Вопросы к экзамену алгоритмам
SPb ITMO MSE, 1-й семестр, 24 января 2025

Общая информация

- pdf-конспект доступен на [wiki]. Есть несколько версий.
- Экзамен: письменный + устный, ≈1 час на подготовку билета, ≈20 минут на ответ записанного и дополнительных вопросов. Экзамен проходит *без* использования конспекта и других источников.
- Курсивом отмечены темы, разобранные на практиках, которые нужно знать к экзамену.
- (a) темы, обязательные к знанию (без них не получить удовл. оценку).

Асимптотика

1. Хорошесть алгоритма: асимптотика; константа; память, кеш; простота идеи, реализации.
(a) 2. Определения: $\mathcal{O}(n)$, $\Theta(n)$.
3. Определения: $o(n)$, $\Omega(n)$, $\omega(n)$.
4. Упражнения на понимание: $\mathcal{O}(\Theta(\mathcal{O}(f)))$, $o(\mathcal{O}(\Theta(f)))$, $\Omega(\Theta(\mathcal{O}(f)))$.
5. Упражнения на доказательства: $\mathcal{O}(\mathcal{O}(f)) = \mathcal{O}(f)$, $\mathcal{O}(C \cdot f) = \mathcal{O}(f)$, $f + o(f) = \Theta(f)$.
6. Стандартные времена работы: полилог, полином, экспонента.
7. Время работы цикла for для чисел Фибоначчи. $\Theta(n)$, $\Theta(n^2)$.
8. Модели вычислений: RAM, RAM-w, $\log n \leqslant w$, bitset, сложение массивов за $\mathcal{O}(1)$.
9. Асимптотика времени работы `for i for j in range(i)` и `for i for j in range(0,n,i)`. Оценка $\sum \frac{1}{i} = \Theta(\log n)$.

Элементарные структуры данных

10. Массив обыкновенный. Интерфейс.
11. Односвязный список `(head, x,next)`. Интерфейс.
12. Двусвязный список. Интерфейс.
13. Вектор. Аллокация с удвоением. Интерфейс.
14. Циклический массив. Аллокация с удвоением.
(a) 15. Интерфейсы стек/очередь/дек.
16. Сравнение реализаций стека/очереди/дека через циклический массив и список.
17. Амортизационный анализ. Метод потенциалов, анализ вектора.
18. Очередь на двух стеках. Очередь с минимумом. Анализ через потенциалы.
19. *Дек с минимумом. Анализ через потенциалы.*
20. *Два указателя: отрезок фиксированной суммы; отрезок* min *длины с k различными.*

Метод разделяй и властвуй

(a) 21. MergeSort. Собственно сортировка.
22. MergeSort. *Подсчёт числа инверсий в массиве.*
23. Рекуррентные соотношения, дерево рекурсии, док-во времени работы MergeSort $\mathcal{O}(n \log n)$.
24. Рекуррентные соотношения: $T(n) = T(\frac{n}{2})+n$, $T(n) = 3T(\frac{n}{2})+n$, обобщение до Мастер-Теоремы.
(a) 25. Двоичный поиск: каноничная 0011-версия.
26. Двоичный поиск: сведение задач `lowerBound`, `sqrt(x)` к 0011-версии.
27. *Двоичный поиск: сколько раз число встречается на отрезке? за* $\mathcal{O}(\log n)$.
28. *Двоичный поиск по ответу: отправка коров в стойла так, чтобы* $\min dist \to \max$.

(a) 29. Умножение многочленов: $\mathcal{O}(n^2)$.

30. Умножение многочленов: рекурсия за $\mathcal{O}(n^2)$, Карацуба за $\mathcal{O}(n^{\log 3})$.

31. Умножение чисел: сведение к многочленам, переносы.

Сортировки и куча

32. Нижняя оценка на сортировки: $\Omega(n \log n)$ сравнений.

33. *Нижняя оценка на время сортировки, корректной для $\frac{1}{100^n}$ доли перестановок:* $\Omega(n \log n)$.

34. InsertionSort, оценка времени работы $\mathcal{O}(n +$ число инверсий$)$.

35. SelectionSort, преимущество над всеми другими сортировками.

(a) 36. Куча: интерфейс (extractMin, add). Сортировка кучей.

37. Бинарная куча: хранение, siftUp, siftDown, add, extractMin.

38. Бинарная куча: построение за $\mathcal{O}(n)$, оценка времени.

39. *Бинарная куча: поддержка медианы массива с помощью кучи.*

40. Inplace версия сортировки кучей (без доппамяти).

41. Куча: decreaseKey, deleteAny, обратные ссылки.

42. `C++`: `prioirity_queue`, `set`; `python`: `heapq.heappush`.

Сортировки-2

43. Понятие вероятностного алгоритма на примере «найти в массиве x: x встречается $>\frac{n}{2}$ раз».

(a) 44. QuickSort. Простая реализация с доппамятью.

45. QuickSort. Простое доказательство через $T(n) \leqslant \frac{1}{2}T(3/4n) + \frac{1}{2}T(1/4n) + n$.

46. QuickSort. Доказательство через оценку числа сравнений $\sum_{i,j} \frac{2}{j-i+1}$.

47. QuickSort. Доказательство заменой суммы на интеграл.

(a) 48. QuickSort. Inplace partition. Алгоритм.

49. QuickSort. Inplace partition. Оценка времени работы.

50. QuickSort. Элиминация хвостовой рекурсии и оценка $\mathcal{O}(\log n)$ на доппамять.

51. QuickSort: IntroSort.

52. k-ая статистика: heap, $\mathcal{O}(n + k \log n)$.

53. k-ая статистика: одноветочный QuickSort за $\mathcal{O}(n)$, оценка времени работы.

54. k-ая статистика: детерминированная с делением на кусочки длины 5.

55. *k-ая статистика: детерминированная с делением на кусочки длины* 3 *и* 7.

56. Inplace алгоритмы: reverse, rotate, swap-половинок-массива.

57. *Inplace stable partition за $\mathcal{O}(n \log n)$.*

58. Inplace stable merge за $\mathcal{O}(n \log n)$.

Сортировки быстрее $n \log n$

(a) 59. CountSort, простая версия.

60. CountSort для сортировка пар $\langle key, object \rangle$. Стабильность.

(a) 61. Частичные (префиксные) суммы.

62. *Структура для n чисел от* 1 *до k за $\langle \mathcal{O}(n+k), \mathcal{O}(1) \rangle$ говорящая «сколько чисел от a до b»?*

63. RadixSort. Реализация за $n \log_n C$.

64. BucketSort для равномерно распределённых вещественных чисел.

65. Несколько CountSort-ов в одном: сортировка массивов $A_1, A_2, \ldots A_k$ над алфавитом m за $\mathcal{O}(m + \sum_i |A_i|)$.

66. Рекурсивный перебор: рюкзак со стоимостями, решение за $\mathcal{O}(2^n)$.

67. Рекурсивный перебор: перебор перестановок за $\mathcal{O}(n!)$.

68. VEB. Куча, которая умеет всё за $\log\log C$: общее устройство, откуда $\log\log$?
69. VEB. Куча, которая умеет всё за $\log\log C$: add, extractMin, lowerBound.

Динамика-1

(a) 70. Рекуррентные соотношения: fib_n, $C_{n,k}$ (биномиальные), $f[L,R] = \min_M f[L,M]\cdot f[M,R]$). Решение всех данных рекуррентных соотношений динамикой.
(a) 71. Перебор с запоминанием и динамика (subsetsum).
72. Задача «калькулятор»: из 1 получить n min числом операций $x \to 2x, 3x, x{+}1$.
73. Задача «кролик»: число путей кролика $x \to x{+}3 \dots x{+}5$ так, что кролик не попадает в дырку.
74. Динамика вперёд, назад, ленивая, граф динамики. Две версии: $dp[v]\colon s \rightsquigarrow v$ и $dp[v]\colon v \rightsquigarrow t$.
75. Динамика, как задача на ацикличном графе: число путей в графе, min/max путь. Решение для графа в явном виде.
(a) 76. НВП за $\mathcal{O}(n^2)$.
77. НВП за $\mathcal{O}(n \log n)$.
78. НОП за $\mathcal{O}(n^2)$.
(a) 79. Рюкзак (subsetsum и knapsack) за $\mathcal{O}(nS)$.
80. Восстановление ответа на примерах НВП, НОП, рюкзак: со ссылками, без ссылок.
81. *Рюкзак с линией памяти. Версии рюкзака, когда предмет можно брать* 1 *или* $+\infty$ *раз.*
82. Рюкзак: линия памяти и восстановление ответа.
83. Рюкзак с bitset.

Динамика-2

(a) 84. DP по дереву: размер поддерева, глубина.
85. DP по дереву: max-independent-set, *паросочетание.*
86. DP на отрезках: игра на массиве – двое берут числа с краёв массива, у кого сумма больше, тот выиграл.
87. DP на подотрезках: перемножение матриц $A_1, \dots, A_n$ за min время.
88. Выбор состояния-функции: версия рюкзака $sumWeight[sumCost, i]$.
(a) 89. Хранение множеств масками. Общий принцип.
90. Операции с множествами за $\mathcal{O}(1)\colon x \in A, A \cap B, A \cup B, A \setminus B, A \subseteq B, \{0, 1, \dots, n{-}1\}$, размер.
91. *Операции с множествами за* $\mathcal{O}(1)$*: найти младший единичный бит.*
(a) 92. Гамильтонов путь за $\mathcal{O}(2^n n^2)$.
93. Гамильтонов путь за $\mathcal{O}(2^n n)$.
94. *Задача SetCover: покрыть множество* B *минимальным числом* A_i*. Решение за* $\mathcal{O}(2^{|B|}m)$.
95. Пребор пар «множество, подмножество» за 3^n.
96. Решение задачи Vertex Coloring за $\mathcal{O}(3^n)$.
97. Замощение доминошками. DP по скошенному профилю. Только рекурсивная версия.
98. Алгоритм Хиршберга восстановления ответа к НОП с $\mathcal{O}(n)$ памяти.

Жадность

99. Жадность, перебор, динамика: различие подходов на примере коммивояжера.
100. Локальные оптимизации. Пример для задачи коммивояжера.
(a) 101. Задача. Непрерывный рюкзак. Жадное решение.
102. Задача. Выполнение всех заданий к дедлайнам. Строгое доказательство корректности.
103. Задача. Покрыть точки на прямой минимальным числом отрезков длины 1.
104. Задача. Выбрать максимальное число непересекающихся отрезков на прямой.

(a) 105. Метод событий на прямой: для каждой точки узнать, сколькими отрезками она покрыта.
106. Метод событий на прямой: жадный выбор заявок для k-аудиторий.
107. Задача: max-independent-set в деревьях, жадное решение.
108. Хаффман. Алгоритм кодирования, декодирования.
109. Хаффман. Хранение дерева/частот.
110. Хаффман. Доказательство минимальности.

MST и DSU

111. MST, алгоритм Прима, доказательство от противного,
(a) 112. MST, алгоритм Прима, реализации за V^2, $E \log V$.
113. MST, алгоритм Прима, реализации за $E \log_{E/V} V$ (d-куча).
114. MST, алгоритм Краскала, доказательство от противного.
115. MST, алгоритм Краскала, реализация, время работы.
116. MST. Свойство любого разреза MST (лемма о min ребре), доказательство от противного
117. DSU. Система Непересекающихся Множеств. Интерфейс.
118. DSU на списках. join-ы за $n \log n$, get за 1.
119. DSU на деревьях, меньшее к большему $\Rightarrow$ join и get за $\log n$.
120. DSU на деревьях, эвристика сжатия путей.
121. DSU. Практически эффективная версия.

Графы. DFS.

122. Определения: граф, смежность, инцидентность, орграф, неорграф.
123. Примеры графов: друзья/знакомства, зависимости между работами/пакетами, страны и дороги, граф состояний в естественной задаче.
(a) 124. Хранение графов: матрица смежности, списки рёбер.
125. Хранение графов: set-ы рёбер, сравнение всех трёх способов.
(a) 126. dfs: проверка достижимости, поиск компонент.
127. dfs: поиск пути, обратный ход рекурсии.
128. dfs и DAG: что такое цикл? topsort, времена входа-выхода.
129. dfs: поиск цикла в орграфе.
130. *dfs: поиск цикла в неорграфе.*
131. dfs: покраска в два цвета.
132. Дерево dfs: классификация рёбер.
133. Дерево dfs: неорграф не содержит перекрёстных рёбер.
134. Компоненты сильной связности. Определение, простая проверка сильной связности.
135. Компоненты сильной связности: веделение всех компонент, построение конденсации.
136. *2-связность: поиск мостов и компонент (dp по дереву).*

Кратчайшие пути

(a) 137. BFS. Две версии: по слоям; с очередью.
138. BFS. *Модификация для целых* 1-k *весов за* $\mathcal{O}(E + kV)$ *и* $\mathcal{O}(E \log k)$.
139. BFS. *Модификация для вещественных* 1-k *весов.*
140. BFS. *Модификация для целых* 0-1 *весов.*
141. Дейкстра. Алгоритм и доказательство корректности.
(a) 142. Дейкстра. Реализации за $\mathcal{O}(n^2)$, $\mathcal{O}(m \log n)$.
143. Реализация на `C++`/`python`: `set`, `prioirity_queue`, `operator<`, `heapq.heappush`.

144. Дейкстра. Время работы при применении куч: d-ичная, VEB, фибоначчиева.
145. Дейкстра. Двусторонняя вариация алгоритма.
146. Алгоритм A*. Пример применения.
147. Алгоритм A*. Доказательство корректности, оценка времени работы.

Амортизация

148. *Алгоритм Борувки.*
149. Skew heap. Алгоритм.
150. Skew Heap, доказательство: лёгкие и тяжёлые рёбра, потенциал.
151. DSU. Доказательство $\log n$ для сжатия путей.
152. DSU. Доказательство $\log^* n$ для двух эвристик.
153. Амортизационный анализ: метод монеток (ростовщика) на примере вектора.
154. Турнирное дерево (аналог дерева отрезков).
155. Списко-куча: `add`, `merge`, `min` за $\mathcal{O}(1)$ и `extractMin` за амортизированный $\mathcal{O}(\log n)$.