Программа полугодового курса "Введение в алгоритмы и структуры данных".

Рассчитана на 10-11 классы IT направления.

1. Введение в **Java**, ООП, паттерны и антипаттерны, кодстайл.
2. Работа с git, хороший тон оформления репозитория.
3. Введение в асимптотику, O-символика. Примеры.
4. Рекуррентные соотношения. Пример задач, в которой они появляются. Бинарный поиск, основная постановка. Тернарный поиск. Доказательство времени работы через метод подстановки.
5. Мастер-теорема. Основной ее смысл и первые 2 случая. Пример: рекуррента для *MergeSort*.
6. Линейные контейнеры. Связные списки.
7. Контейнеры-адаптеры. Стек, его интерфейс, парадигма. Стек с минимумом.
8. Очередь, ее интерфейс, парадигма. Способы реализации: двусвязный список, кольцевой буфер. Преимущество кольцевого буфера. Очередь на двух стеках: мотивация, реализация. Двустороняя очередь.
9. Динамический массив, интерфейс и устройство. Амортизационный анализ. Метод бухгалтерсого учета.
10. Сортировки, постановка задачи. Нижняя оценка на сортировки сравнениями.
11. *MergeSort*. Асимптотика, рекурсивная реализация.
12. Квадратичные сортировки, их мотивация.
13. Интерфейс очереди с приоритетом: *extractMin()*, *getMin()*, *insert()*. Применения.
14. Бинарная куча, способы хранения. Требуемое свойство кучи.

15. Бинарная куча. Вспомогательные операции *SiftUp* и *SiftDown*. Выражение методов кучи через эти операции. Асимптотика.

16. Сортировка кучей (в том числе *inplace*). Процедура *Heapify* с доказательством асимптотики.

17. Быстрая сортировка – описание алгоритма, операция *Partition*. Среднее время работы. Худший случай.

18. Поиск k-ой порядковой статистики – описание алгоритма, среднее время работы.

19. Определение медианы. Детерминированный поиск k-ой порядковой статистики методом медианы медиан.

20. Оценка времени работы поиска k-ой порядковой статистики через рекурренту, детерминированный *Quicksort*.

21. Сортировка подсчетом, поразрядная сортировка (*LSD*).

22. Деревья поиска. Описание интерфейса множества. Наивное дерево поиска, операции на нем. Средняя глубина (без доказательства).

23. *AVL*-дерево. Основное ограничивающее условие. Оценка глубины *AVL* дерева.

24. *AVL*-дерево. Балансировка. Объяснение рекурсивной балансировки, зачем она нужна.

25. Декартово дерево. Определение, средняя глубина (без доказательства). Построение за $O(n)$ по сортированному массиву.

26. Декартово дерево. Вспомогательные операции *split* и *merge*, их реализация.

27. Декартово дерево. Операции *insert* и *erase*, их реализация через вспомогательные операции.

28. B-дерево, определение. Мотивировка, зачем оно нужно. Красно-черное дерево, основная мотивация.

29. Sparse таблица. Задача, которую она решает. Построение + ответ на запрос. Необходимые требования от операции.

30. Дерево отрезков. Модельная задача,которую оно решает. Реализация операций update и sum, доказательство асимптотики.

31. Дерево отрезков с проталкивающими операциями: модельная задача, идея, реализация.

32. Декартово дерево по неявному ключу: модельная задача. Реализация методов *merge* и *SplitByKey*, реализация с помощью их *insert* и *erase*, подсчет суммы.

33. Хэш-таблицы, общий интерфейс и время работы. Различные виды адресаций. Метод цепочек. Основные проблемы, которые могут возникнуть в этом методе, способы их решения.

34. Хэш-таблицы. Универсальное множество хэш-функций (с примером), теорема в среднем времени работы (б/д). Определение load *factor*, *rehash*.

35. Хэш-таблицы с открытой адресацией. Линейное и квадратичное пробирование, двойное хэширование. k-независимое семейство хэш-функций с примером. Теоремы о среднем времени работ хэш-таблиц с открытой адресацией (б/д).

36. Фильтр блума, модельная задача, ее решение. Теорема об оптимальных параметрах для фильтра (б/д). Асимптотика.

37. Строки. Задача поиска подстроки в строке. Полиномиальное хэширование, алгоритм Рабина-Карпа.

38. Строки. Задача поиска подстроки встроке. Префикс-функция,алгоритм Кнута-Морриса-Пратта. Поиск префикс функции за $O(|s|)$.

39. Строки. Задача поиска подстроки в строке. z-функция,алгоритм Кнута-Морриса-Пратта. Поиск z-функции за $O(|s|)$.

40. Графы, основные определения: путь, маршрут, цикл, компоненты (сильной) связности. Способы хранения графов.

41. Графы, *dfs*. Цвета вершин, лемма о белых путях. Топологическая сортировка.

42. Графы, компоненты сильной связности (КСС). Граф конденсации, его ацикличность. Алгоритм Косарайю поиска КСС.

43. Мосты, точки сочленения, ребер на двусвязность. Определение функции *ret*. Критерии поиска мостов и точек сочленения.

44. Задача поиска кратчайшего пути в невзвешенном графе, *BFS*. Двустороняя версия, пример применения.

45. Задача поиска кратчайшего пути в взвешенном ориентированном графе. Алгоритм Дейкстры: способы реализации, асимптотика и корректность.

46. Задача поиска кратчайшего пути в взвешенном ориентированном графе. Алгоритм Флойда: асимптотика и корректность. Оптимизации.

47. Задача поиска кратчайшего пути в взвешенном ориентированном графе. Алгоритм Форда-Беллмана: асимптотика и корректность. Поиск отрицательного цикла в графе, корректность.

48. Минимальные остовные деревья – постановка задачи. Лемма и минимальном разрезе.

49. Минимальные остовные деревья. Алгоритм Прима: асимптотика и корректность.

50. Минимальные остовные деревья. Алгоритм Крускала, сведение к задаче *DSU*.

51. Задача *DSU*: решение с эвристикой по рангу, асимптотика. Эвристика сжатия путей, асимптотика (б/д).

52. Задача о наименьшем общем предке, ее мотивация. Тривиальное решение, метод двоичных подъемов.

53. Задача о наименьшем общем предке, ее мотивация. Решение Эйлеровым подходом и сведение к RMQ.

Примерная стратегия ведения занятий (может меняться по усмотрению автора):

Мною предусмотрено 4 академических часа в неделю: 2 часа дистантом (лекция), 2 часа очно (семинар). На лекции первый астрономический час идёт объяснение теории, последующие 30 минут даются для практики ученикам. На семинаре мы разбираем вместе часть задач по теме, а далее учащиеся практикуются на более сложных задачах в яндекс контесте, предварительно составленным мной. Он же идет в ДЗ, часть задач из него идет на **code review**. Контесты влияют на $\sim$ 70% оценки, остальные 30% выставляются за финальный устный экзамен. Оценку можно поднять за активность на занятиях.

$$\begin{aligned}\text{Оценка} = min\{&0.7 \cdot \text{суммарный балл за контестам} + \\ &0.3 \cdot \text{устный экзамен в конце курса} + \\ &\text{оценка за активность на семинарах}, 100\}.\end{aligned}$$

Далее идет нормировка оценок по результатам учеников.

План на 15 учебных недель:

1. 1–2 и 3–5
2. 6–9 (и далее семинар)
3. 10–12
4. 13–16
5. 17–21
6. 22–24
7. 25–28
8. 29–32
9. 33–36
10. 37–39
11. 40–42
12. 43–45
13. 46–47
14. 48–51
15. 52–53