A Real Elementary Approach to the Master Recurrence and Generalizations*

Chee Yap
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012, USA
and
Korea Institute of Advanced Study
Seoul, Korea

January 20, 2011

Abstract

The master theorem is the solution of a well-known divide-and-conquer recurrence in computer science, called here the master recurrence. This paper proves two generalizations of this master theorem. The first extends the treated class of driving functions to a natural class of exponential-logarithmic functions. The second extends the result to the multiterm master recurrence. Our approach views recurrences as real recurrences, stressing the use of elementary techniques and real induction.

This is the full version of an extended abstract published in TAMC 2011 [37]. It is revised on 18 Aug 2020.

1 Introduction

Recurrences arise naturally in Computer Science from the analysis of algorithms, combinatorial analysis and through probabilistic analysis of computation. Techniques for solving recurrences are among the standard repertoire of algorithmic textbooks (e.g., [21, 2, 9, 27, 28, 8, 26, 3, 19]). See Weide [35] and Lueker [24] for early surveys for solving such recurrences. Recurrences are as varied as there are algorithms. A typical recurrence in combinatorial analysis is the linear recurrence, $F(n) = d(n) + \sum_{i=1}^{k} a_i F(n-i)$. Such $F(n)$'s are normally exponential in n. In the census of combinatorial structures (e.g., [4, 15]), functions that grow double exponentially also arise. But such fast growing functions are atypical in the analysis of efficient algorithms. Here, the proto-typical recurrence has the form

$$T(n) = aT(n/b) + d(n) \tag{1}$$

where $a > 0$ and $b > 1$ are arbitrary real numbers, and $d(n) \geq 1$ is the **driving function**. Solving this recurrence and its generalizations is the focus of this paper.

*This work is partially supported by National Science Foundation Grant #CCF-0728977, and also by the Korea Institute of Advanced Study (KIAS) when the author was visiting as a KIAS Scholar.

Two well-known examples of (1) arise in Mergesort (where $a = b = 2, d(n) = n$) and in Strassen's matrix multiplication (where $a = 7, b = 2, d(n) = n^2$). In attempts to improve Strassen's algorithm, we see some rather exotic examples of (1). E.g., Pan's 1978 matrix multiplication algorithm where $a = 143640, b = 70, d(n) = n^2$ (see [6]). In contrast to linear recurrences, the master recurrence has polynomial-bounded solutions unless $d(n)$ itself is non-polynomial. We will call (1) the **master recurrence** since theorems that provide solutions to this recurrence are now widely known as "master theorems". The solutions depend on the nature of $d(n)$. The case where $d(n)$ is multiplicative, *i.e.*, $d(nm) = d(n)d(m)$, is treated in [3, p. 301]. In an influential note, Bentley, Haken and Saxe proved a master theorem ([6, Table 1, p.39]) under a fairly general hypothesis on $d(n)$. The master recurrence can be generalized to

$$T(n) = \sum_{i=1}^{k} a_i T(n/b_i) + d(n) \tag{2}$$

where $a_i > 0$ and $b_i > 1$ are arbitrary real constants ($k \geq 2$). We will call this the **multiterm master recurrence**. An important case is the 2-term master recurrence $T(n) = T(n/b_1) + T(n/b_2) + n$. When $\frac{1}{b_1} + \frac{1}{b_2} < 1$ the solution is linear, $T(n) = \Theta(n)$. This is the basis of the fast median algorithm of Blum, Floyd, Pratt, Rivest and Tarjan [7]. Variants of their median algorithm led to different values for (b_1, b_2). E.g., $(b_1, b_2) = (5, 10/7)$ in [20] and $(b_1, b_2) = (5, 4/3)$ in [23]. Another 2-term recurrence is $T(n) = T(n/2) + T(n/4) + \log^7 n$. The solution $T(n) = \Theta(n^\alpha)$ where $\alpha = 0.694\ldots$ is the query time for the so-called conjugation tree of Edelsbrunner and Welzl [11].

In discussing the literature, it is useful to begin with the "standard" master theorem which solves the master recurrence (1). This is given as Proposition 1 below. There are two kinds of generalizations of this result: (A) The first kind, as in Verma [33], extends the class of driving functions $d(n)$ that are captured by the standard master theorem. Verma's main result [33, Theorem 13] provided integral bounds on solutions when the driving functions $d(n)$ satisfy some general convergence or growth properties. (B) The second kind of generalization comes from extending the master recurrence itself. Thus, Wang and Fu [34, Theorem 3.5] gave an integral bound on solutions to a parametric form of (1) in which the constants a, b are now functions of n. Of course, the multiterm master recurrence (2) represents a generalization of the second kind. An early treatment is found in Purdom and Brown [28]. The first multiterm master theorems in the literature are from Kao [17] and Akra and Bazzi [5]. Akra and Bazzi used an "order transform" of real functions to prove an integral bound [5, Theorem 3] on solutions to (2); their master theorem is inferred from this bound. Leighton [22] provides an alternative exposition of [5], and discussed the removal of the ceiling and floor functions. Roura [31, Theorem 2.3] provides a general multiterm master theorem; but in addition, he introduced a "continuous master recurrence" to treat recurrences arising from probabilistic analysis (see also [30, 29]). We will compare our results with these papers at the appropriate junctures.

We prove two main results: Theorem A in Section 4 solves the master recurrence for an infinite family of driving functions $d(n)$. It can be viewed as a natural completion of several known extensions of the standard master theorem. Thus Theorem A belongs to the first kind of generalization. Theorem B in Section 6 extends the master theorem to the multiterm master recurrence. Thus it belongs to the second kind of generalization. All our proofs use only elementary arguments.

The approach in this paper has two special emphases.

- Our first emphasis is on "real" recurrences: we regard (1) and (2) as *real recurrences* in the sense that n is a real variable, $T(n)$ a real function and all constants a, b, a_i, b_i are real numbers. The standard approach in the algorithms literature is to regard n as an integer variable. An example is the following multiterm recurrence taken from Kao [17]:

$$T(n) = \begin{cases} c \cdot n^\alpha \cdot \log^\beta n + \sum_{i=1}^{k} a_i T(\lceil b_i n \rceil) & \text{for } n \geq n_0 \\ c_n & \text{for } n < n_0 \end{cases} \tag{3}$$

where n and k are positive integers, c, c_n, a_i are positive constants, α, β are non-negative constants, $b_i \in (0,1)$ and $n_0 \geq \max_{i=1}^{k} \frac{1}{1-b_i}$. The integer viewpoint is also found in the cited work of Wang-Fu, Akra-Bazzi and Roura.

Although the original interpretation of $T(n)$ is only meaningful for integer values of n, we note that most driving functions are naturally real functions. E.g., $d(n) = \sqrt{n}$ and $d(n) = n \log n$. Hence our real extension remains well-defined if we simply omit the integer-valued functions such as ceilings or floors; e.g., in (3), we simply write "$T(b_i n)$" instead of "$T(\lceil b_i n \rceil)$". Since the recursive treatment of ceiling/floor is ugly, there is another standard approach, by domain restriction. Taking master recurrence (1) as example, we restrict the domain of $T(n)$ to only positive powers of b (e.g., [8, p. 145, Problem 4.44] or [33]). Of course, b is also restricted to be integer. Finally, to enlarge this restricted domain to all n, one use special arguments involving floors and ceilings (e.g., [9, pp. 81–84]) or makes smoothness assumptions on $T(n)$. But this restricted domain approach is somewhat problematic for multiterm master recurrences (2). We hope to show the simplicity and beauty of the real approach in this paper. Although the idea of real recurrences is nascent in several of the cited papers (e.g., [5, 33, 31]), it seldom takes on a full-blown form. In this paper, we develop basic tools to rigorously treat real recurrences.

- The second emphasis is to solve $T(n)$ up to Θ-order. This is conventional wisdom in the algorithmic literature (e.g., [2, 6]) because it yields more robust, implementation-independent conclusions about complexity. In fact, the above omission of ceiling or floor functions is justified on the ground that such actions preserve the Θ-order of solutions under very mild and usually automatic restrictions. Despite their Θ-order emphasis, many authors do not fully exploit the simplifications that it affords (cf. [33, 34, 31]). But up to Θ-order, solutions are insensitive to the initial conditions and simple variations in the recurrences – these are the "Θ-robustness" properties of the solution space. For this reason, we generally do not include initial conditions in our recurrences, but assume the following **default initial condition** (DIC):

$$T(n) = 1, \qquad (n \leq n_0) \tag{4}$$

for some real n_0, and the recurrence equation is assumed to be operative for $n > n_0$. For instance, using real recurrences under DIC, Kao's recurrence (3) would be simply written as

$$T(n) = n^{\alpha} \log^{\beta} n + \sum_{i=1}^{k} a_i T(b_i n)$$

Roura [31] and Leighton [22] also discuss robustness issues. Furthermore, the analysis tools in textbooks are much sharper than what is strictly needed for Θ-order analysis. For instance, a summation is often bounded via the exact Euler-Maclaurin formula (14) (Section 2). We seek to replace such calculus tools by cruder but elementary tools which suffice for Θ-order bounds. Here "elementary" means the avoidance of calculus or continuous methods like limits.

Our use of elementary methods has two advantanges: the pedagogical advantage of avoiding calculus is obvious, but the other methodological advantage is that our driving function $d(n)$ need not be differentiable or even continuous (weaker Lipschitz type bounds suffice). As another remark: we are not interested in generalizations that leaves the solution $T(n)$ as an integral or as an open sum (cf. [34]) The usefulness of the Master theorem is that it provides ready-to-use closed form solutions.

These two emphases explain the title of this paper. In summary, we believe the main advantage of our overall approach is simplicity and power.

¶1. On the Master Theorem. It is important to understand the "standard" master theorem as it provides the basic motif for generalizations. With respect to the master recurrence (1), let us define a **watershed constant**

$$\alpha := \log_b a \tag{5}$$

and an associated **watershed function** $w(n) = n^\alpha$. The watershed terminology derives from the fact that the **master theorem** says that the solution to the ma to the master recurrence has three cases, obtained by comparing $d(n)$ to the watershed function $w(n)$:

Proposition 1 (Master Theorem) *The solution to (1) has the form:*

$$T(n) = \begin{cases} \Theta(n^\alpha) & \text{if } \ d(n) = \mathcal{O}(w(n)n^{-\varepsilon}) \ \textit{for some } \varepsilon > 0 & \text{[CASE } (-)] \\ \Theta(n^\alpha \log n) & \text{if } \ d(n) = \Theta(w(n)) & \text{[CASE } (0)] \\ \Theta(d(n)) & \text{if } \ \text{“}d(n) = \Omega(w(n)n^{\varepsilon})\text{”} \ \textit{for some } \varepsilon > 0 & \text{[CASE } (+)]. \end{cases} \tag{6}$$

This is taken from [9, p. 73], except we now interpret $T(n)$ as the solution to a real recurrence (so we allow $a > 0$ instead of their $a \geq 1$). The three cases of this theorem correspond to, respectively, $d(n)$ being polynomially-slower [CASE $(-)$], being Θ-order of [CASE (0)], and being polynomially-faster [CASE $(+)$] than the watershed function. The condition for CASE $(+)$ is written in quotes in (6) because the original paper of Bentley et al [6, p. 39] defines the Ω-notation in an unusual way: "$d(n) = \Omega(w(n)n^\varepsilon)$" if there exists $K > 0$ such that for all $c > 1$,

$$d(n) \geq K \cdot c^{\alpha+\varepsilon} \cdot d(n/c) \text{ (ev. } n). \tag{7}$$

Here, the qualification "(ev. n)" reads as "eventually n", and asserts that the statement is true for sufficiently large n. Cormen et al [9] replaces this by the weaker **regularity condition**: there exists $C > 1$ such that

$$d(n) \geq C \cdot a \cdot d(n/b) \text{ (ev. } n). \tag{8}$$

Note that (7) implies (8). Assuming that $d(n) \geq 1$ for $n > 0$, (8) implies that for some $\varepsilon > 0$, $d(n) \geq Kw(n)n^\varepsilon$ (ev.), which is the standard definition of "$d(n) = \Omega(w(n)n^\varepsilon)$" (see definitions below).

By treating (1) as a real recurrence, we can verify the master theorem relatively easily as follows. By induction on $i = 0, 1, \ldots$, we see that

$$T(n) = a^{i+1}T\left(n/b^{i+1}\right) + \sum_{j=0}^{i} a^j \cdot d\left(\frac{n}{b^j}\right).$$

Assuming $T(n) = 0$ for $n < 1$ (this is justified below), and setting $m = \lceil \log_b n \rceil$, we obtain the open sum,

$$T(n) = \sum_{j=0}^{m} a^j d(n/b^j). \tag{9}$$

The 3 cases can now be verified by plugging in the corresponding bounds for $d(n)$.

It is well-known that the 3 cases of the standard master theorem are non-exhaustive. For instance, the case $d(n) = w(n)\log^\delta n$ ($\delta \neq 0$) is not covered. However, the master recurrence is amenable to two general transformation techniques called **domain transformation** (or change of variable) and **range transformation** (or summation factor). See, e.g., [8, pp. 130-137]. Using these transformation methods, we obtain a transformed version of (9), and as a result we can easily provide a solution for the case $d(n) = w(n)\log^\delta n$. This yields the following extension of Proposition 1:

Proposition 2 (Brassard-Bratley) *Let $T(n)$ be the solution to the master recurrence (1).*

$$T(n) = \begin{cases} \Theta(d(n)) & \text{if } \ d(n) \text{ satisfies the regularity condition (8)} & \text{[CASE } (+)]\ , \\ \Theta(d(n)\log n) & \text{if } \ d(n) = \Theta(n^\alpha \log^\delta n)) \ \textit{for some } \delta > -1 & \text{[CASE } (0)]\ , \\ \Theta(d(n)\log n \log\log n) & \text{if } \ d(n) = \Theta(n^\alpha \log^\delta n)) \ \textit{where } \delta = -1 & \text{[CASE } (1)]\ , \\ \Theta(n^\alpha) & \text{if } \ d(n) = \mathcal{O}(n^\alpha \log^\delta n) \ \textit{for some } \delta < -1 & \text{[CASE } (-)]\ . \end{cases}$$

REMARKS:
1. Proposition 2 generalizes the standard master theorem (6) because the original CASE $(-)$ is subsumed by the new CASE $(-)$, the original CASE (0) is subsumed by the new CASE (0), and CASE $(+)$ is unchanged (assuming we originally use the regularity condition (8)). But CASE (1) is new.
2. Proposition 2 is from [8, p. 145] (cf. [9, p.84, Ex.4.4-2]), slightly sharpened. In particular, we state CASE $(+)$ in terms of the regularity condition, thus allowing $d(n)$ to be arbitrarily complex. Because [8] focuses on integer recurrences, they assume that n has the form $n_0 b^i$ for integers $n_0 \ge 1$ and $b \ge 2$ (but $a > 0$ real).
3. Wang and Fu also has a version of this Proposition (see §4.3 and the last row of Table 1 in [34]). Roura [30] noted an abbreviated form of Proposition 2 but missed CASE (1). Theorem 1 of Verma [33] is also a form of Proposition 2, but his case 3 is weaker than our CASE (0) because he assumes $\delta \ge 0$.
4. What is the next step to take? Observe that Proposition 2 is silent when the driving functions are

$$d_0(n) = n^\alpha \log n \log\log n, \quad d_1(n) = n^\alpha (\log\log n)^r, \quad d_2(n) = n^\alpha \frac{(\log\log\log n)^s)}{\log n \log\log n} \tag{10}$$

for all $r \in \mathbb{R}$ and $s > -1$. Note that $d_0(n)$ appears in the recurrence for the Schönhage-Strassen multiplication algorithm [32] with $\alpha = 1$. Theorem A below will provide solutions for an infinite class of driving functions that are natural generalizations of such examples. The solutions corresponding the driving functions of (10) are

$$T_0(n) = \Theta(n^\alpha \cdot \log^2 n \log\log n), \quad T_1(n) = \Theta(n^\alpha \cdot \log n (\log\log n)^r), \quad T_2(n) = \Theta(n^\alpha (\log\log\log n)^{s+1}).$$

But another generalization of Proposition 2 is to prove its multiterm analogue; this is our Theorem B.

¶2. Notations for Partial Functions. We recall some basic concepts of complexity analysis. We take special care in discussing partial functions since they are usually not systematically treated (cf. [36]). Let $f : \mathbb{R} \to \mathbb{R}$ be a partial real function. If f is undefined at $x \in \mathbb{R}$, we write $f(x)\uparrow$; otherwise write $f(x)\downarrow$. Let $\mathtt{domain}(f) := \{x \in \mathbb{R} : f(x)\downarrow\}$ and $\mathtt{range}(f) := \{f(x) : x \in \mathbb{R}, f(x)\downarrow\}$ denote the **proper domain** and **proper range** of f. We may call $\mathbb{R}$ the **nominal domain** and **nominal range** of f. A **real predicate** is a partial function $P : \mathbb{R}^k \to \{0,1\}$. We say the predicate P **holds at $\boldsymbol{x} \in \mathbb{R}^k$**, written "$P(\boldsymbol{x})$ holds", if $P(\boldsymbol{x})\uparrow$ or $P(\boldsymbol{x}) = 1$. Thus, there are two possibilities when P holds at $\boldsymbol{x}$: (1) Either $P(\boldsymbol{x})\uparrow$, in which case we say $P(\boldsymbol{x})$ **holds vacuously** (2) Or $P(\boldsymbol{x}) = 1$, in which case we say P is **valid** at $\boldsymbol{x}$. We say P is **eventually valid**, if there is some c such that $P(\boldsymbol{x}) = P(x_1, \ldots, x_k) = 1$ for all $x_i > c$. Thus "validity" always implies the underlying functions in the definition of P are defined. For example, we say $f(x)$ is **eventually increasing** if $x < y$ implies $f(x) < f(y)$ is eventually valid. In particular, this implies that f is eventually defined.

Examples. Our real predicates $P(x)$ are typically constructed from partial functions f. E.g., "$f(x) \ge 0$" represents the predicate P_f where $P_f(x)\uparrow$ if $f(x)\uparrow$, otherwise $P_f(x)\downarrow$ and $P_f(x) = 1$ iff $f(x) \ge 0$. This generalizes in the obvious way to the multivariate setting and possibly involving several partial functions. E.g., $P(x)$ is "$f_0(x) \le f_1(x)$" or $P(x,y)$ might be "$f_0(x,y) = f_1(x) + f_2(y)$". Thus, the predicate "$\sqrt{x} \ge 0$" is valid since $\sqrt{x}$ is non-negative whenever defined. But the predicate "$\log x \ge 0$" is (only) eventually valid.

A **complexity function** is a partial real function f such that $f(x)\downarrow$ (ev.). The running time $T(n)$ of an algorithm is often defined only for $n \in \mathbb{N}$. We turn T into a complexity function f by defining $f(x) := \sup\{T(n) : n \le x,\ T(n)\downarrow\}$; thus, $f(x)\uparrow$ iff for all $n \le x$, $T(n)\uparrow$.

For complexity functions f and g, say f **dominates** g, written $f \succeq g$ if there is some $K > 0$ such that $f(x) \ge Kg(x)$ is eventually valid. We write $g = \mathcal{O}(f)$ (big-Oh) if $g : f \succeq g \succeq 0$. Here, we follow [9] in requiring $g \succeq 0$ (i.e., g is eventually non-negative). Similarly, $g = \Omega(f)$ (big-Omega) if $g \succeq f$ and $g \succeq 0$. and $g = \Theta(f)$ (big-Theta) if $g = \mathcal{O}(f)$ and $f = \mathcal{O}(g)$. The notation $f = o(g)$ (small-oh), is usually defined using limits. In the spirit of avoiding calculus, we can define it as follows: $f = o(g)$ if for all $C > 0$, $0 \le f \le C \cdot g$ (ev.).

Let $\log x$ denote the real-valued logarithm function, to some unspecified base $b > 1$. Thus, we have $\log x \uparrow$ for $x \le 0$. Let ln and lg denote the logarithm to the natural base $b = e$ and base $b = 2$, respectively. We mostly use lg because it suits our elementary arguments (this is the "computer science logarithm"). The **iterated logarithm function** (to base 2) is defined as

$$\ell\ell g_k(x) := \underbrace{\lg(\lg(\cdots(\lg(x))\cdots))}_{k \text{ times}}$$

for $k \in \mathbb{N}$. E.g., $\ell\ell g_0(x) = x$, $\ell\ell g_1(x) = \lg x$, $\ell\ell g_2(x) = \lg\lg x$. We may extend the index k to range over all the integers: for $k \in \mathbb{N}$, define $\ell\ell g_{-(k+1)}(x) := 2^{\ell\ell g_{-k}(x)}$. Thus $\ell\ell g_{-1}(x) = 2^x$ and $\ell\ell g_{-2}(x) = 2^{2^x}$. When we compose these functions, the following is valid: $\ell\ell g_m(\ell\ell g_n(x)) = \ell\ell g_{m+n}(x)$. Note that the left-hand side may be undefined even when the right-hand side is defined; but by our definition of predicates constructed from partial functions, this counts as satisfying the predicate (vacuously).

2 Elementary Summation Techniques

¶3. Summing Values of Real Functions. Given a partial real function $f : \mathbb{R} \to \mathbb{R}$, we consider two kinds of discrete summation on values of f between arbitrary real limits $a, b \in \mathbb{R}$:

$$\left.\begin{array}{lcll} \sum_{x\ge a}^{b} f(x) & := & f(b) + f(b-1) + f(b-2) + \cdots + f(b - \lfloor b-a \rfloor), & \text{(descending)} \\ \sum_{x=a}^{b} f(x) & := & f(a) + f(a+1) + f(a+2) + \cdots + f(a + \lfloor b-a \rfloor) & \text{(ascending)} \end{array}\right\} \tag{11}$$

Both sums are by definition equal to 0 when $b < a$. If $b \ge a$, the descending sum will include the term $f(b)$ while the ascending sum will include $f(a)$. But neither will not include both $f(a)$ and $f(b)$ unless $b - a \in \mathbb{N}$. The distinction between ascending and descending sums is indicated by way we write their lower limits: "$\sum_{x\ge a}$" versus "$\sum_{x=a}$". For instance, $\sum_{x\ge 1}^{\pi} f(x) = f(\pi) + f(\pi - 1) + f(\pi - 2)$ while $\sum_{x=1}^{\pi} f(x) = f(1) + f(2) + f(3)$ where $\pi = 3.14159\ldots$. Another convention concerning summation over partial functions is this: if a summand is undefined, it is replaced by 0. Thus, the sums $\sum_{x\ge a}^{b} f(x)$ and $\sum_{x=a}^{b} f(x)$ are always defined. E.g., $\sum_{x\ge 1}^{4} \lg\lg\lg\lg(x) = 0$ since $\lg\lg\lg\lg(x) \uparrow$ for $x \le 4$. The following identity connects these 2 summation conventions:

$$\sum_{x\ge 0}^{b} f(x) \equiv \sum_{x=0}^{b} f(b - x). \tag{12}$$

These conventions afford rigorous tools for manipulation of real valued discrete sums in the algorithmic literature.

Our main focus will be descending sums of the form

$$S_f(n) := \sum_{x\ge 1}^{n} f(x) \tag{13}$$

for real values of n. Traditionally, the sum $S_f(n)$ (for $n \in \mathbb{N}$) is bounded using the Euler-Maclaurin summation formula,

$$\sum_{i=1}^{n-1} f(i) = \int_i^n f(x)dx + \left(\sum_{i=1}^{\infty} \frac{B_i f^{(i-1)}(x)}{i!} \right)_{x=1}^{x=n} \tag{14}$$

where B_i is the ith Bernoulli number (e.g., [13, p. 217]). But we seek cruder but easier-to-use elementary methods which can determine $S_f(n)$ up to Θ-order.

¶4. Summation Rules. We classify complexity functions f by their growth properties:

- We say f is **polynomial-type** if $f \geq 0$, f is non-decreasing, and for some $K > 0$,

$$f(x) \leq K f(x/2) \text{ (ev.).} \tag{15}$$

  E.g., $f(x) = 1$ is polynomial-type. So are the following functions:

$$f_0(x) = x, \quad f_1(x) = \log \max\{1, x\}, \quad f_2(x) = f_0(x) f_1(x), \quad f_3(x) = (f_0(x))^a \ (a > 0).$$

  The requirement "$f \geq 0$" says that, whenever $f(x) \downarrow$, it must be non-negative. Hence the partial function $\log x$ is not polynomial-type; its modified version $f_1(x)$ is polynomial-type.

- We say f is **increasing exponential-type** if $f > 0$ and there are real numbers $C > 1$ and $k > 0$ such that

$$f(x) \geq C \cdot f(x - k) \text{ (ev.).} \tag{16}$$

  E.g., the following functions are increasing exponential-type.

$$g_0(x) = 2^x, \quad g_1(x) = x!, \quad g_2(x) = g_0(x) g_1(x), \quad g_3(x) = 2^{g_0(x)}$$

- We say f is **decreasing exponential-type** if $f > 0$ and there are real numbers $0 < c < 1$ and $k > 0$ such that

$$f(x) \leq c \cdot f(x - k) \text{ (ev.).} \tag{17}$$

  E.g., the following functions are decreasing exponential-type.

$$h_0(x) = 2^{-x}, \quad h_1(x) = x^{-x}, \quad h_2(x) = h_0(x) h_1(x), \quad h_3(x) = 2^{h_0(x)}$$

We say f is **exponential-type** if it is an increasing or decreasing exponential-type. The following is immediate from the above definition of the growth types:

- If f is polynomial-type then $f(x) = \mathcal{O}(x^{\lg K})$.
- If increasing exponential-type then $f(x) = \Omega(C^{x/k})$.
- If decreasing exponential-type then $f(x) = \mathcal{O}(c^{x/k})$.

We say the discrete sum $S_f(n)$ given by (13) is **polynomial-type** or **exponential-type**, according to the above classification of f. We may also say "increases/decreases exponentially" to describe the two exponential-types.

Theorem 3 (Summation Rules)
(i) If f is polynomial-type, then $S_f(n) = \Theta(n f(n))$.
(ii) If f is exponential-type,

$$S_f(n) = \begin{cases} \Theta(f(n)) & \text{if } f \text{ increases exponentially,} \\ \Theta(1) & \text{if } f \text{ decreases exponentially.} \end{cases}$$

Proof. (i) For a polynomial-type sum, using the fact that f is non-decreasing, we get the upper bound $S_f(n) \leq \sum_{x \geq 1}^{n} f(n) = n f(n)$. For lower bound, we also use the fact that $f(x) \leq C f(x/2)$ (ev.) for some

$C > 0$. Eventually, we have

$$\begin{aligned} S_f(n) &\geq \sum_{x \geq n/2}^{n} f(x) \\ &\geq \sum_{x \geq n/2}^{n} f(n/2) \geq \lfloor n/2 \rfloor f(n/2) \\ &\geq \lfloor n/2 \rfloor \frac{f(n)}{C} = \Omega(nf(n)). \end{aligned}$$

(iia) For an increasing exponential sum, there are constants $C > 1$, $k > 0$ and $n_0 > 0$ such that for all $n \geq n_0 + k$, we have $f(n) \downarrow$ and $f(n) \geq Cf(n-k)$. By increasing n_0 and k if necessary, we may assume k is an integer. Next, for each $n \geq n_0 + 2k$, we may pick $n_1 \in [n_0 + k, n_0 + 2k)$ so that $n - n_1$ is divisible by k. Note that although n_1 depends on n, it is at most $n_0 + 2k$. Thus,

$$S_f(n) = f(n) + f(n-1) + \cdots + f(n_1 + 1) + S_f(n_1)$$

We subdivide the sum $f(n) + f(n-1) + f(n-2) + \cdots + f(n_1 + 1)$ into k different sub-sums, each of the form

$$T_\kappa := f(n-\kappa) + f(n-\kappa-k) + f(n-\kappa-2k) + \cdots + f(n_1 - \kappa + k) \tag{18}$$

for some $\kappa = 0, 1, \ldots, k-1$. Each sub-sum T_κ satisfies

$$\begin{aligned} T_\kappa &\leq f(n-\kappa)\left[1 + \frac{1}{C} + \frac{1}{C^2} + \cdots\right] \\ &= f(n-\kappa)\frac{C}{C-1} \\ &= \mathcal{O}(f(n-\kappa)) = \mathcal{O}(f(n)). \end{aligned}$$

Thus $S_f(n) = S_f(n_1) + \sum_{\kappa=0}^{k-1} T_\kappa = \mathcal{O}(f(n))$, since $n_1 \leq n_0 + 2k$. The lower bound $S_f(n) \geq f(n)$ is immediate.

(iib) For a decreasing exponential sum, there is some $c < 1$, $k > 0$ and $n_0 > 0$ such that for all $n \geq n_0$, we have $f(n) \downarrow$ and $f(n+k) \leq cf(n)$ and $f(n_0) \geq f(n)$. Again we may choose k and n_0 such that $k \in \mathbb{N}$. Next, for each $n \geq n_0 + 1$, we may choose $n_1 \in [n_0, n_0 + 1)$ such that $n - n_1 \in \mathbb{N}$ (again, n_1 depends on n). Then,

$$S_f(n) \leq S_f(n_1) + f(n_1 + 1) + f(n_1 + 2) + \cdots = S_f(n_1) + \sum_{i=1}^{\infty} f(n_1 + i).$$

The sum $\sum_{i=0}^{\infty} f(n_1 + i)$ can be broken up into k sub-sums, as in (18). Each sub-sum is bounded as

$$\begin{aligned} & f(n_1 - \kappa + k) + f(n_1 - \kappa + 2k) + f(n_1 - \kappa + 3k) + \cdots \\ \leq\ & f(n_1 - \kappa + k)\left[1 + c + c^2 + \cdots\right] \\ \leq\ & f(n_1)\frac{1}{1-c} = \mathcal{O}(1). \end{aligned}$$

Thus $S_f(n) = \mathcal{O}(1)$. But $S_f(n) \geq f(n_1) > 0$ and hence $S_f(n) = \Theta(1)$. **Q.E.D.**

Our polynomial-type functions is just Verma's "slowly growing functions" [33]. But we do not have his "slowly diminishing functions" because their sums do not seem to allow any simple Θ-order rule as in Theorem 3. For instance, $f(x) = x, f(x) = 1, f(x) = 1/x, f(x) = 1/x^2, f(x) = 1$ are all slowly diminishing, but $S_f(n)$ is $\Theta(n^2), \Theta(n), \Theta(\log n), \Theta(1)$. Our theorem is elementary and easy to use: once we have determined the type of the sum, we can mechanically write down its Θ-order. This is illustrated in the following:

- Polynomial sums:

$$\sum_{x\ge 1}^{n} x^k = \Theta(n^{k+1}) \text{ for } \; k>0, \qquad \sum_{x\ge 1}^{n} \log^2 x = \Theta(n\log^2 n), \qquad \sum_{x\ge 1}^{n} x^2/\log x = \Theta(n^3/\log n). \tag{19}$$

- Increasing Exponential Sums.

$$\sum_{x\ge 1}^{n} 2^x = \Theta(2^n), \qquad \sum_{x\ge 1}^{n} x^{-5}2^{2^x} = \Theta(n^{-5}2^{2^n}), \qquad \sum_{x\ge 1}^{n} x! = \Theta(n!) \quad . \tag{20}$$

- Decreasing Exponential Sums.

$$\sum_{x\ge 1}^{n} 2^{-x} = \Theta(1), \qquad \sum_{x\ge 1}^{n} x^2 x^{-x} = \Theta(1), \qquad \sum_{x\ge 1}^{n} x^{-x} = \Theta(1) \quad . \tag{21}$$

¶5. Closure Properties. The usefulness of the summation rules comes from shifting the burden of estimating sums $S_f(n)$ to verifying the growth type of f. To determine growth types, we exploit the fact that many functions are constructed out of simpler functions, and the constructions preserve or systematically transform growth types. Here are some common construction methods:

Lemma 4 *Let $a \in \mathbb{R}$.*
(i) Polynomial-type functions f, g are closed under addition, multiplication, raising to any positive power $a > 0$, summation $S_f(n)$, and function composition $f \circ g$.
(ii) Exponential-type functions g are closed under addition, multiplication, raising to any power a. If g is increasing exponential-type, it is closed under summation $S_g(n)$. In case $a > 0$, the function g^a will not change its subtype (increasing or decreasing). In case $a < 0$, the function g^a will change its subtype.
(iii) If f is polynomial-type then the composition $f \circ \log$ is also polynomial-type. If, in addition we have $f > 1$ (ev.), then $\log \circ f$ is also polynomial-type. If g is exponential-type and $a > 1$ then so is a^g.

Proof. All the inequalities in the following derivations are assumed to be eventually valid. In brief:
(i) Assume $f(n) \le Kf(n/2)$ and $g(n) \le Kg(n/2)$ for some $K > 1$. Then $f(n)+g(n) \le K(f(n/2)+g(n/2))$, $f(n)g(n) \le K^2 f(n/2)g(n/2)$, and for any $a > 0$, $f(n)^a \le K^a f(n/2)^a$. For summation, we have

$$S_f(n) = \sum_{x\ge 1}^{n} f(x) \le \sum_{x\ge 1}^{n} Kf(x/2) \le \sum_{x\ge 1/2}^{n/2} K[f(x-\tfrac{1}{2}) + f(x)] \le \sum_{x\ge 1/2}^{n/2} 2Kf(x) = \mathcal{O}(S_f(n/2)).$$

For function composition, we may assume K is a power of 2:

$$f(g(n)) \le f(K\cdot g(n/2)) \le K \cdot f(K/2 \cdot g(n/2)) \le \cdots \le K^{\lg K} \cdot f(g(n/2)).$$

(ii) For $i = 0, 1$, assume $g_i(n) \ge Cg_i(n-k)$ and $h_i(n) \le ch_i(n-k)$ for some $C > 1, c < 1$. Also, write $g = g_0$, $h = h_0$. Closure under addition: $g_0(n)+g_1(n) \ge C(g_0(n-k)+g_1(n-k))$ and $h_0(n)+h_1(n) \le c(h_0(n-k)+h_1(n-k))$. Closure under product: $g_0(n)g_1(n) \ge C^2 g_0(n-k)g_1(n-k))$ and $h_0(n)h_1(n) \le c^2 h_0(n-k)h_1(n-k)$. Closure under raising to power a: If $a > 0$, then $g^a(n) \ge C^a g^a(n-k)$ and $h^a(n) \le c^a h^a(n-k)$ where $C^a > 1$ and $c^a < 1$. If $a < 0$, then $g^a(n) \le C^a g^a(n-k)$ and $h^a(n) \ge c^a h^a(n-k)$ where $C^a < 1$ and $c^a > 1$. For summation of increasing exponential-types, we have

$$S_g(n) = \sum_{x\ge 1}^{n} g(x) \ge \sum_{x\ge 1}^{n} C\cdot g(x-k) \ge C \sum_{x\ge 1-k}^{n-k} g(x) \ge CS_g(n-k).$$

(iii) Assume $f(n) \le Kf(n/2)$ (ev.). Then $f(\log(n)) \le Kf(\log(n)/2) \le Kf(\log(n/2))$, showing $f \circ \log$ is polynomial-type. If $f > 1$ (ev.), then using the fact that f is non-decreasing, we have $\log(f(n/2)) \ge \varepsilon$ (ev.) for some $\varepsilon > 0$. So $\log(f(n)) \le (\log K) + \log(f(n/2)) \le (1 + (\log K)/\varepsilon)\log(f(n/2))$. Thus proves $\log \circ f$ is polynomial-type. **Q.E.D.**

Although the growth type of f is usually easy to determine, there are exceptions. Moreover, the following example shows that our two growth types are not exhaustive:

Lemma 5 *The function $f(x) = x^{\ln x}$ is not polynomial-type and not exponential-type.*

Proof. It is easily seen that $f(x)$ is not polynomial-type. By way of contradiction, suppose $f(x)$ is exponential-type. It must be increasing exponentially, so there exists $C_0 > 1$ and $k > 0$ such that

$$f(x) \ge C_0 f(x-k) \text{ (ev.)}. \tag{22}$$

We will use the elementary estimate

$$\ln x \le \ln(x-k) + (2k/x) \text{ (ev.)}. \tag{23}$$

Eventually, we have

$$\begin{array}{rcll} f(x) & = & \left[(x-k)(1+\frac{k}{x-k})\right]^{\ln x} & \\ & \le & (x-k)^{\ln(x-k)+(2k/x)}(1+\frac{k}{x-k})^{\ln x} & \text{(by (23))} \\ & = & f(x-k)\cdot(x-k)^{2k/x}\cdot e^{\ln(1+\frac{k}{x-k})\ln x} & \\ & \le & f(x-k)\cdot e^{2k\ln(x-k)/x}\cdot e^{\frac{k\ln x}{x-k}} & \text{(since } \ln(1+y) < y \text{ for } |y|<1) \\ & = & f(x-k)\cdot C_1(x) & \end{array}$$

where $C_1(x) := e^{2k\ln(x-k)/x} \cdot e^{\frac{k\ln x}{x-k}}$. Since $\ln C_1(x) = (2k\ln(x-k)/x) + (k\ln x/(x-k)) \to 0$ as $x \to \infty$, we conclude that $C_1(x) < C_0$ (ev.). This show $f(x) < f(x-k)C_0$ (ev.), contradicting (22). **Q.E.D.**

3 Summation in Elementary Terms

¶6. Exponential-Logarithmic Functions. To generalize Proposition 2, we now introduce a family of partial real functions, based on iterated logarithms or exponentials. For $a \in \mathbb{R}$, the **a-th power** of $\ell\ell g_k(x)$ is denoted $\ell\ell g_k^a(x) := (\ell\ell g_k(x))^a$. An **exponent sequence** is any finite sequence $\boldsymbol{e} = (e_0, e_1, \ldots, e_\ell)$, $\ell \ge 0$, of real numbers. We say $\boldsymbol{e}$ is **normalized** if either $\ell = 0$ or $e_0 e_\ell \ne 0$. Given $k \in \mathbb{Z}$ and $\boldsymbol{e} = (e_0, \ldots, e_\ell)$, we define[1] an **elementary term** to be an expression of the form

$$\mathrm{EL}_k^{\boldsymbol{e}}(x) := \prod_{i=0}^{\ell} \ell\ell g_{k+i}^{e_i}(x). \tag{24}$$

E.g., $\mathrm{EL}_0^{(-2,0,\pi)}(x) = x^{-2}(\lg\lg(x))^\pi$ and $\mathrm{EL}_1^{(-2,0,\pi)}(x) = \lg^{-2}(x)(\lg\lg\lg(x))^\pi$.

Each elementary term represents a partial real function, called an **EL-function**. The **trivial EL-function** is $\mathrm{EL}_k^{(0)}(x) = 1$ for all x and k. Conversely, each EL-function $f(x)$ can be represented by an elementary term (24). If $f(x)$ is non-trivial, the elementary term (24) is unique when $\boldsymbol{e}$ is normalized. The

[1]Here, "elementary" refers to functions composed from logarithms and exponentials, not in the sense of avoiding calculus. EL is mnemonic for both *EL*ementary and *E*xp-*L*og.

order of f is k, and its **exponent** is $\boldsymbol{e}$ where $\mathrm{EL}_k^{\boldsymbol{e}}(x)$ is the unique elementary term for f. If f is trivial, we define its order to be 0 and its exponent to be (0). Denote the order and exponent of f by $Ord(f)$ and $Exp(f)$, respectively.

The set of EL-functions is totally ordered by the **lexicographical ordering** $<_L$ of their exponents and order. Given an order k and exponent $\boldsymbol{e} = (e_0, \ldots, e_\ell)$, let the ($k$-shifted) **augmented exponent** $\sigma_k(\boldsymbol{e})$ be the doubly infinite sequence $\sigma_k(\boldsymbol{e}) = (\ldots, d_{-1}; d_0, d_1, d_2, \ldots)$ where $d_{i+k} = e_i$ for $i = 0, \ldots, \ell$, and $d_{i+k} = 0$ for $i > \ell$ or $i < 0$. We mark the "origin" d_0 of the doubly infinite sequence with a semicolon, $(\ldots, d_{-1}; d_0, \ldots)$, viewing $\sigma_k(\boldsymbol{e})$ as the pair of infinite sequences, $(d_0, d_1, d_2, \ldots)$ and $(d_{-1}, d_{-2}, \ldots)$. If $\boldsymbol{e}$ is not all zeros, and if i is is the smallest index such that $e_i \neq 0$, we call e_i the **leading power** in $\boldsymbol{e}$. Say $\boldsymbol{e}$ is **negative** (resp., **positive**) if its leading power is negative (resp., positive). Clearly, $\sigma_k(\boldsymbol{e})$ is negative or positive according as $\boldsymbol{e}$ is negative or positive, independent of k. Given two order-exponent pairs $(k, \boldsymbol{d})$ and $(\ell, \boldsymbol{e})$, we order them as

$$(k, \boldsymbol{d}) <_L (\ell, \boldsymbol{d})$$

if the component-wise difference $\sigma_k(\boldsymbol{d}) - \sigma_\ell(\boldsymbol{e})$ is negative. We write $f <_L g$ iff $(Ord(f), Exp(f)) <_L (Ord(g), Exp(g))$. Also write $f \leq_L g$ for $f = g$ or $f <_L g$. The following ordering of EL-functions is well-known:

Lemma 6 *Let f, g be EL-functions.*
(i) f is a complexity function. Moreover, $f \geq 0$ and is monotone increasing or monotone decreasing.
(ii) If $Exp(f)$ is positive, then f is strictly monotone increasing (ev.)*, and if $Exp(f)$ is negative, f is strictly monotone decreasing* (ev.)*.*
(iii) $f \prec g$ iff $(Ord(f), Exp(f)) <_L (Ord(g), Exp(g))$
(iv) If $Ord(f) \neq Ord(g)$ then $f \prec g$ iff $f = o(g)$
(v) An EL-function is polynomial-type if its order is non-negative; otherwise, it is exponential-type.
(vi) If f is exponential-type, then it is increasing exponentially if its leading power is positive, and it is decreasing exponentially if its leading power is negative.

We omit the simple proofs.

Lemma 6(iii) implies that the EL-functions are totally ordered under the domination relation; this is a special case of the total ordering of algebraico-logarithmic functions of Hardy [16]. From Lemma 6(v) and Lemma 5, we conclude[2] that $f(x) = x^{\lg x}$ is not an EL-function. Nevertheless, $\lg f(x)$ is an EL-function. The following transformation of EL-functions is immediate from the definitions: $\mathrm{EL}_k^{\boldsymbol{e}}(2^x) = \mathrm{EL}_{k-1}^{\boldsymbol{e}}(x)$ and $\mathrm{EL}_k^{\boldsymbol{e}}(\lg x) = \mathrm{EL}_{k+1}^{\boldsymbol{e}}(x)$. More generally, we have

$$\mathrm{EL}_k^{\boldsymbol{e}}(\ell\ell g_\ell(x)) = \mathrm{EL}_{k+\ell}^{\boldsymbol{e}}(x). \tag{25}$$

¶7. Error Notation.

In the following proofs, we use a convenient notation for additive error terms from from [36]: in any numerical expression, each occurrence of the symbol "$\pm$" is replaced by "$+\theta$" where θ is an anonymous variable satisfying $|\theta| \leq 1$. Thus, we write $x = y \pm \epsilon$ to mean that $x = y + \theta\epsilon$ for some θ where $|\theta| \leq 1$. This is equivalent to saying $|x - y| \leq |\epsilon|$. Each occurrence of $\pm$ introduces a new θ variable. Like the big-Oh notation, we view it as a variable hiding device.

For instance, consider the summation $S = \sum_{i=1}^n f(n \pm c)$ for some constant c. The ith summand $f(n \pm c)$ in S may depend on i, but it is hidden by $\pm$ convention (i.e., $S = \sum_{i=1}^n f(n + \theta_i c)$ where $|\theta_i| \leq 1$). Then we

[2]Of course, the expression $x^{\ln x}$ is not an EL-expression, but it might be possible to express the same function using some other EL-expression.

have

$$S = nf(n \pm c). \tag{26}$$

This follows easily from the fact if f is continuous then

$$a \cdot f(x) + b \cdot f(y) = (a+b) f(x \pm (y-x)) \tag{27}$$

whenever a, b are non-negative. If f is monotone (as in our applications), there is a unique solution for $x \pm (y-x)$ in (27). For $k \geq -1$, we also have

$$\mathrm{EL}_k^{\boldsymbol{e}}(n) = \Theta(\mathrm{EL}_k^{\boldsymbol{e}}(n \pm c)) \tag{28}$$

for any $c \in \mathbb{R}$. For $k \geq 0$, this follows from the fact that $f(n) = \mathcal{O}(f(n \pm c))$ if f is polynomial-type. This bound is also valid for $k = -1$ since $(2^n)^e = \Theta((2^{n \pm c})^e)$.

¶8. Elementary Sums. Call $S_f(n)$ an **elementary sum** when f is an EL-function. If $f(x) = \mathrm{EL}_k^{\boldsymbol{e}}(x)$, then we denote the elementary sum by

$$S_k^{\boldsymbol{e}}(n) := \sum_{x \geq 1}^{n} \mathrm{EL}_k^{\boldsymbol{e}}(x). \tag{29}$$

For any exponent sequence $\boldsymbol{e} = (e_0, e_1, \ldots, e_\ell)$, we use the convenient notation,

$$\boldsymbol{e} + 1 := (1 + e_0, e_1, \ldots, e_\ell). \tag{30}$$

The following transformation of elementary sums is the key to our elementary proofs:

Lemma 7 *Assume $k \geq 0$ and $\boldsymbol{e} = (e_0, e_1, \ldots, e_\ell)$. For all real n,*

$$S_k^{\boldsymbol{e}}(n) = \Theta(S_{k-1}^{\boldsymbol{e}+1}(\lg n)).$$

Proof. Let $\boldsymbol{e}' = (e_1, \ldots, e_\ell)$ with the convention that $\boldsymbol{e}' = (0)$ if $\ell = 0$. Initially, assume $n+1$ is a power of 2:

$$\begin{array}{rcll}
S_k^{\boldsymbol{e}}(n) = \sum_{x \geq 1}^{n} \mathrm{EL}_k^{\boldsymbol{e}}(x) & = & \sum_{p=1}^{\lg(n+1)} \sum_{x=2^{p-1}}^{2^p-1} \mathrm{EL}_k^{\boldsymbol{e}}(x) & \text{(since } n+1 \text{ is a power of 2)} \\
& = & \sum_{p=1}^{\lg(n+1)} \sum_{x=2^{p-1}}^{2^p-1} \mathrm{EL}_k^{\boldsymbol{e}}(2^{p \pm 1}) & \\
& = & \sum_{p=1}^{\lg(n+1)} 2^{p-1} \mathrm{EL}_k^{\boldsymbol{e}}(2^{p \pm 1}) & \text{(by (26))} \\
& = & \sum_{p=1}^{\lg(n+1)} 2^{p-1} 2^{(p \pm 1)e_0} \mathrm{EL}_{k+1}^{\boldsymbol{e}'}(2^{p \pm 1}) & \\
& = & \Theta\left(\sum_{p=1}^{\lg(n+1)} 2^{(p \pm 1)(1+e_0)} \mathrm{EL}_{k+1}^{\boldsymbol{e}'}(2^p) \right) & \text{(by (28), } \mathrm{EL}_{k+1}^{\boldsymbol{e}'}(2^{p \pm 1}) = \Theta(\mathrm{EL}_k^{\boldsymbol{e}'}(p))) \\
& = & \Theta\left(\sum_{p=1}^{\lg(n+1)} \mathrm{EL}_k^{\boldsymbol{e}+1}(2^{p \pm 1}) \right) & \text{(by definition of } \boldsymbol{e}+1) \\
& = & \Theta\left(\sum_{p=1}^{\lg(n+1)} \mathrm{EL}_{k-1}^{\boldsymbol{e}+1}(p \pm 1) \right) & \text{(by (25))} \\
& = & \Theta\left(\sum_{p=1}^{\lg(n+1)} \mathrm{EL}_{k-1}^{\boldsymbol{e}+1}(p) \right) & \text{(by (28))} \\
& = & \Theta(S_{k-1}^{\boldsymbol{e}+1}(\lg(n+1))) & \text{(by definition of } S_{k=1}^{\boldsymbol{e}+1})
\end{array}$$

Thus we have shown that

$$S_k^{\boldsymbol{e}}(n) = \Theta(S_{k-1}^{\boldsymbol{e}+1}(\lg(n+1))), \tag{31}$$

when $n+1$ is a power of 2. Next let $n \geq 3$ be arbitrary. So there is some n' such that $(n-1)/2 \leq n' < n$ where $n'+1$ is the power of two. Thus $2n'+2$ is also a power of 2. Hence

$$S_k^{\boldsymbol{e}}(n') \leq S_k^{\boldsymbol{e}}(n) \leq S_k^{\boldsymbol{e}}(2n'+1).$$

From (31), we have

$$\begin{aligned} S_k^{\boldsymbol{e}}(2n'+1) &= \Theta(S_{k-1}^{\boldsymbol{e}+1}(\lg(2n'+2))) && \text{(by (31))} \\ &= \Theta(S_{k-1}^{\boldsymbol{e}+1}(\lg(n'+1)\pm 1)) \\ &= \Theta(S_{k-1}^{\boldsymbol{e}+1}(\lg(n'+1))) \\ &= \Theta(S_k^{\boldsymbol{e}}(n')) && \text{(by (31))}. \end{aligned}$$

This shows that $S_k^{\boldsymbol{e}}(n) = \Theta(S_k^{\boldsymbol{e}}(2n'+1))$. Moreover,

$$\begin{aligned} S_k^{\boldsymbol{e}}(n) &= \Theta(S_k^{\boldsymbol{e}}(2n'+1)) \\ &= \Theta(S_{k-1}^{\boldsymbol{e}+1}(\lg(2n'+2))) \\ &= \Omega(S_{k-1}^{\boldsymbol{e}+1}(\lg n)). \qquad (32) \\ S_k^{\boldsymbol{e}}(n) &= \Theta(S_k^{\boldsymbol{e}}(n')) \\ &= \Theta(S_{k-1}^{\boldsymbol{e}+1}(\lg(n'+1))) \\ &= \mathcal{O}(S_{k-1}^{\boldsymbol{e}+1}(\lg n)). \qquad (33) \end{aligned}$$

From (32),(33), we see that (31) is valid for all n. **Q.E.D.**

The next theorem shows that, up to Θ-order, elementary functions are closed under summation. In other words, for any order k and exponent sequence $\boldsymbol{e}$, there is an elementary function f such that $S_k^{\boldsymbol{e}}(n) = \Theta(f)$. To explicitly describe f, we make two preparatory steps:

- An elementary term $\mathrm{EL}_k^{\boldsymbol{e}}(n)$ is said to be in **reduced form** if k and $\boldsymbol{e} = (e_0, \ldots, e_\ell)$ satisfy

$$k \le 0 \quad \text{and} \quad (k = 0 \text{ or } \; e_0 \ne 0). \qquad (34)$$

  Note that every elementary function can be represented by a reduced form by repeated application of the transformation

$$\mathrm{EL}_k^{\boldsymbol{e}}(n) \leftrightarrow \mathrm{EL}_{k-1}^{(0,\boldsymbol{e})}(n)$$

  where $(0, \boldsymbol{e}) = (0, e_0, \ldots, e_\ell)$. This transform does not change the value of the function, and can be applied either direction. If $k > 0$, we use the forward direction to replace k by $k-1$. If $k < 0$ and the exponent sequence has the form $(0, \boldsymbol{e})$, we use the backward direction to replace $(0, \boldsymbol{e})$ with $\boldsymbol{e}$. The elementary term is reduced iff neither transformation is applicable. Termination of this process is clear. Also, an elementary sum $S_k^{\boldsymbol{e}}(n)$ is in **reduced form** if k and $\boldsymbol{e}$ satisfy (34).

- Given $\boldsymbol{e} = (e_0, \ldots, e_\ell)$, let h be the largest index among $\{0, 1, \ldots, \ell+1\}$ such that $e_0 = e_1 = \cdots = e_{h-1} = -1$ and either $h = \ell + 1$ or $e_h \ne -1$. We may extend $\boldsymbol{e}$ to $(e_0, \ldots, e_\ell, 0) = (e_0, \ldots, e_\ell, e_{\ell+1})$ where $e_{\ell+1} = 0$. Then we define

$$\boldsymbol{e}^* := \begin{cases} (\underbrace{0, 0, \ldots, 0}_{h \text{ times}}, 1 + e_h, e_{h+1}, \ldots, e_{\ell+1}) & \text{if } \; e_h > -1 \\ (0) & \text{if } \; e_h < -1 \end{cases}. \qquad (35)$$

  In particular, $h = \ell + 1$ then $\boldsymbol{e} = (\underbrace{-1, -1, \ldots, -1}_{h \text{ times}})$ and $\boldsymbol{e}^* = (\underbrace{0, 0, \ldots, 0}_{h \text{ times}}, 1)$. The next notation is also helpful:

$$LL_h(n) := \prod_{i=1}^{h} \ell\ell g_i(n) = \lg n \cdot \lg \lg n \cdot \ell\ell g_3(n) \cdots \ell\ell g_h(n). \qquad (36)$$

  Then, in case $e_h > -1$, we see that

$$\mathrm{EL}_0^{\boldsymbol{e}^*}(n) = \mathrm{EL}_0^{\boldsymbol{e}}(n) LL_h(n). \qquad (37)$$

Theorem 8 *Let $S^{\boldsymbol{e}}_k(n)$ be an elementary sum in reduced form with $\boldsymbol{e} = (e_0, \ldots, e_\ell)$.*

$$S^{\boldsymbol{e}}_k(n) = \Theta\begin{cases} \mathrm{EL}^{\boldsymbol{e}}_k(n) & \text{if } (k \le -1 \wedge e_0 > 0), \\ \mathrm{EL}^{\boldsymbol{e}^*}_0(n) & \text{if } k = 0, \\ 1 & \text{if } (k \le -1 \wedge e_0 < 0) \end{cases} \tag{38}$$

$$= \Theta\begin{cases} \mathrm{EL}^{\boldsymbol{e}}_k(n) & \text{if } (k \le -1 \wedge e_0 > 0), \\ \mathrm{EL}^{\boldsymbol{e}}_0(n) LL_h(n) & \text{if } (k = 0 \wedge e_h > -1) \\ 1 & \text{if } (k \le -1 \wedge e_0 < 0) \text{ or } (k = 0 \wedge e_h < -1) \end{cases} \tag{39}$$

where $(e_0, \ldots, e_\ell, 0) = (-1, \ldots, -1, e_h, \ldots, e_{\ell+1})$ *and* $e_h \neq -1$.

Proof. Suppose $k \le -1$. Since the elementary sum is in reduced form, $e_0 \neq 0$. If $e_0 > 0$, then $S^{\boldsymbol{e}}_k n$ is an increasing exponential sum; if $e_0 < 0$, it is decreasing exponential sum. So the formula (38) for the two cases of $k \le -1$ follows by our summation rules for exponential sums (Theorem 3).

Suppose $k = 0$. Let $\boldsymbol{e}' = (e_1, \ldots, e_\ell)$ be obtained from $\boldsymbol{e}$ by dropping e_0. In case $\ell = 0$, $\boldsymbol{e}' = (0)$. We have:

$$\begin{aligned} S^{\boldsymbol{e}}_0(n) &= \Theta(S^{\boldsymbol{e}+1}_{-1}(\lg n)) && \text{(by Lemma 7)} \\ &= \Theta(\textstyle\sum_{p\ge1}^{\lg n} 2^{(1+e_0)p} \mathrm{EL}^{\boldsymbol{e}'}_0(p)). \end{aligned}$$

Now the function $F(p) := (2^{1+e_0})^p \mathrm{EL}^{\boldsymbol{e}'}_0(p)$ is equal to $\Theta(\mathrm{EL}^{\boldsymbol{e}'}_0(p))$ if $e_0 = -1$; it is increasing exponentially if $e_0 > -1$; and it is decreasing exponentially if $e_0 < -1$. Applying the summation rules of Theorem 3 to $S_F(\log n)$, we conclude:

$$S^{\boldsymbol{e}}_0(n) = \Theta\begin{cases} (2^{1+e_0})^{\lg n} \mathrm{EL}^{\boldsymbol{e}'}_0(\lg n) & \text{if } e_0 > -1, \\ \sum_{p=1}^{\lg n} \mathrm{EL}^{\boldsymbol{e}'}_0(p) & \text{if } e_0 = -1, \\ 1 & \text{if } e_0 < -1. \end{cases}$$

$$= \Theta\begin{cases} \mathrm{EL}^{\boldsymbol{e}+1}_0(n) & \text{if } e_0 > -1, \\ S^{\boldsymbol{e}'}_0(\lg n) & \text{if } e_0 = -1, \\ 1 & \text{if } e_0 < -1. \end{cases} \tag{40}$$

Note that (40) includes the case where $\boldsymbol{e} = (0)$, in which case $\boldsymbol{e}+1 = (1)$ and $S^{(0)}_0(n) = \Theta(\mathrm{EL}^{(1)}_0(n)) = \Theta(n)$.

We may write $\boldsymbol{e} = (-1, -1, \ldots, -1, e_h, e_{h+1}, \ldots, e_\ell)$ for some $h \in \{0, 1, \ldots, \ell+1\}$. Let

$$\boldsymbol{d} = \begin{cases} (e_h, e_{h+1}, \ldots, e_\ell) & \text{if } h \le \ell \\ (0) & \text{if } h = \ell + 1 \end{cases}$$

We view (40) (case $e_0 = -1$) as a transformation rule for $S^{\boldsymbol{e}}_0(n)$. Applying this transformation for h times, we obtain

$$\begin{aligned} S^{\boldsymbol{e}}_0(n) &= \Theta(S^{\boldsymbol{e}'}_0(\lg n)) \\ &= \cdots \\ &= \Theta(S^{\boldsymbol{d}}_0(\ell\ell g_h(n))). \end{aligned}$$

Applying transformation (40) once more, to $S^{\boldsymbol{d}}_0(\ell\ell g_h(n))$, we have three possibilities:
(a) If $e_h < -1$, then

$$S^{\boldsymbol{d}}_0(\ell\ell g_h(n)) = \Theta(1).$$

(b) If $h = \ell + 1$, then $\boldsymbol{d} = (0)$ and

$$S^{\boldsymbol{d}}_0(\ell\ell g_h(n)) = \sum_{p\ge1}^{\ell\ell g_h(n)} \mathrm{EL}^{(0)}_0(p) = \ell\ell g_h(n) + \mathcal{O}(1). \tag{41}$$

(c) If $e_h > -1$, then

$$\begin{aligned} S_0^{\boldsymbol{d}}(\ell\ell g_h(n)) &= \mathrm{EL}_0^{\boldsymbol{d}+1}(\ell\ell g_h(n)) \\ &= \mathrm{EL}_0^{\boldsymbol{e}^*}(n) \end{aligned}$$

where $\boldsymbol{e}^* = (\underbrace{0, 0, \ldots, 0}_{h \text{ times}}, 1 + e_h, e_{h+1}, \ldots, e_\ell)$. The three possibilities (a)-(c) are captured by our formula (38):

$$S_k^{\boldsymbol{e}}(n) = \Theta(\mathrm{EL}_0^{\boldsymbol{e}^*}(n)).$$

We can also regroup the cases of (38) as in (39), to show the three distinct solution functions. **Q.E.D.**

We note some consequences of Theorem 8. The simplest case is when $k = 0$ and $\boldsymbol{e} = (c)$, and $f(x) = x^c$ (for any $c \in \mathbb{R}$). Then (38) becomes

$$S_f(n) = \sum_{x=1}^{n} x^c = \Theta \begin{cases} n^{c+1} & \text{if } c > -1, \\ \lg n & \text{if } c = -1, \\ 1 & \text{if } c < -1. \end{cases}$$

This result combines, within a common proof framework, three standard estimates: (i) the "generalized arithmetic sum" $\sum_{i=1}^n i^k = \Theta(n^{k+1})$ $(k \in \mathbb{N})$, (ii) the harmonic numbers $H_n = \Theta(\lg n)$, and (iii) the decreasing exponential series $\sum_{i=1}^n b^i = \Theta(1)$ $(0 < |b| < 1)$. Other instances of (38) include:

$$\sum_{x \geq 1}^{n} \frac{(\lg\lg\lg x)^c}{x \lg x \lg\lg x} = \Theta((\lg\lg\lg n)^{c+1}), \qquad (c > -1)$$

and

$$\sum_{x \geq 1}^{n} \frac{1}{x \lg x \cdot \ell\ell g_2(x) \cdots \ell\ell g_k(x)} = \Theta(\ell\ell g_{k+1}(n)).$$

The last Θ-bound corresponds to exact integral formulas from calculus (e.g., Goursat [14, p. 349]. [1, p. 69, §4.1.52] has the formula for $k = 1$).

4 Generalized Master Theorem

Proposition 2 extended the standard master theorem to include driving functions of the form $d(n) = n^\alpha \log^c n$. Our generalized[3] master theorem next completes this generalization to allowing $d(n)$ to be any EL-function.

Theorem A – Generalized Master Theorem.
Consider the master recurrence (1) with $\alpha = \log_b a$ as the watershed constant, and $d(n) = \mathrm{EL}_k^{\boldsymbol{e}}(n)$ as driving function. Assume $k \leq 0$ and $\boldsymbol{e} = (e_0, \ldots, e_\ell)$ is in reduced form. Let $(e_1, \ldots, e_\ell, 0) = (-1, -1, \ldots, -1, e_h, \ldots, e_{\ell+1})$ for some $h = 1, \ldots, \ell + 1$ where $e_{\ell+1} = 0$ and $e_h \neq -1$. Then

$$T(n) = \Theta \begin{cases} d(n) & \text{if } (k < 0 \wedge e_0 > 0) \text{ or } (k = 0 \wedge e_0 > \alpha), & \text{[CASE } (+)] \\ d(n) LL_h(n) & \text{if } (k = 0 \wedge e_0 = \alpha \wedge e_h > -1), & \text{[CASE } (h-1)] \\ n^\alpha & \text{if } (k < 0 \wedge e_0 < 0) \text{ or } (k = 0 \wedge e_0 < \alpha) \text{ or } (k = 0 \wedge e_0 = \alpha \wedge e_h < -1). & \text{[CASE } (-)] \end{cases} \tag{42}$$

Before giving the proof, we make a few remarks:

1. Theorem A has infinitely many cases, corresponding to $h = 1, 2, 3, \ldots$. Note that CASE (0) and CASE

[3]We are generalizing the (standard) master theorem, not solving a "generalized master recurrence". As there are many generalizations of the master recurrence, we decline to call any particular formulation "the" generalized master recurrence.

(1) of Proposition 2 correspond to $h = 1$ and $h = 2$, respectively. By the definition of $LL_h(n)$ (see (36)), the solution to CASE $(h-1)$ can also be written $T(n) = \Theta(n^\alpha \ell\ell g_h(x)$ instead of $T(n) = \Theta(d(n)LL_h(n))$.
2. Although the driving functions covered by these infinitely many cases are in some sense complete, there are many driving functions that are not covered; Verma's Theorem 13 [33] can capture some of these.
3. To make Theorem A fully comparable to Proposition 1, we could have formulated CASE (+) to include the possibility of $d(n)$ satisfying the regularity condition (8).
Proof. Let $n = b^m$. From (9), we have

$$\begin{aligned} T(n) = \sum_{i=0}^{m} a^i d(n/b^i) &= \sum_{i\geq 0}^{m} a^{m-i} d(b^i) \qquad \text{(by (12))} \\ &= n^\alpha \sum_{i\geq 0}^{m} a^{-i} d(b^i) \\ &= n^\alpha \sum_{i\geq 0}^{m} a^{-i} \mathrm{EL}_k^{\boldsymbol{e}}(b^i). \end{aligned} \tag{43}$$

If $k \leq -1$, the function $F(i) := a^{-i}\mathrm{EL}_k^{\boldsymbol{e}}(b^i)$ is increasing exponentially when $e_0 > 0$, and decreasing exponentially when $e_0 < 0$. Applying our summation rules (Theorem 3) to (43),

$$\begin{aligned} T(n) &= n^\alpha \cdot \Theta \begin{cases} a^{-m}\mathrm{EL}_k^{\boldsymbol{e}}(b^m) & \text{if } e_0 > 0 \\ 1 & \text{if } e_0 < 0 \end{cases} \\ &= n^\alpha \cdot \Theta \begin{cases} n^{-\alpha}\mathrm{EL}_k^{\boldsymbol{e}}(n) & \text{if } e_0 > 0 \\ 1 & \text{if } e_0 < 0 \end{cases} \\ &= \Theta \begin{cases} \mathrm{EL}_k^{\boldsymbol{e}}(n) & \text{if } e_0 > 0 \\ n^\alpha & \text{if } e_0 < 0 \end{cases}. \end{aligned}$$

This proves our theorem in case $k \leq -1$. Next assume $k = 0$. Then (43) becomes Let $\boldsymbol{d} = (e_1, \ldots, e_\ell, 0) = (e_1, \ldots, e_{1+\ell})$, i.e., $\boldsymbol{e}$ without the leading entry e_0, with an appended $0 = e_{1+\ell}$).

$$\begin{aligned} T(n) &= n^\alpha \sum_{i\geq 0}^{m} a^{-i}\mathrm{EL}_0^{\boldsymbol{e}}(b^i) \\ &= n^\alpha \sum_{i\geq 0}^{m} a^{-i} \cdot b^{e_0 i} \cdot \mathrm{EL}_1^{\boldsymbol{d}}(b^i) \\ &= n^\alpha \sum_{i\geq 0}^{m} (b^{e_0}/a)^i \cdot \mathrm{EL}_1^{\boldsymbol{d}}(b^i). \end{aligned} \tag{44}$$

Now,

$$(b^{e_0}/a) \begin{cases} > 1 & \text{if } e_0 > \alpha, \\ = 1 & \text{if } e_0 = \alpha, \\ < 1 & \text{if } e_0 < \alpha. \end{cases}$$

Hence the sum (44) is exponential-type when $e_0 \neq \alpha$, and is polynomial-type when $e_0 = \alpha$. Again, our summation rules yield

$$\begin{aligned} T(n) &= n^\alpha \cdot \Theta \begin{cases} a^{-m}\mathrm{EL}_0^{\boldsymbol{e}}(b^m) & \text{if } e_0 > \alpha, \\ \sum_{i\geq 0}^{m} \mathrm{EL}_1^{\boldsymbol{d}}(b^i) & \text{if } e_0 = \alpha, \\ 1 & \text{if } e_0 < \alpha \end{cases} \\ &= \Theta \begin{cases} \mathrm{EL}_0^{\boldsymbol{e}}(n) & \text{if } e_0 > \alpha, \\ n^\alpha \cdot \sum_{i\geq 0}^{m} \mathrm{EL}_0^{\boldsymbol{d}}(i \lg b) & \text{if } e_0 = \alpha, \\ n^\alpha & \text{if } e_0 < \alpha. \end{cases} \end{aligned} \tag{45}$$

Thus, the theorem is shown for the case $k = 0$ and $e_0 \neq \alpha$. It remains to consider the case $k = 0$ and $e_0 = \alpha$. With h as in the theorem, let $\boldsymbol{d}^+ = (\underbrace{0, 0, \ldots, 0}_{h \text{ times}}, 1 + e_h, \ldots, e_{1+\ell})$. Starting from (45),

$$\begin{array}{rcll} T(n) & = & \Theta(n^\alpha \cdot \sum_{i\geq 0}^{m} \mathrm{EL}_0^{\boldsymbol{d}}(i \lg b)) & \\ & = & \Theta(n^\alpha \cdot \sum_{i\geq 0}^{m} \mathrm{EL}_0^{\boldsymbol{d}}(i)) & (\lg b \text{ is constant in a polynomial-type sum}) \\ & = & \Theta(n^\alpha \cdot S_0^{\boldsymbol{d}}(m)) & (\text{definition of } S_0^{\boldsymbol{d}}(m)) \\ & = & \Theta(n^\alpha \cdot \mathrm{EL}_0^{\boldsymbol{d}^+}(m)) & (\text{by Theorem } 8) \end{array}$$

If $e_h < -1$, $\boldsymbol{d}^+ = (0)$ (by (35)) and this proves our theorem in this case.

Q.E.D.

5 Real Induction

The principle of natural induction, or induction on $\mathbb{N}$, is well-known. Briefly, if $P(n)$ is a predicate on natural numbers, then to prove the validity of $P(n)$, we must show that $P(0)$ holds (basis), and that if $P(n)$ holds, then $P(n+1)$ holds (inductive step). In strong natural induction, we deduce $P(n+1)$ from $P(0), \ldots, P(n)$. Properties of integer recurrences can often be proved with this principle. But in our transition to real recurrences, we appear to have lost this tool. We now provide such a substitute.

Induction on real numbers is seldom found in the literature. But the need for such induction principles arises in automatic correctness proofs of programs involving real numbers [10], or involving timing logic [25], or in the programming language Real PCF [12].

PRINCIPLE OF (ARCHIMEDEAN) REAL INDUCTION.
Let $P(x)$ be a real predicate (see ¶2). Suppose there exist real numbers x_1 and $\gamma > 0$ satisfying the following conditions:

Real Basis: *For all $x < x_1$, $P(x)$ holds.*

Real Induction Step: *For all $y \geq x_1$, if $(\forall x \leq y - \gamma)[\ P(x)$ holds$\ ]$ then $P(y)$.*

Then $P(x)$ is valid.

It is helpful to re-formulate the Real Induction Step as follows: For all $y \geq x_1$,

$$P^*(y) \Rightarrow P(y)$$

where $P^*(y) \equiv (\forall x \leq y - \gamma)[P(x)$ holds$]$. We call $P^*(y)$ the **Real Induction Hypothesis**.

We call $\gamma > 0$ the **gap constant** of this instance of Real Induction; it plays the role of the constant 1 in natural induction. Similarly, x_1 plays the role of 0 in natural induction, and is called the **cutoff parameter**. This principle is called "Archimedean" because its justification exploits the Archimedean property of real numbers: let x_1 and $\gamma > 0$ be any real constant. Then for all real x, there is smallest natural number $n = n(x)$ such that $x \leq x_1 + n\gamma$. Showing the validity of Real Induction can be reduced to a strong natural induction on $n = n(x)$. We omit the straightforward proof.

To use this principle to prove the validity of a predicate $P(x)$, we need to show (1) the real basis, and (2) the real induction step. Typically, it is the real basis that is tedious; the real induction step is often quite intuitive. The predicate "$(\forall x \le y-\gamma)[\, P(x)$ holds $]$" in the real induction step is called the **real induction hypothesis**. In showing the real induction step, we would need to invoke the real induction hypothesis. In our applications below, there will be a constant x_0 such that $P(x)$ is vacuously true for $x \le x_0$. E.g., $P(x)$ has the form "$x \ge x_0 \Rightarrow \ldots$" Typically, we need to choose the cutoff parameter such that $x_1 > x_0$.

We next illustrate the use of this real induction principle.

¶9. Multiterm Regularity Condition. Consider the multiterm recurrence (2). We say its driving function $d(n)$ satisfies the **multiterm regularity condition** if the following is valid:

$$\sum_{i=1}^{k} a_i d\left(\frac{n}{b_i}\right) \le c \cdot d(n) \tag{46}$$

for some $0 < c < 1$. We may again associate with (2) a **watershed constant** α satisfying the equation

$$\sum_{i=1}^{k} \frac{a_i}{b_i^{\alpha}} = 1. \tag{47}$$

Clearly α exists and is unique. Equation (47) is known as the characteristic equation of (2) [17, 5].

Lemma 9 *The multiterm regularity condition implies that $d(n) = \Omega(n^{\alpha+\varepsilon})$ for some $\varepsilon > 0$.*

Proof. For some $D > 0$ and n_0, we want to prove the validity of the following predicate $P(n)$: "for all $n \ge n_0$, $d(n) \ge D \cdot n^{\alpha+\varepsilon}$". The real basis says there is some $n_1 > n_0$ such that the predicate $P(n)$ holds whenever $n < n_1$. The real induction step is:

$$\begin{array}{rcll} d(n) & \ge & \frac{1}{c}\sum_{i=1}^{k} a_i d(n/b_i) & \text{(by multiterm regularity)} \\ & \ge & \frac{1}{c}\sum_{i=1}^{k} a_i D\cdot (n/b_i)^{\alpha+\varepsilon} & \text{(by real induction hypothesis)} \\ & = & \frac{D}{c} n^{\alpha+\varepsilon} \sum_{i=1}^{k} \frac{a_i}{b_i^{\alpha+\varepsilon}} & \\ & = & D n^{\alpha+\varepsilon}. & \end{array}$$

The last equality follows if we choose ε to satisfy the equation

$$c = \sum_{i=1}^{k} \frac{a_i}{b_i^{\alpha+\varepsilon}}.$$

Since $c < 1$, such a choice is clearly possible.

It remains to show the real basis for induction, and how to choose D, n_0, n_1. Our derivation so far imposes no conditions on these, so they can be determined from the initial conditions. Choose $n_0 \ge 1$ large enough so that $d(n)$ is defined for $n \ge n_0$. There is some Archimedean constant $\gamma \in (0,1)$ such that for all $n \ge n_0$,

$$n\gamma < \min_{i=1}^{k} \{n/b_i\} \le \max_{i=1}^{k} \{n/b_i\} \le n - \gamma. \tag{48}$$

We choose the cutoff parameter to be $n_1 = n_0/\gamma$. Finally, choose $D > 0$ such that $1 \ge D n^{\alpha+\varepsilon}$ holds for all $n \in [n_0, n_1]$. With these choices, the real basis holds non-vacuously because, by assumption, $d(n) \ge 1$.
Q.E.D.

This lemma shows that regularity condition on $d(n)$ implies that $d(n)$ is lower bounded by the watershed function n^{α}: Variations of the parameters n_0 and n_1 in this proof will be used again in the proof of Theorem B.

6 The Multiterm Master Theorem

Our next theorem has the same structure as Proposition 2, but generalized to the multiterm master recurrence. We are no longer in possession of a summation formula for its solution $T(n)$ analogous to (9), and so the previous proof breaks down. One alternative approach is to use integral calculus, but it seems that real induction gives the most direct argument.

THEOREM B – MULTITERM MASTER THEOREM.
Let $T(n)$ satisfy the multiterm recurrence (2), with driving function $d(n)$ and watershed constant α. Then

$$T(n) = \begin{cases} \Theta(d(n)) & \text{if } d(n) \text{ satisfies the multiterm regularity condition (46)} & \text{[CASE (+)]}\,, \\ \Theta(d(n)\lg n) & \text{if } d(n) = \Theta(n^\alpha \lg^\delta n)) \textit{ for some } \delta > -1 & \text{[CASE (0)]}\,, \\ \Theta(d(n)\lg n \lg\lg n) & \text{if } d(n) = \Theta(n^\alpha \lg^\delta n)) \textit{ where } \delta = -1 & \text{[CASE (1)]}\,, \\ \Theta(n^\alpha) & \text{if } d(n) = \mathcal{O}(n^\alpha \lg^\delta n) \textit{ for some } \delta < -1 & \text{[CASE (−)]}\,. \end{cases}$$

Note that restricted versions of Theorem B have appeared in the literature: they all have 3 cases, without our CASE (1), in analogy with Proposition 1. Kao [17] assumes $d(n) = n^\alpha \log^\delta n$, and gave an inductive proof for the case $k = 2$ only. A full proof in his setting (3) of integer recurrences and initial conditions would be quite involved. Roura [31, Theorem 2.3] treats driving functions $d(n)$ that are somewhat more general than Kao's. Akra and Bazzi [5] deduced their result from a more general integral bound with a long proof. Here we give a direct proof based real induction. Leighton's [22] simplification of Akra and Bazzi also uses induction.

Proof. First we first prove the real induction steps for each of the cases, leaving the real bases for the end.

CASE (+): This is the easiest case. The lower bound $T(n) = \Omega(d(n))$ is trivial. For the upper bound, we will show $T(n) \le D_1 d(n)$ (ev.), for some D_1:

$$\begin{array}{rcll} T(n) & = & d(n) + \sum_{i=1}^k a_i T\left(\frac{n}{b_i}\right) & \\ & \le & d(n) + \sum_{i=1}^k a_i D_1 d(n/b_i) & \text{(by real induction hypothesis)} \\ & = & d(n) + D_1 c d(n) & \text{(by regularity)} \\ & \le & D_1 d(n) & \text{(choosing } D_1 \ge 1/(1-c)) \end{array}$$

CASE (0): Assume that $d(n) = n^\alpha \lg^\delta n$ for some $\delta > -1$. We first show that $T(n) \le D_1 d(n) \lg n$. We have

$$\begin{array}{rcll} T(n) & = & d(n) + \sum_{i=1}^k a_i T\left(\frac{n}{b_i}\right) & \\ & \le & n^\alpha \lg^\delta n + \sum_{i=1}^k a_i D_1 \left(\frac{n}{b_i}\right)^\alpha \lg^{\delta+1}\left(\frac{n}{b_i}\right) & \text{(by real induction hypothesis)} \\ & = & n^\alpha \lg^\delta n + D_1 n^\alpha \lg^{\delta+1} n \left[\sum_{i=1}^k \frac{a_i}{b_i^\alpha}\left(1 - \frac{\lg b_i}{\lg n}\right)^{\delta+1}\right] & \\ & = & D_1 n^\alpha \lg^{\delta+1} n \left[\frac{1}{D_1 \lg n} + \sum_{i=1}^k \frac{a_i}{b_i^\alpha}\left\{1 - (\delta+1)\frac{\lg b_i}{\lg n}(1+o(1))\right\}\right] & \text{(where } o(1) \to 0 \text{ as } n \to \infty) \\ & = & D_1 n^\alpha \lg^{\delta+1} n \left[1 + \frac{1}{\lg n}\left\{\frac{1}{D_1} - (\delta+1)\sum_{i=1}^k \frac{a_i \lg b_i}{b_i^\alpha}(1+o(1))\right\}\right] & \\ & \le & D_1 n^\alpha \lg^{\delta+1} n & \end{array}$$

provided D_1 is sufficiently large to verify

$$\frac{1}{D_1} < (\delta+1)\sum_{i=1}^k \frac{a_i \lg b_i}{b_i^\alpha}.$$

Here the condition $\delta > -1$ is necessary. Similarly, we show the lower bound $T(n) \geq D_2 d(n) \lg n$ using the same derivation above, but with reversed inequalities. The provision is that D_2 is small enough to verify

$$\frac{1}{D_2} > (\delta+1) \sum_{i=1}^{k} \frac{a_i \lg b_i}{b_i^\alpha}.$$

CASE (1): Assume that $d(n) = n^\alpha / \lg n$. We first show that $T(n) \leq D_1 d(n) \lg\lg n$.

$$\begin{array}{rcll}
T(n) & = & d(n) + \sum_{i=1}^{k} a_i T\left(\frac{n}{b_i}\right) & \\
& \leq & \frac{n^\alpha}{\lg n} + \sum_{i=1}^{k} a_i D_1 \left(\frac{n}{b_i}\right)^\alpha \lg\lg\left(\frac{n}{b_i}\right) & \text{(by real induction hypothesis)} \\
& = & D_1 n^\alpha \left[\frac{1}{D_1 \lg n} + \sum_{i=1}^{k} \frac{a_i}{b_i^\alpha} \lg\left\{(\lg n)\left(1 - \frac{\lg b_i}{\lg n}\right)\right\}\right] & \\
& = & D_1 n^\alpha \left[\frac{1}{D_1 \lg n} + \sum_{i=1}^{k} \frac{a_i}{b_i^\alpha} \left\{\lg\lg n + \lg\left(1 - \frac{\lg b_i}{\lg n}\right)\right\}\right] & \\
& = & D_1 n^\alpha \left[\lg\lg n + \frac{1}{D_1 \lg n} + \sum_{i=1}^{k} \frac{a_i}{b_i^\alpha} \lg\left(1 - \frac{\lg b_i}{\lg n}\right)\right] & \\
& = & D_1 n^\alpha \left[\lg\lg n + \frac{1}{D_1 \lg n} - \sum_{i=1}^{k} \frac{a_i}{b_i^\alpha} \frac{\lg b_i}{\lg n}(1+o(1))\right] & \\
& = & D_1 n^\alpha \left[\lg\lg n + \frac{1}{\lg n}\left\{\frac{1}{D_1} - \sum_{i=1}^{k} \frac{a_i \lg b_i}{b_i^\alpha}(1+o(1))\right\}\right] & \\
& \leq & D_1 n^\alpha \lg\lg n &
\end{array}$$

provided D_1 is large enough to verify

$$\frac{1}{D_1} < \sum_{i=1}^{k} \frac{a_i \lg b_i}{b_i^\alpha}.$$

Similarly, the lower bound $T(n) \geq D_2 n^\alpha \lg\lg n$ uses the above derivation with inequalities reversed, and D_2 small enough to verify

$$\frac{1}{D_2} > \sum_{i=1}^{k} \frac{a_i \lg b_i}{b_i^\alpha}.$$

CASE $(-)$: Assume $0 \leq d(n) \leq n^\alpha \lg^\delta n$ for some $\delta < -1$. This is the trickiest case: to show $T(n) = \mathcal{O}(n^\alpha)$, the hypothesis $T(n) \leq D_1 n^\alpha$ will not do. The trick is to use a stronger hypothesis of the form $T(n) \leq D_1 n^\alpha \left[1 - K \lg^{\delta+1} n\right]$ (ev.) for some positive D_1, K. Note that this hypothesis requires $\delta < -1$.

$$\begin{array}{rcll}
T(n) & = & d(n) + \sum_{i=1}^{k} a_i T\left(\frac{n}{b_i}\right) & \\
& \leq & n^\alpha \lg^\delta n + \sum_{i=1}^{k} a_i D_1 \left(\frac{n}{b_i}\right)^\alpha \left[1 - K \lg^{\delta+1}\left(\frac{n}{b_i}\right)\right] & \text{(by induction hypothesis)} \\
& = & D_1 n^\alpha \left[\frac{\lg^\delta n}{D_1} + 1 - K \sum_{i=1}^{k} \frac{a_i}{b_i^\alpha} \lg^{\delta+1}\left(\frac{n}{b_i}\right)\right] & \\
& = & D_1 n^\alpha \left[\frac{\lg^\delta n}{D_1} + 1 - K \lg^{\delta+1} n \sum_{i=1}^{k} \frac{a_i}{b_i^\alpha} \left(1 - \frac{\lg b_i}{\lg n}\right)^{\delta+1}\right] & \\
& = & D_1 n^\alpha \left[1 - K \lg^{\delta+1} n \left\{-\frac{1}{K D_1 \lg n} + \sum_{i=1}^{k} \frac{a_i}{b_i^\alpha} \left(1 - (\delta+1)\frac{\lg b_i}{\lg n}\right)(1+o(1))\right\}\right] & \\
& = & D_1 n^\alpha \left[1 - K \lg^{\delta+1} n \left\{1 - \frac{1}{K D_1 \lg n} - \frac{(\delta+1)}{\lg n} \sum_{i=1}^{k} \frac{a_i \lg b_i}{b_i^\alpha}(1+o(1))\right\}\right] & \\
& = & D_1 n^\alpha \left[1 - K \lg^{\delta+1} n \left\{1 - \frac{1}{\lg n}\left(\frac{1}{K D_1} + (\delta+1) \sum_{i=1}^{k} \frac{a_i \lg b_i}{b_i^\alpha}(1+o(1))\right)\right\}\right] & \\
& \leq & D_1 n^\alpha \left[1 - K \lg^{\delta+1} n\right] &
\end{array}$$

provided KD_1 is small (sic) enough to verify

$$\frac{1}{K D_1} > -(\delta+1) \sum_{i=1}^{k} \frac{a_i \lg b_i}{b_i^\alpha}. \tag{49}$$

The introduction of K is crucial: otherwise, if K is fixed global constant, then condition (49) imposes a lower bound on D_1 which may be difficult to satisfy in the real basis.

For the lower bound, we show that $T(n) \geq D_2 n^\alpha(1+\lg^{\delta+1} n)$. The derivations is essentially the same, except inequalities are reversed:

$$\begin{aligned}
T(n) &= d(n) + \textstyle\sum_{i=1}^k a_i T\left(\frac{n}{b_i}\right) \\
&\geq n^\alpha \lg^\delta n + \textstyle\sum_{i=1}^k a_i D_2 \left(\frac{n}{b_i}\right)^\alpha \left[1+\lg^{\delta+1}\left(\frac{n}{b_i}\right)\right] && \text{(by induction hypothesis)} \\
&= \vdots \\
&= D_2 n^\alpha \left[1+\lg^{\delta+1} n \left\{1 + \tfrac{1}{\lg n}\left(\tfrac{1}{D_2} - (\delta+1)\textstyle\sum_{i=1}^k \frac{a_i \lg b_i}{b_i^\alpha}(1+o(1))\right)\right\}\right] \\
&\geq D_2 n^\alpha \left[1+\lg^{\delta+1} n\right]
\end{aligned}$$

since $\delta+1<0$. This time, there are no restrictions on D_2 except through the initial conditions. This completes the real inductive step for the 4 cases.

We now provide the real bases for each of the above real induction. Choose n_0 so that $d(n)$ is defined and the recurrence (2) for $T(n)$ holds non-vacuously, for all $n \geq n_0$. Let $\gamma = \gamma(n_0)$ be the Archimedean constant defined as in (48). The cutoff parameter n_1 will be chosen to be at least n_0/γ in the cases below, ensuring that the real induction hypothesis holds nonvacuosly.

- CASE (+): Choose $n_1 = n_0/\gamma$. Finally, let
$$D_1 := \max\left(\frac{1}{1-c}, \sup_{n_0\leq n\leq n_1}\left\{\frac{T(n)}{d(n)}\right\}\right)$$
where $1/(1-c)$ is the constant noted in the derivation of this case.
- CASE (0): For upper bound, first let n_2 be large enough so that the $o(1)$ term have absolute value $< 1/2$. Then choose $n_1 = \max\{n_2, n_0/\gamma\}$. Choose
$$D_1 = \max\left(\frac{2}{(\delta+1)\sum_i \frac{a_i \lg b_i}{b_i}}, \sup_{n_0\leq n\leq n_1}\left\{\frac{T(n)}{d(n)\lg n}\right\}\right).$$
For lower bound, we choose analogous n_2 and then n_1. Choose
$$D_2 = \min\left(\frac{2}{3(\delta+1)\sum_i \frac{a_i \lg b_i}{b_i}}, \inf_{n_0\leq n\leq n_1}\left\{\frac{T(n)}{d(n)\lg n}\right\}\right).$$
- CASE (1): This case is similar to the previous, and is omitted.
- CASE (−): For upper bound, we first choose the product KD_1 to be equal to the reciprocal of $-(\delta+1)\sum_{i=1}^k \frac{3a_i \lg b_i}{2b_i^\alpha}$. Now, let n_2 be large enough so that the $o(1)$ term have absolute value $< 1/2$, and such that for $n \geq n_2$, the function
$$f(n) = n^\alpha - (KD_1)\lg^{\delta+1} n$$
is increasing and ≥ 1. Then n_1 is chosen to be $\max n_0/\gamma, n_2$. Finally, choose
$$D_1 := \sup_{n_0\leq n\leq n_1}\left\{\frac{T(n)}{f(n)}\right\}.$$
Note that $f(n) \leq D_1 n^\alpha(1-K\lg^{\delta+1} n)$ and hence $T(n) \leq f(n) \leq D_1 n^\alpha(1-K\lg^{\delta+1} n)$ for $n \in [n_0, n_1]$.

Q.E.D.

7 Robustness Results

In this section, we make prove some theorems about the insensitivity of our Θ-order solutions to variations in (2). There are 3 specific issues: (i) choice of initial conditions, (ii) Θ-order modifications to the driving functions, and (iii) perturbations of the recursive arguments. For example, we may perturb the recursive argument n/b_i in (2) into $\lceil n/b_i \rceil + e$ for some small constant e.

Item (i) is especially useful for simplifying multiterm recurrences because keeping track of the initial conditions can be tedious (cf. Leighton [22]). In real induction proofs, this tedium translates into providing a real basis for induction. Intuitively, item (i) exploits the that (2) is a non-homogeneous recurrence with a positive driving function, $d(n) \geq 1$ (ev.). This is natural in algorithms since $d(n)$ represents the cost of "subdividing and combining" subproblems. Item (ii) justifies the standard disregard for multiplicative constants, and item (iii) justifies discarding the ceilings or floor functions.

By a **divide-and-conquer scheme** we mean a sequence of complexity functions of the form

$$(d, G, b_1, \ldots, b_k) = (d(n), G(n, x_1, \ldots, x_k), b_1(n), \ldots, b_k(n)).$$

This scheme defines the set

$$SOL = SOL(d, G, b_1, \ldots, b_k)$$

of complexity functions $T(n)$ that eventually satisfies the following recurrence

$$T(n) = d(n) + G(n,\ T(b_1(n)), \ldots, T(b_k(n))). \tag{50}$$

The scheme $(d, G, b_1, \ldots, b_k)$ is **regular** if the following properties are eventually satisfied non-vacuously:

- $d(n) \geq 1$ and non-decreasing;
- $G(n, x_1, \ldots, x_k) > 0$ and is increasing in each component;
- $G(n, x_1, \ldots, x_k)$ is **sublinear**: this means for all $C > 0$,

$$G(n, Cx_1, \ldots, Cx_k) \leq C \cdot G(n, x_1, \ldots, x_k).$$

- Each b_i is **Archimedean**: this means b_i increases unboundedly, and $0 < b_i(n) < n - \gamma$ for a fixed constant $\gamma > 0$.

We call $\gamma > 0$ the **gap constant** of the regular scheme. For a regular scheme, there exists a constant $n_0 = n_0(d, G, b_1, \ldots, b_k)$ such that for all $n \geq n_0$ and $x_i \geq n_0$ $(i = 1, \ldots, k)$, all the above properties are non-vacuously satisfied. We call $d(n)$ the **driving function**, each $b_i(n)$ a **divide function** and $G(n, x_1, \ldots, x_k)$ the **combine function** of the scheme.

Example and non-examples of regular schemes:

- Typical driving functions include $d_1(n) = 1$, $d_2(n) = \log n$ or $d_2(n) = n^k$ $(k \geq 1)$. However, $d(n) = 0$ or $d(n) = 1/n$ would be irregular, and disallowed.
- The combine function in the multiterm master recurrence is $G(n, x_1, \ldots, x_k) = \sum_{i=1}^{k} a_i x_i$ where a_i are positive constants. Clearly G is sublinear and increasing in each component.

We may generalize G so that each a_i is be replaced by polynomials in n with positive leading coefficients: e.g., $G(n,x,y) = n^2x + (2n-3)y$. This is basically the class of **parametric multiterm master recurrences** introduced by Wang and Fu [34]:

$$G(n, x_1, \ldots, x_k) = \sum_{i=1}^{k} a_i(n)x_i \tag{51}$$

where the a_i's are complexity functions satisfying $a_i(n) > 0$ and increasing (ev.).

However, non-linear terms in the x's such as $G(n,x,y) = x^2y$ would be irregular.

- The following divide functions are easily seen to be Archimedean: for constants $C > 1, c > 0$,

  $$b_1(n) = n/C, \quad b_2(n) = n - c, \quad b_3(n) = \frac{n}{2} + \frac{n}{\lg n}.$$

  The function $b_1(n)$ occurs in the Master Recurrence, $b_2(n)$ occurs in Fibonacci Recurrence, and $b_3(n)$ occurs in a recurrence of [22].

- Our divide-and-conquer schemes do not capture the "full history" recurrences of Roura [31], where $T(n)$ may depend on all previous values $T(n-1), T(n-2), \ldots, T(n - \lfloor n \rfloor)$. Nevertheless, such recurrences can sometimes be transformed into the parametric form (51). For instance, the quicksort recurrence [31, equation (4)] is a full history of the form:

  $$T(n) = n + 1 + \frac{2}{n}\sum_{i \geq 0}^{n-1} T(i) \tag{52}$$

  Note that we view (52) as the real version of the usual integer recurrence of quicksort. By subtracting $(n-1)T(n-1)$ from $nT(n)$, the full-history recurrence (52) becomes a (single-term) parametric recurrence,

  $$T(n) = 2 + \frac{n+1}{n}T(n-1). \tag{53}$$

Let K, K' be two classes of complexity functions. We say K is **big-Oh** of K', written $K = \mathcal{O}(K')$, if for all $f \in K$ and $f' \in K'$, we have $f' = \mathcal{O}(f)$. If $K = \mathcal{O}(K)$ (i.e., $K = K'$ in the previous definition), then we say K is **Θ-robust**. Note Θ-robustness of K means that any two functions in K are Θ-order of each other. We now establish some robustness properties of the class

$$SOL = SOL(d, G, b_1, \ldots, b_k),$$

of solutions to a regular scheme.

- **Lemma 10** *Each $T \in SOL$ is eventually increasing.*

  Proof. Assume for all $n \geq n_1$, T satisfies its defining recurrence. We may assume that $n_1 > n_0 = n_0(d, G, b_1, \ldots, b_k)$. If $n' > n \geq n_1$, then:

  $$\begin{array}{rcll} T(n') & = & d(n') + G(n', T(b_1(n')), \ldots, T(b_k(n'))) & (n' > n_1) \\ & \geq & d(n) + G(n, T(b_1(n)), \ldots, T(b_k(n))) & (G \text{ is increasing, and by Real Induction Hypothesis}) \\ & \geq & T(n) & (n > n_1). \end{array}$$

  Q.E.D.

- **Lemma 11** *SOL is non-empty*

Proof. It suffices to construct one solution $T_0 \in SOL$. Let $n_1 := \max\left\{b_i^{-1}(n_0) : i = 1, \ldots, k\right\}$ where $n_0 = n_0(d, G, b_1, \ldots, b_k)$. Note that n_1 is uniquely defined since $b_i(n) < n$ and is increasing for $n > n_0$. It follows that $n_0 < n_1$. Let $x_1 := n_1 + \gamma$. We define T_0 recursively as follows:

$$T_0(n) = \begin{cases} 0 & \text{if } n < x_1, \\ d(n) + G(n, T(b_1(n)), \ldots, T(b_k(n))) & \text{if } n \geq x_1. \end{cases}$$

We must show that $T_0(n)$ is well-defined. We use the Principle of Real Induction: the Real Basis is trivially true, as it corresponds to the case $n < x_1$. When $n \geq x_1 = n_1 + \gamma$, the Archimedean Property implies $b_i(n) \leq n - \gamma$. So we may invoke the Real Induction Hypothesis which says that each $T(b_i(n))$ is well-defined. It follows that $T(n)$ is well-defined, as claimed. **Q.E.D.**

- **Theorem 12 (Robustness)** *SOL is* Θ*-robust.*

  Proof. The previous lemma has shown that SOL is non-empty. Suppose $T_1, T_2 \in SOL$. We must show that $T_1 = O(T_2)$. Choose n_1 larger than $n_0 = n_0(d, G, b_1, \ldots, b_k)$ so that for all $n \geq n_1$, T_1 and T_2 satisfy the recurrences

$$\begin{aligned} T_1(n) &= d(n) + G(n, T_1(b_1(n)), \ldots, T_1(b_k(n))), \\ T_2(n) &= d(n) + G(n, T_2(b_1(n)), \ldots, T_2(b_k(n))). \end{aligned}$$

  Let $n_2 := \max\left\{b_i^{-1}(n_1) : i = 1, \ldots, k\right\}$ and finally $x_1 := n_2 + \gamma$. For $n \in [n_1, x_1)$ we see that $T_j(n) \geq 1$ $(j = 1, 2)$ because $n \geq n_0$. Hence we may define the positive constant $C := \max\left\{T_1(n)/T_2(n) : n \in [n_1, x_1)\right\}$. Consider the predicate:

$$P(n) \equiv [n \geq n_1 \Rightarrow T_1(n) \leq CT_2(n)]$$

  We prove the validity of $P(n)$ by real induction. Let x_1 be the cutoff constant of real induction. Clearly, $P(n)$ is valid for $n < x_1$, by definition of C. For $n \geq x_1$, we see that

$$\begin{aligned} T_1(n) &= d(n) + G(n, T_1(b_1(n)), \ldots, T_1(b_k(n))) && (n \geq n_1) \\ &\leq d(n) + G(n, C \cdot T_2(b_1(n)), \ldots, C \cdot T_2(b_k(n))) && \text{(by Real Induction Hypothesis)} \\ &\leq d(n) + C \cdot G(n, T_2(b_1(n)), \ldots, T_2(b_k(n))) && \text{(by sublinearity of } G) \\ &\leq C[d(n) + G(n, T_2(b_1(n)), \ldots, T_2(b_k(n)))] && \text{(we may assume } C \geq 1) \\ &= CT_2(n) && (n \geq n_1). \end{aligned}$$

  The Real Induction Hypothesis needs some justification, and it amounts to the claim that $T_1(b_i(n)) \leq C \cdot T_2(b_i(n))$ for each i. To justify it, we must ensure that $b_i(n)$ is (i) not too large $(b_i(n) \leq n - \gamma)$ and (ii) not too small $(b_i(n) \geq n_1)$. Condition (i) follows from the Archimedean Property of $b_i(n)$. We need condition (ii) because our predicate $P(n)$ has the requirement "$n \geq n_1$". This is justified by $b_i(n) > b_i(n_2) \geq b_i(b_i^{-1}(n_1)) = n_1$. This proves that $T_1 = O(T_2)$, as desired. **Q.E.D.**

- **Lemma 13** *Let* $SOL' = SOL(d', G, b_1, \ldots, b_k)$*. If* $d' = \mathcal{O}(d)$ *then* $SOL' = \mathcal{O}(SOL)$

  We omit the proof as it is similar to the previous proof.

  (hide)

- Next consider perturbations in the divide functions: let b_i' and b_i be related as follows

$$b_i'(n + \delta(n)) \overset{(a)}{\leq} b_i(n) + \delta(n) \overset{(b)}{\leq} n - \gamma \tag{54}$$

  where $\delta(n) \geq 0$ and $\gamma > 0$ is a constant. We call b' a **δ-perturbation** (or simply, a perturbation) of b, and $\delta = \delta(n)$ is called a **perturbation function**.

  We illustrate some possible perturbation functions. Consider the divide functions in multiterm recurrences with

$$b_i(n) = n/c_i, \quad b_i'(n) = \lceil n/c_i \rceil + e, \qquad (i = 1, \ldots, k)$$

where $e > 0$ and $c_i > 1$. Define $\delta(n)$ as the constant $\frac{(e+1)c}{c-1}$ where $c = \min\{c_i : i = 1, \ldots, k\}$. Then (54)(a) is verified as follows:

$$b_i'(n+\delta(n)) = \left\lceil \frac{n+\delta(n)}{c_i} \right\rceil + e < \frac{n+(e+1)c/(c-1)}{c_i} + (1+e) = \frac{n}{c_i} + \left(\frac{(e+1)c}{c_i(c-1)} + 1 + e\right) \leq b_i(n) + \delta(n).$$

Moreover (54)(b) clearly holds eventually. Or consider Leighton's divide function [22]

$$b'(n) = \frac{n}{2} + \frac{n}{\lg n}. \tag{55}$$

Let $b(n) := n/2$ and define the perturbation function $\delta(n) := \frac{3n}{\log n}$. We may verify (54)(a) will eventually hold:

$$\begin{aligned} b'(n+\delta(n)) = \frac{n+\delta(n)}{2} + \frac{n+\delta(n)}{\lg(n+\delta(n))} &< \frac{n+\delta(n)}{2} + \frac{n+\delta(n)}{\lg n} \\ &= \frac{n}{2} + \frac{\delta(n)}{2} + \frac{n+\delta(n)}{\lg n} \\ &< \frac{n}{2} + \frac{3n}{\lg n} \\ &= b(n) + \delta(n). \end{aligned}$$

Moreover, $b(n) + \delta(n) < n - \gamma$ (ev.).

- **Theorem 14 (Perturbation Theorem)**
  Let $T \in SOL(d, G, b_1, \ldots, b_k)$ and $T' \in SOL(d, G, b_1', \ldots, b_k')$ where each $b_i'(n)$ is a $\delta(n)$-perturbation of $b_i(n)$ as in (54). Then there exists a constant $C = C(T, T') > 0$ such that

  $$T'(n+\delta(n)) \leq C \cdot T(n) \quad \text{(ev.)}.$$

  Before giving the proof of this theorem, we state its main corollary: Since $\delta(n) \geq 0$, $T'(n) = O(T(n))$, i.e.,

  $$SOL' = \mathcal{O}(SOL).$$

  The point is that $\delta(n)$ does not appear in this conclusion. Nevertheless, the proof requires the trick of proving a stronger assertion.

  Proof. The proof is by Real Induction. We first do the Real Induction Step:

  $$\begin{array}{rcll} T'(n+\delta(n)) & = & d(n) + G(n, T'(b_1'(n+\delta(n))), \ldots, T'(b_k'(n+\delta(n)))) & \text{(ev.)} \\ & \leq & d(n) + G(n, T'(b_1(n)+\delta(n)), \ldots, T'(b_k(n)+\delta(n))) & \text{(by (54))} \\ & \leq & d(n) + G(n, CT(b_1(n)), \ldots, CT(b_k(n))) & \text{(Real Induction Hypothesis; } G \text{ is increasing)} \\ & \leq & C[d(n) + \cdot G(n, T(b_1'(n)), \ldots, T(b_k'(n)))] & \text{(by sublinearity of } G\text{; } C \geq 1) \\ & = & CT(n) & \text{(ev.)} \end{array}$$

  As usual, it is a bit tedious to set up the "eventual" conditions to justify the Real Induction Hypothesis; it is through this process that the reader will discover how to choose the constant C. However, we will omit the details as it is similar to the proof of Theorem 12 above. **Q.E.D.**

- We may apply Theorem 14 to show the robustness of our Multiterm Master Theorem (Theorem B). It frequently happens that multiterm recurrences involve ceiling or flour functions and small additive constants in the recursive calls, say

  $$T'(n) = d(n) + \sum_{i=1}^{k} a_i T'(\lceil n/b_i \rceil + e).$$

  Then using the perturbation function $\delta(n) = (1+e)c/(c-1)$, we conclude that $T'(n+\delta(n)) \leq C \cdot T(n)$. Hence $T'(n) = \mathcal{O}(T(n))$, i.e., up to Θ-order, the original bounds are valid.

- Our formulation of $\delta(n)$ leads to some surprisingly large perturbations. Leighton [22] considered perturbations of the form
$$\delta(n) = \mathcal{O}(\log^{\alpha} n), \qquad (\alpha < -1).$$
Moreover, it is suggested that if we use a larger perturbation such as $\alpha = -1/2$, we obtain qualitatively different results. More precisely, Leighton suggested that if we use
$$b'(n) = \frac{n}{2} + \frac{n}{\sqrt{\lg n}} \tag{56}$$
instead of $b(n) = n/2$ in the mergesort recurrence
$$T(n) = n + 2T(b(n))$$
then the solution $T'(n)$ would not be Θ-order of $T(n)$. We contradict this by showing that $b'(n)$ is a $\delta(n)$-perturbation of $b(n)$ if we define
$$\delta(n) := \frac{3n}{\sqrt{\lg n}}.$$
It is justified in the same way as (55). Therefore we conclude $T'(n) = O(T(n)) = O(n \log n)$.

REMARKS:
1. Since our solutions are insensitive to initial conditions, we can exploit the ability to choose initial conditions to find *exact solutions* in very simple forms. E.g., the recurrence $T(n) = 2T(n/2) + n$ has the elegant solution of $T(n) = n \lg n$ if we choose $T(n) = n \lg n$ for all $1 \le n < 2$, and use the recurrence for $n \ge 2$.
2. Our treatment could be extended to treat recurrence inequalities, e.g., $T(n) \ge d(n) + G(n,\ T(g_1(n)), \ldots, T(g_k(n)))$. This is useful when we are unable to prove a Θ-bound, but may settle for some upper and lower bound on $T(n)$ i.e., $\mathcal{O}$- and Ω-bounds.

8 Conclusion

As noted by Karp [18], we seek "cookbook theorems" or easy-to-use solutions of recurrences. That is the appeal of the standard master theorem. Theorem B certainly has the same cookbook qualities of the master theorem. Theorem A is still cookbook, though the generality of its class of driving functions calls for some deciphering of the notations. Clearly further generalizations of Theorems A and B are possible, some of which are suggested by the cited references; the challenge is to make them more cookbook.

Cormen et al [9, p. 90] noted that some generalized master theorems are not easy use. One reason is that some bounds are left in the form of integrals. But another reason may be be the presence of tedious details that ought to be factored out, and relegated to general robustness properties of solutions. As illustrated in this paper, we can go a long way to distill the essence of such recurrences when we exploit Θ-robustness and embrace real recurrences whole-heartedly. A tool that ought to be used more widely in this context are principles of real induction. The focus on Θ-robust results ought also lead to much greater use of elementary techniques.

We feel these ideas to be pedagogically sound. For instance, the elementary summation rules in Section 2 are easily taught in an introductory class in algorithms, eschewing the calculus that are often used to prove such bounds. Indeed, our perspectives have been developed from classroom experience.

Acknowledgments

The robustness section has been greatly improved thanks to the feedback from Charles Leiserson and William Kuszmaul who pointed an error in an earlier proof of Theorem 14.

References

[1] M. Abramovitz and I. A. Stegun. *Handbook of Mahtmeatical Functions with Forulas, Graphs, and Mathematical Tables.* Dover, New York, 9th dover printing, 10th gpo printing edition, 1964.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Massachusetts, 1974.

[3] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms.* Addison-Wesley, Reading, Massachusetts, 1983.

[4] A. V. Aho and N. J. A. Sloane. Some doubly exponential sequences. *Fibonacci Quarterly*, 11:429–437, 1973.

[5] M. Akra and L. Bazzi. On the solution of linear recurrences. *Computational Optimizations and Applications*, 10(2):195–210, 1998.

[6] J. L. Bentley, D. Haken, and J. B. Saxe. A general method for solving divide-and-conquer recurrences. *ACM SIGACT News*, 12(3):36–44, 1980.

[7] M. Blum, R. Floyd, V. Pratt, R. Rivest, and R. Tarjan. Time bounds for selection. *J. Computer and System Sciences*, 7(4):448–461, 1973.

[8] G. Brassard and P. Bratley. *Fundamentals of Algorithms.* Prentice-Hall, Inc, Englewood Cliffs, 1996.

[9] T. H. Corman, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* The MIT Press and McGraw-Hill Book Company, Cambridge, Massachusetts and New York, second edition, 2001.

[10] G. Dowek. Preliminary investigations on induction over real numbers, 2003. Manuscript, http://www.lix.polytechnique.fr/ dowek/publi.html.

[11] H. Edelsbrunner and E. Welzl. Halfplanar range search in linear space and $O(n^{0.695})$ query time. *Info. Processing Letters*, 23:289–293, 1986.

[12] M. H. Escardó and T. Streicher. Induction and recursion on the partial real line with applications to Real PCF. *Theoretical Computer Science*, 210(1):121–157, 1999.

[13] G. H. Gonnet. *Handbook of Algorithms and Data Structures.* International Computer Science Series. Addison-Wesley Publishing Company, London, 1984.

[14] É. Goursat. *A Course in Mathematical Analysis*, volume 1. Ginn & Co., Boston, 1904. Trans. by Earle Raymod Hedrick. Available from Google books.

[15] D. H. Greene and D. E. Knuth. *Mathematics for the Analysis of Algorithms.* Birkhäuser, 2nd edition, 1982.

[16] G. H. Hardy. *Orders of Infinity.* Cambridge Tracts in Mathematics and Mathematical Physics, No. 12. Reprinted by Hafner Pub. Co., New York. Cambridge University Press, 1910.

[17] M. Kao. Multiple-size divide-and-conquer recurrences. *SIGACT News*, 28(2):67–69, June 1997. Also: Proc. 1966 Intl. Conf. on Algorithms, National Sun Yat-Sen University, Kaohsiung, Taiwan, pp. 159–161.

[18] R. M. Karp. Probabilistic recurrence relations. *J. ACM*, 41(6):1136–1150, 1994.

[19] J. Kleinberg and É. Tardos. *Algorithm Design.* Addison Wesley, Boston, 2005.

[20] D. E. Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley, Boston, 1972.

[21] D. E. Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 1-3. Addison-Wesley, Boston, 1973-1981.

[22] T. Leighton. Notes on better master theorems for divide-and-conquer recurrences, 1996. Class notes.

[23] H. R. Lewis and L. Denenberg. *Data Structures and their Algorithms.* Harper Collins Publishers, New York, 1991.

[24] G. S. Lueker. Some techniques for solving recurrences. *Computing Surveys*, 12(4):419–436, 1980.

[25] B. P. Mahony and I. J. Hayes. Using continuous real functions to model timed histories. In *Proc. 6th Australian Software Engineering Conf. (ASWEC91)*, pages 257–270. Australian Comp. Soc., 1991.

[26] U. Manber. *Introduction to Algorithms: A Creative Approach.* Addison-Wesley, Reading, Mass., 1989.

[27] K. Mehlhorn. *Datastructures and Algorithms*, volume Volumes 1-3. Springer-Verlag, Berlin, 1984.

[28] J. Paul Walton Purdom and C. A. Brown. *The Analysis of Algorithms.* Holt, Rinehart and Winston, New York, 1985.

[29] S. Roura. *Divide-and-Conquer Algorithms and Data Structures.* PhD thesis, Universitat Politècnica de Catalunya, Departament de Llenguatges i Systemes Informàtics, Barcelona, Catalonia, Spain, 1997.

[30] S. Roura. An improved master theorem for divide-and-conquer recurrences. In *Proc. 24th Int. Coll. on Automata, Lang. and Prog. (ICALP'97)*, pages 449–459. Springer-Verlag, 1997.

[31] S. Roura. Improved master theorems for divide-and-conquer recurrences. *J. ACM*, 48(2):170–205, 2001.

[32] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.

[33] R. M. Verma. A general method and a master theorem for divide-and-conquer recurrences with applications. *J. Algorithms*, 16:67–79, 1994.

[34] X. Wang and Q. Fu. A frame for general divide-and-conquer recurrences. *Info. Processing Letters*, 59:45–51, 1996.

[35] B. Weide. Survey of analysis techniques for discrete algorithms. *Computing Surveys*, 9(4):291–313, 1977.

[36] C. K. Yap. Theory of real computation according to EGC. In P. Hertling, C. Hoffmann, W. Luther, and N.Revol, editors, *Reliable Implementation of Real Number Algorithms: Theory and Practice*, number 5045 in LNCS, pages 193–237. Springer, 2008.

[37] C. K. Yap. A real elementary approach to the master recurrence and generalizations. In M. Ogihara and J. Tarui, editors, *8th Conf. on Theory and Applic. of Models of Computation (TAMC)*, pages 14–26. Springer, May 2011. LNCS No. 6648. May 23-25, Tokyo, Japan. The full paper may be obtained from the author's website.