Общие сведения

1. Будет три курсовые контрольные, две обязательные и одна “утешительная”.
2. Студент, сдавший обе обязательные контрольные на оценку не ниже уд. 3, может получить максимум из полусуммы оценок за контрольные и оценки семинариста. Если вы не согласны с полученной оценкой, этот вопрос может быть решён комиссией из преподавателей курса.
3. Студент, не сдавший какую-либо из обязательных контрольных или пропустивший её, должен писать утешительную. Если студент сдал утешительную контрольную, вопрос его зачёта решается при показе работы. В противном случае, студент отправляется на пересдачу.
4. Важные материалы по курсу будут публиковаться в группе курса вк. Решения нужно присылать на странице курса в classroom, они должны быть оформлены в виде `pdf` файла, также нужно приложить оригинальную редактируемую версию файла.

Зачёт по курсу

Ваша оценка будет высчитана по следующей формуле:

$$S = \max([0.5 \cdot (K_1 + K_2)], [0.3 \cdot (K_1 + K_2 + \text{Д} + \text{Б})])$$

Округление происходит к ближайшему целому по математическим правилам. Обозначения:

1. K_1 и K_2 – оценки, полученные на семестровых контрольных (целые числа от 3 до 10).
2. Д – оценка за домашние работы (вещественное число от 0 до 12). Сумма баллов по основным задачам в каждом задании будет отнормирована на 12, затем будет взято среднее арифметическое по всем заданиям. Если оценка выходит меньше, чем 4, то задание считается несданным и при усреднении его оценка берётся равной 0.

   По приёму заданий не будет жёстких ограничений по времени, но через две недели после того, как задание выложено, сдать или дополнить его можно будет только устно. Это распространяется и на тех, кто пропустил срок по уважительной причине.
3. Б – бонусная оценка (вещественное число от 0 до 12). В неё могут входить:
   (a) Бонусные задачи. Баллы за задачу будут непосредственно идти в бонусный балл. Эти задачи отмечены символом * и не учитываются при подсчёте Д. Сдавать бонусные задачи можно только устно, сумма бонусных баллов за задачи не ограничена.
   (b) Устный зачёт в конце семестра. Оценка будет отнормирована на 6 и внесена в бонус.
   (c) Алгоритмический техкурс. Оценка будет отнормирована на 6 и внесена в бонус.
   (d) Прочее. Баллы присваиваются по усмотрению преподавателя, но не более 6 баллов.

Для внесения в бонус оценки за устный зачёт или техкурс, она должна быть не ниже, чем $[0.5 \cdot (K_1 + K_2)] - 1$. Для приёма устного зачёта могут приглашаться другие преподаватели. При подсчёте S величина Б будет взята как минимум из 12 и суммы указанных величин.

Основные понятия и обозначения

Один из основных вопросов при работе с алгоритмами – “за какое время алгоритм завершит свою работу?”. Ответ зависит от свойств устройства, на котором алгоритм выполняется и не может быть универсальным. Поэтому вместо подсчёта физического времени будем считать его пропорциональным числу произведённых *элементарных операций*, которое и будем исследовать. Это даст нам возможность объективно оценивать время работы алгоритма, опираясь только на сам алгоритм и *модель вычислений*, в рамках которой он задан.

Модели вычислений

Выбранный набор *элементарных операций* определяет *модель вычислений*, в которой мы работаем. В рамках курса мы по умолчанию считаем, что работа происходит на Random Access Machine (RAM) с некоторыми ограничениями. Напомним, что у RAM есть входная лента, выходная лента и память, состоящая из последовательности регистров.

Элементарные операции RAM зависят от конкретной постановки, но обычно в них входят чтение со входной и запись чисел в выходную ленты, простейшие побитовые операции (логические И, ИЛИ, НЕ, сдвиги) и целочисленная арифметика (сложение, умножение, деление нацело, вычисление остатка), ветвления, циклы и чтение/запись чисел в произвольные регистры, к которым можно обращаться по индексам напрямую или через числа, записанные в других регистрах.

Дешёвая работа с числами сколь угодно большой длины вызывает проблемы, о которых будет сказано позже, поэтому применяются модели с дополнительными ограничениями:

1. Битовая. Элементарными считаются только операции, которые выполняются над двумя битами, а число x занимает $\lceil \log_2 x \rceil$ единиц памяти. Таким образом, сложение и умножение чисел a и b, сделанные "в столбик", займут $\lceil \log_2 a \rceil + \lceil \log_2 b \rceil$ и $\lceil \log_2 a \rceil \cdot \lceil \log_2 b \rceil$ элементарных операций соответственно.

2. Word-RAM. Считаем, что напрямую машина может работать только с числами длины не больше размера машинного слова w, то есть, с числами не больше 2^w. При этом дополнительно считается, что $w \geq \lceil \log_2 n \rceil$, где n – длина слова, записанного на входной ленте. В такой модели операции RAM будут считаться элементарными, если проводятся над машинными словами, а при необходимости работать с большими числами, их можно будет разбивать на машинные слова.

Есть также модели вещественнозначной арифметики, но их мы подробно рассматривать не будем. В задачах курса, если не указано обратное, будет использоваться Word-RAM модель.

Время работы алгоритма

Пусть мы определились с моделью $\mathcal{M}$ и рассматриваем алгоритм $\mathcal{A}$, которому на вход подаётся слово x. *Временем работы* $T_{\mathcal{A}}(x)$ алгоритма $\mathcal{A}$ на слове x назовём число элементарных операций, совершённых алгоритмом. Нас интересует зависимость этой величины от размера входа, поэтому введём агрегированные величины:

1. *Время работы в худшем случае* на входе длины n: $f_{\max}(n) = \max\limits_{|x|=n} T_{\mathcal{A}}(x)$

2. *Время работы в лучшем случае* на входе длины n: $f_{\min}(n) = \min\limits_{|x|=n} T_{\mathcal{A}}(x)$

3. *Время работы в среднем* на входе длины n: $f_{\text{avg}}(n) = \mathop{\mathbb{E}}\limits_{|x|=n} T_{\mathcal{A}}(x)$

4. *Рандомизированное время работы* на входе длины n: $f_{\text{rand}}(n) = \mathop{\mathbb{E}}\limits_{w} \max\limits_{|x|=n} T_{\mathcal{A}}(x, w)$

Здесь и далее $\mathop{\mathbb{E}}\limits_{x} f(x)$ обозначает математическое ожидание величины $f(x)$. Если распределение величины x нам не известно, его стоит считать равномерным. В определении рандомизированного времени работы у алгоритма, помимо входа, есть источник случайности, дающий ему доступ к конечной последовательности w из независимо и равномерно выбранных бит.

Ассимптотические оценки

Нотация $O(\cdot), o(\cdot), \Omega(\cdot), \omega(\cdot), \Theta(\cdot)$

Как уже было сказано, мы считаем, что реальное время работы алгоритма будет пропорционально числу элементарных операций, поэтому разумным будет определять это число с точностью до постоянного множителя. Формально для этого будут использоваться асимптотические обозначения, известные по курсу математического анализа[1]:

1. $f(n) \in O(g(n)) \iff$ "$f(n)$ *асимптотически ограничена сверху* функцией $g(n)$":
$$\exists c > 0 : \exists n_0 : \forall n > n_0 \hookrightarrow f(n) \leq c \cdot g(n)$$
2. $f(n) \in \Omega(g(n)) \iff$ "$f(n)$ *асимптотически ограничена снизу* функцией $g(n)$":
$$\exists c > 0 : \exists n_0 : \forall n > n_0 \hookrightarrow c \cdot g(n) \leq f(n)$$
3. $f(n) \in o(g(n)) \iff$ "$g(n)$ *асимптотически доминирует* над $f(n)$":
$$\forall c > 0 : \exists n_0 : \forall n > n_0 \hookrightarrow f(n) < c \cdot g(n)$$
4. $f(n) \in \omega(g(n)) \iff$ "$f(n)$ *асимптотически доминирует* над $g(n)$":
$$\forall c > 0 : \exists n_0 : \forall n > n_0 \hookrightarrow c \cdot g(n) < f(n)$$
5. $f(n) \in \Theta(g(n)) \iff$ "$f(n)$ *асимптотически ограничена сверху и снизу* функцией $g(n)$":
$$\exists c_1, c_2 > 0 : \exists n_0 : \forall n > n_0 \hookrightarrow c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$
6. $f(n) \sim g(n) \iff$ "$f(n)$ *асимптотически эквивалентна* $g(n)$":
$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$$

Определения оценок можно упростить, записав их через частичные верхние пределы:

$$f(n) \in O(g(n)) \iff \overline{\lim_{n\to\infty}} \frac{f(n)}{g(n)} < \infty, \quad f(n) \in \Omega(g(n)) \iff \overline{\lim_{n\to\infty}} \frac{g(n)}{f(n)} < \infty,$$
$$f(n) \in o(g(n)) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0, \quad f(n) \in \omega(g(n)) \iff \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$$

Выражение $f(n) \in O(g(n))$ может также читаться как "$f(n)$ есть O-большое от $g(n)$" и записываться как $f(n) = O(g(n))$. Последняя запись удобна, но не совсем корректна, так как из $f(n) = O(g(n))$ не следует $g(n) = O(f(n))$. Строго говоря, $O(g(n))$ – это семейство функций $f(n)$, для которых выполнено указанное определение.

Неформально эти обозначения описывают скорость роста функций: $f(n) \in O(g(n))$ значит, что с точностью до постоянного множителя $f(n)$ растёт не быстрее, чем $g(n)$, а $f(n) \in o(g(n))$ значит, что $f(n)$ будет расти строго медленнее, чем $g(n)$, даже если уменьшить $g(n)$ в любое заранее заданное число раз. При исследований алгоритмов нас, чаще всего, будет интересовать либо Θ-, либо O-оценки времени работы алгоритма.

[1] В матанализе, сравнивая функции асимптотически, нужно также указывать точку, в которой они сравниваются и говорить о том, что неравенства над *модулями* функций выполняются в некоторой её окрестности. Здесь же $f(n)$ и $g(n)$ стоит воспринимать как последовательности натуральных чисел, а не как произвольные функции, поэтому можно считать, что точкой сравнения мы всегда будем считать $+\infty$ со всеми вытекающими.

1. (2б) $f(n)$ и $g(n)$ – неотрицательные неубывающие функции. Расставьте $\Longleftarrow$, $\Longrightarrow$ или $\Longleftrightarrow$ в записи снизу (следствие надо доказать, отсутствие следствия подтвердить примером):

$$f(n) \in O(g(n)) \overset{?}{\Longleftrightarrow} f(n) \notin \omega(g(n)) \overset{?}{\Longleftrightarrow} g(n) \notin o(f(n)) \overset{?}{\Longleftrightarrow} g(n) \in \Omega(f(n))$$

Изменится ли ответ, если убрать требование неубывания?

2. (3б) Найдите Θ-оценки следующих величин при $n \to \infty$:

1. $\binom{n}{k}$ 2. $\binom{n^8 + 2n^4}{n^4}$ 3. $\binom{2n}{\lfloor \frac{n}{3} \rfloor}$

Напомним, что для факториала имеет место формула Стирлинга $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

1*. (0.5б) Рассмотрим следующее альтернативное определение:

$$f'_{\text{rand}}(n) = \max_{|x|=n} \mathop{\mathbb{E}}_{w} T_{\mathcal{A}}(x, w)$$

Верно ли, что для любой неотрицательной неубывающей по $|x|$ функции $T_{\mathcal{A}}(x, w)$ верно:

1. Что $f'_{\text{rand}}(n) = \Theta(f_{\text{rand}}(n))$?
2. Что $f'_{\text{rand}}(n) \sim f_{\text{rand}}(n)$?

Это – ваша первая бонусная задача.

Оценки для конечных сумм

Пусть $f(n)$ – неубывающая функция и нужно оценить $\sum\limits_{k=1}^{n} f(k)$. Для этого можно оценить данную сумму сверху и снизу соответствующими интегралами. Из $f(\lfloor x \rfloor) \le f(x) \le f(\lceil x \rceil)$ следует, что:

$$\sum_{k=1}^{n-1} f(k) = \int_1^n f(\lfloor x \rfloor)dx \le \int_1^n f(x)dx \le \int_1^n f(\lceil x \rceil)dx = \sum_{k=2}^{n} f(k)$$

Используя данные неравенства, можно получить, что:

$$\int_1^n f(x)dx \le \sum_{k=1}^{n} f(k) \le \int_1^{n+1} f(x)dx$$

Рассмотрим $f(k) = k^2 \ln k$. Её первообразная это $\dfrac{3k^3(3\ln k - 1)}{9}$, поэтому:

$$\frac{3n^3(\ln n - 1)}{9} \le \sum_{k=1}^{n} k^2 \ln k \le \frac{3(n+1)^3(\ln(n+1) - 1)}{9}$$

$$\sum_{k=1}^{n} k^2 \ln k \sim \frac{k^3 \ln k}{3}$$

3. (2б) Найдите Θ-оценки для $\sum\limits_{k=1}^{n} \dfrac{1}{k}$ и $\sum\limits_{k=1}^{n} \ln k$ в виде функции от n.

4. (2б) Найдите Θ-оценки для $\sum\limits_{k=1}^{n} 2^{\sqrt{k}}$ и $\sum\limits_{k=1}^{n} 2^{k^2}$ в виде функции от n.

Основная теорема о рекуррентных оценках

При анализе времени работы часто нужно решать рекуррентные уравнения вида:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

Так происходит когда функция совершает a рекурсивных вызовов, разбивая исходную задачу на b равных, а процесс разбиения (и слияния результатов) требует $f(n)$ действий. Например:

1. Двоичное возведение в степень: $T\left(\frac{n}{2}\right) + \Theta(1)$
2. Сортировка слияниями: $2T\left(\frac{n}{2}\right) + \Theta(n)$
3. Метод Карацубы умножения чисел: $3T\left(\frac{n}{4}\right) + \Theta(n)$
4. Метод Штрассена умножения матриц: $7T\left(\frac{n}{2}\right) + \Theta(n^2)$

Постепенно раскрывая соотношение, мы можем записать его в виде конечной суммы:

$$T(n) = f(n) + af\left(\frac{n}{b}\right) + a^2T\left(\frac{n}{b^2}\right) = \sum_{i=0}^{?} a^i f\left(\frac{n}{b^i}\right)$$

При $i > \log_b n$ аргумент $f(\cdot)$ будет меньше единицы, поэтому получаем решение

$$T(n) = \sum_{i=0}^{\lfloor \log_b n \rfloor} a^i f\left(\frac{n}{b^i}\right)$$

Основная теорема фактически является рассмотрением ряда частных случаев этой ситуации.

1. $f(n) = O(n^c)$ и $c < \log_b a$. Тогда $T(n) = \Theta(n^{\log_b a})$:

$$T(n) = n^c \sum_{i=0}^{\lfloor \log_b n \rfloor} O\left(\frac{a^i}{b^{ic}}\right) = n^c O\left(\frac{\left(\frac{a}{b^c}\right)^{1+\lfloor \log_b n \rfloor} - 1}{\frac{a}{b^c} - 1}\right) = n^c O\left(\left[\frac{a}{b^c}\right]^{\log_b n}\right)$$

$$a^{\log_b c} = c^{\log_b a} \Longrightarrow T(n) = n^c O\left(\frac{n^{\log_b a}}{n^{\log_b b^c}}\right) = O(n^{\log_b a})$$

   Чтобы получить оценку снизу, достаточно проделать те же операции, взяв $f(n) = \Omega(1)$.

2. $f(n) = \Theta(n^c \log^k n)$, где $c = \log_b a$. Тогда $T(n) = \Theta(n^c \log^{k+1} n)$.
3. $f(n) = \Omega(n^c)$, $c > \log_b a$, и для некоторых n_0 и $k < 1$ выполнено условие регулярности:

$$af\left(\frac{n}{b}\right) \le kf(n)$$

   В этом случае $T(n) = \Theta(f(n))$.

5. (2б) Приведите неубывающую $f(n)$, показывающую, что условие регулярности значимо.

6. (2б) Оцените трудоёмкость рекурсивного алгоритма, разбивающего исходную задачу размера n на n задач размеров $\left\lceil \frac{n}{2} \right\rceil$ каждая, используя для этого $O(n)$ операций.

7. (2б) Оцените трудоёмкость рекурсивного алгоритма, разбивающего исходную задачу размера n на 2 задачи размера $\left\lceil \frac{n}{2} \right\rceil$ каждая, используя для этого $\frac{n}{\log n}$ операций.

2*. (0.5б) Чтобы посчитать некоторую функцию $f(n)$, нужно сперва посчитать её значение в точках $\left\lceil \frac{n}{4} \right\rceil$ и $\left\lceil \frac{3n}{4} \right\rceil$. Оцените количество точек, в которых нужно сперва вычислить функцию, чтобы было возможно посчитать значение в точке n.

Линейные рекуррентные уравнения

Общая теория

Рассмотрим линейные однородные рекуррентные соотношения порядка k:

$$f_{n+k} = \sum_{i=0}^{k-1} a_i f_{n+i}$$

У них много общего с линейными диффурами, но работаем мы теперь не с функциями, а с последовательностями. Например, если у соотношения порядка k заданы начальные k значений, все остальные элементы заданной последовательности определяются однозначно. Примеры:

1. Числа Фибоначчи: $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$

$$\begin{cases} F_1 = F_2 = 1, \\ F_n = F_{n-1} + F_{n-2} \end{cases}$$

2. Последовательность квадратов: $c_n = n^2$.

$$\begin{cases} c_0 = 0, c_1 = c_{-1} = 1, \\ c_n = 3c_{n-1} - 3c_{n-2} + c_{n-3} \end{cases}$$

Как и линейные диффуры, линейные рекурренты могут быть решены с помощью характеристических уравнений. Рассмотрим общий подход, стоящий за этим. Обозначим символом Δ линейный оператор сдвига, который сопоставляет последовательности a_n последовательность, у которой на k-ом месте стоит элемент a_{k+1}. Рассмотрим также степени этого оператора Δ^k, ставящие на позицию n элемент a_{n+k}. Теперь можно записать исходное уравнение как:

$$\Delta^k f = \sum_{i=0}^{k-1} a_i \Delta^i f \iff \left(\Delta^k - \sum_{i=0}^{k-1} a_i \Delta^i \right) f = 0$$

Решение уравнения образуют ядро получившегося справа линейного оператора[2], значит, множество решений образует линейное пространство. При этом размерность данного пространства не превышает k, т.к. нам достаточно указать $a_0, \ldots, a_{k-1}$ чтобы однозначно задать последовательность. Считая, что коэффициенты взяты из $\mathbb{C}$, можно представить уравнение в виде:

$$\left(\prod_i (\Delta - \lambda_i)^{p_i} \right) f = 0$$

Где λ_i – корни характеристического уравнения с учётом их кратностей p_i:

[2]Решения линейного однородного диффура в свою очередь образуют ядро оператора $\left(\frac{d^k}{dx^k} - \sum\limits_{i=0}^{k-1} a_i \frac{d^i}{dx^i} \right)$.

$$\prod_i (\lambda - \lambda_i)^{p_i} = \lambda^k - \sum_{i=0}^{k-1} a_i \lambda^i = 0$$

Это позволит нам свести задачу решения произвольного уравнения к более простой задаче:

$$(\Delta - \lambda_i)^{p_i} f = 0$$

Как из решений таких уравнений получить общее? Если линейные операторы A и B коммутируют, то $\ker A + \ker B \subset \ker AB$, поэтому можно рассмотреть линейную комбинацию всех решений примитивных уравнений. При $p_i = 1$ решением будет $f_n = f_0 \lambda_i^n$. Уравнения более высокого порядка решаются по индукции. Пусть f^k – решение уравнения при $p_i = k$. Тогда:

$$[(\Delta - \lambda_i)^{k-1}](\Delta - \lambda_i) f^k = 0 \implies (\Delta - \lambda_i) f^k = f^{k-1} \implies \Delta f^k = \lambda_i f^k + f^{k-1}$$

Таким образом, мы свели решение произвольного однородного рекуррентного соотношения к неоднородному соотношению, но первого порядка. Найдём решение f^2 для $(\Delta - \lambda_i)^2 f = 0$. В коэффициентах линейное уравнение будет иметь вид $f_{n+1} = \lambda_i f_n + f_0 \lambda_i^n$. По аналогии с диффурами, можно найти общее решение неоднородного уравнения, варьируя постоянную. Будем искать решение в виде $f_n = C_n \lambda_i^n$, где C_n – неизвестная последовательность:

$$C_{n+1} \lambda_i^{n+1} = C_n \lambda_i^{n+1} + f_0 \lambda_i^n$$

$$C_{n+1} - C_n = \frac{f_0}{\lambda_i} \Longrightarrow C_n = C_0 + n \cdot \frac{f_0}{\lambda_i} = c_1 + c_2 n$$

Значит, f^2 может быть представлено как $(c_1 + c_2 n)\lambda_i^n$, где c_1, c_2 – произвольные константы. Продолжая данные рассуждения по индукции можно показать, что $f^k = P(n)\lambda_i^n$, где $P(n)$ – многочлен степени не выше $k-1$. Коэффициенты многочленов можно найти из $a_0, \ldots, a_{k-1}$.

Общее решение однородного уравнения

Подытожим полученный результат общим алгоритмом решения линейного однородного рекуррентного уравнения с постоянными коэффициентами. Пусть у нас есть уравнение $f_{n+k} = \sum_{i=0}^{k-1} a_i f_{n+i}$. Выпишем его характеристическое уравнение:

$$\lambda^k - \sum_{i=0}^{k-1} a_i \lambda^i = \prod_i (\lambda - \lambda_i)^{p_i}$$

Где λ_i – корни уравнения, а p_i – их кратности. Общее решение уравнения будет иметь вид:

$$f_n = \sum_i P_i(n) \lambda_i^n$$

Где $P_i(n)$ – произвольные многочлены степени не выше $p_i - 1$.

Числа Фибоначчи Решим уравнение $f_{n+1} = f_n + f_{n-1}$. Его характеристическое уравнение имеет вид $\lambda^2 - \lambda - 1 = 0$. Его корни: $\lambda_{1,2} = \frac{1 \pm \sqrt{5}}{2}$. Отсюда следует общее решение:

$$f_n = C_1 \left(\frac{1 - \sqrt{5}}{2} \right)^n + C_2 \left(\frac{1 + \sqrt{5}}{2} \right)^n$$

Найдём частное решение для случая $f_0 = 0, f_1 = 1$:

$$\begin{cases} C_1 + C_2 = 0, \\ C_1\left(\dfrac{1-\sqrt{5}}{2}\right) + C_2\left(\dfrac{1+\sqrt{5}}{2}\right) = 1 \end{cases} \implies C_1\left[\frac{1-\sqrt{5}}{2} - \frac{1+\sqrt{5}}{2}\right] = 1 \implies C_1 = -C_2 = -\frac{1}{\sqrt{5}}$$

То есть,

$$f_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

Что совпадает с формулами Бине.

Комплексные корни Пусть среди корней характеристического уравнения есть комплексные числа. Так как исходное уравнение обычно имеет вещественные коэффициенты, можем считать, что вместе с решением $\rho e^{i\varphi}$ всегда также входит решение $\rho e^{-i\varphi}$. От них можно линейным преобразованием перейти к $\rho\cos\varphi = \dfrac{\rho e^{i\varphi} + \rho e^{-i\varphi}}{2}$ и $\rho\sin\varphi = \dfrac{\rho e^{i\varphi} - \rho e^{-i\varphi}}{2i}$, так как множество решений линейной рекурренты образуют линейное пространство. Соответственно, если мы захотим запараметризовать все вещественные решения, в общем решении нужно будет вместо $P_1(n)\lambda_i^n = P_1(n)\rho^n e^{in\varphi}$ и $P_2(n)\hat{\lambda}_i^n = P_2(n)\rho^n e^{-in\varphi}$ рассматривать $P_1(n)\rho^n\cos n\varphi$ и $P_2(n)\rho^n\sin n\varphi$.

Неоднородные уравнения Решение неоднородного уравнения следует искать как сумму общего решения однородного уравнения и частного решения неоднородного. Частное решение в общем случае можно искать методом вариации постоянной. Но для некоторых случаев можно просто искать их в подходящем виде.

Решим для примера уравнение $f_{n+1} = 4f_n - 4f_{n-1} + 2^n$.

$$\lambda^2 - 4\lambda + 4\lambda = 0 \to \lambda_{1,2} = 2$$

Общее решение однородного уравнения будет иметь вид $f_n = (C_1 + C_2 n)2^n$. Неоднородность у нас имеет вид 2^n, основание совпадает с собственным числом λ кратности 2, что порождает резонанс, поэтому частное решение стоит искать в виде $Cn^2 2^n$.

$$C(n+1)^2 2^{n+1} = 4Cn^2 2^n - 4C(n-1)^2 2^{n-1} + 2^n$$

$$C(n+1)^2 2^2 = 4Cn^2 \cdot 2 - 4C(n-1)^2 + 2$$

$$4C(n^2 + 2n + 1) = 8Cn^2 - 4C(n^2 - 2n + 1) + 2 \Longrightarrow C = \frac{1}{4} = 2^{-2}$$

Таким образом, общее решение исходного имеет вид $f_n = (C_1 + C_2 n)2^n + 2^{n-2}n^2$.

8. (1б) Сформулируйте рекуррентное соотношение порядка 4, задающее $f_n = n^3$.

9. (2б) Найдите общее решение неоднородного рекуррентного уравнения:

$$f_{n+1} = 3f_n + 4f_{n-1} + n^2$$

10. (1б) Найдите общее решение рекуррентного уравнения:

$$f_{n+1} + f_n + f_{n-1} = 0$$

В итоговом ответе не должно быть комплексных чисел.

3*. (3б) Пусть A – некоторый линейный оператор. Докажите или опровергните, что:

$$\ker \prod_{i=1}^{k} (A - \lambda_i)^{d_i} = \sum_{i=1}^{k} \ker (A - \lambda_i)^{d_i}$$

Здесь d_i, k – натуральные числа, λ_i – попарно различные вещественные числа.

При доказательстве допустимо ввести на оператор и пространства, над которыми он определён какие-либо дополнительные разумные ограничения. В таком случае баллы будут пересчитаны в зависимости от содержательности новой постановки. Решение для случая когда A – матрица оценивается в 1 балл.

Алгоритмы умножения. Методы Карацубы и Штрассена

Умножение длинных чисел

Все знают алгоритм, работающий за $O(n^2)$: умножение в столбик. Долгое время предполагалось, что ничего быстрее придумать нельзя. Первым эту гипотезу опроверг Анатолий Карацуба, в дальнейшем были разработаны ещё более быстрые методы, основанные на быстром преобразовании Фурье. Пусть мы перемножаем $A = a_0 + a_1 x$ и $B = b_0 + b_1 x$. Тогда:

$$A \cdot B = a_0 b_0 + (a_0 b_1 + a_1 b_0)x + a_1 b_1 x^2 =$$
$$= a_0 b_0 + [(a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1]x + a_1 b_1 x^2$$

Пусть для простоты числа нам даны в двоичной системе счисления и имеют длину n. Тогда если мы возьмём $x = 2^k, k \approx n/2$, то мы сведём задачу к трём вызовам той же задачи, но в два раза меньшего размера: для $a_0 b_0$, $a_1 b_1$ и $(a_0 + a_1)(b_0 + b_1)$. Оценим время работы:

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n) = \Theta\left(n^{\log_2 3}\right) = \Theta(n^{1.584\ldots})$$

11. (2б) Пусть у вас есть k чисел, суммарное количество цифр в которых равно n. Предложите алгоритм на основе метода Карацубы с как можно меньшим временем работы для нахождения произведения этих чисел.

Алгоритм Штрассена

Пусть нам нужно найти матрицу $C = AB$, где A и B – матрицы размера $n \times n$. По определению можно считать эту матрицу по формуле $C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$. Время работы такого алгоритма будет равно $\Theta(n^3)$. Алгоритм Штрассена, улучшающий этот результат, был предложен в 1969 году. Будем считать, что n – степень двойки, если это не верно, можно дополнить матрицы нулевыми столбцами и строками, чтобы это выполнялось. Представим матрицы в блочном виде:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Где A_{ij}, B_{ij}, C_{ij} – матрицы размера $\frac{n}{2}$. В таких обозначениях можно записать

$$C_{ij} = A_{i1} B_{1j} + A_{i2} B_{2j}$$

Это сводит задачу умножения больших матриц к умножению мелких. Однако суммарно требует 8 умножений матриц и 4 сложения, по два умножения и одному сложению на каждый из

четырёх C_{ij}. При этом переходить мы будем от матриц порядка n к матрицам порядка $\frac{n}{2}$, то есть, общее время работы будет равно

$$T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2) = \Theta(n^{\log_2 8}) = \Theta(n^3)$$

Чтобы решить эту проблему, было предложено ввести следующие вспомогательные матрицы:

$$\begin{gathered} P_1 = (A_{11} + A_{22})(B_{11} + B_{22}) \\ P_2 = (A_{21} + A_{22})B_{11} \\ P_3 = A_{11}(B_{12} - B_{22}) \\ P_4 = A_{22}(B_{21} - B_{11}) \\ P_5 = (A_{11} + A_{12})B_{22} \\ P_6 = (A_{21} - A_{11})(B_{11} + B_{12}) \\ P_7 = (A_{12} - A_{22})(B_{21} + B_{22}) \end{gathered}$$

Можно проверить, что все C_{ij} можно выразить через P_k:

$$\begin{gathered} C_{11} = P_1 + P_4 - P_5 + P_7 \\ C_{12} = P_3 + P_5 \\ C_{21} = P_2 + P_4 \\ C_{22} = P_1 - P_2 + P_3 + P_6 \end{gathered}$$

Таким образом, нам потребуется всего 7 умножений матриц и 18 сложений. Время работы будет:

$$T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2) = \Theta(n^{\log_2 7}) = \Theta(n^{2.81\ldots})$$

Порядковая статистика и быстрая сортировка, медиана

Рассмотрим следующий алгоритм сортировки: выберем некоторый элемент массива, затем отнесём в одну половину все элементы меньше него, а в другую – все элементы, которые больше либо равны ему. Затем запустимся рекурсивно в эти две части и отсортируем их. В результате получим отсортированный массив.

Аналогичный алгоритм можно выразить для поиска k-ой порядковой статистики в массиве. Мы разбиваем массив на две части относительно некоторого элемента и смотрим, сколько элементов перешли в первую часть. Если их a и $a < k$, то мы рекурсивно запустимся во вторую часть с $k' = k - a$, иначе в первую с тем же k. В конце у нас будет только один элемент – ответ.

Чтобы понять за сколько алгоритм работает нужно определиться с тем, как выбирать опорный элемент. Из комбинаторных соображений можно понять, что лучше всего будет брать медиану. Тогда время работы первого алгоритма будет $T(n) = 2T\left(\frac{n}{2}\right) + O(n) = \Theta(n \log n)$, а второго – $T(n) = T\left(\frac{n}{2}\right) + O(n) = \Theta(n)$. Оказывается, можно искать не точную медиану, а элемент, который точно будет иметь достаточно много элементов как меньше, так и больше него.

Один из известных линейных алгоритмов поиска k-ой статистики (BFPRT-Алгоритм) имеет следующий вид: разобьём массив на группы из пяти элементов. В каждой из групп выберем медиану за $O(1)$. Затем найдём медиану этих медиан (т.е. $n/2$-ю статистику), обозначим её s и разделим множество на два по ключам относительно неё. Половина пятёрок будут иметь медиану не больше s, а другая половина имеет медиану не меньше s, значит, хотя бы $\frac{1}{2}\frac{3}{5}n = \frac{3}{10}n$ элементов будут в каждом из множеств. Сравним теперь количество элементов в левой половине a с требумой статистикой k. Если элементов больше либо равно, найдём k-ую статистику в левой половине. В противном случае либо s – ответ если $k - a = 1$ либо нужно вызваться в правую

половину с числом $k - a - 1$. Так как размер обеих половин будет не больше, чем $\frac{7n}{10}$, получим в итоге следующее соотношение:

$$T(n) = \Theta(n) + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) = \Theta(n)$$

12. (2б) Докажите, что указанную рекурренту можно оценить как $\Theta(n)$. Что случится с асимптотикой если бить на группы по три элемента? По четыре?