Основные алгоритмы, комментарии по третьему листку (Алгоритмы «разделяй и властвуй»)

Шибаев Иннокентий

March 1, 2021

1 По поводу теории

Основным теоретическим результатом в данном листке была т.н. основная теорема о рекуррентных соотношениях (a.k.a Master theorem for divide-and-conquer recurrences, или, коротко, Master theorem). Важно отметить, что в нашем курсе используется именно этот вариант Master theorem, их много, на Википедии к примеру вариант похожий, но более сильный для второго случая (там рассматривается случай $f(n) = \Theta(n^d \log^k n)$, и доказывается что тогда $T(n) = \Theta(n^d \log^{k+1} n)$, у нас же только для $k = 0$). Есть и другое обобщение - теорема Акра-Баззи, которая может также решать случаи с неодинаковым ветвлением, но опять же – в курсе (и на контрольных) мы используем именно вариант который приведен ниже и взят из лекций:

Теорема 1.1 (Master theorem)**.** Пусть $n, a, b \in \mathbb{N}\backslash\{0\}$ и $d = \log_b a$. Для рекуррентных соотношений вида

- $T(n) = aT\left(\frac{n}{b}\right) + f(n)$
- $T(n) = aT\left(\left\lfloor\frac{n}{b}\right\rfloor\right) + f(n)$
- $T(n) = aT\left(\left\lceil\frac{n}{b}\right\rceil\right) + f(n)$

справедливо следующее утверждение:

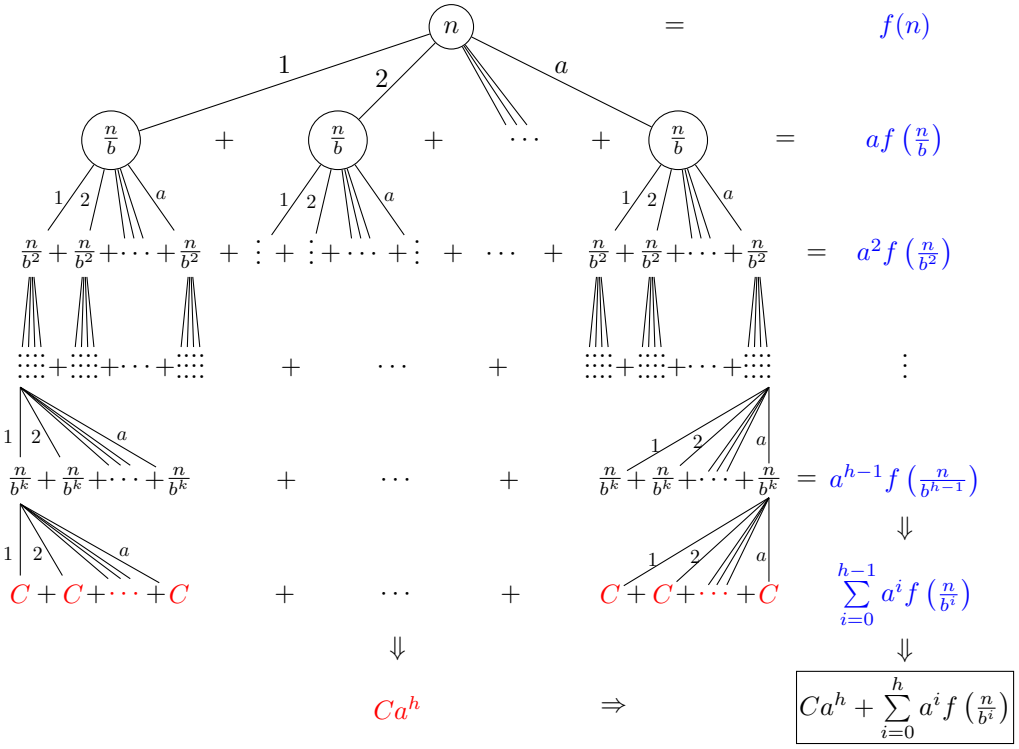
1. Если $f(n) = O(n^{d-\varepsilon})$ для некоторого $\varepsilon > 0$, то $T(n) = \Theta(n^d)$.
2. Если $f(n) = \Theta(n^d)$, то $T(n) = \Theta(n^d \log n)$.
3. Если

   а) $f(n) = \Omega(n^{d+\varepsilon})$ для некоторого $\varepsilon > 0$ и

   б) $\exists c : 0 < c < 1, af\left(\frac{n}{b}\right) \leqslant cf(n)$,

   то $T(n) = \Theta(f(n))$.

Схема доказательства. На самом деле она не сильно отличается от метода решения этой рекурренты который мы применяли ранее. Рассмотрим дерево рекурсии:



На каждом уровне дерева рекурсии (кроме листьев) мы совершаем некоторую работу (некоторый объем вычислений) связанную с $f(n)$ (в узлах дерева), в листьях мы суммарно совершаем работу C умноженную на кол-во листьев (a^h). Высота дерева h задается как $\log_b n$. Таким образом получаем что суммарное число операций имеет вид (дополнительно предположим что $n = b^k$, для простоты)

$$Ca^{\log_b n} + \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) = Cn^{\log_b a} + \sum_{i=1}^{\log_b n} a^{\log_b n - i} f\left(\frac{n}{b^{\log_b n - i}}\right) =$$

$$= Cn^d + n^d \sum_{i=1}^{\log_b n} \frac{f\left(b^i\right)}{a^i} =$$

$$= n^d \left(C + \sum_{i=1}^{\log_b n} \frac{f\left(b^i\right)}{a^i}\right)$$

и дальше уже можно рассматривать варианты, но уже из этого видно что результат как минимум $\Omega\left(n^d\right)$ (что мы и видим в условии теоремы). Кстати второй вариант из этого получается очень просто – если $f(n) = \Theta\left(n^d\right)$ то так как $d = \log_b a$ то дробь в сумме превращается в константу, а именно

$$n^d \left(C + \sum_{i=1}^{\log_b n} \frac{f\left(b^i\right)}{a^i}\right) \leqslant n^d \left(C + \sum_{i=1}^{\log_b n} \frac{C_2 b^{i \log_b a}}{a^i}\right) = n^d \left(C + \sum_{i=1}^{\log_b n} \frac{C_2 a^i}{a^i}\right) =$$

$$= n^d \left(C + C_2 \log_b n\right) \leqslant 2 \max\{C, C_2\} n^d \log_b n$$

и таким же образом оцениваем снизу. □

Примечание 1.1. Обсудим смысл различных случаев теоремы. Сложность алгоритма у нас формируется из двух вещей – вклада от листьев, и вклада от работы на уровнях выше листьев (к примеру при слиянии массивов в MergeSort мы делаем $O(n)$ операций). Тогда можно сказать следующее:

- Первый случай, когда $f(n) = O(n^{d-\varepsilon})$ – это случай когда сумма в скобках это константа, мы фактически получаем что основной вклад у нас идет от листьев. Это, к примеру, происходит в алгоритме быстрого перемножения чисел Карацубы – само объединение результатов (сложение чисел) намного легче чем их перемножение, и в результате основным становится вклад именно листьев.
- Второй случай, когда $f(n) = \Theta(n^d)$ – и тут уже вклад от работы на уровнях больше – сумма в скобках превращается просто в высоту дерева. Это случай алгоритма MergeSort, к примеру.
- Последний случай говорит о том когда работа на одном верхнем уровне сопоставима с работой по всему дереву.

Примечание 1.2. В данной формулировке важно что в первом случае есть $\varepsilon > 0$. К примеру вот для такого случая

$$T(n) = 4f\left(\frac{n}{2}\right) + \frac{n^2}{\log n}$$

имеем $d = 2$, но при этом неверно что $f(n) = O\left(n^{d-\varepsilon}\right)$ (т.к. логарифм растет медленнее любого n^ε, это можно к примеру через правило Лопиталя легко получить). Неверно также и что $f(n) = \Omega\left(n^{d+\varepsilon}\right)$, и под второй случай такое $f(n)$ очевидным образом не подходит. Опять же, если использовать здесь вариант теоремы с wiki то это будет относится ко второму случаю, однако в нашем варианте теоремы этот случай не покрывается. Еще можно ломать применимость добавляя зависимость от уровня, к примеру $f(n) = n^{2+\cos(\pi n)}$, для такого можно только сказать что $f(n) = O(n^3)$ и $f(n) = \Omega(n)$. Есть и другие контрпримеры, список на wiki.

Примечание 1.3. Через основную теорему можно получить оценки для некоторых стандартных алгоритмов. К примеру:

Алгоритм	Рекуррента	d	Случай теоремы	Результат
Бинарный поиск	$T(n) = T\left(\frac{n}{2}\right) + C$	$\log_2 1 = 0$	2 случай	$\Theta(\log n)$
MergeSort	$T(n) = 2T\left(\frac{n}{2}\right) + Cn$	$\log_2 2 = 1$	2 случай	$\Theta(n \log n)$
Алгоритм Карацубы	$T(n) = 3T\left(\frac{n}{2}\right) + Cn$	$\log_2 3 > 1$	1 случай	$\Theta\left(n^{\log_2 3}\right)$
Алгоритм Штрассена	$T(n) = 7T\left(\frac{n}{2}\right) + Cn^2$	$\log_2 7 > 2$	1 случай	$\Theta\left(n^{\log_2 7}\right)$

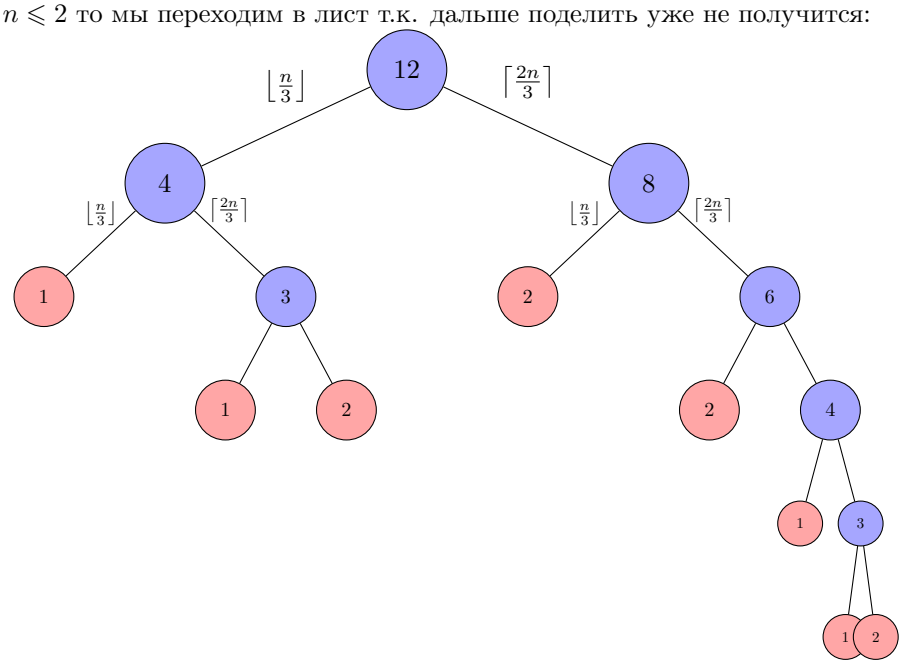
В этой таблице нет примеров использования третьего случая, и я как-то не могу вспомнить алгоритм который бы проходил по третьему случаю. В определенном смысле таковым является поиск k-й порядковой статистики, т.к. там мы имея $T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + Cn$ фактически (из-за того что $f(n)$ здесь линейна) переходим к чему-то вроде $T(n) = 9T\left(\frac{n}{10}\right) + Cn$ что как раз проходит по третьему случаю и дает нам искомое $\Theta(n)$, но это **очень** неформально, и использовать такое рассуждение ни в коем случае не надо. Есть еще одна теорема из второго издания Кормена (Theorem 28.8: (Inversion is no harder than multiplication)) в которой третий случай используется в одном из переходов в оценках.

2 По поводу задач

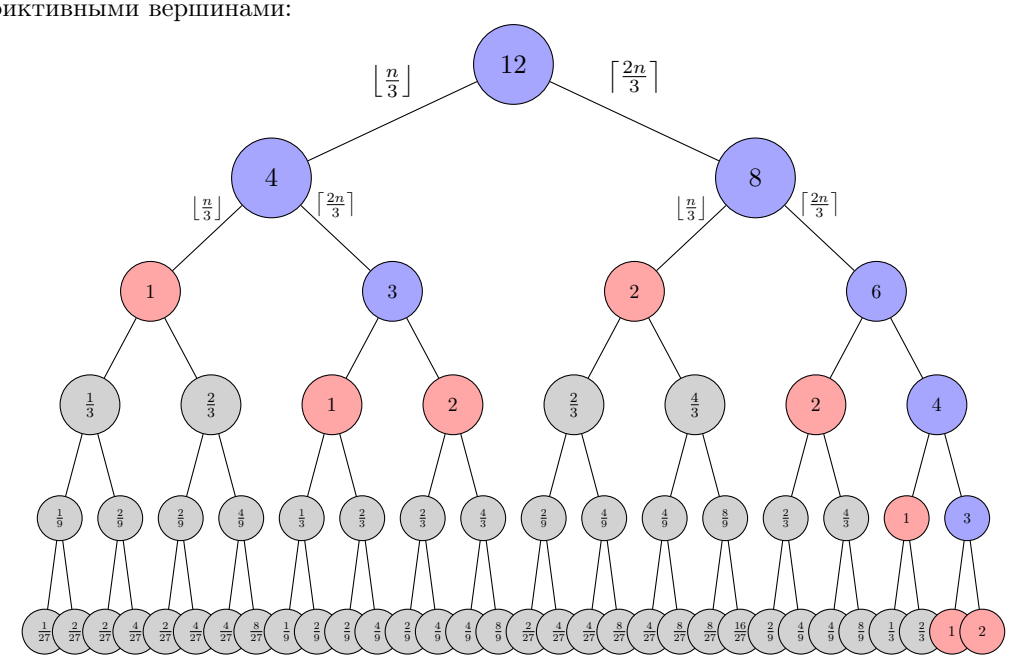
Рассмотрим пример с неравновесным деревом из третьего листка

Задача 2.1. Найти асимптотическую оценку $T(n)$,используя деревья рекурсии: **а)** $T(n) = T\left(\lfloor\frac{n}{3}\rfloor\right) + T\left(\lceil\frac{2n}{3}\rceil\right) + cn$; **б)** $T(n) = 4T\left(\lfloor\frac{n}{2}\rfloor\right) + cn^2 \log n$.

Решение (Пункт **а**)**.** Давайте нарисуем дерево рекурсивных вызовов для $n = 12$ (при этом если $n \leqslant 2$ то мы переходим в лист т.к. дальше поделить уже не получится:



Красным в данном дереве помечены листья, а синим – вершины в которые приходит $n > 2$, т.е. вершины где еще происходит деление на подзадачи. Оценка здесь тем же способом как раньше в принципе возможна, но надо очень аккуратно считать листья, и суммы по уровням будут выглядеть также заметно сложнее. Поэтому на семинаре мы попытались упростить себе жизнь, и достроили это дерево до сбалансированного фиктивными вершинами:



На картинке данные фиктивные вершины выделены серым. И дальше мы пытались оценить результат используя уже формулу $Ca^h + \sum_{i=0}^{h} cn$ (где cn в сумме это сумма действий по уровню, аналог $a^i f(n/b^i)$ для нашего случая), для оценки сверху используя

$b = \frac{3}{2}$, а для оценки снизу – $b = 3$:

$$C2^{\log_3 n} + cn\log_3 n \leqslant T(n) \leqslant C2^{\log_{3/2} n} + cn\log_{3/2} n$$
$$Cn^{\log_3 2} + cn\log_3 n \leqslant T(n) \leqslant Cn^{\log_{3/2} 2} + cn\log_{3/2} n$$

и мы получали следующую проблему – оценки сверху и снизу имеют разный порядок. Снизу мы оцениваем как $n\log n$, а сверху как $n^{\log_{3/2} 2} \approx n^{1.7}$ значит у нас не получается оценить θ. При этом мы видим, что этот большой вклад пришел от листьев, от члена Ca^h. Вопрос: как решить эту проблему?

1. Вариант первый, который мы рассматривали на семинаре, заключается в "переносе сложности" из листьев в вершины, а именно в том чтобы доказать что можно почти что выбросить из рассмотрения вклад от листьев в этом конкретном случае. Понятно что наша проблема приходит из того что мы неправильно оцениваем кол-во листьев (как легко видеть на последней картинке вершин на нижнем уровне гораздо больше (32) чем реальных листьев (красных, 8), и с ростом n это отличие будет только увеличиваться).

   С другой стороны, у нас граф это бинарное дерево, в нем число вершин не-листьев всего на 1 меньше числа листьев. И мы в определенном смысле считаем эти самые не листовые вершины, когда считаем суммы по уровням – да, мы их там считаем с некоторой константой, в ходе подсчета вклада $f(n)$ но суть не меняется – у нас уже есть они посчитанные в каком-то виде. Ну, или как минимум оценка сверху на их кол-во – ведь в сумму мы также включаем фиктивные вершины, которых на самом деле нет.

   Итак, идея в следующем – возьмем это C из листьев, и добавим его к сложности в вершинах не являющимися листьями. Однако это надо сделать аккуратно – если мы просто рассмотрим новую $f'(n) = f(n) + C$ то у нас получится ровно то же что и было.

   Нам дано что $f(n) = cn$. Вершина является не листом если $n > 2$. Значит если мы добавим C следующим образом: $f'(n) = (c + C)n$ то мы к каждой реальной вершине добавим минимум по $3C$ (т.е. примерно трехкратно учтем листья)! Т.е. мы можем оценить

   $$T(n) \leqslant C \cdot (\text{реальное кол-во листьев}) + cn\log_{3/2} n \leqslant$$
   $$\leqslant 0 \cdot (\text{реальное кол-во листьев}) + (c + \text{C})n\log_{3/2} n = (c + \text{C})n\log_{3/2} n$$

   и мы наконец получили $T(n) = \Theta(n\log n)$. Иными словами, мы добавили сложность из листьев (из реальных листьев, а не последнего слоя) прямо к константе в f, и воспользовались тем что в реальных вершинах $n > 2$, а из вида f поняли куда именно вставить константу чтобы учесть сложность листьев достаточно аккуратно. В фиктивных вершинах она как бы тоже учитывается, но там происходит довольно быстрое уменьшение, поэтому там нам эта константа ничего не портит.

2. Другой вариант это просто подстановка. Мы уже получили оценку снизу, вида $\Omega(n\log n)$, давайте докажем так же оценку сверху, вида $O(n\log n)$. Для этого нам надо по индукции доказать что $T(n) < Dn\log n$:

   $$T(n) = T\left(\left\lfloor\frac{n}{3}\right\rfloor\right) + T\left(\left\lceil\frac{2n}{3}\right\rceil\right) + cn \overset{\text{Инд. переход}}{\leqslant}$$
   $$\leqslant D\left\lfloor\frac{n}{3}\right\rfloor\log\left\lfloor\frac{n}{3}\right\rfloor + D\left\lceil\frac{2n}{3}\right\rceil\log\left\lceil\frac{2n}{3}\right\rceil + cn$$

теперь аккуратно оцениваем $\left\lfloor \frac{n}{3} \right\rfloor < \frac{n}{3}$ и $\left\lceil \frac{2n}{3} \right\rceil < \frac{2n}{3} + 1$, тогда

$$\begin{aligned} T(n) &\leqslant D\frac{n}{3}\log\frac{n}{3} + D\left(\frac{2n}{3}+1\right)\log\left(\frac{2n}{3}+1\right) + cn = \\ &= D\frac{n}{3}\log\frac{n}{3} + D\left(\frac{2n}{3}+1\right)\log\left(\frac{2}{3}+\frac{1}{n}\right) + D\left(\frac{2n}{3}+1\right)\log n + cn = \\ &= Dn\log n - D\frac{n}{3}\log 3 + D\left(\frac{2n}{3}+1\right)\log\left(\frac{2}{3}+\frac{1}{n}\right) + D\log n + cn \end{aligned}$$

заметим, что при $n > 3$ часть $D\left(\frac{2n}{3}+1\right)\log\left(\frac{2}{3}+\frac{1}{n}\right)$ становится отрицательной, т.е. ее можно оценить сверху 0, значит

$$\begin{aligned} T(n) \leqslant Dn\log n - D\frac{n}{3}\log 3 + D\log n + cn &\overset{?}{\leqslant} Dn\log n \\ -D\frac{n}{3}\log 3 + D\log n + cn &\overset{?}{\leqslant} 0 \end{aligned}$$

и при $D = \frac{6c}{\log 3}$ получаем

$$-cn + \frac{6c}{\log 3}\log n \leqslant 0$$

что верно при достаточно больших n, т.к. логарифм растет медленнее n. Таким образом $T(n) \leqslant Dn\log n$ и индуктивный переход доказан, тем самым $T(n) = O(n\log n)$ и с учетом нашей нижней оценки мы получаем $T(n) = \Theta(n\log n)$.

Второй вариант универсальнее, но требует угадывания ответа и довольно аккуратных выкладок в процессе. В данном конкретном случае мы глобально стремились к тому чтобы сохранить как можно аккуратнее убывающий линейно член $-D\frac{n}{3}\log 3$ т.к. наша $f(n) = cn$ и только он мог ее "перетянуть" подбором нужного D.

Решение (Пункт **б)**). Заодно разберем и второй пункт. Там дана рекуррента вида $T(n) = 4T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + cn^2\log n$, дерево уже симметричное, и первое что надо проверить – подходит ли она под условия основной теоремы. В данном случае $d = \log_2 4 = 2$, $f(n) = cn^2\log n > n^2$ но при этом у нас неверно что $f(n) = \Omega(n^{d+\varepsilon})$, так что напрямую теорема в данном случае неприменима.

Воспользуемся формулой которая у нас получалась в обсуждении д-ва этой теоремы:

$$\begin{aligned} Ca^{\log_b n} + \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) &= Cn^{\log_b a} + \sum_{i=0}^{\log_b n - 1} a^i c\frac{n^2}{b^{2i}}\log\left(\frac{n}{b^i}\right) = \\ &= Cn^2 + c\sum_{i=0}^{\log_2 n - 1} 4^i\frac{n^2}{2^{2i}}\log\left(\frac{n}{2^i}\right) = \\ &= Cn^2 + c\sum_{i=0}^{\log_2 n - 1}\left(n^2\log(n) - in^2\log(2)\right) = \\ &= Cn^2 + cn^2\log^2(n) - \frac{\log(n)(\log(n)-1)}{2}n^2\log(2) = \\ &= \Theta\left(n^2\log^2 n\right) \end{aligned}$$

и все, получили $T(n) = \Theta\left(n^2\log^2 n\right)$ (это кстати можно было бы получить из второго случая Master theorem с Википедии, который как раз расписан для $f(n) = \Theta(n^d\log^k n)$, но нам пришлось это сделать руками).