Страницы указаны для 2-го издания (печатного)
Глава 3 Оценивание числа шагов алгоритма

1. **Алгоритм Евклида. (E)** Число шагов в алгоритме Евклида <= минимального числа из 2-х рассматриваемых **(стр 73)**
   Сложность по числу делений - $T_E(a1) \leq 2log_2 a1 + 1$ ---- a1 – меньшее из чисел
   Точность этой оценки $O(\log_2 a1)$ доказывается через числа Фибоначи (стр 82)
   Ограничение снизу (стр 83) $\frac{1}{4}\log_\phi a1 + C$, где C – некоторая константа
   Ограничение снизу (стр 86) $\frac{1}{2}\log_\phi a1 + C$, где C – некоторая константа
   Расширенный алгоритм Евклида (EE) - $T_{EE}(a1) \leq 6\log_2 a1 + 3$
   Ограничение снизу (стр 83) $T_{EE}(a1) = \theta(\log_2 a1)$
2. **Бинарный поиск места элемента в упорядоченном массиве из n элементов (BS) (стр 77)** (суть в делении массива пополам на каждом шаге)
   $T_{BS}(n) = \lfloor \log_2 n \rfloor + 1 = \lceil \log_2 n + 1 \rceil$
   На этом алгоритме основан алгоритм определения принадлежности точки некоторому выпуклому многоугольнику
3. **Сложность сортировки Фон-Нейманом (Сортировки Слиянием) (стр 79)**
   По числу сравнений меньше, а по числу присваиваний равна $T_{BS}(n) = n\lceil \log_2 n \rceil$
   $T_{vN}(n) \sim n\log_2 n$ (стр. 107)
4. **Сложность по числу сравнений сортировки бинарными вставками (стр 81)**
   $T_B(n) = (n-1)\lceil \log_2 n \rceil$
   (стр 87) $T_B(n) \approx n\log_2 n$
5. **Поиск корней уравнения (стр 81)**
   Метод деления пополам (число итераций) – $\log_2 1/\varepsilon + O(1)$
   Метод Ньютона (касательных) (число итераций) – $O(\log_2 \log_2 1/\varepsilon)$
6. **Завершимость работы алгоритма (стр 88)**
7. **Вложенные циклы.** пример асимптотики при вложенных циклах (стр 93)
8. **Нецелые размеры входа (стр 95). Разрывность сложности для алгоритма Евклида (стр 96)**

Глава 4 Нижняя граница сложности алгоритмов некоторого класса. Оптимальные алгоритмы

9. Сложность **алгоритма поиска наименьшего элемента** массива длинны n по числу (стр. 105) сравнений – (n-1)
10. **Для всех алгоритмов сортировки выполнено**: (стр. 106)
    $2^{T(n)} \geq n! => T(n) \geq \lceil \log_2 n! \rceil > n\log_2 n - 2n$, где T(n) – временная сложность сортировки сравнениями
11. Если **сложность сортировки по числу сравнений** не превосходит $n*\log_2 n + cn$, тогда $T(n) \sim n\log_2 n$ (стр. 107)
12. **Алгоритм поиска места элемента в массиве** (нижняя граница) (стр 108)
    $f(n) = \lceil \log_2(n+1) \rceil$
13. **Алгоритм вычисления a^n с помощью умножения** (нижняя граница) (стр 108)
    $f(n) = \lceil \log_2 n \rceil$
14. Остановился на странице 108
15. **Алгоритм одновременного выбора наибольшего и наименьшего элементов массива длинны n (стр. 109)**
    $f(n) = \left\lceil \frac{3n}{2} \right\rceil - 2$ – сложность по числу сравнений, приведённый алгоритм - оптимален

16. **Вычисление значения полинома в данной точке (стр 111)** (лишь упомянуто)
    Схема Горнерна – оптимальный алгоритм
17. Оптимального алгоритма может не существовать (стр. 111)
18. Объяснение не оптимальности алгоритмов сортировки – бинарный алгоритм возведения в степень (стр. 112) (об оптимальной сортировке см приложение F)
19. **Достаточное условие оптимальности алгоритма: (стр 114)**
    Если f(n) – является асимптотической нижней границей сложности, и если $T_A(n) = O(f(n))$ то этот алгоритм оптимален по порядку сложности и $T_A(n) = \Theta(f(n))$
20. **Бинарный алгоритм возведения в степень (стр. 114)** оптимален по порядку сложности в классе алгоритмов вычисления a^n с помощью умножений
21. **Алгоритм построения Эйлерова цикла данного ориентированного графа (стр. 114)** со сложностью O(|E|) - оптимален
22. **Алгоритм построения остовного дерева (Алгоритм Прима) (стр 115)** – оптимален по порядку сложности, и его сложность – $\Theta(|V|^2)$
23. Любая сортировка допускающая оценку $O(\log_2 n!)$ или $O(n \log_2 n)$ являбется оптимальной по порядку сложности (стр. 116)
24. **Сортировка Бинарными вставками** – оптимальны по порядку сложности по числу сравнений (стр. 116)
25. **Сортировка Фон-Неймана** – оптимальны по порядку сложности по числу сравнений (стр. 116)
26. Функция $\log_2 n!$ Является нижней границей сложности в среднем для класса алгоритмов сортировки массивов длинны n с помощью представлений (стр 117)
27. **Сортировка Бинарными вставками** и **Сортировка фон-Неймана** и **Быстрая сортировка –** оптимальны по порядку сложности в среднем по числу сравнений (стр. 119)
28. **Нижняя граница сложности в среднем алгоритмов одновременного выбора наибольшего и наименьшего элементов массива длинны n >= 2, с помощью сравнений (стр. 121)**
    $$f(n) = \begin{cases} \frac{3}{2}n - 2, \text{если } n - \text{чётно} \\ \frac{3}{2n} - 2 + \frac{1}{2n}, \text{если } n - \text{нечётно} \end{cases}$$
    Эта оценка – оптимальна, и алгоритм оптимален
29. **Принцип Яо (стр. 127)**
30. Для любой рандомизированной сортировки (которую теоретически можно задать, как конечное множество детерминированных алгоритмов сортировки) её сложность не может быть меньше, чем **$\log_2 n!$ (стр. 128)**
31. **Рандомизированная быстрая сортировка** оптимальна по порядку сложности в класе рандомизированных сортировок (стр. 128)

Глава 5 Битовая сложность

«*» - означает, что мы рассматриваем битовую сложность

32. **Алгоритм сложения столбиком** при использовании m в качестве размера входа (число бит максимального из чисел) $T^*_{Add}(m) = \theta(m) = \theta(\max\{m1, m2\})$ (стр. 135)
    Этот алгоритм является оптимальным (ибо мы не можем игнорировать содержимое битов)
33. Сверхнаивное умножение (стр. 136) (умножение посредством сложения) битовая сложность = $\Theta(2^m m)$

34. Наивное умножение - $T^*_{NM}(m) = \theta(m^2) = \theta(m_1 * m_2)$ (стр. 137)
    $S^*_{NM}(m) = 2m + O(1) = m_1 + m_2 + O(1)$
    $T_{NM}(m) = O(\log b * \log b)$
35. **Вычисление n! с помощью пошаговых наивных умножений** имеет битовую временную сложность $O((n \log n)^2)$ (стр. 139)
36. **Сложность умножения чисел a1 … an при пошаговом наивном умножении.** Пусть M – суммарная битовая длинна этих чисел, тогда временная сложность допускает верхнюю оценку $O(M^2)$ (стр. 139)
37. **Сложность наивного деления (стр. 140)** – битовая сложность = $T^*_{ND}(m) = \theta(m^2) = \theta((m_1 - m_2 + 1) * m_2) = O(\log b * \log b)$
38. **Построение k-ичной записи числа n (перевод из 2-ичной системы счисления в k-ичную) (стр. 141)**
    битовая сложность = $O(\log^2 n)$
39. **Алгоритм Евклида** битовая сложность (стр. 142)
    $O(\log a_0 \log a_1)$ или $O(\log^2 a_0)$ или $O(m^2)$ при $m = \lambda(a_1)$
    Если **алгоритм Евклида** (или **расширенный Алгоритм Евклида**) основывается на делении с остатком, затраты которого оцениваются O(log v log u), то его битовая оценка имеет вид $\Theta(\log^2 a_0)$
40. **Обращение в поле (нахождение обратного элемента в поле) (стр. 147)**
    Если расширенный алгоритм Евклида основывается на алгоритме деления и умножения со сложностью O(log v log u), то битовая сложность обращения числа в поле Z_p допускает оценку $O(\log^2 p)$
41. **Малая теорема Ферма (стр. 147)** – p – простое, a – произвольное, тогда a^p==a (mod p)
42. Пусть a – целое, p – натуральное, и они взаимно просты, тогда n – просто, тогда и только тогда, когда $(x-a)^n = x^n - a \pmod n$ (т.е. числовые коэффициенты при одинаковых степенях в полиномах, расположенных в левой и правой части – сравнимы по модулю n) (стр. 148)
43. Алгоритм AKS (Агравал, Кайал, Саксена) – алгоритм определения простоты числа,
    сложность – O с волной $(m^{21/2})$, если ещё допустить некоторые, пока не доказанные в математике гипотезы, то O с волной (m^6) (стр 148)
44. **Возведение в степень булевой матрицы (стр 152)** –
    Обычным умножением – $\Theta(n^3 \log n)$
    Если B(n) – сложность используемого алгоритма умножения булевых матриц, то сложность возведения в степень = $n + B(n)(\lambda(n) + \lambda^*(n) - 2)$ (первое λ – битовая длинна числа n, второе – количество единиц в его двоичной записи)
45. **Построение транзитивно-рефлексивного замыкания ориентированного графа (Алгоритм Уоршелла) (стр. 153)** – (n вершин) затрачивая $2*n^3+n$ битовых операций
    Пространственная сложность ограничена константой

Глава 6. Рекуррентные соотношения, как средство анализа сложности алгоритмов

46. Если мы будет добавлять по единице к некоторому числу начиная с нуля, то для достижения 2^n-1 потребуется 2^n-n-1 переносов (стр. 158)
47. Лучше если рекуррентная формула зависит независимо только от одного предыдущего значения, потому что если она зависит от нескольких, то это приводит к повторным вычислениям (хотя странно, почему нельзя просто сохранять значения) (стр. 160)
48. Пусть дан рекурсивный алгоритм Yn=U(Yn-1, …, Yn-k) и k >= 2, тогда количество вычислений этой функции при нахождении $Yn = \Theta(\alpha^n_k)$, где $2 - 1/k <= \alpha^n_k < 2$ (стр. 161)

49. **Рекурсивная сортировка слияниями (стр. 162) -**
$(\lfloor \log_2 n \rfloor - 1)2^{\lfloor \log_2 n \rfloor} + 1 \leq T_{MS}(n) \leq (\lceil \log_2 n \rceil - 1)2^{\lceil \log_2 n \rceil} + 1$
Пространственная сложность = Ω(n)
50. **Теорема о рекуррентном неравенстве (случай <=) (стр. 166)**

Теорема 26.1 (о рекуррентном неравенстве, случай $\leqslant$). *Пусть неотрицательная вещественная функция f натурального аргумента удовлетворяет неравенству*

$$f(n) \leqslant \begin{cases} u, & \text{если } n = 1, \\ vf\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + wf\left(\left\lceil \frac{n}{2} \right\rceil\right) + cn^d, & \text{если } n > 1, \end{cases} \qquad (26.1)$$

где u, v, w, d — неотрицательные вещественные числа, причем $v + w \geqslant 1$; c — положительное вещественное число. Тогда

$$f(n) = \begin{cases} O(n^d \log n), & \text{если } d = \log_2(v+w), \\ O(n^d), & \text{если } d > \log_2(v+w), \\ O(n^{\log_2(v+w)}), & \text{если } d < \log_2(v+w). \end{cases} \qquad (26.2)$$

51. **Теорема о рекуррентном неравенстве (случай >=) (стр. 167)**

Теорема 26.2 (о рекуррентном неравенстве, случай $\geqslant$). *Пусть вещественная функция $f(n)$ натурального аргумента удовлетворяет неравенству*

$$f(n) \geqslant \begin{cases} u, & \text{если } n = 1, \\ vf\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + wf\left(\left\lceil \frac{n}{2} \right\rceil\right) + cn^d, & \text{если } n > 1, \end{cases}$$

где u, v, w, d — неотрицательные вещественные числа, причем $v + w \geqslant 1$; c — положительное вещественное число. Тогда

$$f(n) = \begin{cases} \Omega(n^d \log n), & \text{если } d = \log_2(v+w), \\ \Omega(n^{\log_2(v+w)}), & \text{если } d > \log_2(v+w), \\ \Omega(n^d), & \text{если } d < \log_2(v+w). \end{cases} \qquad (26.4)$$

52. **Сложность рекурсивной сортировки слияниями (стр. 168)** – (и по числу сравнений, и по числу перемещений) (стр. 168) Θ(n log n)
53. **Бинарное возведение в степень n (стр. 168)** $T_{RS}(n) = \Theta(\log n)$
54. **Построение выпуклой оболочки объединения 2-х выпуклых многоугольников (стр. 169)** O(n log n)
55. **Умножение Карацубы (стр. 171)**
56. **Умножение 2-х чисел методом Карацубы (стр. 173)** - $T_{KM}(m) = \Theta(m^{\log_2 3})$
57. **Умножение 2-х матриц со стороной порядка 2^k – (Метод Штрассена) (стр . 173)** – $T_{St}(n) = \Theta(n^{\log_2 7})$
58. **Умножение 2-х булевых матриц с применением алгоритма Штрассена и арифметики по модулю n+1. (стр. 175)** Битовая сложность O с волной $(n^{\log_2 7})$
59. Есть вроде бы что-то более зубодробительное начиная с стр 176, но вроде это лишнее

Глава 7 Сводимость

60. **Сведение умножения к возведению в квадрат (стр. 184)**
61. **Сведение задачи построения транзитивно-рефлексивного замыкания графа к задаче умножения 2-х булевых матриц порядка n (стр. 185)**
62. **Нахождение транзитивно-рефлексивного замыкания ориентированного графа (стр. 188)** Существует алгоритм со сложностью $O(n^{2,82})$ а точнее O с волной $(n^{\log_2 7})$

63. **Нижняя грань сложности алгоритмов сортировки. (стр. 191)** Функция $f(n) = \lceil \log_2 n! \rceil$ Является нижней гранью сложности по числу сравнений алгоритмов сортировки массивов длинны попарно-различных вещественных числе с помощью сравнений и 4-х арифметических операций
64. **Сложность алгоритма построения выпуклой оболочки (стр. 193)** с помощью арифметических операций и сравнений, который имеет сложность O(n log n)
    Равна Θ(n log n) и является оптимальным
    Алгоритм Грехема - оптимален

Этого не будет:

65. Задача NP – это задача, на которую ответ либо «да», либо «нет»
66. Задача класса P – задача распознавания свойства с полиномиальной сложностью
67. P вложен в NP, но не равен
68. **Теорема Фишера-Рабина (стр. 197)** – она доказывает, основываясь на арифметике Пресбургера, что класс P не равен классу NP
69. **Стр 199 – определения Полиномиальной сводимости, NP-полных задач**
70. **Задача Sat (стр. 200)** – задача выполнимости заданной в КНФ булевой формулы (задача выполнимости КНФ)
71. **Задача Sat полиномиально сводится к задаче о существовании «клики с m вершинами» (т.е. существует в данном графе набор из m вершин, любые 2 из которых соединены ребром) (стр. 200)**
    Задача распознавания гамильтоновости графа является NP-полной
72. Пример задачи из NP, но не P – Для заданных k и l неотрицательных целых и k < n выяснить, имеется ли у числа l делитель n, такой что 1 < l <= k