Алгоритмы для разработчиков

Курс о разработке и использовании разнообразных алгоритмов и структур данных

Длительность курса: 174 академических часа

1 Введение в алгоритмы и структуры данных

1	**Математика для разработчиков**	освежить основные разделы математики, использующиеся в курсе "алгоритмы для разработчиков"
2	**Введение в алгоритмы, RAM-модель. Порядок роста функций.**	Студенты ознакомятся с эмулятором RAM-машины, поймут состав элементарных операций и научатся оценивать сложность алгоритмов. Домашние задания 1 Реализация алгоритма сортировки на RAM Цель: 1. Написать на RAM алгоритм умножения двух целых неотрицательных

чисел через сложение
2. Написать на RAM алгоритм деления двух целых неотрицательных чисел через вычитание
3. Написать на RAM алгоритм простой сортировки

1. Написать на RAM алгоритм умножения двух целых неотрицательных чисел через сложение (+1 балл).
На входе два числа.
На выходе результат умножения.
Оптимизировать количество итераций по меньшему из множителей
Загрузить текст программы на гитхаб и приложить ссылку на репозиторий.
Протестировать работу на следующих данных (+1 балл):
а. 5 * 3 = 15
в. 1111 * 111 = 123321
б. 1 * 1000000 = 1000000 (быстро)
г. 1000000 * 1 = 1000000 (быстро)
д. 123 * 0 = 0
е. 0 * 456 = 0
ё. предложить свой тест (+1 балл)
Написать, какой из приведённых тестов не прошёл с первого раза и как вы исправили ошибку.
Написать, сколько времени ушло на выполнение задания.

2. Написать на RAM алгоритм деления двух целых неотрицательных чисел через вычитание (+1 балл).
На входе делимое и делитель.
На выходе частное и остаток.
Загрузить текст программы на гитхаб и приложить ссылку на репозиторий.
Протестировать работу на следующих данных (+1 балл):
а. 50 / 30 = 1 и 20
б. 25 / 25 = 1 и 0

в. 5 / 30 = 0 и 5
г. 100001 / 2 = 50000 и 1
д. 0 / 123456 = 0 и 0
е. 123456 / 0 = нет ответа (выход пустой, программа остановлена)
ё. предложить свой тест (+1 балл)
Написать, какой из приведённых тестов не прошёл с первого раза и как вы исправили ошибку.
Написать, сколько времени ушло на выполнение задания.

3. Написать на RAM алгоритм простой сортировки (+1 балл).
На входе целое неотрицательное число N и потом N целых чисел.
На выходе N чисел отсортированных по возрастанию.
В доп. материалах есть пример считывания начальных данных.
Псевдокод сортировки:
для каждого i от 0 до N-1
для каждого k от i+1 до N-1
если элемент[i] > элемент[k]
элемент[k] <=> элемент[i];
* Оптимизировать программу, чтобы код был без повторений и лишних действий (+1 балл).
Загрузить текст программы на гитхаб и приложить ссылку на репозиторий.
Протестировать работу на следующих данных (+1 балл):
а. 5 4 3 0 2 1
б. 8 0 1 2 3 4 5 6 7
в. 8 7 6 5 4 3 2 1 0
г. 0
д. 1 8
е. 7 7 7 7 7 7 7 7
ё. предложить свой тест (+1 балл)
Написать, какой из приведённых тестов не прошёл с первого раза и как вы исправили ошибку.
Написать, сколько времени ушло на

выполнение задания.

---

3 **Базовые структуры данных: массив, динамический массив, список, стек, очередь, очередь с приоритетами**

Студенты ознакомятся с реализацией базовых структур данных, и научатся правильно выбирать структуру данных под задачу

Домашние задания

1 Динамические массивы, неполный массив, очередь с приоритетом.

Цель: Создание разных алгоритмов для реализации Динамического массива и сравнение их производительности. Создание приоритетной очереди или неполного массива.

1 задание. Динамические массивы.
Написать метод добавления и удаления элементов:
void add(T item, int index);
T remove(int index); // возвращает удаляемый элемент
по индексу во все варианты динамических массивов:
SingleArray, VectorArray, FactorArray, MatrixArray *.
+3 байта

2 задание. Таблица сравнения производительности.
Сравнить время выполнения разных операций
для разных массивов с разным порядком значений.
* сделать обёртку над ArrayList() и тоже сравнить.
Составить таблицу и приложить её скриншот.
Сделать выводы и сформулировать их в нескольких предложениях.
+3 байта

3 задание. Приоритетная очередь (на выбор).
Написать реализацию PriorityQueue - очередь с приоритетом.
Варианты реализации - список списков, массив списков
Методы к реализации:
enqueue(int priority, T item) - поместить элемент в очередь
T dequeue() - выбрать элемент из очереди
+4 байта

4 задание. Неполный массив (на выбор).
Написать Реализацию класса SpaceArray массив массивов с неполным заполнением.
Делать на основе одного из уже созданных массивов и/или списков.
+4 байта дополнительно.

Напишите сколько времени ушло на домашнее задание.

Для реализации можно использовать только стандартные массивы [], созданные классы массивов и/или списков. Стандартные коллекции не используем!

ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

---

4 **Алгебраические алгоритмы: алгоритм Евклида, быстрое возведение в степень, решето Эратосфена, быстрое вычисление чисел Фибоначчи**

Студенты ознакомятся с использованием и реализацией некоторых популярных алгебраических алгоритмов.

Домашние задания

1 НОД, Степень, Простые числа, Числа Фибоначчи.

Цель: Написать и сравнить разные алгоритмы

нахождения наибольшего общего делителя, возведения числа в целую степень, поиска простых чисел и вычисления чисел фибоначчи.

Написать следующие алгоритмы и сравнить их быстродействие и результаты.
Приложить скриншоты/ссылки на сравнительную таблицу проведённых экспериментов.

На выбор: 1 или 2 задание, а также 3 или 4.

1. Алгоритм Евклида поиска НОД макс. 4 байта
1a. Через вычитание
+1 байт 1b. Через остаток
+1 байт 1c. Через битовые операции
+2 байт Составить сравнительную таблицу времени работы алгоритмов для разных начальных данных.
Написать, какие пункты выполнены и сколько времени ушло на выполнение домашнего задания.

2. Алгоритм возведения в степень макс. 4 байта
2a. Итеративный (n умножений)
+1 байт 2b. Через степень двойки с домножением
+1 байт 2c. Через двоичное разложение показателя степени.
+2 байт Составить сравнительную таблицу времени работы алгоритмов для разных начальных данных.
Написать, какие пункты выполнены и сколько времени ушло на выполнение домашнего задания.

3. Алгоритмы поиска кол-ва простых чисел до N макс. 6 байт
3a. Через перебор делителей.

+1 байт 3b. Несколько оптимизаций перебора делителей, с использованием массива.
+1 байт 3c. Решето Эратосфена со сложностью O(n log log n).
+1 байт 3d. Решето Эратосфена с оптимизацией памяти: битовая матрица, по 32 значения в одном int
+1 байт 3e. Решето Эратосфена со сложностью O(n)
+2 байт Составить сравнительную таблицу времени работы алгоритмов для разных начальных данных.
Написать, какие пункты выполнены и сколько времени ушло на выполнение домашнего задания.

4. Алгоритм поиска чисел Фибоначчи макс. 6 байта
4a. Через рекурсию
+1 байт 4b. Через итерацию
+1 байт 4c. По формуле золотого сечения
+2 байт 4d. Через умножение матриц
+2 байт Составить сравнительную таблицу времени работы алгоритмов для разных начальных данных.
Написать, какие пункты выполнены и сколько времени ушло на выполнение домашнего задания.

ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

---

5 **Шахматное программирование**

Студенты повторят все правила игры в шахматы, получат техническое задание по их кодированию.

Домашние задания

1 Система тестирования, Работа с битами, Генерация ходов.

Цель: Подготовить систему тестирования.

Работа с битами на примере Bitboard.
Парсинг, генерация FEN и выполнение хода.
* Написать алгоритм генерации допустимых ходов.

Задания выполнять только в порядке их расположения!
После выполнения каждого раздела написать в ЧАТ сообщение:
Какие задачи выполнены,
Какие были ошибки и удалось ли их исправить
Сколько времени ушло на выполнение раздела
Приложить ссылку на репозиторий с решением.

Основная часть домашнего задания (+10 байт).

!. Подготовить систему тестирования. +2 байта.
Начальные данные и результаты хранятся в файлах.
Запустить программу, дать на вход начальные данные,
Сохранить результат, сравнить с правильным ответом.
Проверить на примере задачи нахождения длины строки.
Написать, сколько времени ушло на выполнение.

0. Работа с битами на примере Bitboard. +4 байта.
Прочитать статью "Ход конём: bitboard и чёрные дыры".
Решить 4 задачи, проверить решение по тестам.
1744.0.Bitboard - FEN +1 байт.
3717.0.Bitboard - Конь +1 байт.
3718.0.Bitboard - Ферзь +1 байт.

3733.0.Bitboard - Король +1 байт.
Написать, какие задачи выполнены, какие были ошибки, сколько времени ушло на выполнение.

1. Парсинг, генерация FEN и выполнение хода. +4 байта.
Решить 9 последовательных задач, проверить решение по тестам.
1743.1.FEN - ASCII
1745.1.Сборка и разборка +1 байт
1746.1.Счётчик ходов
3694.1.Счётчик полуходов
3711.1.Перемещение фигуры +1 байт
3713.1.Превращение пешки
3714.1.Взятие на проходе +1 байт
3715.1.Королевские флаги
3716.1.Рокировка +1 байт
Написать, какие задачи выполнены, какие были ошибки, сколько времени ушло на выполнение.

Факультативная часть домашнего задания (+20 байт).

2. * Написать алгоритм генерации ходов фигурами. +6 байт.
Решить 7 последовательных задач, проверить решение по тестам.
3719.2.Сортировка ходов +1 байт
3721.2.Ход конём +1 байт
3722.2.Ход слоном +1 байт
3723.2.Ход ладьёй +1 байт
3724.2.Ход ферзём +1 байт
3725.2.Хоровод
3726.2.Все ходы записаны +1 байт
Написать, какие задачи выполнены, какие были ошибки, сколько времени ушло на выполнение.

3. * Написать алгоритм генерации ходов пешками и королём. +6 байт.

Решить 6 последовательных задач,
проверить решение по тестам.
3727.3.Белая пешка +1 байт
3728.3.Белая Королева +1 байт
3729.3.Взятие на проходе +1 байт
3730.3.Чёрные пешки +1 байт
3731.3.Королевский шаг +1 байт
3732.3.Рокировка +1 байт
Написать, какие задачи выполнены, какие
были ошибки, сколько времени ушло на
выполнение.

4. * Реализовать проверку на шах, мат, пат,
ничью. +8 байт.
Решить 6 последовательных задач,
проверить решение по тестам.
3737.4.Шах +1 байт
3738.4.Настоящие ходы +1 байт
3739.4.Мат и пат +1 байт
3740.4.Мат в 1 ход +1 байт
3741.4.Ничья +2 байта
3742.4.Мат в 2 хода +2 байта
Написать, какие задачи выполнены, какие
были ошибки, сколько времени ушло на
выполнение.

2 Сортировки

1 Сортировка вставками, сортировка Шелла, сортировка выбором, пузырьковая сортировка

Студенты освоят алгоритмы сортировки вставками, выбором, пузырьком, сортировку Шелла. По окончании занятия студенты смогут реализовывать и правильно применять данные алгоритмы.

Домашние задания

1 Реализовать Insertion Sort и Shell Sort

Реализовать алгоритмы insertion sort и Shell sort. Дополнительно: протестировать работу алгоритмов на случайных массивах и на практически упорядоченных массивах, сравнить производительность. Сравнить производительнсоть сортировки Шелла с разной последовательностью шагов.

Как сравнивать производительность:

Для алгоритмов, как минимум, Insertion Sort, Shell Sort с двумя вариантами последовательностей шагов, сделать для каждого следующие измерения:

Создать массивы размера от 20, 40, 80, 160 ... и до~100.000 элементов для C++/Java или ~10.000 элементов в случае python. Для каждого размера сделать по три массива: случайный (функцией для shuffle из упорядоченного), массив, в котором перемешаны ~10% элементов, и массив, в котором перемешаны 5 элементов.

Замерять процессорное время для каждого алгоритма и каждого массива, составить табличку в формате .csv либо график, приложить к коду.

Варианты последовательностей "шагов" (gap sequences) Shell Sort можно брать отсюда,

любые, но интереснее будет с O() < O(n^2): https://en.wikipedia.org/wiki/Shellsort#Gap_sequences

ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

---

2 **Пирамидальная сортировка (heap sort), tree sort**

Студенты смогут реализовывать и применять пирамидальную сортировку, tree sort.

Домашние задания

1 Реализация heapsort

- Реализовать *Drown* - 1 балл
- Реализовать *BuildHeap* - 1 балл
- Реализовать алгоритм *Heapsort* - 1 балл
- Можно все inline и без отдельных процедур, тогда 3 балла, если все иделаьно работает
- *Heapsort* должен работать для получения зачета
- Дополнительно 1: реализовать итеративно, а не через рекурсию - 1 балл
- Дополнительно 2: реализовать удаление элемента с сохранением свойст пирамиды - 1 балл
- Дополнительно 3: реализовать очередь с приоритетами при помощи heap, сравнить с тем, что было ранее сделано - без баллов, по своему желанию (т.к. нам сложно гарантировать быструю проверку)

ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

---

3 **Сортировка слиянием, timsort.**

Студенты освоят и смогут реализовать алгоритмы быстрой сортировки, сортировки слиянием и timsort.

Быстрая сортировка

Домашние задания

1 Реализовать quicksort или mergesort, дополнительно - усовершенствовать

Реализовать:

MergeSort или QuickSort (рандомизированный и обычный) -
- алгоритм работает корректно, используется insertion sort: 3 балла
- алгоритм работает корректно, но выполняет много лишних действий / отклоняется от реализации не в лучшую сторону - 2 балл

Дополнительно:
- quicksort - сравнить рандомизированный и обычный варианты - 1 балл
- mergsesort - добавить обработку run'ов - 1 балл

- распараллелить алгоритм - 1 балл (если считаете, что этот материал надо разобрать отдельно - напишите в слак) - 1 балл

- делать оба алгоритма не обязательно, но при желании можно, баллы ставятся за лучший

** ТЕСТИРОВАНИЕ QUICKSORT **
Массивы для тестирования Quicksirt: тестируйте алгоритм так, как объяснено в конце занятия; при этом тестируйте на "случайно" неупорядоченных массивах и отдельно на массивах с упорядоченными данными, которые будут "ломать" Partition и приводить к квадратичному времени работы алгоритма.

ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

4	**Сортировка за линейное время. Поиск порядковых статистик за линейное время.**	Студенты освоят и смогут реализовать сортировку подсчетом, поразрядную сортировку, блочную сортировку. Будет разобран научатся алгоритм для нахождения порядковых статистик за линейное время.

3 Деревья

1 **Двоичные деревья поиска, декартовы деревья, АВЛ-деревья**

Студенты узнают, что такое дерево, двоичные дерево поиска и декартово дерево. Изучат структуры данных и алгоритмы, реализующие операции над ними.

Домашние задания

1 Реализовать АВЛ или Декартово дерево

Вариант 1. Написать реализацию АВЛ-дерева
Методы к реализации:
smallLeft/RightRotation - малое левое/правое вращение
bigLeft/RightRotation - большое левое/правое вращение, написать через вызов малых вращений
insert
remove
rebalance

Вариант 2. Написать реализацию Декартово дерева.
Методы:
merge
split
add
remove

ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

2 **Красно-черные деревья, расширяющиеся деревья, рандомизированные деревья**

Студенты освоят и смогут применять красно-черные деревья, расширяющиеся деревья и рандомизированные деревья

Домашние задания

1 Реализация красно-чёрного дерева, вставки и поиска

Домашнее задание выполняется по желанию.

Вариант A. Реализовать рандомизированное дерево с операциями вставки, поиска и удаления.
Вариант B. Реализовать расширяющееся дерево с операциями вставки и поиска и удаления.
Вариант C. Реализовать красно-чёрное дерево с операциями вставки и поиска. Сравнить производительность операций с AVL деревом.

ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

3 **B-деревья, B+-деревья. Деревья отрезков**

Студенты освоят и смогут применять B-деревья и B+-деревья. Ознакомятся с деревьями отрезков.

Домашние задания

1 Реализовать оптимальное дерево поиска

Реализовать оптимальное дерево поиска
- Алгоритм 1
- Алгоритм 2

Сравнить
1) время построения, включая сортировку
2) время поиска

Тест провести на dataset из прошлого занятия. Включить в глобальный тест (только dataset)

Опционально 1:
Написать реализацию бинарного дерева поиска или расширяющегося дерева или рандомизированного дерева.
Включить в глобальный тест

Опционально 2:
У кого была реализация декартового дерева использовать его как оптимальное дерево поиска. Включить в тест на dataset

ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

4 Хеш-таблицы

1 **Таблицы с прямой адресацией. Хэш-таблицы, хэш-функции. Метод цепочек (chaining).**

Студенты смогут реализовывать хэш-таблицы с прямой адресацией, а также изучат работу хэш-функций и хэш-таблиц. Борьба с коллизиями будет разобрана на примере метода цепочек.

Домашние задания

1 Хэш-таблицы, часть I

- Домашняя работа одна на всю неделю. Это _первая половина_, которую можно начать выполнять сейчас.
- Вторая половина будет после занятия в среду.

1. Реализовать хеш-таблицу, использующую метод цепочек
- дополнительно: для хранения внутри цепочек при достижении значительного числа элементов (~32) заменять их на BST

2. _Или_: реализовать хеш-таблицу с открытой адресацией
- дополнительно: реализовать "ленивое" удаление
- реализовать квадратичный пробинг

ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

2	**Хеш-функции. Стратегии поиска. Универсальное хеширование**	Студенты смогут реализовывать хэш-таблицы с открытой адресацией. Будут рассмотрены различные стратегии поиска, универсальное хеширование, различные хеш-функции. Домашние задания 1 Домашнего задания нет Домашнего задания нет
3	**Универсальное и идеальное хэширование.**	Будет рассмотрено универсальное и идеальное хеширование, области их применения. Домашние задания 1 Домашнего задания нет Домашнего задания нет

5 Графы

1 **Поиск в ширину. Поиск в глубину, поиск компонент сильной связности. Алгоритм Косарайю.**

Студенты освоят, смогут реализовывать и применять поиск в ширину и поиск в глубину. Будут разобраны алгоритмы поиска компонент сильной связности.

Домашние задания

1 Реализовать алгоритм Косарайю

Реализовать алгоритм Косарайю

Входные данные:
Граф задан вектором смежности int A[N][Smax]. Это п.5 в структурах данных в лекции. Отличие только в том, что вершины нумеруются от 0 а не от 1, и номера самой вершины первым столбцом в матрице не будет, будут только номера смежных вершин

Задание:
Реализовать алгоритм Косарайю, рекурсивный вариант, как он был дан в лекции
Если понадобится использование стека/очереди обязательно применение собственных структур данных из предыдущих занятий

Выходные данные:
Результат должен быть представлен в виде массива int[] component где элемент с номером вершины содержит номер компонента

Дополнительное задание 1
Реализовать итеративный поиск в глубину с сохранением состояния, что бы уже пройденные уровни повторно не проходились

Дополнительное задание 2
Реализовать поиск по критерию стоимости

ВАЖНО! При размещении ответа укажите, на

каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

---

2 **Топологическая сортировка**

Студенты освоят, смогут реализовывать и применять топологическую сортировку.

Домашние задания

1 Алгоритм Демукрона

Реализовать алгоритм Демукрона

Граф задан вектором смежности int A[N][Smax]. Это п.5 в структурах данных в лекции. Отличие только в том, что вершины нумеруются от 0 а не от 1, и номера самой вершины первым столбцом в матрице не будет, будут только номера смежных вершин

Задание:
Реализовать алгоритм Демукрона
Если понадобится использование стека/очереди обязательно применение собственных структур данных из предыдущих занятий
Можно использовать стандартный массив [] встроенный в язык

Выходные данные:
Результат должен быть представлен в виде массива int[][] level где первый индекс - номер уровня, на каждом уровне массив, с номерами вершин, принадлежащих этому уровню

Дополнительное задание 1
Реализовать алгоритм Тарьяна

Дополнительное задание 2
Реализовать алгоритм поиска мостов или точек сочленения

ВАЖНО! При размещении ответа укажите, на

каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

3	**Минимальные остовные деревья. Алгоритмы Крускала и Прима**	Студенты освоят, смогут реализовывать и применять алгоритмы нахождения минимальных остовных деревьев.
		
		Домашние задания
		
		1 Реализовать алгоритм нахождения минимального остовного дерева
		
		Реализовать алгоритм Краскала
		
		Граф задан вектором смежности int A[N][Smax]. Это п.5 в структурах данных в лекции. Отличие только в том, что вершины нумеруются от 0 а не от 1, и номера самой вершины первым столбцом в матрице не будет, будут только номера смежных вершин
		
		Задание:
		Реализовать алгоритм Краскала
		Структура Union-Find собственной реализации.
		Если понадобится использование стека/очереди обязательно применение собственных структур данных из предыдущих занятий
		Можно использовать стандартный массив [] встроенный в язык
		
		Выходные данные:
		Результат должен быть представлен в виде массива Edge[] edges где Edge - класс, содержащий пару вершин, которые соединяет это ребро
		Edge
		{
		int v1;
		int v2;
		}
		Для любителей компактного хранения можно

упаковать в long два int-а :)
Тогда результат будет long[] edges

Дополнительное задание 1
Реализовать алгоритм Прима

Дополнительное задание 2
Реализовать алгоритм Борувки

ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

---

4 **Поиск кратчайшего пути в графе. Алгоритмы Дейкстры, Беллмана-Форда, Флойда-Уоршалла**

Студенты освоят, смогут реализовывать и применять алгоритмы поиска кратчайшего пути в графе.

Домашние задания

1 Реализовать алгоритм Дейкстры

Реализовать классику всех времен и народов, алгоритм Дейкстры :)

Граф задан вектором смежности int A[N][Smax]. Это п.5 в структурах данных в лекции. Отличие только в том, что вершины нумеруются от 0 а не от 1, и номера самой вершины первым столбцом в матрице не будет, будут только номера смежных вершин

Задание:
Реализовать алгоритм Дейкстры
Если понадобится использование дерева/кучи обязательно применение собственных структур данных из предыдущих занятий
Можно использовать стандартный массив [] встроенный в язык

Выходные данные:
Результат должен быть представлен в виде массива Edge[] edges где Edge - класс, содержащий пару вершин, которые соединяет

это ребро
Edge
{
int v1;
int v2;
}
Для любителей компактного хранения можно упаковать в long два int-а :)
Тогда результат будет long[] edges

Дополнительное задание 1
"Расскажи своей бабушке".
Рассказать идею алгоритма Дейкстры совей бабушке так, что бы она это поняла. Поделиться своим опытом в слаке. Не забыть приложить ссылку на пост в задании

Дополнительное задание 2
Реализовать алгоритм Флойда-Уоршалла или Беллама-Форда на выбор

ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

5 **Алгоритмы Джонсона, A*, и способы решения задачи коммивояжера**

Студенты освоят, смогут реализовывать и применять алгоритмы Джонсона, A* и ознакомятся со способами решения задачи коммивояжера.

Домашние задания

1 Предложить вариант алгоритма для решения задачи

Задача: Нужно быстро строить кратчайший маршрут на большие расстояния по реальной дорожной сети, например от Лиссабона до Владивостока. Можно взять данные OSM.

Предложить вариант решения задачи, работающий быстрее, чем применение A* влоб.

Идею и описание алгоритма прислать в виде небольшой записки.

Дополнительное задание 1: Закодить алгоритм A*
Дополнительное задание 2: Закодить алгоритм Джонсона

ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

---

6 **Heap manager, Garbage collector**

Будет рассмотрена работа garbage collector'a на примере современных языков программирования.

6 Алгоритмы на строках

1	**Алгоритм Бойера-Мура**	Студенты освоят, смогут реализовывать и применять алгоритм Бойера-Мура.
2	**Алгоритм Кнута-Морриса-Пратта**	Студенты освоят, смогут реализовывать и применять алгоритм Кнута-Морриса-Пратта. Домашние задания 1 Реализовать алгоритмы Бойера-Мура и Кнута-Морриса-Пратта Цель: ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.
3	**Алгоритм Ахо-Корасика**	Студенты освоят, смогут реализовывать и применять алгоритм Ахо-Корасика.
4	**Код Хаффмана, алгоритм Лемпела-Зива. Run-length encoding.**	Будет разобран run-length encoding (RLE). Студенты освоят кодирование Хаффмана, алгоритм Лемпела-Зива.
5	**Шифрование данных, базовые принципы и алгоритмы.**	Будут разобраны основные алгоритмы шифрования данных.

7 Динамическое программирование

1 **Кэширование**

Кеширование: структура кеша, стратегии кеширования, кеширование данных из БД, кеширование выполнения функций, кеширование в многопоточных системах.

---

2 **Динамическое программирование: задачи динамического программирования**

Студенты освоят и смогут применять метод динамического программирования для решения практических задач.

Домашние задания

1 Решить задачу на динамическое программирование

   Цель: ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.

8 Вероятностные алгоритмы и структуры данных

1	**Фильтр Блума**	Студенты освоят, смогут реализовывать и применять фильтр Блума.
2	**Алгоритмы MinHash, SimHash**	Студенты освоят, смогут реализовывать и применять алгоритмы MinHash, SimHash. Домашние задания 1 Реализовать фильтр Блума, вероятностные алгоритмы Цель: ВАЖНО! При размещении ответа укажите, на каком языке вы выполнили ДЗ. Это поможет нам ускорить его проверку.
3	**Алгоритмы HyperLogLog, Count-Min Sketch**	Студенты освоят, смогут применять и реализовывать алгоритмы HyperLogLog и Count-Min Sketch.

9 Численные методы оптимизации

1	**Поиск экстремума функции**	Потск эстремума функции, основные методы. Многокритериальные задачи, область Поретто, свертка критериев.
2	**Нейронные сети. Алгоритм обратного распространения ошибки (backpropagation)**	Будет рассмотрено устройство нейросетей, алгоритм backpropagation и подходы к созданию современных фреймворков для работы с нейросетями.

10 Проектная работа

1	**Урок о проекте**	Урок о проекте - какие могут быть проекты, что для этого надо сделать + вопросы по несделанным домашкам. Домашние задания 1 Проектная работа
2	**Промежуточная встреча**	Обзор состояния дел по проекту, текущие вопросы студентов по их проектам.
3	**Презентация проектов**	Презентация проектов, рефлексия.
4	**Заключительное занятие**	Обзор пройденных тем. Студенты вспомнят пройденный материал.