекурсия, методы сортировки	2			4		2	Отчет по лабораторной работе с устной защитой, контрольная работа
5	Стек, очередь, бинарная куча	2			4			Отчет по лабораторной работе с устной защитой

6	Словари, множества, графы	2			4			Отчет по лабораторной работе с устной защитой
7	Жадные алгоритмы, перебор с возвратом, динамическое программирование	4			6			Отчет по лабораторной работе с устной защитой

ИНФОРМАЦИОННО-МЕТОДИЧЕСКАЯ ЧАСТЬ

Перечень основной литературы

1. Котов, В.М. Теория алгоритмов. Организация перебора и приближенные алгоритмы : учебно-методическое пособие для студ. учреждений высшего образования, обуч. по спец. "Информатика" / В. М. Котов, Е. П. Соболевская, Г. П. Волчкова ; БГУ. - Минск : БГУ, 2022. - 151 с. - URL: https://elib.bsu.by/handle/123456789/312810.
2. Павлов, Л. А. Структуры и алгоритмы обработки данных : учебник / Л. А. Павлов, Н. В. Первова. - Изд. 2-е, испр. и доп. - Санкт-Петербург ; Москва ; Краснодар : Лань, 2020. - 254 с. - URL: https://e.lanbook.com/book/156929.
3. Апанасевич, С. А. Структуры и алгоритмы обработки данных. Линейные структуры : учебное пособие / С. А. Апанасевич. - Санкт-Петербург ; Москва ; Краснодар : Лань, 2019. - 134 с. - URL: https://e.lanbook.com/book/206261.
4. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для вузов, для студентов высших учебных заведений, обучающихся по инженерно-техническим направлениям / Д. Ю. Федоров. - 3-е изд., перераб. и доп. - Москва : Юрайт, 2021. - 210 с.
5. Игнашева, Е. П. Системы счисления, алгоритмизация и программирование : учебное пособие / Е. П. Игнашева ; Черноморское высшее военно-морское училище имени П. С. Нахимова. - Москва : ИНФРА-М, 2020. - 224 с. - URL: https://znanium.com/catalog/product/1078360.

Перечень дополнительной литературы

1. Ахмад И. 40 алгоритмов, которые должен знать каждый программист на Python / Пер. с англ. — СПб: Питер, 2023.
2. Дасгупта С., Пападимитриу Х., Вазирани У. Алгоритмы / Пер. с англ. — М.: МЦНМО, 2014.
3. Красиков И. В., Красикова И. Е. Алгоритмы. Просто как дважды два — М.: Эксмо, 2007.
4. Курносов М. Г. Введение в методы машинной обработки данных — Новосибирск: Автограф, 2020.
5. Круз Р. Л. Структуры данных и проектирование программ, 2-е издание / Пер. с англ. — М.: БИНОМ. Лаборатория знаний, 2014.
6. Рафгарден Т. Совершенный алгоритм. Основы / Пер. с англ. — СПб: Питер, 2019.
7. Рафгарден Т. Графовые алгоритмы и структуры данных / Пер. с англ. — СПб: Питер, 2019.
8. Рафгарден Т. Совершенный алгоритм. Жадные алгоритмы и динамическое программирование / Пер. с англ. — СПб: Питер, 2020.
9. Рафгарден Т. Совершенный алгоритм. Алгоритмы для NP-трудных задач / Пер. с англ. — СПб: Питер, 2021.

10. Рубио-Санчес М. Введение в рекурсивное программирование / Пер. с англ. — М.: ДМК Пресс, 2019.
11. Седжвик Р., Уэйн К., Дондеро Р. Программирование на языке Python: учебный курс / Пер. с англ. — СПб: ООО Альфа-книга, 2017.
12. Скиена С. С. Алгоритмы. Руководство по разработке / Пер. с англ. — СПб: Питер, 2022.
13. Стивенс Р. Алгоритмы. Теория и практическое применение / Пер. с англ. — М.: Издательство Э, 2016.
14. Феррейра Фило В. Теоретический минимум по Computer Science. Все, что нужно программисту и разработчику / Пер. с англ. — СПб: Питер, 2022.
15. Хайнеман Дж. Алгоритмы. С примерами на Python / Пер. с англ. — СПб: Питер, 2023.
16. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих / Пер. с англ. — СПб: Питер, 2017.
17. Кормен Т. Х., Лейзерсон Ч. И., Ривест Р. Л., Штайн К. Алгоритмы: построение и анализ, 3-е издание / Пер. с англ. — М.: Вильямс, 2019.
18. Лааксонен А. Олимпиадное программирование. Изучение и улучшение алгоритмов на соревнованиях / Пер. с англ. — М.: ДМК Пресс, 2018.

Рекомендуемое учебно-лабораторное оборудование

Для проведения занятий требуется исполняемая среда Python 3.

Перечень рекомендуемых средств диагностики и методика формирования итоговой отметки

Объектом диагностики компетенций студентов являются знания, умения, полученные ими в результате изучения учебной дисциплины. Выявление учебных достижений студентов осуществляется с помощью мероприятий текущего контроля и промежуточной аттестации.

Для диагностики компетенций могут использоваться следующие средства текущего контроля: отчет по лабораторной работе, контрольная работа.

Задания к лабораторным и контрольным работам составляются согласно содержанию учебного материала.

При защите лабораторных работ оценивается корректность программного кода, его производительность, оптимальный выбор структур данных, соответствие ввода и вывода заданной спецификации, самостоятельность выполнения заданий. Также ценится знание студентом теоретических сведений, полученных на лекциях, поэтому студенту при выполнении лабораторных заданий необходимо знание лекционных материалов.

Формой промежуточной аттестации по дисциплине «Алгоритмы и структуры данных» учебным планом предусмотрен **зачет.**

Зачет по дисциплине проходит в форме контрольного опроса в устной или письменной форме, выполнения заданий на компьютере.

Примерный перечень заданий для управляемой самостоятельной работы студентов

Тема 3. Интерактивный ввод, файлы и графика (2 часа)

Интерактивный ввод данных позволяет вести диалог с программой.
Задание. Напишите программу oracle.py, которая использует бинарный поиск для отгадывания заданного пользователем числа. На входе подается число *n*, после чего программа сначала предлагает задать целое число не превышающее *n*, а затем последовательно выводит свою догадку, а пользователь отвечает либо 0 (меньше), либо 1 (больше), либо 2 (угадано). Пользователь может запутаться, что приведет к логическому противоречию – отследите это.

Python считывает ввод пользователя при помощи функции `input()`. Чтобы организовать диалог, вызовите эту функцию в бесконечном цикле, сравнивая ее значение со случайным числом, ранее “загаданным” компьютером и сообщая пользователю, что больше. Выход из цикла произойдет после угадывания числа.

Ниже приведен пример вывода программы.

```
% python3 oracle.py 80
Загадайте целое число от 1 до 80.

Это число 40?
[0 = число < 40, 1 = число > 40, 2 = угадано]
1

Это число 60?
[0 = число < 60, 1 = число > 60, 2 = угадано]
1

Это число 70?
[0 = число < 70, 1 = число > 70, 2 = угадано]
1

Это число 75?
[0 = число < 75, 1 = число > 75, 2 = угадано]
0

Это число 72?
[0 = число < 72, 1 = число > 72, 2 = угадано]
2

Ваше число 72 угадано за 5 попыток.
```

Напишите программу ifs.py, которая реализует систему итерационных функций. Программа считывает файл, содержащий правила построения фрактала, а затем применяет эти правила и анимирует результат.

Ниже приведен пример файла fern.txt для построения папоротника Барнсли:

```
0.01 0.85 0.07 0.07 # вероятности

0.0 0.0 0.5 # матрица для вычисления x
0.85 0.04 0.075
0.2 -0.26 0.4
-0.15 0.28 0.575

0.0 0.16 0.0 # матрица для вычисления y
-0.04 0.85 0.18
0.23 0.22 0.045
0.26 0.24 -0.086
```

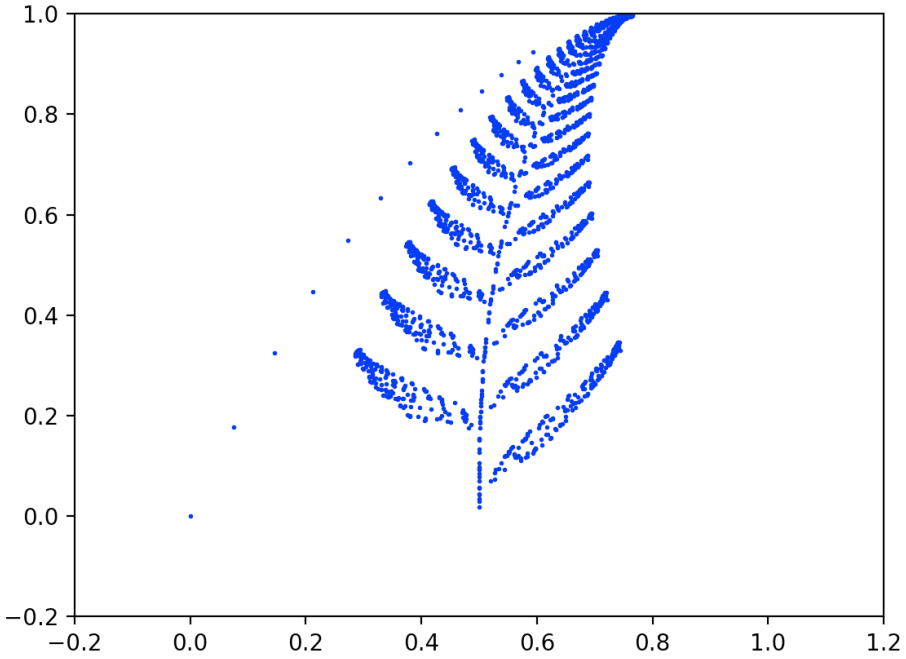
Алгоритм построения фрактала:

- начать с точки (0, 0)
- далее в цикле:
    - выбрать нужную колонку в матрицах для x и y с вероятностями, заданными в первой строке
    - получить новую точку (x, y) применив преобразование с коэффициентами из выбранной колонки

В материалах к лабораторным работам вы найдете файлы с коэффициентами для построения треугольника Серпинского (sierpinsky.txt), папоротника Барнсли (fern.txt) и дерева (tree.txt). Обратите внимание, что матрицы этих фракталов имеют разный размер, и программа должна уметь правильно их считывать.

Ниже показан один из кадров анимации.

```
% python3 ifs.py 'fern.txt'
```



Форма контроля – контрльная работа.

Тема 4. Сложность алгоритмов, рекурсия, методы сортировки (2 часа)

Задание. Напишите программу sorting.py, которая реализует несколько методов сортировки и сравнивает их по времени выполнения:

- `def selection_sort(lst)` – сортировка выбором
- `def bubble_sort(lst)` – пузырьковая сортировка
- `def insertion_sort(lst)` – сортировка вставками
- `def quick_sort(lst)` – быстрая сортировка
- `def merge_sort(lst)` – сортировка слиянием
- `def python_sort(lst)` – встроенная сортировка Python (timsort)

На входе программа получает три целых числа n, m, k. Программа генерирует список длины n случайных целых чисел от 1 до m, сортирует его, после чего выводит время выполнения и k *последних* элементов отсортированного списка (если k = 0, то элементы не выводятся, если k > n, то выводятся все элементы).

Ниже приведен пример вывода программы.

```
% python3 sorting.py 1000 10000 10

bubble sort
0.06780314445495605 [9867, 9870, 9919, 9931, 9931, 9937, 9942,
9944, 9952, 9983]

insertion sort
0.028393030166625977 [9867, 9870, 9919, 9931, 9931, 9937, 9942,
9944, 9952, 9983]
```

```
selection sort
0.019643783569335938 [9867, 9870, 9919, 9931, 9931, 9937, 9942,
9944, 9952, 9983]

quick sort
0.0009739398956298828 [9867, 9870, 9919, 9931, 9931, 9937, 9942,
9944, 9952, 9983]

merge sort
0.0020749568939208984 [9867, 9870, 9919, 9931, 9931, 9937, 9942,
9944, 9952, 9983]

python sort
8.893013000488281e-05 [9867, 9870, 9919, 9931, 9931, 9937, 9942,
9944, 9952, 9983]
```

В этой программе вам нужно сортировать один и тот же массив, но так как массив является изменяемой структурой, то уже после первой сортировки он станет упорядоченным. Поэтому нужно перед вызовом очередного метода создавать копию исходного массива. Ниже приведен пример такого кода на примере встроенной сортировки:

```
def python_sort(lst, k=0):
     print('\npython sort')
     arr = copy.deepcopy(lst)
     t0 = time.time()
     arr.sort()
     if k > 0:
          print(time.time() - t0, arr[-k-1:-1])
     else:
          print(time.time() - t0)
…
python_sort(numbers, k)
```

Форма контроля – контрольная работа.

Примерная тематика лабораторных занятий

Лабораторная работа № 1. Создание простейших программ

Знакомство с языком программирования Python. Написание и запуск программ. Прием и обработка аргументов. Переменные, присваивания и сравнения. Типы данных. Ветвления и циклы. Модули math, random. Вычисление логических выражений. Генерация случайных чисел. Поиск кратчайшего расстояния на глобусе. Алгоритм Евклида. Нахождение нулей функции. Метод Монте-Карло. Системы счисления. Разложение числа на множители.

Лабораторная работа № 2. Массивы и функции

Списки, кортежи и операции над ними. Вложенные циклы. Двумерные списки и матрицы. Форматированный вывод значений. Функции и модули. Вычисление средних. Построение таблицы истинности. Решето Эратосфена. Вычисление биномиальных коэффициентов. Приведение матрицы к ступенчатому виду. Построение матрицы Хаара. Численное интегрирование. Матричное произведение. Метод Гаусса.

Лабораторная работа № 3. Интерактивный ввод, файлы и графика

Чтение и запись файлов. Обработка текстовых данных. Удаление дубликатов. Частотность символов и слов. Сдвиг субтитров. Построение таблицы турнира. Алгоритмические парадигмы. Линейный и бинарный поиск. Поиск в словаре. Метод грубой силы. «Быки и коровы». Простейшая графика. Пакет matplotlib. Построение розы ветров Минска. Случайное блуждание.

Лабораторная работа № 4. Сложность алгоритмов, рекурсия, методы сортировки

Алгоритмы и их сложность. Асимптотический анализ. Абстрактные структуры данных. Представление списка в Python. Рекурсивные алгоритмы. Генерация перестановок. Возведение в степень. Обход директории. Построение кривой Гильберта. Принцип "Разделяй и властвуй". Сортировка слиянием. Поиск общих суффиксов. Методы сортировки. Сортировки квадратичной сложности. Быстрая сортировка.

Лабораторная работа № 5. Стек, очередь, бинарная куча

Дек, стек, очередь и операции над ними. Реализация стека в виде класса. Обратная польская запись. Вычисление арифметических выражений. Проверка скобочных выражений. Реализация очереди в виде класса. Моделирование работы касс магазина. Структура данных «куча». Реализация бинарной кучи в виде класса. Сортировка кучей.

Лабораторная работа № 6. Словари, множества, графы

Словарь, множество и операции над ними. Мемоизация. Задача о лестнице. Построение LRU кэша. Графы и их представление в Python. Поиск в ширину и глубину. Нахождение кратчайшего расстояния между актерами. Поиск кратчайшего пути в лабиринте. Топологическая сортировка. Поиск кратчайших путей в графе. Алгоритм Дейкстры.

Лабораторная работа № 7. Жадные алгоритмы, перебор с возвратом, динамическое программирование

Жадные алгоритмы. Размен монет. Оптимальное распределение задач. Построение минимального остовного дерева (алгоритм Прима). Построение лабиринта. Перебор с возвратом. Задача о N ферзях. Решение судоку. Поиск пути в лабиринте. Динамическое программирование. Поиск наибольшей общей подпоследовательности. Задача о рюкзаке 0-1.

Описание инновационных подходов и методов к преподаванию учебной дисциплины

При организации образовательного процесса используется ***эвристический подход***, который предполагает демонстрацию многообразия решений большинства профессиональных задач и жизненных проблем.

При организации образовательного процесса используется ***практико-ориентированный подход,*** который предполагает освоение содержания через решения практических задач.

При организации образовательного процесса ***используются методы и приемы развития критического мышления,*** которые представляют собой систему, формирующую навыки работы с информацией в процессе чтения и письма; понимании информации как отправного, а не конечного пункта критического мышления.

Методические рекомендации по организации самостоятельной работы обучающихся

Для организации самостоятельной работы студентов по учебной дисциплине рекомендовано разместить на образовательном портале или сайте кафедры учебно-методические материалы: курсы лекций и лабораторные практикумы, методические указания к лабораторным занятиям, вопросы для подготовки к зачету, перечень рекомендуемой литературы, информационные ресурсы.

Самостоятельная работа студента включает в себя работу с учебной литературой по заданным темам дисциплины, поиск новейшей учебной и научной информации в указанных областях знаний и знакомство с ней, а также выполнение поставленных заданий.

Примерный перечень вопросов к зачёту

Структуры данных –необходимо показать их реализацию в Python, знать основные методы и их производительность (асимптотику):

1. Список
2. Стек
3. Очередь
4. Бинарная куча
5. Словарь
6. Множество
7. Граф

Математические алгоритмы – необходимо рассказать идею, а также быть готовым написать код на Python:

1. Нахождение нулей функции методом Ньютона
2. Вычисление корня числа бинарным поиском
3. Представление числа в заданной системе счисления
4. Разложение числа на множители
5. Генерация булеана (множества всех подмножеств списка)

6. Генерация матрицы Адамара
7. Нахождение списка простых чисел используя решето Эратосфена
8. Приведение матрицы к ступенчатому виду
9. Вычисление интеграла функции (любой из методов)
10. Вычисление произведения матриц
11. Вычисление степени числа при помощи рекурсии
12. Генерация всех перестановок множества элементов

Другие алгоритмы – необходимо рассказать идею, знать асимптотику, а также быть готовым написать код на Python:

1. Бинарный поиск
2. Медленные сортировки: пузырьком, вставками, выбором
3. Сортировка слиянием
4. Быстрая сортировка
5. Вычисление арифметического выражения (обратная польская запись)
6. Вычисление арифметического выражения (инфиксная запись)
7. Реализация очереди с приоритетом при помощи бинарной кучи
8. Сортировка кучей
9. Реализация LRU кэша
10. Обход графа в ширину
11. Обход графа в глубину
12. Топологическая сортировка
13. Нахождение кратчайших расстояний в графе методом Дейкстры
14. Жадная стратегия (на примере размена монет)
15. Построение минимального остовного дерева алгоритмом Прима
16. Перебор с возвратом (бэктрекинг)
17. Динамическое программирование
18. Решение задачи о рюкзаке 0-1

ПРОТОКОЛ СОГЛАСОВАНИЯ УЧЕБНОЙ ПРОГРАММЫ УВО

Название учебной дисциплины, с которой требуется согласование	Название кафедры	Предложения об изменениях в содержании учебной программы УВО по учебной дисциплине	Решение, принятое кафедрой, разработавшей учебную программу (с указанием даты и номера протокола)[1]
Компьютерная математика.	Кафедра дифференциальных уравнений и системного анализа	нет	Вносить изменения не требуется (протокол № 13 от 14.06.2023)

Кафедра дифференциальных уравнений и системного анализа _____ Л. Л. Голубева

[1] При наличии предложений об изменениях в содержании учебной программы УВО.