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 - ---- a1 – меньшее из чисел

Точность этой оценки доказывается через числа Фибоначи (стр 82)

Ограничение снизу (стр 83)

Ограничение снизу (стр 86)

**Расширенный алгоритм Евклида (EE)** -

Ограничение снизу (стр 83)

1. **Бинарный поиск места элемента в упорядоченном массиве из n элементов (BS) (стр 77)**

(суть в делении массива пополам на каждом шаге)

На этом алгоритме основан алгоритм определения принадлежности точки некоторому выпуклому многоугольнику

1. **Сложность сортировки Фон-Нейманом (Сортировки Слиянием) (стр 79)**

По числу сравнений меньше, а по числу присваиваний равна

(стр. 107)

1. **Сложность по числу сравнений сортировки бинарными вставками (стр 81)**

(стр 87)

1. **Поиск корней уравнения (стр 81)**

Метод деления пополам (число итераций) –

Метод Ньютона (касательных) (число итераций) –

1. **Завершимость работы алгоритма (стр 88)**
2. **Вложенные циклы.** пример асимптотики при вложенных циклах (стр 93)
3. **Нецелые размеры входа (стр 95). Разрывность сложности для алгоритма Евклида (стр 96)**

Глава 4 Нижняя граница сложности алгоритмов некоторого класса. Оптимальные алгоритмы

1. Сложность **алгоритма поиска наименьшего элемента** массива длинны n по числу (стр. 105) сравнений – (n-1)
2. **Для всех алгоритмов сортировки выполнено**: (стр. 106)

, где T(n) – временная сложность сортировки сравнениями

1. Если **сложность сортировки по числу сравнений** не превосходит n\*log2n +cn, тогда (стр. 107)
2. **Алгоритм поиска места элемента в массиве** (нижняя граница) (стр 108)
3. **Алгоритм вычисления an с помощью умножения** (нижняя граница) (стр 108)
4. Остановился на странице 108
5. **Алгоритм одновременного выбора наибольшего и наименьшего элементов массива длинны n (стр. 109)**

– сложность по числу сравнений, приведённый алгоритм - оптимален

1. **Вычисление значения полинома в данной точке (стр 111)** (лишь упомянуто)

Схема Горненра – оптимальный алгоритм

1. Оптимального алгоритма может не существовать (стр. 111)
2. Объяснение не оптимальности алгоритмов сортировки – бинарный алгоритм возведения в степень (стр. 112) (об оптимальной сортировке см приложение F)
3. **Достаточное условие оптимальности алгоритма: (стр 114)**

Если f(n) – является асимптотической нижней границей сложности, и если TA(n) = O(f(n)) то этот алгоритм оптимален по порядку сложности и TA(n) = Θ(f(n))

1. **Бинарный алгоритм возведения в степень (стр. 114)** оптимален по порядку сложности в классе алгоритмов вычисления an с помощью умножений
2. **Алгоритм построения Эйлерова цикла данного ориентированного графа (стр. 114)** со сложностью O(|E|) - оптимален
3. **Алгоритм построения остовного дерева (Алгоритм Прима) (стр 115)** – оптимален по порядку сложности, и его сложность – Θ(|V|2)
4. Любая сортировка допускающая оценку явля6ется оптимальной по порядку сложности (стр. 116)
5. **Сортировка Бинарными вставками** – оптимальны по порядку сложности по числу сравнений (стр. 116)
6. **Сортировка Фон-Неймана** – оптимальны по порядку сложности по числу сравнений (стр. 116)
7. Функция log2n! Является нижней границей сложности в среднем для класса алгоритмов сортировки массивов длинны n с помощью представлений (стр 117)
8. **Сортировка Бинарными вставками** и **Сортировка фон-Неймана** и **Быстрая сортировка –** оптимальны по порядку сложности в среднем по числу сравнений (стр. 119)
9. **Нижняя граница сложности в среднем алгоритмов одновременного выбора наибольшего и наименьшего элементов массива длинны n >= 2, с помощью сравнений (стр. 121)**

Эта оценка – оптимальна, и алгоритм оптимален

1. **Принцип Яо (стр. 127)**
2. Для любой рандомизированной сортировки (которую теоретически можно задать, как конечное множество детерминированных алгоритмов сортировки) её сложность не может быть меньше, чем **log2n! (стр. 128)**
3. **Рандомизированная быстрая сортировка** оптимальна по порядку сложности в класе рандомизированных сортировок (стр. 128)

Глава 5 Битовая сложность

«\*» - означает, что мы рассматриваем битовую сложность

1. **Алгоритм сложения столбиком** при использовании m в качестве размера входа (число бит максимального из чисел) (стр. 135)

Этот алгоритм является оптимальным (ибо мы не можем игнорировать содержимое битов)

1. Сверхнаивное умножение (стр. 136) (умножение посредством сложения) битовая сложность = Θ(2mm)
2. Наивное умножение - (стр. 137)
3. **Вычисление n! с помощью пошаговых наивных умножений** имеет битовую временную сложность O((n log n)2) (стр. 139)
4. **Сложность умножения чисел a1 … an при пошаговом наивном умножении.** Пусть M – суммарная битовая длинна этих чисел, тогда временная сложность допускает верхнюю оценку O(M2) (стр. 139)
5. **Сложность наивного деления (стр. 140)** – битовая сложность =
6. **Построение k-ичной записи числа n (перевод из 2-ичной системы счисления в k-ичную) (стр. 141)**

битовая cложность = O(log2n)

1. **Алгоритм Евклида** битовая сложность (стр. 142)

O(log a0 log a1) или O(log2a0) или O(m2) при m = λ(a1)

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 Θ(log2a0)

1. **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 Zp допускает оценку O(log2p)

1. **Малая теорема Ферма (стр. 147)** – p – простое, a – произвольное, тогда ap==a (mod p)
2. Пусть a – целое, p – натуральное, и они взаимно просты, тогда n – просто, тогда и только тогда, когда (x-a)n=xn-a(mod n) (т.е. числовые коэффициенты при одинаковых степенях в полиномах, расположенных в левой и правой части – сравнимы по модулю n) (стр. 148)
3. Алгоритм AKS (Агравал, Кайал, Саксена) – алгоритм определения простоты числа,

сложность – O с волной (m21/2), если ещё допустить некоторые, пока не доказанные в математике гипотезы, то O с волной (m6) (стр 148)

1. **Возведение в степень булевой матрицы (стр 152)** –

Обычным умножением – Θ(n3log n)

Если B(n) – сложность используемого алгоритма умножения булевых матриц, то сложность возведения в степень = n + B(n)( λ(n)+ λ\*(n) -2) (первое λ – битовая длинна числа n, второе – количество единиц в его двоичной записи)

1. **Построение транзитивно-рефлексивного замыкания ориентированного графа (Алгоритм Уоршелла) (стр. 153)** – (n вершин) затрачивая 2\*n3+n битовых операций

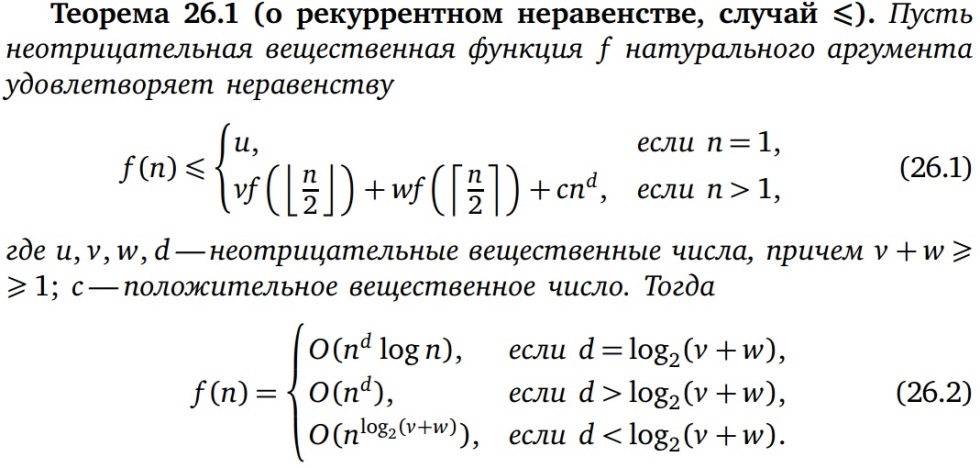
Пространственная сложность ограничена константой

Глава 6. Рекуррентные соотношения, как средство анализа сложности алгоритмов

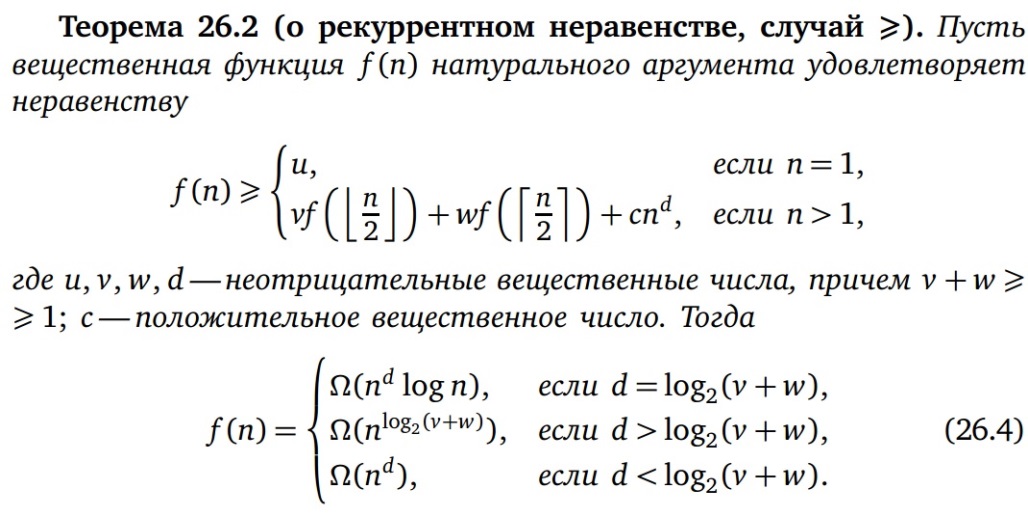
1. Если мы будет добавлять по единице к некоторому числу начиная с нуля, то для достижения 2n-1 потребуется 2n-n-1 переносов (стр. 158)
2. Лучше если рекуррентная формула зависит независимо только от одного предыдущего значения, потому что если она зависит от нескольких, то это приводит к повторным вычислениям (хотя странно, почему нельзя просто сохранять значения) (стр. 160)
3. Пусть дан рекурсивный алгоритм Yn=U(Yn-1, …, Yn-k) и k >= 2, тогда количество вычислений этой функции при нахождении Yn = Θ(αnk), где 2 – 1/k <= αnk < 2 (стр. 161)
4. **Рекурсивная сортировка слияниями (стр. 162) -**

Пространственная сложность = Ω(n)

1. **Теорема о рекуррентном неравенстве (случай <=) (стр. 166)**

****

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

****

1. **Сложность рекурсивной сортировки слияниями (стр. 168)** – (и по числу сравнений, и по числу перемещений) (стр. 168) Θ(n log n)
2. **Бинарное возведение в степень n (стр. 168)** TRS(n)= Θ(log n)
3. **Построение выпуклой оболочки объединения 2-х выпуклых многоугольников (стр. 169)**

O(n log n)

1. **Умножение Карацубы (стр. 171)**
2. **Умножение 2-х чисел методом Карацубы (стр. 173)**  -

TKM(m)= Θ(mlog2 3)

1. **Умножение 2-х матриц со стороной порядка 2k – (Метод Штрассена) (стр . 173)** –

TSt(n)= Θ(nlog2 7)

1. **Умножение 2-х булевых матриц с применением алгоритма Штрассена и арифметики по модулю n+1. (стр. 175)** Битовая сложность O с волной (nlog2 7)
2. Есть вроде бы что-то более зубодробительное начиная с стр 176, но вроде это лишнее

Глава 7 Сводимость

1. **Сведение умножения к возведению в квадрат (стр. 184)**
2. **Сведение задачи построения транзитивно-рефлексивного замыкания графа к задаче умножения 2-х булевых матриц порядка n (стр. 185)**
3. **Нахождение транзитивно-рефлексивного замыкания ориентированного графа (стр. 188)**

Существует алгоритм со сложностью O(n2,82) а точнее O с волной (nlog2 7)

1. **Нижняя грань сложности алгоритмов сортировки. (стр. 191)** Функция Является нижней гранью сложности по числу сравнений алгоритмов сортировки массивов длинны попарно-различных вещественных числе с помощью сравнений и 4-х арифметических операций
2. **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:**

1. Задача NP – это задача, на которую ответ либо «да», либо «нет»
2. Задача класса P – задача распознавания свойства с полиномиальной сложностью
3. P вложен в NP, но не равен
4. **Теорема Фишера-Рабина (стр. 197)** – она доказывает, основываясь на арифметике Пресбургера, что класс P не равен классу NP
5. **Стр 199 – определения Полиномиальной сводимости, NP-полных задач**
6. **Задача Sat (стр. 200)** – задача выполнимости заданной в КНФ булевой формулы (задача выполнимости КНФ)
7. **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

1. 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