NOTES ON ASYMPTOTIC ANALYSIS AND RECURSIONS

PETER BRASS

VERSION OF 12.05.2009

DEDICATED
TO THE MEMORY
OF MY MOTHER
GISELA BRASS
25.4.1935–26.9.2008

Contents

0. Preface

The O-notation and its associates $\Omega, \Theta, o, \omega$ are a central tool in subjects related to discrete mathematics or algorithms, as well as many other parts of mathematics. Its textbook treatment is, however, limited to a chapter at the beginning of each algorithms textbook, with some standard definitions and exercises. But the use, even in the same textbook, tends not to agree with the definition, and in the more technical literature the handling of O-expressions is, on second thought, amazingly messy. We are just used to this very convenient tool, and assume that anyway everyone knows what we mean.

This book aims to summarise and make concise the theory of O-notation, and its use in the context of recursions, as needed by a graduate student of computer science interested in algorithms analysis. This is also a resource book for any professor who needs material and exercises on O-notation beyond the material in the standard algorithms textbooks. The book is not intended as stand-alone textbook; the subject does not justify a full semester course, but it does perhaps deserve more than the chapter of standard material and questions included in any algorithms textbook. I believe there is currently no book covering this selection of material, although asymptotic analysis and O-notation has, of course,been discussed in a number of mathematics textbooks, in view of mathematical applications.

Every book has its history; this book was started on october 23rd 2008, after in september my second book had appeared (Advanced Data Structures), my son was born, and my mother died. This book is dedicated to her memory. I thank my wife Mijin for all her love and support.

This book project has not been supported by any funding agency, neither have my two previous books. It is not supported by my college, the City College of New York, or my department either; my teaching load is now three courses per semester, and the release time for unsponsored research has been abolished. I wish to put on record the general hostile attitude to scientific work not supported by external funding.

1. Definition of Growth Classes

In this chapter the notations O, Ω, Θ, o, ω are introduced, and derived notations are discussed.

The O-notation is a tool for the simple representation of growth rates of functions. It is a tool that has been found extremely convenient by many people in different scientific subcommunities; unfortunately, the convenience of use led to much carelessness in the notation, which might be permissible in the hands of the expert, but is dangerous and confusing for the beginner.

When working with O-expressions, it is most important to first get clear the meaning of the statement. At least initially, the first step is always to translate the O-statement into an inequality.

1.1 Definition and Notation

Any O-statement compares the growth rate of two functions under a limit, and translates into and inequality between the function values, under some scaling factor, for any function argument sufficiently far along in the limit process. Usually, the limit is for the function argument to ∞.

As an example, we give the classical definitions. If f and g are *nonnegative* functions defined on the integers, we say

- $f(n)$ is $O(g(n))$ if there is a constant C such that for all n sufficiently large holds $f(n) \leq Cg(n)$.
- $f(n)$ is $o(g(n))$ if for each constant $C > 0$ and all n sufficiently large holds $f(n) \leq Cg(n)$.
- $f(n)$ is $\Omega(g(n))$ if there is a constant $C > 0$ such that for all n sufficiently large holds $f(n) \geq Cg(n)$.
- $f(n)$ is $\omega(g(n))$ if for each constant C and all n sufficiently large holds $f(n) \geq Cg(n)$.
- $f(n)$ is $\Theta(g(n))$ if there are constants $C_1, C_2 > 0$ such that for all n sufficiently large holds $C_1 g(n) \leq f(n) \leq C_2 g(n)$.

Thus O, o describe upper bounds, Ω, ω describe lower bounds, and Θ simultaneous upper and lower bounds.

For a given function g, we can interpret $O(g)$ as the class of all functions which satisfy that inequality; then we should write $f \in O(g)$, which is consistent, but notationally inconvenient. Instead, one writes $f(n) = O(g(n))$, but this is not an equality, but an abbreviation for the definition above:

$$\exists C, \exists n_0 \forall n > n_0 \colon f(n) \leq Cg(n).$$

Another notation for the same, making the upper bound aspect more visible, is $f(n) \leq O(g(n))$.

1.2 Extension of Standard Notation

With this standard definition, we have assigned a meaning only to expressions of the form $f(n) = O(g(n))$, not to the numerous other forms of expressions in which O is used, e.g., $f(n) = g(n) + O(h(n))$. There are at least two natural ways to extend the standard notation, unfortunately both with their separate unexpected details, but equivalent for most applications in this book. If we want to distinguish them, we use an $=$ or an $\leq$ sign in the O-expression, as in $f(n) = g(n) + O(h(n))$ versus $f(n) \leq g(n) + O(h(n))$. In either case, we always assume that the comparison function inside the $O(\cdot)$ is nonnegative.

A general interpretation for such terms would be to transform them until they have the form we originally defined, e.g., $f(n) - g(n) = O(h(n))$, and then interpret them as we defined in the previous section. But we defined this only for nonnegative functions, so we have to insure that after the transformation, the function to the left of the $=$-sign is nonnegative. Also this requires that functions can be inverted, so that the required transformation can be performed.

This suggests a more general definition: $f(n) = g(n, O(h(n)))$ is an abbreviation for

$$\exists C \exists n_0 \forall n > n_0 \exists m < Ch(n) \colon f(n) = g(n, m).$$

If for all $n > n_0$ the function $g(n, m)$ is monotone increasing in m, and $f(n) \geq g(n, 0)$, this is the same as $f(n) \leq g(n, Ch(n))$, But if g is not monotone increasing in m, this is not an upper bound anymore. In a statement like $f(n) = g(n) - O(h(n))$, this becomes a lower bound for f: for some $C > 0$ we have $f(n) \geq g(n) - Ch(n)$. But even when the function g is monotone, this violates our expectation for an upper bound, since with this interpretation, $(n+1)^2 = n^2 + O(n)$ holds, but $(n-1)^2 = n^2 + O(n)$ fails.

An alternative interpretation is to keep the inequality as central concept, and define $f(n) \leq g(n, O(h(n)))$ as an abbreviation for

$$\exists C \exists n_0 \forall n > n_0 \exists m < Ch(n) \colon f(n) \leq g(n, m).$$

In that case, we can increase the right-hand side of an inequality, $(n-1)^2 \leq n^2 + O(n)$ and $(n+1)^2 \leq 2n^2 + O(n)$ hold, but any statement in which the O-term carries a negative coefficient becomes meaningless: $f(n) \leq g(n) - O(h(n))$ is the same as $f(n) \leq g(n)$.

Both definitions can be further extended to multiple Os:

$$f(n) = g(n, O(h_1(n)), \dots, O(h_k(n)))$$

is an abbreviation for either

$$\exists C \exists n_0 \forall n > n_0 \exists m_1 < Ch_1(n), \dots m_k < Ch_k(n) \colon f(n) = g(n, m_1, \dots, m_k),$$

or the same with a $\leq$ instead of $=$. Both definitions reduce for expressions $f(n) = O(g(n))$ to the standard definition, and for $f(n) = g(n, O(h(n)))$ to $f(n) \leq g(n, Ch(n))$ for some C, if $f(n) \geq g(n, 0)$ and $g(n, m)$ is monotone increasing in m. If we want to use specifically one or the other version, we will indicate by using $=$ or $\leq$ in the expression.

Corresponding definitions can be made for o, Ω, ω, Θ. Even more complicated forms of expressions occur in literature, e.g., nested Os; in each case one needs to think about the actual meaning of the expression.

In all previous definitions, the O was constrained to occur only on the right side of the $=$ sign, but it frequently is used in chains of 'equations'. We define that $f_1(n, O(g_1(n))) = f_2(n, O(g_2(n)))$ is an abbreviation for

$$\forall C_1 \exists C_2 \exists n_0 \forall n > n_0 \forall m_1 < C_1 g_1(n) \exists m_2 < C_2 g_2(n) \colon f_1(n, m_1) = f_2(n, m_2),$$

or the same with $\leq$ instead of $=$. So for any function h with $h(n) = O(g_1(n))$, the previously defined statement $f_1(n, h(n)) = f_2(n, O(g_2(n)))$ should hold. This asymmetry, with $\forall$ on the left and $\exists$ on the right allows us to chain these O-'equations'.

In parametrized families of such O-statements, like $f(n+1) = f(n) + O(n)$, we have to use the same multiplicative constant in each O-statement of this family, otherwise they become meaningless.

1.3 Nonstandard Variants

Any O-statement involves a limit process with respect to a variable, so this notation should be used most carefully, or not at all, when there are several variables present. Almost never is a uniform bound for all multi-variable assignments intended; instead usually one variable is kept fixed, and the limit process performed for the other variable. This must be made clear when using such a multi-variable notation. As example, the statement "for fixed k holds $f(n, k) = O(g(n, k))$" must be interpreted as

$$\forall k \exists C(k), n_o(k) \forall n > n_0(k) \colon f(n, k) \leq C(k) g(n, k).$$

In our original definition, we restricted our O-statements to nonnegative functions, and used always O, o for upper bounds and Ω, ω for lower bounds. A different interpretation is widespread in some parts of mathematics, which uses O, o for bounds on the absolute value. In that case, the function need not be nonnegative (although the comparison function still must be), and there are no Ω, ω, Θ. This variant is normally not used in the algorithms or discrete mathematics communities, but it is standard in number theory, and the more analytical-oriented parts of mathematics, e.g., numerical analysis.

We wil use the symbols $O_{|\cdot|}$ and $0_{|\cdot|}$ when we intend this interpretation. In this interpretation, $f(n) = O_{|\cdot|}(g(n))$ is an abbreviation for

$$\exists C, n_0 \forall n > n_0 \colon |f(n)| \leq Cg(n),$$

and $f(n) = o_{|\cdot|}(g(n))$ for

$$\forall C \exists n_0 \forall n > n_0 \colon |f(n)| \leq Cg(n),$$

which then can again be extended to other expressions by transforming them like normal equalities into this form, and then interpreting them as O-expressions. So $f(n) = g(n, O_{|\cdot|}(h(n)))$ means

$$\exists C, n_0 \forall n > n_0 \exists m \colon |m| < Ch(n) \wedge f(n) = g(n, m).$$

1.4 Exercise Problems

EXERCISE 1.1: What is the meaning of $f(n) = n^{O(1)}$?

EXERCISE 1.2: What is meant by the following form of the Stirling formula $n! = n^n e^{-n} \sqrt{2\pi n}(1 + o_{|\cdot|}(1))$?

EXERCISE 1.3: What is meant by $n^{O(1)} \leq O\left((\log n)^{\log n}\right)$?

2. Connection to Limits and the Comparison of Explicitely Given Functions

In this chapter the connection of statements $f(n) = O(g(n))$ to the limit $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ is introduced.

2.1 Results

If f and g are two explicitely given functions, then the comparison of their growth rates for $n \to \infty$ can be performed by evaluating the limit $\lim_{n\to\infty} \frac{f(n)}{g(n)}$, if it exists.

THEOREM 2.1: If f and g are two nonnegative functions, then

- if $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, then $f(n) = \omega(g(n))$, $f(n) = \Omega(g(n))$, but $f(n) \neq O(g(n))$, $f(n) \neq o(g(n))$, $f(n) \neq \Theta(g(n))$
- if $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} = C \in]0, \infty[$, then $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, $f(n) = \Theta(g(n))$ but $f(n) \neq \omega(g(n))$, $f(n) \neq o(g(n))$.
- if $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} = 0$, then $f(n) = O(g(n))$, $f(n) = o(g(n))$, but $f(n) \neq \Theta(g(n))$, $f(n) \neq \Omega(g(n))$, $f(n) \neq \omega(g(n))$.

PROOF: All statements reduce directly to the definition: if $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, then for each C there is an $n_0(C)$ with $\frac{f(n)}{g(n)} \geq C$ for $n \geq n_0(C)$, which is the definition of $f(n) = \omega(g(n))$, and the negation of the definition of $f(n) = O(g(n))$, which therefore cannot hold. For any specific C, this gives $f(n) = \Omega(g(n))$. All other statements are similar.

If the limit does not exist, we can express the properties by lim sup and lim inf.

THEOREM 2.2:

- $f(n) = \omega(g(n))$ if and only if $\liminf\limits_{n\to\infty} \frac{f(n)}{g(n)} = \infty$.
- $f(n) = \Omega(g(n))$ if and only if $\liminf\limits_{n\to\infty} \frac{f(n)}{g(n)} > 0$.
- $f(n) = O(g(n))$ if and only if $\limsup\limits_{n\to\infty} \frac{f(n)}{g(n)} < \infty$.
- $f(n) = o(g(n))$ if and only if $\limsup\limits_{n\to\infty} \frac{f(n)}{g(n)} = 0$.
- $f(n) = \Theta(g(n))$ if and only if $\liminf\limits_{n\to\infty} \frac{f(n)}{g(n)} > 0$ and $\limsup\limits_{n\to\infty} \frac{f(n)}{g(n)} < \infty$.

So the only case when none of these relations holds is when $\liminf\limits_{n\to\infty} \frac{f(n)}{g(n)} = 0$ and $\limsup\limits_{n\to\infty} \frac{f(n)}{g(n)} = \infty$.

PROOF: Again each statement reduces directly to the definition.

In the lim sup and lim inf-form, it is especially easy to see the symmetry of these operations: $f(n) = O(g(n))$ is the same as $g(n) = \Omega(f(n))$, $f(n) = o(g(n))$ is the same as $g(n) = \omega(f(n))$, and $f(n) = \Theta(g(n))$ is the same as $g(n) = \Theta(f(n))$.

In the previous chapter, other types of O-statements beyond the simple $f(n) = O(g(n))$ were defined, but these do not have a simple equivalent limit version.

2.2 Exercise Problems

Exercise 2.1: Is $n + \log n = O(n)$?

Exercise 2.2: Is $2^{n+\log n} = O(2^n)$?

Exercise 2.3: Is $2^{n/2} = \Omega(2^n)$?

Exercise 2.4: Is $2^n - 2^{n-1} - 2^{n-2} = \Omega(2^n)$?

Exercise 2.5: Bring the following functions in increasing order according to their growth rate: $n^2 2^n$, $\log(n^2)$, $\frac{n}{\log n}$, $\sqrt{n}$, $(\log n)^2$, $(\sqrt{n})^{\log n}$, $2^{\log n}$, 2^{2n}, $n2^{\sqrt{n}}$, $n!$.

Exercise 2.6: Is $f(n) = (\log n)^{\log\log n} = O(n)$?

Exercise 2.7: Is $f(n) = (\log n)^{\log n} = O(n)$?

Exercise 2.8: Is $n^{\frac{1}{\log n}} = \Omega(\log n)$?

3. General Properties and Counterexamples

In this chapter we discuss some general O-statements, some of which are false

3.1 Results

The importance and usefulness of the O-notation and its relatives is because they allow to compare and abstract growth rates of functions. The growth rate comparison estabishes a partial order on the nonnegative functions, or at least on equivalence classes of functions.

THEOREM 3.1: The relations $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, $f(n) = \Theta(g(n))$, $f(n) = o(g(n))$, $f(n) = \omega(g(n))$ are all transitive.
The relation $f(n) = \Theta(g(n))$ is an equivalence relation, and the relations o and ω define inverse partial orders on the Θ-equivalence classes.

PROOF: These statements follow immediately from the definition, e.g., if $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) \leq C_1 g(n)$ and $g(n) \leq C_2 h(n)$ for n sufficiently large, so $f(n) \leq C_1 C_2 h(n)$.

This partial order is not a linear order, some functions are not comparable under the O-relation.

EXAMPLE: The functions f, g defined by $f(n) = (2k+1)!$ for $n = 2k, 2k+1$, and $g(n) = (2k)!$ for $n = 2k-1, 2k$ satisfy $\frac{f(2k)}{g(2k)} = 2k+1$ and $\frac{g(2k+1)}{f(2k+1)} = 2k+2$, so neither $f(n) = O(g(n))$ nor $g(n) = O(f(n))$ holds.

Many further simple properties of O-statements follow directly from the definition, and can be immediately seen in the equivalent lim sup- or lim inf-version. We list some below.

THEOREM 3.2: Let $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$, then
- $f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$,
- $f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n)))$,
- $f_1(n) f_2(n) = O(g_1(n) g_2(n))$.

The same statements hold for $o, \omega, \Omega, \Theta$.

But for other arithmetic operations, this simple compatibility with $O(\cdot)$, does not work. $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ does not imply either $f_1(n) - f_2(n) = O(g_1(n) - g_2(n))$ nor $\frac{f_1(n)}{f_2(n)} = O(\frac{g_1(n)}{g_2(n)})$. For subtraction, there is no relation among the differences.

EXAMPLE: The functions $f_1(n) = 2n^2$, $g_1(n) = n^2 + n$, $f_2(n) = g_2(n) = n^2$ satisfy $f_i(n) = \Theta(g_1 in))$, $i = 1, 2$, and $f_1(n) - f_2(n)$ and $g_1(n) - g_2(n)$ are nonnegative, but $f_1(n) - f_2(n) = n^2$ and $g_1(n) - g_2(n) = n$.

For division, there is a relation if we mix lower and upper bounds.

Theorem 3.3: If $f_1(n) = O(g_1(n))$ and $f_2(n) = \Omega(g_2(n))$, then $\frac{f_1(n)}{f_2(n)} = O(\frac{g_1(n)}{g_2(n)})$.
If $f_1(n) = \Theta(g_1(n))$ and $f_2(n) = \Theta(g_2(n))$, then $\frac{f_1(n)}{f_2(n)} = \Theta(\frac{g_1(n)}{g_2(n)})$.

If the bounds are in the same direction, the quotients are unrelated.

Example: The functions $f_1(n) = n$, $g_1(n) = n^3$, $f_2(n) = g_2(n) = n^2$ satisfy $f_i(n) = O(g_1 in))$, $i = 1, 2$, but $\frac{f_1(n)}{f_2(n)} = \frac{1}{n}$ and $\frac{g_1(n)}{g_2(n)} = n$.

3.2 Exercise Problems

Exercise 3.1: Is it true that for a monotone increasing function f holds $f(n) = O(f(n-1)))$?

Exercise 3.2: Is it true that for monotone increasing functions f, g_1, g_2 holds if $f(n) = O(g_1(n) + g_2(n))$ then $f(n) = O(g_1(n))$ or $f(n) = O(g_2(n))$?

Exercise 3.3: Is it true that for monotone increasing functions f_1, f_2, g holds if $f_1(n) = O(f_2(n))$ then $f_1(g(n)) = O(f_2(g(n)))$?

Exercise 3.4: Is it true that for monotone increasing functions f, g_1, g_2 holds if $g_1(n) = O(g_2(n))$ then $f(g_1(n)) = O((g_2(n)))$?

Exercise 3.5: Is it true that for monotone increasing functions f, g holds if $f(n) = O(g(n))$ then $f(f(n)) = O(g(g(n)))$?

Exercise 3.6: Is it true that for monotone increasing functions f, g holds if $f(n) = O(g(n))$ then $f^{-1}(n) = \Omega(g^{-1}(n))$?

Exercise 3.7: Show that if f satisfies $f(n) \geq n^2 - f(n-1)$, and f is monotone increasing, then $f(n) = \Omega(n^2)$, and give an example of a function f that satisfies the same inequality, but is not monotone increasing and not $\Omega(n^2)$.

4. Polynomials

In this chapter we discuss recursions and O-statements involving polynomials, especially that the partial sums of a polynomial are again a polynomial, of degree one higher.

4.1 Results

Most natural operations, when applied to polynomials, give again a polyomial. If we know a function is a polynomial, and also know the degree, we can determine the function from its staring values by solving a system of linear equations for the coefficients of the polynomial.

THEOREM 4.1: If $f(n)$ is a polynomial of degree k and $g(n) = a_0 f(n) + a_1 f(n-1) + a_2 f(n-2) + \cdots + a_l f(n-l)$, then

- $g(n)$ is also a polynomial of degree k,
- if $a_0 + \cdots a_l = 0$, then $g(n)$ is a polynomial of degree $k-1$.

PROOF: If $f(n)$ is a polynomial of degree k, then so is $f(n-1), f(n-2), \ldots$. So any linear combination of these functions is again a polynomial. These functions have all the same leading coefficient; so a linear combination whose coefficients sum to zero will make the leading term vanish.

Any recursion of the form $f(n) = f(n-1) + g(n)$ with $g(n)$ a polynomial of degree k describes itself a polynomial of degree $k+1$:

THEOREM 4.2: If $g(n)$ is a polynomial of degree k, then $f(n) = \sum\limits_{i=1}^{n} g(i)$ is a polynomial of degree $k+1$.

PROOF: We prove this by a dimension argument. We cnsider the space of all polynomials of degree k, which is a linear space of dimension $k+1$, and on this linear space the difference operator, which maps the polynomial p to the function $p(n) - p(n+1)$, which by theorem 4.1 is a polynomial of degree $k-1$. Two polynomials p_1, p_2 that are mapped on the same difference function $q(n) = p_1(n) - p_1(n-1) = p_2(n) - p_2(n-1)$ differ only by a constant, since $p_1(n) = p_1(1)+q(1)+q(2)+\ldots+q(n)$ and $p_2(n) = p_2(1)+q(1)+q(2)+\ldots+q(n)$. So the kernel of this linear map from the $k+1$-dimensional space of polynomials of degree k to the k-dimensional space of polynomials of degree $k-1$ is one-dimensional, so the linear map is surjective. Thus there is a polynomial p_k^* of degree k with $p_k^*(n) - p_k^*(n-1) = n^{k-1}$. So $\sum\limits_{i=1}^{n} i^{k-1} = p_k^*(n)$, for any k. If $g(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_0$, then $\sum\limits_{i=1}^{n} g(i) = a_k p_{k+1}^*(n) + a_{k-1} p_k^*(n) + \cdots + a_0 p_1^*(n)$.

4.2 Exercise Problems

Exercise 4.1: If f is a polynomial of degree k, is then sum $\sum_{i=n}^{2n} f(i)$ also a polynomial? If yes, of what degree?

Exercise 4.2: If f is a polynomial of degree k, is then sum $\sum_{i=1}^{n^2} f(i)$ also a polynomial? If yes, of what degree?

Exercise 4.3: The function f is given by $f(n) = f(n-1) + 2n$ and $f(0) = 0$. Determine f.

5. Functions of Polynomial Growth

In this chapter we discuss some general properties of functions whose growth rate is bounded by a polynomial.

5.1 Results

We say a function is of *polynomial growth* if it is monotone increasing for sufficiently large n, and satisfies $f(2n) = O(f(n))$. Every polynomial is of polynomial growth, but so are many other fuctions, like $\sqrt{n}$, $\log n$ and so on.

THEOREM 5.1: If f is a function of polynomial growth, then $f(n) = O(n^k)$ for some k.

PROOF: Let $f(2n) \leq Cf(n)$ for $n \geq n_0$, then $f(n) \leq f(2^{\lceil \log n \rceil} n_0) \leq C^{\lceil \log n \rceil} f(n_0) \leq 2f(n_0) 2^{\log C \log n} = 2f(n_0) n^{\log C}$.

The opposite direction does not hold; it is possible that a function is bounded by a polynomial without being itself of polynomial growth.

EXAMPLE: The function $f(n)$ which is defined by $f(n) = k!$ for $k! \leq n < (k+1)!$ satisfies $f(n) \leq n$, but for $n = (k+1)! - 1$ we have $f(2n) = (k+1)f(n)$, so $f(2n) \neq O(f(n))$.

It is not even sufficient for polynomial growth that the increments of a function are bounded.

EXAMPLE: The function $f(n)$ defined by $f(n) = f(n-1)$ for $(2k-1)! < n \leq (2k)!$ and $f(n) = f(n-1)+1$ for $(2k)! < n \leq (2k+1)!$ has $f((2k)!) = (2k-1)! - (2k-2)! + (2k-3)! - (2k-4)! + \cdots \leq (2k-1)!$ but $f(2(2k)!) = f((2k)!) + (2k)! > (2k)!$, so $f(2(2k)!) > (2k)f((2k)!)$, and again $f(2n) \neq O(f(n))$.

If the function is bounded from both sides by a polynomial, or any function of polynomial growth, then it is itself of polynomial growth.

THEOREM 5.2: If $f(n) = \Theta(g(n))$, and g is a function of polynomial growth, then f is also of polynomial growth.

PROOF: Let $C_1 g(n) \leq f(n) \leq C_2 g(n)$, and $g(2n) \leq C_3 g(n)$ for $n > n_0$, then $f(2n) \leq C_2 g(2n) \leq C_2 C_3 g(n) \leq \frac{C_2 C_3}{C_1} f(n)$.

The factor two in the definition could be replaced by any other factor.

THEOREM 5.3: If f is a monotone increasing function
- and $f(2n) = O(f(n))$, then for any fixed a we have $f(an) = O(f(n))$.
- and $f(an) = O(f(n))$ for some $a > 1$, then $f(2n) = O(f(n))$.

Proof: Let $f(2n) \le Cf(n)$ for $n \ge n_0$, then $f(4n) \le C^2 f(n)$ and $f(an) \le f(2^{\lceil \log a \rceil} n) \le C^{\lceil \log a \rceil} f(n)$. In the other direction, let $f(an) \le Cf(n)$ for $n \ge n_0$, then $f(2n) \le f(a^{\lceil \log_a 2 \rceil} n) \le Ca^{\lceil \log_a 2 \rceil} f(n)$.

Many combinations of functions of polynomial growth are again of polynomial growth.

Theorem 5.4: If f and g are functions of polynomial growth, then $f+g$, fg, and $f(g(\cdot))$ are again functions of polynomial growth.

The property of polynomial growth is important since allows us to exchange functions in some composition of functions.

Theorem 5.5: If f_1 is a function of polynomial growth, and $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$, then $f_1(f_2(n)) = O(g_1(g_2(n)))$.

Proof: Let $f_1(n) \le C_1 g_1(n)$ and $f_2(n) \le C_2 g_2(n)$ then $f_1(f_2(n)) \le f_1(C_2 g_2(n)) \le C_3 f_1(g_2(n)) \le C_1 C_3 g_1(g_2(n))$.

In general, without the assumption of polynomial growth, this relation fails to hold: if $f_1(n) = g_1(n) = 2^n$ and $f_2(n) = 2n$, $g_2(n) = n$, we have $f_1(f_2(n)) = 4^n$ and $g_1(g_2(n)) = 2^n$.

Polynomial growth is not sufficient to make ant two functions comparable in their growth rate.

Example: The function $f(n)$ defined by $f(n) = n^2/k!$ for $(k!)^2 \le n \le (k+1)(k!)^2$ and $f(n) = ((k+1)!)^2$ for $(k+1)(k!)^2 \le n \le ((k+1)!)^2$ is of polynomial growth, indeed $f(2n) \le 4f(n)$, and $\liminf_{n\to\infty} \frac{f(n)}{n} = 1$ but $\limsup_{n\to\infty} \frac{f(n)}{n} = \infty$, so although $f(n) = \Omega(n)$ we have neither $f(n) = \Theta(n)$ nor $f(n) = \omega(n)$.

5.2 Exercise Problems

Exercise 5.1: Is the function $f(n) = \binom{n}{n-3}$ of polynomial growth?

Exercise 5.2: Is the function $f(n) = 100^{\log n}$ of polynomial growth?

Exercise 5.3: Is the function $f(n) = (\log n)^{\log n}$ of polynomial growth?

Exercise 5.4: Is $f(n+1) = O(f(n))$ if f is of polynomial growth?

Exercise 5.5: Is $f(n^2) = O((f(n))^2)$ if f is of polynomial growth?

Exercise 5.6: Construct functions f, g which are both of polynomial growth, but neither $f(n) = O(g(n))$ nor $f(n) = \Omega(g(n))$ holds.

EXERCISE 5.7: Show that if $f(n)$ is a function of polynomial growth, then $g(n) = \sum_{i=1}^{n} f(i)$ is also of polynomial growth.

EXERCISE 5.8: If f and g are functions of polynomial growth, and their difference is monotone increasing, is that difference $f - g$ then also of polynomial growth?

6. Geometric Series

In this chapter we discuss the importance of the geometric series and its relatives

6.1 Results

The geometric series $1+x+x^2+\cdots+x^k = \frac{x^{k+1}-1}{x-1}$ is among the most important sums in mathematics; it is easily proved by multiplying the sum with $(x-1)$ and observing that almost everything cancels. We can take derivatives of this identity, and find $1+2x+3x^2+\cdots+kx^{k-1} = \frac{(k+1)x^k(x-1)-(x^{k+1}-1)}{(x-1)^2}$, and obtain by taking suitable linear combinations of derivatives sum formulas for all sums $\sum\limits_{i=0}^{n} p(i)x^i$ for any polynomial p, e.g., $x+2x^2+3x^3+\cdots+nx^n = x\frac{nx^{n+1}-(n+1)x^n+1}{(x-1)^2}$. Any such sum can be evaluated exactly. If we are interested only in the growth rate, then for $0<x<1$ the sum is $O(1)$, and for $x>1$ the sum is $O(n^k x^n)$ if the coefficient polynomial p is of degree k. In general holds

Theorem 6.1: If $f(n) = \sum\limits_{i=0}^{n} g(i)x^i$, then

- if $0<x<1$, and g is of polynomial growth, then $f(n) = O(1)$,
- if $x>1$, and g is monotone increasing, then $f(n) = O(g(n)x^n)$,

Proof: If $0<x<1$, and g is of polynomial growth, then $g(n) \leq Cn^k$ for some C,k and all $n>n_0$. For all $n>n_1$, we additionally have $\frac{(n+1)^k}{n^k}x < 1-\varepsilon$ for some $\varepsilon>0$. So $\sum\limits_{i=0}^{n} g(i)x^i < \sum\limits_{i=0}^{\infty} g(i)x^i < \sum\limits_{i=0}^{n_1} g(i)x^i + \sum\limits_{i=n_1+1}^{\infty} Ci^k x^i < \sum\limits_{i=0}^{n_1} g(i)x^i + Cn_1^k x_1^n \sum\limits_{i=1}^{\infty}(1-\varepsilon)^i = O(1)$.

If $x>1$, then $f(n) = \sum\limits_{i=0}^{n} g(i)x^i \leq g(n)\sum\limits_{i=1}^{n} x^i = g(n)\frac{x^{n+1}-1}{x-1}$.

Theorem 6.2: If $f(n) = \sum\limits_{i=0}^{n} g(n-i)x^i$, then

- if $0<x<1$ and g is monotone increasing, then $f(n) = O(g(n))$,
- if $x>1$ and g is of polynomial growth, then $f(n) = O(x^n)$,

Proof: We can reduce this sum to the previous type by observing that $\sum\limits_{i=0}^{n} g(n-i)x^i = x^n\sum\limits_{i=0}^{n} g(n-i)x^{i-n} = x^n\sum\limits_{i=0}^{n} g(n-i)(\frac{1}{x})^{n-i} = x^n\sum\limits_{i=0}^{n} g(i)(\frac{1}{x})^i$.

6.2 Exercise Problems

EXERCISE 6.1: How large is $f(n) = 1 + 3 + 9 + \cdots + 3^{\log n}$? The logarithm is, as always, to the base 2.

EXERCISE 6.2: If $f(n)$ satisfies $f(n) = 2f(n-1) + O(n)$, how large is $f(n)$?

7. Sums of a Given Function

In this chapter we discuss O-statements that concern the growth of partial sums of a given function.

7.1 Results

The simplest type of recursion is $f(n) = f(n-1)+g(n)$, which can be expanded to $f(n) = f(0) + \sum_{i=1}^{n} g(i)$, so, unless the sum $\sum_{i=1}^{\infty} g(i)$ is bounded, the growth rate of f is the same as that of $\sum_{i=1}^{n} g(i)$. This type of recursion occurs whenever we reduce a problem of size n to a problem of size $n-1$, and it is in most cases easy to find the growth rate of that function

The central insight that allows to determine the growth rate of many sums $f(n) = \sum_{i=1}^{n} g(i)$ is this.

THEOREM 7.1: If $g(n)$ is a nonnegative monotonically increasing function, and $f(n) = \sum_{i=1}^{n} g(i)$, then

- $f(n) = O(ng(n))$ and $f(n) = \Omega(g(n))$
- if g is of polynomial growth, then $f(n) = \Theta(ng(n))$.
- if g grows at least exponentially, so $g(n) \geq (1+\varepsilon)g(n-1)$ for some $\varepsilon > 0$, then $f(n) = \Theta(g(n))$.

PROOF: $g(n) \leq \sum_{i=1}^{n} g(i) \leq ng(n)$, since g is nonnegative and $g(n)$ is the largest summand; this proves the first statement. Also g is of polynomial growth, so $g(n) \leq Cg(\frac{n}{2})$ for some C, or $g(\frac{n}{2}) \geq \frac{1}{C}g(n)$, and $\frac{1}{2C}ng(n) \leq \frac{n}{2}g(\frac{n}{2}) \leq \sum_{i=\frac{n}{2}}^{n} g(i) \leq f(n) = \sum_{i=1}^{n} g(i) \leq ng(n)$, which proves the second statement. In the third statement, with $\alpha = \frac{1}{1+\varepsilon} < 1$ we have $g(n-1) \leq \alpha g(n)$ and $g(n) \leq \sum_{i=1}^{n} g(i) \leq g(n) \sum_{i=0}^{n-1} \alpha^i < \frac{1}{1-\alpha}g(n)$, so in that case the last summand dominates the sum.

The situation gets more difficult to analyze if the function g that we sum is not increasing, but decreasing. If $g(n)$ is a monotone decreasing nonnegative function, then $f(n) = \sum_{i=1}^{n} g(i)$ satisfies $g(1) \leq f(n) \leq ng(1)$, so $f(n) = \Omega(1)$ and $f(n) = O(n)$. Anything in this range is possible. A technique which sometimes helps to bound sums is to cut the sequence of $g(i)$ into blocks of exponentially increasing length, and bound the function on these blocks by the first or last element.

THEOREM 7.2: If $g(n)$ is a nonnegative monotonically decreasing function, and $f(n) = \sum_{i=1}^{n} g(i)$, then $g(1) + \sum_{i=1}^{k} 2^{i-1} g(2^i) < f(2^k) < \sum_{i=1}^{k} 2^i g(2^i) + g(2^n)$, so $f(2^k) = \Theta\left(\sum_{i=1}^{k} 2^i g(2^i)\right)$.

PROOF: For the lower bound, we divide the sequence $g(1), \ldots, g(2^k)$ into blocks $g(1) \mid g(2) \mid g(3), g(4) \mid g(5), \ldots, g(8) \mid \ldots$, and observe that the second block is at least $2^0 g(2)$, the third block is at least $2^1 g(4)$, and in general the $i+1$-th block is at least $2^{i-1} g(2^i)$. For the upper bound, we divide the sequence into blocks $g(1) \mid g(2), g(3) \mid g(4), \ldots g(7) \mid g(8), \ldots, g(15) \mid \ldots$, and observe that the first block is at most $2^0 g(1)$, the second block is at most $2^1 g(2)$, and in general the i-th block is at most $2^i g(2^i)$.

This method gives several classical proofs for sums.

THEOREM 7.3:

$$
\begin{aligned}
\sum_{i=1}^{n} \tfrac{1}{i} &= \Theta(\log n) \\
\sum_{i=1}^{n} \tfrac{1}{i \log i} &= \Theta(\log \log n) \\
\sum_{i=1}^{n} \tfrac{1}{i^{1+\varepsilon}} &= \Theta(1) \qquad \text{for any } \varepsilon > 0 \\
\sum_{i=1}^{n} \tfrac{1}{i^{1-\varepsilon}} &= \Theta(n^{\varepsilon}) \qquad \text{for any } 0 < \varepsilon < 1
\end{aligned}
$$

PROOF: Each of these bounds follows from theorem 7.2. For $n = 2^k$ we have

$$\sum_{i=1}^{n} \tfrac{1}{i} = \Theta\left(\sum_{i=1}^{k} 2^i \tfrac{1}{2^i}\right) = \Theta(k).$$

If we iterate this, for $n = 2^{2^k}$ we have

$$\sum_{i=1}^{n} \tfrac{1}{i \log i} = \Theta\left(\sum_{i=1}^{2^k} 2^i \tfrac{1}{2^i \log(2^i)}\right) = \Theta\left(\sum_{i=1}^{2^k} \tfrac{1}{i}\right) = \Theta\left(\sum_{j=1}^{k} 2^j \tfrac{1}{2^j}\right) = \Theta(k).$$

And for any positive ε, and $n = 2^k$ we have

$$\sum_{i=1}^{n} \tfrac{1}{i^{1+\varepsilon}} = \Theta\left(\sum_{i=1}^{k} 2^i \tfrac{1}{2^{i(1+\varepsilon)}}\right) = \Theta\left(\sum_{i=1}^{k} \tfrac{1}{2^{\varepsilon i}}\right) = \Theta(1),$$

since the last term is a geometric series, and

$$\sum_{i=1}^{n} \tfrac{1}{i^{1-\varepsilon}} = \Theta\left(\sum_{i=1}^{k} 2^i \tfrac{1}{2^{i(1-\varepsilon)}}\right) = \Theta\left(\sum_{i=1}^{k} 2^{i\varepsilon}\right) = \Theta\left((2^k)^{\varepsilon}\right),$$

again a geometric series.

If the function $g(n)$ is decreasing only slowly, then a trivial argument gives again a good bound

Theorem 7.4: If $g(n)$ is a nonnegative monotonically decreasing function, and $g(ng(n)) = \Theta(g(n))$, then $\sum_{i=1}^{n} g(i) = \Theta(ng(n))$.

Proof: $ng(n) < \sum_{i=1}^{n} g(i) = \sum_{i=1}^{ng(n)} g(i) + \sum_{i=ng(n)}^{n} g(i) < ng(n)g(1) + (n - ng(n))g(ng(n)) < n(g(n)g(1) + g(ng(n)))$.

This shows that $\sum_{i=2}^{n} \frac{1}{\log i} = \Theta\left(\frac{n}{\log n}\right)$.

Sometimes occur weighted sums of form $\sum_{i=1}^{n} h(i)g(i)$; we observe here

Theorem 7.5: If $g(n)$ is a monotonically increasing function, we have $\sum_{i=1}^{n} \frac{1}{i} g(i) = O(g(n)\log n)$; if $g(n) = \Omega(n^{\varepsilon})$ for some $\varepsilon > 0$, then $\sum_{i=1}^{n} \frac{1}{i} g(i) = O(g(n))$.

Proof: $\sum_{i=1}^{n} \frac{1}{i} g(i) \leq g(n) \sum_{i=1}^{n} \frac{1}{i} = O(g(n)\log n)$. If $g(n) = n^{\varepsilon} h(n)$ with $h(n)$ increasing for sufficiently large n, then we have $\sum_{i=1}^{n} \frac{1}{i} g(i) = \sum_{i=1}^{n} \frac{i^{\varepsilon}}{i} h(i) \leq h(n) \sum_{i=1}^{n} \frac{1}{i^{1-\varepsilon}} \leq h(n)O(i^{\varepsilon}) \leq O(g(n))$.

7.2 Exercise Problems

Exercise 7.1: How large is $\sum_{i=1}^{n} \sqrt{n+i}$?

Exercise 7.2: How large is $\sum_{i=1}^{n} \log(i \log i)$?

Exercise 7.3: How large is $(\log n)^3 + (\log n - 1)^3 + \cdots + 1^3$?

Exercise 7.4: Show that if $f(n+1) \leq f(n) + O(\log n)$, then $f(n) = O(n \log n)$.

Exercise 7.5: Show that $\sum_{i=1}^{n} 2^{(\log i)^2} = \Omega\left(\frac{n}{\log n} 2^{(\log n)^2}\right)$

Exercise 7.6: Show that $\sum_{i=1}^{n} \frac{\log i}{i} = \Theta\left((\log n)^2\right)$

8. Harmonic Series and the Approximation of Sums by Integrals

In this chapter we discuss the harmonic series, and the approximation of sums by integrals

8.1 Results

The harmonic series $H_n = 1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n}$ is another frequently occuring series; its sum is $\Theta(\log n)$. It is a good illustration of the approximation of sums by integrals: we can bound $\int\limits_1^{n+1} \frac{1}{x}dx$ from above by $1+\frac{1}{2}+\cdots+\frac{1}{n}$, dividing the interval $[1, n+1]$ into n subintervals $[i, i+1]$, and bounding the integral on each subinterval by the maximum, $\frac{1}{i}$; and we can bound the integral from below by $\frac{1}{2}+\frac{1}{3}\cdots+\frac{1}{n+1}$, with the same technique, using on each subinterval the minimum $\frac{1}{i+1}$. By this we get $H_n > \frac{1}{\log e}\log(n+1) > H_n - 1 + \frac{1}{n+1}$, so $H_n = \frac{1}{\log e}\log(n+1) - O(1)$ or $H_n = \Theta(\log n)$. The same technique can be used for many other sums $\sum\limits_{i=1}^{n} f(i)$, if we know the integral $\int\limits_1^n f(x)dx$.

Theorem 8.1: If f is a monotone decreasing nonnegative function, then $\sum\limits_{i=1}^{n} f(i) = \Theta\left(\int\limits_1^n f(x)dx\right)$.

Proof: If f is monotone decreasing and nonnegative, then

$$\sum_{i=1}^{n} f(i) \geq \sum_{i=1}^{n}\int_i^{i+1} f(x)dx = \int_1^{n+1} f(x)dx \geq \int_1^n f(x)dx \qquad \text{and}$$

$$\sum_{i=1}^{n} f(i) \leq f(1) + \sum_{i=2}^{n}\int_{i-1}^{i} f(x)dx = f(1) + \int_1^n f(x)dx \leq \frac{f(1)+f(2)}{f(2)}\int_1^n f(x)dx.$$

If we apply the same argument for increasing functions f, we also get that $\int\limits_1^n f(x)dx \leq \sum\limits_{i=1}^{n} f(i) \leq \int\limits_1^{n+1} f(x)dx$, but if f grows fast enough, it is possible that $\int\limits_1^{n+1} f(x)dx \neq O(\int\limits_1^n f(x)dx$. We need here a bound on the growth rate of f, which should be at most exponential, or other additional property of f.

Example: Let $f(x) = \lfloor x \rfloor!$, then $\int\limits_1^n f(x) = \sum\limits_{i=0}^{n-1} f(i) \neq \Theta\left(\sum\limits_{i=1}^{n} f(i)\right)$.

Example: Let f be the function that linearly interpolates $f(n) = n!$. Then $\int\limits_1^n f(x)dx = \Theta(n!)$, so $\int\limits_1^{n+1} f(x)dx \neq O(\int\limits_1^n f(x)dx$, but $\sum\limits_{i=1}^{n} f(i) = \Theta\left(\int\limits_1^n f(x)dx\right)$.

By switching the sign, we get similar examples for unbounded decreasing functions.

THEOREM 8.2: If f is monotone increasing and $f(n+1) = O(f(n))$, then $\sum\limits_{i=1}^{n} f(i) = \Theta\left(\int\limits_{1}^{n} f(x)dx\right)$.

PROOF: If $f(n+1) \leq Cf(n)$, then

$$\begin{aligned}\sum_{i=1}^{n} f(i) &\leq \int_{1}^{n+1} f(x)dx \leq \int_{1}^{n} f(x)dx + f(n+1)\\ &\leq \int_{1}^{n} f(x)dx + C^2 f(n-1) \leq (1+C^2)\int_{1}^{n} f(x)dx \qquad \text{and}\\ \sum_{i=1}^{n} f(i) &\geq \sum_{i=2}^{n} f(i) \geq \int_{1}^{n} f(x)dx.\end{aligned}$$

The relation between sums and integrals has been studied in much more detail than necessary for our applications; it is expressed by the Euler-Maclaurin sum formula, of which we give only a simple version, and without proof.

THEOREM 8.3: Let f be a function with continuous k-th derivative, and $(B_j)_{j=0}^{\infty}$ be the sequence of Bernoulli numbers, and $k \geq 1$. If $\int_a^\infty f^{(k)}(x)dx$ is bounded, then

$$\sum_{i=a}^{b-1} f(i) = \int_{a}^{b} f(x)dx + \sum_{j=0}^{k-1} \frac{B_{j+1}}{(j+1)!}\left(f^{(j)}(b) - f^{(j)}(a)\right) + O_{|\cdot|}(1).$$

The sequence of Bernoulli number is defined by $B_0 = 1$, $\sum_{k=0}^{n-1}\binom{n}{k}B_k = 0$, the first values are $1, -\frac{1}{2}, \frac{1}{6}, 0, \ldots$. This theorem provides much more than the statement that the integral and the sum are of the same order; it a parametrized family of bounds on their difference.

8.2 Exercise Problems

EXERCISE 8.1: If $f(n) = \frac{1}{n} + \frac{1}{n-1} + f(n-2)$, how large is $f(n)$?

EXERCISE 8.2: Show that $\sum\limits_{i=1}^{n} \sqrt{i} = 2\sqrt{n} - O(1)$.

9. Iteration-Counting Functions

In this chapter we discuss functions defined by iteration, like the iterated logarithm function.

9.1 Results and Examples

One of the simplest recursions is $f(n) = f(g(n))+1$; for any function $g(n)$ with $g(n) < n$ for $n \geq 2$; with $f(1) = 0$ this just counts the number of iterations of g that we need to get from n down to 1. If g is a slowly growing function, then this iteration-counting function will grow even slower; the function f can be anything between $n-1$, for $g(n) = n-1$, and constant one, for $g(n) = 1$, but most functions defined this way grow very slow.

THEOREM 9.1: If $g(n)$ is a function with $1 \leq g(n) < n$ for $n \geq 2$, and $f(n) = f(g(n)) + 1$, then $f(n) = O(g^k(n))$ for any fixed $k \geq 1$, where $g^k(n) = g(g^{k-1}(n))$ is the k-fold iterate of g.

PROOF: Since there are at most $n-1$ decreasing steps from n to 1, we have $f(n) \leq n-1$, and $f(n) = f(g(n))+1 = f(g^k(n))+k \leq g_k(n)-1+k \leq kg^k(n)$.

As simplest nontrivial example, we observe that for $g(n) = \frac{1}{2}n$ we get $f(n) = \Theta(\log n)$. More general

THEOREM 9.2: If $f(n) \leq f(g(n)) + 1$ for some function g with $g(n) \leq \alpha n$ for some $\alpha < 1$, then $f(n) = O(\log n)$.

PROOF: This follows by straightforward induction: if $f(m) \leq C \log m$ for $m < n$, with $C \log \frac{1}{\alpha} > 1$, then $f(n) \leq f(\alpha n) + 1 \leq C \log(\alpha n) + 1 = C \log \alpha + C \log n + 1 \leq C \log n$.

Another important case is $g(n) = n^\alpha$ for $0 < \alpha < 1$; we have

THEOREM 9.3: If $f(n) \leq f(g(n)) + 1$ for some function g with $g(n) \leq n^\alpha$ for some $0 < \alpha < 1$, then $f(n) = O(\log \log n)$.

PROOF: Again this is a simple inductive proof: if $f(m) \leq C \log \log m$ for $m < n$, with $C \log \frac{1}{\alpha} > 1$, then $f(n) \leq f(n^\alpha) + 1 \leq C \log \log(n^\alpha)) + 1 = C \log(\alpha \log n) + 1 \leq C \log \alpha + C \log \log n + 1 \leq C \log \log n$.

But the most important special case is a function not defined in any other way, the iterated logarithm function $\log^*(n)$, which we obtain for $g(n) = \log n$. To appreciate the slow growth of $\log^*(n)$, we look at its inverse: we have $\log^*(n) = k$ if n is less than an exponential tower of 2s of height k. We collect some simple properties

Theorem 9.4: The function $\log^*$ efined by $\log^*(n) = \log^*(\log(n)) + 1$ has the following properies:

- $2^{(\log^*(n))} = O(\log^k(n))$ for any fixed $k \geq 1$, so $2^{(\log^*(n))}$ grows slower than any fixed number of iterations of log.
- $\log^*(2^n) = \log^*(n) + 1$.

Proof: Let $\text{exptower}(l)$ denote an exponential tower of height l, so $\text{exptower}(l) = 2^{\text{exptower}(l-1)}$. We have $\log^*(n) \leq \log^{k+1}(n)$ for n sufficiently large, since for $n = \text{exptower}(l)$ we have $\log^*(n) = l$ and $\log^{k+1}(n) = \text{exptower}(l-k-1)$. From this follows $2^{(\log^*(n))} \leq 2^{(\log^{k+1}(n))} = \log^k(n)$, the first claim of the theorem. The second claim follows directly from the definition.

9.2 Exercise Problems

Exercise 9.1: How large is $\log^*(n!)$?

Exercise 9.2: Does $(\log^*(n))!$ still grow slower than all fixed iterations of log ?

Exercise 9.3: Is $2^{\log^*(n)} = O(\log^*(n))$?

10. Simple Divide-and-Conquer type Recursions

In this chapter recursions that express or bound $f(n)$ in terms of some $f(m)$, for m much smaller than n, plus some overhead, are discussed.

10.1 Results

The prototype of these recursions is $f(n) = 2f(\frac{n}{2}) + O(n)$, which occurs, e.g., in the analysis of mergesort, where one problem of size n is reduced to two problems of size $\frac{n}{2}$, with some $O(n)$ overhead; we solve it as follows.

Theorem 10.1: If $f(n)$ satisfies $f(n) \leq 2f(\frac{n}{2}) + O(n)$ for all $n > n_0$, then $f(n) \leq O(n \log n)$

Proof: We prove it by induction: let $f(n) \leq 2f(\frac{n}{2}) + Cn$ for all $n > n_0$, and let $f(m) \leq Dm \log m$ for all $m < n$, with $D > C$, then

$$\begin{aligned} f(n) &\leq 2f(\tfrac{n}{2}) + Cn \leq 2D\tfrac{n}{2} \log \tfrac{n}{2} + Cn \\ &= Dn \log n - Dn + Cn \leq Dn \log n \end{aligned}$$

This is an instance of the general type of recursions $f(n) \leq \alpha_1 f(\beta_1 n) + \cdots + \alpha_k f(\beta_k n) + g(n)$ with $0 < \beta_1, \ldots, \beta_k < 1$ and $\alpha_1, \ldots, \alpha_k \geq 0$. If we consider first the form without overhead term, $f(n) = \alpha_1 f(\beta_1 n) + \cdots + \alpha_k f(\beta_k n)$ we observe that there is a critical exponent γ with $1 = \alpha_1 \beta_1^\gamma + \cdots + \alpha_k \beta_k^\gamma$, since this expression is a monotone decreasing function of γ, wih limit 0 for $\gamma \to \infty$ and $+\infty$ for $\gamma \to -\infty$. With this critical exponent, any fuction $f(n) = Cn^\gamma$ is a solution to the recursion, since $f(n) = \alpha_1 f(\beta_1 n) + \cdots + \alpha_k f(\beta_k n) = C\alpha_1 \beta_1^\gamma n^\gamma + \cdots + C\alpha_k \beta_k^\gamma n^\gamma = Cn^\gamma$. If we now add the overhead term $g(n)$ to the recursion, the behavior of the solutions depends on the growth rate of $g(n)$ compared to that critical exponent γ.

Theorem 10.2: If $f(n)$ satisfies $f(n) \leq \alpha_1 f(\beta_1 n) + \cdots + \alpha_k f(\beta_k n) + g(n)$ for $n > n_0$, and γ with $1 = \alpha_1 \beta_1^\gamma + \cdots + \alpha_k \beta_k^\gamma$ is the critical exponent, then
- if $g(n) = O(n^\sigma)$ for some $\sigma < \gamma$, this implies $f(n) = O(n^\gamma)$.
- if $g(n) = O(n^\gamma)$, this implies $f(n) = O(n^\gamma \log n)$.
- if $g(n) = O(n^\tau)$ for some $\tau > \gamma$, this implies $f(n) = O(n^\tau)$.

Proof: We prove each case by induction, starting with the first case, with $g(n) < Cn^\sigma$ for some $\sigma < \gamma$; then $\alpha_1 \beta_1^\sigma + \cdots + \alpha_k \beta_k^\sigma > 1$, so we can choose some $E > 0$ sufficiently large that $(C - E(\beta_1^\sigma + \cdots + \beta_k^\sigma - 1)) < 0$. Let

$f(m) \leq Dm^\gamma - Em^\sigma$ for all $m < n$ and some sufficiently large $D > 0$, then we have

$$\begin{aligned} f(n) &\leq \alpha_1 f(\beta_1 n) + \cdots + \alpha_k f(\beta_k n) + g(n) \\ &\leq \alpha_1(D\beta_1^\gamma n^\gamma - E\beta_1^\sigma n^\sigma) + \cdots + \alpha_k(D\beta_k^\gamma n^\gamma - E\beta_k^\sigma n^\sigma) + Cn^\sigma \\ &= Dn^\gamma - E(\beta_1^\sigma + \cdots + \beta_k^\sigma)n^\sigma + Cn^\sigma \\ &= Dn^\gamma - En^\sigma + (C - E(\beta_1^\sigma + \cdots + \beta_k^\sigma - 1))n^\sigma \\ &< Dn^\gamma - En^\sigma. \end{aligned}$$

The proof of the last cae is similar: if $g(n) < Cn^\tau$ for some $\tau > \gamma$, then $\alpha_1\beta_1^\tau + \cdots + \alpha_k\beta_k^\tau < 1$, so we can choose some $E > 0$ sufficiently large that $(C - E(1 - \beta_1^\tau - \cdots - \beta_k^\tau)) < 0$. Let $f(m) \leq Dm^\gamma + Em^\tau$ for all $m < n$ and some sufficiently large $D > 0$, then we have

$$\begin{aligned} f(n) &\leq \alpha_1 f(\beta_1 n) + \cdots + \alpha_k f(\beta_k n) + g(n) \\ &\leq \alpha_1(D\beta_1^\gamma n^\gamma + E\beta_1^\tau n^\tau) + \cdots + \alpha_k(D\beta_k^\gamma n^\gamma + E\beta_k^\tau n^\tau) + Cn^\tau \\ &= Dn^\gamma + E(\beta_1^\tau + \cdots + \beta_k^\tau)n^\tau + Cn^\tau \\ &= Dn^\gamma + En^\tau + (C - E(1 - \beta_1^\tau - \cdots - \beta_k^\tau))n^\tau \\ &< Dn^\gamma + En^\tau. \end{aligned}$$

In the middle case, we have $g(n) < Cn^\gamma$. Then $\alpha_1\beta_1^\gamma \log\beta_1 + \cdots + \alpha_k\beta_k^\gamma \log\beta_k < 0$, so for D sufficiently large, we have $D(\alpha_1\beta_1^\gamma \log\beta_1 + \cdots + \alpha_k\beta_k^\gamma \log\beta_k) + C < 0$. Let $f(m) \leq Dm^\gamma \log m$ for all $m < n$ and a sufficiently large $D > 0$, then we have

$$\begin{aligned} f(n) &\leq \alpha_1 f(\beta_1 n) + \cdots + \alpha_k f(\beta_k n) + g(n) \\ &\leq \alpha_1(D\beta_1^\gamma n^\gamma \log n + D\beta_1^\gamma n^\gamma \log\beta_1) + \cdots + \\ &\quad \alpha_k(D\beta_k^\gamma n^\gamma \log n + D\beta_k^\gamma n^\gamma \log\beta_k) + Cn^\gamma \\ &\leq Dn^\gamma \log n + Dn^\gamma(\alpha_1\beta_1^\gamma \log\beta_1 + \cdots + \alpha_k\beta_k^\gamma \log\beta_k) + Cn^\gamma \\ &\leq Dn^\gamma \log n. \end{aligned}$$

Another type of recursion arises when the size of the subproblem is not just a constant fraction of n, or the number of subproblems depends on n. The most general form of this recursion is $f(n) \leq g(n)f(h(n)) + k(n)$. Only few special cases of this admit simple solutions.

10.2 Exercise Problems

Exercise 10.1: Show that if f satisfies $f(n) \leq 3f(\frac{1}{2}n) + 4f(\frac{1}{4}n)$, then $f(n) = O(n^2)$.

EXERCISE 10.2: Does $f(n) \leq 2f(\frac{1}{3}n) + O(n)$, imply $f(n) = O(n)$?

EXERCISE 10.3: Does $f(n) \leq f(\frac{1}{2}n) + O(\log n)$, imply $f(n) = O((\log n)^2)$?

EXERCISE 10.4: Show that if f satisfies $f(n) \leq 2f(\sqrt{n})$, then $f(n) = O(\log n)$.

EXERCISE 10.5: Show that if f satisfies $f(n) \leq \sqrt{n} f(\sqrt{n})$, then $f(n) = O(n)$.

EXERCISE 10.6: Show that if f satisfies $f(n) \leq \sqrt{n} f(\sqrt{n}) + O(1)$, then $f(n) = O(n)$.

EXERCISE 10.7: Show that if f satisfies $f(n) \leq \sqrt{n} f(\sqrt{n}) + O(n)$, then $f(n) = O(n \log \log n)$.

EXERCISE 10.8: Show that if f satisfies $f(n) \leq \sqrt{n} f(\sqrt{n}) + O(n \log n)$, then $f(n) = O(n \log n)$.

EXERCISE 10.9: Show that if f satisfies $f(n) \leq n^{2/3} f(n^{2/3}) + O(n^2)$, then $f(n) = O(n^2 \log n)$.

EXERCISE 10.10: Show that if f satisfies $f(n) \leq \frac{1}{2} n^2 f(\frac{1}{2}n)$, then $f(n) = O(2^{(\log n)^2})$.

11. Parametrized Divide-and-Conquer type Recursions

In this chapter recursions are discussed that give a parametrized family of bounds on $f(n)$ in terms of some $f(m(n,k))$ plus some overhead $g(k)$. The function satisfies all these bounds simultaneously.

11.1 Results

The simplest parametrized recursion is $f(n) \leq g(k)f(n/k)$ for $k = 2, \ldots, n$. It is simple to solve.

Theorem 11.1: If $f(n) \leq g(k)f(n/k)$ for all n and $k = 2, \ldots, n$, and $\alpha = \min_{k=2,\ldots,\infty} \frac{\log(g(k))}{\log k}$ then $f(n) = O(n^\alpha)$, and the function n^α satisfies the recursion.

Proof: If the sequence $\frac{\log(g(k))}{\log k}$ has a minimum $k_{\min}$, then $g(k_{\min}) = k_{\min}^\alpha$, and we have $f(n) \leq k_{\min}^\alpha f(n/k_{\min})$, which implies $f(n) = O(n^\alpha)$.

If the sequence has no minimum, the situation becomes more complicated. The lim inf cannot replace the minimum; $f(n) \leq g(k)f(n/k)$ with $\alpha = \liminf_{k\to\infty} \frac{\log(g(k))}{\log k}$ does not imply $f(n) = O(n^\alpha)$.

Example: The functions $g(k) = 2k\log k$ and $f(n) = n\log 2n$ satisfy the recursion $f(n) \leq g(k)f(n/k)$, since $\log 2n \leq 2\log k \log\frac{2n}{k}$ for $k = 2, \ldots, n$, so $n\log 2n \leq 2k\log k\frac{n}{k}\log\frac{2n}{k}$; $\liminf_{k\to\infty}\frac{\log(g(k))}{\log k} = 1$, but $f(n) \neq O(n)$.

A simple sufficient condition is

Theorem 11.2: If $f(n) \leq g(k)f(n/k)$ for all n and $k = 2, \ldots, n$, and $g(k) = O(k^\alpha)$ then $f(n) = O(n^\alpha)$

Proof: If $f(n) \leq g(k)f(n/k)$, then especially $f(n) \leq g(n)f(1)$, so $f(n) = O(g(n))$.

Frequently the parameter range for k will be restricted, with k being significantly smaller than n. If $\frac{\log(g(k))}{\log k}$ has a minimum $k_{\min}$, and $k_{\min}$ is allowed for sufficiently large n, then that determines the growth rate, as in theorem 10.1. If that sequence has no minimum, but is decreasing with $\liminf_{k\to\infty}\frac{\log(g(k))}{\log k} = \alpha$, and $g(k) = O(k^\alpha)$, then the implied bound on the growth rate of $f(n)$ depends on the allowed parameter range of k. If $k = 2, \ldots, n/a$ is allowed, for some constant a, then as in theorem 11.2 we have $f(n) \leq g(n/a)f(a) = O(g(n))$. But if only the range $k = 2, \ldots, \sqrt{n}$ is allowed, then the same recursion does not imply $f(n) = O(g(n))$.

EXAMPLE: The functions $f(n) = n \log n$ and $g(k) = 2k$ satisfy $f(n) \leq g(k)f(n/k)$ for $k \leq \sqrt{n}$; there $g(n) = O(n)$ but $f(n) \neq O(n)$.

In general, we have

THEOREM 11.3: If $f(n) \leq g(k)f(n/k)$ for all n and $k = 2, \dots, \frac{n}{r(n)}$, and $g(k) = O(k^\alpha)$ then $f(n) = n^\alpha 2^{O(r^*(n))}$, where $r^*(n) = \min\{i \mid r^i(n) = O(1)\}$.

If $k = 2, \dots, n^\beta$ is allowed, for some $0 < \beta < 1$, then $f(n) = n^\alpha (\log n)^{O(1)}$.

PROOF: If $g(k) = O(k^\alpha)$, then $g(k) \leq Ck^\alpha$ for some C. Then

$$\begin{aligned}
f(n) &\leq C\left(\frac{n}{r(n)}\right)^\alpha f(r(n)) \\
&\leq C^2\left(\frac{n}{r(n)}\right)^\alpha \left(\frac{r(n)}{r(r(n))}\right)^\alpha f(r(r(n))) \\
&= C^2\left(\frac{n}{r(r(n))}\right)^\alpha f(r(r(n))) \\
&\leq C^3\left(\frac{n}{r(r(n))}\right)^\alpha \left(\frac{r(r(n))}{r(r(r(n)))}\right)^\alpha f(r(r(r(n)))) \\
&= C^3\left(\frac{n}{r(r(r(n)))}\right)^\alpha f(r(r(r(n)))) \\
&\leq \dots \leq C^{r^*(n)} n^\alpha f(O(1))
\end{aligned}$$

If $r(n) = n^{1-\beta}$, then $r^*(n) = O(\log\log(n))$, and $C^{O(\log\log(n))} = (\log n)^{O(1)}$, which gives the second part of the theorem.

If our recursion contains additionally an overhead term, like $f(n) \leq g(k)f(f/k) + h(n,k)$, we can again reduce it to the results of the previous chapter by choosing a value k. If $\frac{\log g(k)}{\log k}$ has a minimum $k_{\min}$, with $g(k_{\min}) = k_{\min}^\alpha$, and $h(n, k_{\min}) = O(n^\beta)$ for some $\beta < \alpha$, we get from theorem 10.2 that $f(n) = O(n^\alpha)$, and this is the best bound we can derive from he recursion since it is reached even without the overhead term. More general, we can look for a $k_{\min}^*$ that minimizes $\max(\frac{\log g(k)}{k}, \lim\limits_{n\to\infty} \frac{\log(h(n,k))}{\log n})$, and apply theorem 10.2 for that $k_{\min}^*$.

Overhead terms can have the same effect as limiting the range of admissible k if the overhead grows sufficiently fast with k.

EXAMPLE: The functions $f(n) = n \log n$, $g(k) = 2k$ and $h(k) = k^2 \log k$ satisfy the recursion $f(n) \leq g(k)f(n/k) + h(k)$ for all $k = 2, \dots, n$.

11.2 Exercise Problems

EXERCISE 11.1: Show that $f(n) \leq \sqrt{k} f(\frac{n}{k}) + O(\sqrt{n})$ for $k \leq \sqrt{n}$ implies $f(n) = O(n \log\log n)$, and $n \log\log n$ does satisfy this recursion for $k \leq \sqrt{n}$.

12. Weighted Sum Recursions

In this chapter we discuss recursions in which $f(n)$ is bounded by a weighted sum of all previous $f(i)$.

12.1 Examples and Results

This type of recursion occurs especially in the analysis of randomized algorithms. As an example, when performing quicksort, a problem of size n is divided with probability $\frac{1}{n}$ in a problem of size i and a problem of size $n-1-i$, for $i = 1, \ldots, n-1$, so the expected complexity satisfies the recursion $f(n) \le O(n) + \sum\limits_{i=1}^{n-1} \frac{2}{n} f(i)$. We bound this as follows

Theorem 12.1: If the nonnegative function f satisfies
$f(n) \le O(n) + \sum\limits_{i=1}^{n-1} \frac{2}{n} f(i)$ for all $n > n_0$, then $f(n) \le O(n \log n)$.

Proof: We obtain this result by induction. Let $f(n) \le Cn + \sum_{i=1}^{n-1} \frac{2}{n} f(i)$ for all $n > n_0$, and $f(m) \le Dm \log m$ for all $m < n$, where $n > n_0$, with $\frac{2}{9} D > C$. Then

$$\begin{aligned} f(n) &\le Cn + \sum_{i=1}^{n-1} \tfrac{2}{n} f(i) \le Cn + \sum_{i=1}^{n-1} \tfrac{2}{n} Di \log i \\ &= Cn + D \sum_{i=1}^{n-1} 2\tfrac{i}{n} \log \tfrac{i}{n} + D \sum_{i=1}^{n-1} 2\tfrac{i}{n} \log n \\ &< Cn + D \sum_{i=n/3}^{2n/3} 2\tfrac{i}{n} \log \tfrac{i}{n} + D \log n \tfrac{2}{n} \sum_{i=1}^{n-1} i \\ &< Cn + D\tfrac{n}{3} 2\tfrac{2}{3} \log \tfrac{2}{3} + D \log n \tfrac{2}{n} \sum_{i=1}^{n-1} i \\ &< Cn + D\tfrac{n}{3} \tfrac{-2}{3} + D \log n \tfrac{2}{n} \tfrac{(n-1)n}{2} < Cn - D\tfrac{2n}{9} + Dn \log n \\ &< Dn \log n \end{aligned}$$

This bound is best possible, since the function $f(n) = n \log n$ does satisfy the recursion. This recursion is an example of the general type $f(n) \le g(n) + \sum\limits_{i=1}^{n-1} \alpha_{i,n} f(i)$ with an overhead term $g(n)$ and nonnegative weights $\alpha_{i,n}$. Some simple general insights for this type of recursion are

Theorem 12.2: If the nonnegative function f satisfies $f(n) \le \sum\limits_{i=1}^{n-1} \alpha_{i,n} f(i)$ with $\sum\limits_{i=1}^{n-1} \alpha_{i,n} \le 1$ for all n, then $f(n) = O(1)$.

PROOF: $f(n) \le \sum_{i=1}^{n-1} \alpha_{i,n} f(i) \le \sum_{i=1}^{n-1} \left(\alpha_{i,n} \max_{j=1,\dots,n-1} f(j) \right) < \max_{j=1,\dots,n-1} f(j)$

THEOREM 12.3: If the nonnegative monotone increasing function f satisfies $f(n) \ge \sum_{i=1}^{n-1} \alpha_{i,n} f(i)$ with $\sum_{i=n-k}^{n-1} \alpha_{i,n} > 1 + \varepsilon$ for all n and some $\varepsilon > 0$, then $f(n) = \Omega((1+\varepsilon)^{n/k})$.

PROOF: $f(n) \ge \sum_{i=1}^{n-1} \alpha_{i,n} f(i) \ge \sum_{i=n-k}^{n-1} \alpha_{i,n} f(i) \ge f(n-k) \sum_{i=n-k}^{n-1} \alpha_{i,n}$, so $f(n) \ge (1+\varepsilon) f(n-k)$.

So for interesting growth rates, the weight sum must be larger than one, but should not be concentrated near the end. As an example we state

THEOREM 12.4: If the nonnegative function f satisfies $f(n) \le \sum_{i=1}^{n-1} \frac{C}{n} f(i)$ for some $C > 1$ and all $n > n_0$, then $f(n) = O(n^{C-1})$.

PROOF: Let $f(m) \le Dm^{C-1}$ for $m < n$, then $f(n) \le \sum_{i=1}^{n-1} \frac{C}{n} f(i) \le \frac{C}{n} \sum_{i=1}^{n-1} Di^{C-1} \le \frac{C}{n} D \int_1^n x^{C-1} dx = \frac{C}{n} D \frac{1}{C}(n^C - 1) \le Dn^C$.

This has a more important version with an overhead term.

THEOREM 12.5: If the nonnegative function f satisfies $f(n) \le g(n) + \sum_{i=1}^{n-1} \frac{C}{n} f(i)$ for some integer $C \ge 1$ and all $n > n_0$, then

- if $g(n) = O(n^D)$, for some $D > C - 1$, then $f(n) = O(n^D)$.
- if $g(n) = O(n^{C-1})$, then $f(n) = O(n^{C-1} \log n)$.
- if $g(n) = O(n^D)$, for some $D < C - 1$, then $f(n) = O(n^{C-1})$.

This contains theorem 12.1 as a special case. We prove it by unrolling the recursion.

Proof: Let $f(n) \le g(n) + \sum_{i=1}^{n-1} \frac{C}{n} f(i)$, then

$$\begin{aligned}
f(n) &\le g(n) + \tfrac{C}{n} f(n-1) + \tfrac{C}{n} \sum_{i=1}^{n-2} f(i) \\
&\le g(n) + \tfrac{C}{n} g(n-1) + \tfrac{C^2}{n(n-1)} \sum_{i=1}^{n-2} f(i) + \tfrac{C}{n} \sum_{i=1}^{n-2} f(i) \\
&= g(n) + \tfrac{C}{n} g(n-1) + \tfrac{C(n+C-1)}{n(n-1)} \sum_{i=1}^{n-2} f(i) \\
&= g(n) + \tfrac{C}{n} g(n-1) + \tfrac{C(n+C-1)}{n(n-1)} f(n-2) + \tfrac{C(n+C-1)}{n(n-1)} \sum_{i=1}^{n-3} f(i) \\
&\le g(n) + \tfrac{C}{n} g(n-1) + \tfrac{C(n+C-1)}{n(n-1)} g(n-2) \\
&\quad + \tfrac{C(n+C-1)(n+C-2)}{n(n-1)(n-2)} \sum_{i=1}^{n-3} f(i) \\
&\le g(n) + \tfrac{C}{n} g(n-1) + \tfrac{C(n+C-1)}{n(n-1)} g(n-2) \\
&\quad + \cdots + \tfrac{C(n+C-1)(n+C-2)\cdots(n+C-k)}{n(n-1)(n-2)\cdots(n-k)} g(n-k-1) \\
&\quad + \tfrac{C(n+C-1)\cdots(n+C-k)(n+C-k-1)}{n(n-1)(n-2)\cdots(n-k)(n-k-1)} \sum_{i=1}^{n-k-2} f(i) \\
&\le g(n) + \tfrac{C}{n} g(n-1) + \tfrac{C(n+C-1)}{n(n-1)} g(n-2) \\
&\quad + \cdots + \tfrac{C(n+C-1)(n+C-2)\cdots(n+C-k)}{n(n-1)(n-2)\cdots(n-k)} g(n-k-1) \\
&\quad + \cdots + \tfrac{C(n+C-1)(n+C-2)\cdots(C+3)}{n(n-1)(n-2)\cdots(3)} g(2) \\
&\quad + \tfrac{C(n+C-1)(n+C-2)\cdots(C+2)}{n(n-1)(n-2)\cdots(2)} f(1).
\end{aligned}$$

Thus

$$\begin{aligned}
f(n) &\le g(n) + C \sum_{k=1}^{n-2} \tfrac{(n+C-1)\cdots(n+C-k+1)}{n(n-1)\cdots(n-k+1)} g(n-k) + O(1) \\
&= g(n) + C(n+C-1)(n+C-2)\cdots(n+1) \\
&\qquad \sum_{k=1}^{n-2} \tfrac{1}{(n+C-k)\cdots(n-k+1)} g(n-k) + O(1).
\end{aligned}$$

Now let $g(n) = O(n^D)$, then $\frac{1}{(n+C-k)\cdots(n-k+1)} g(n-k) = O\left((n-k)^{D-C}\right)$, and $f(n) \le g(n) + O(n^{C-1}) \sum_{k=1}^{n-2} O\left((n-k)^{D-C}\right)$

- If $D > C-1$, then $\sum_{k=1}^{n-2} O\left((n-k)^{D-C}\right) = O(n^{D-C+1})$, so $f(n) \le O(n^D)$.
- If $D = C-1$, then $\sum_{k=1}^{n-2} O\left((n-k)^{D-C}\right) = \sum_{k=1}^{n-2} O\left(\frac{1}{n-k}\right) = O(\log n)$, so $f(n) \le O(n^{C-1} \log n)$.

- If $D < C - 1$, then $\sum_{k=1}^{n-2} O\left((n-k)^{D-C}\right) = O(1)$, so $f(n) \leq O(n^{C-1})$.

12.2 Exercise Problems

EXERCISE 12.1: Show that $f(n) \leq f(\frac{n}{2}) + f(\frac{n}{3}) + f(\frac{n}{4}) + \cdots + f(\frac{n}{n})$ implies $f(n) = O(n^{\alpha})$ for some α.

EXERCISE 12.2: Show that $f(n) \leq 2f(\frac{n}{2}) + 4f(\frac{n}{4}) + \cdots + 2^{\log n} f(\frac{n}{2^{\log n}})$ implies $f(n) = O(n^2)$.

EXERCISE 12.3: Does $f(n) \leq \sum_{i=1}^{n} \frac{1}{2^i} f(n-i)$ imply $f(n) = O(1)$?

EXERCISE 12.4: Does $f(n) \leq \sum_{i=1}^{n} \frac{1}{i} f(i)$ imply $f(n) = O(n)$?

EXERCISE 12.5: Does $f(n) \leq \sum_{i=1}^{n} \frac{1}{n-i} f(i)$ imply $f(n) = O(n)$?

13. Linear Recursions with Constant Coefficients

In this chapter we describe how to solve linear recursions with constant coefficients

13.1 Examples and Results

A linear recursion with constant coefficients of order k is of the form $f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_k f(n-k)$; given k consecutive initial values, e.g., $f(1), \ldots, f(k)$, this recursion uniquely defines a function. The best known recursion of this type is the recursion defining the Fibonacci numbers $f(n) = f(n-1)+f(n-2)$ with $f(1) = f(2) = 1$. These recursions occur whenever some object of order n is constructed from several objects of order $n-1, n-2, \ldots$, as e.g., height-balanced trees. The useful property of these recursions is that we can solve them; the space of solutions of such a recursion is a k-dimensional linear space, and we can obtain a basis for this space, and then solve a system of linear equations to select the right element that fits our initial conditions. A consequence of this structure is that, since for almost all initial conditions the coefficients of all basis vectors will be nonzero, the growth rate is normally determined by the fastest-growing basis function.

The fundamental insight is the connection of the complex solutions of $f(n) = a_1 f(n-1)+a_2 f(n-2)+\cdots+a_k f(n-k)$ to the zeros of the characteristic polynomial $p(x) = x^n - a_1 x^{n-1} - a_2 x^{n-2} - \cdots - a_{n-1} x^1 - a_n$.

Theorem 13.1: Let $p(x)$ be the characteristic polynomial of the recursion $f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_k f(n-k)$. If a complex number c is a zero of p, then $f_c(n) = c^n$ is a solution of the recursion. If c is a r-fold zero, then c^n, nc^n, n^2c^n,$ldots$, $n^{r-1}c^n$ are solutions of the recursion. All complex solutions of the recursion are complex linear combinations of these basic solutions.

Proof: If $p(c) = 0$ then $c^k = a_1 c^{k-1} + a_2 c^{k-2} + \cdots + a_k c^0$, so

$$\begin{aligned} f_c(n) = c^n = c^k c^{n-k} &= (a_1 c^{k-1} + a_2 c^{k-2} + \cdots + a_k c^0) c^{n-k} \\ &= a_1 f_c(n-1) + a_2 f_c(n-2) + \cdots + a_k f_c(n-k), \end{aligned}$$

which shows the first claim.

If c is a r-fold zero of p, so $p(x) = (x-c)^r q(x)$ for some polynomial q, then c is also a zero of the derivatives $p', \ldots, p^{(r-1)}$. The i-th derivative of p has the form

$$\begin{aligned} p^{(i)}(x) = k(k-1)\cdots(k-i+1)x^{k-i} &- a_1(k-1)(k-2)\cdots(k-i)x^{k-i-1} \\ &- \cdots - a_{k-i} i(i-1)\cdots 1 x^0, \end{aligned}$$

so with $s_i(x) = x(x-1)\cdots(x-i+1)$ we have $p^{(i)}(x) = s_i(k)x^{k-i} - a_1 s_i(k-1)x^{k-i-1} - \cdots - a_{k-i}s_i(i)x^0$, and using that $s_i(i-1) = \ldots = s_i(0) = 0$, we find

$$\begin{aligned} x^i p^{(i)}(x) = s_i(k)x^k - a_1 s_i(k-1)x^{k-1} - \cdots - a_{k-i}s_i(i)x^i \\ - a_{k-i+1}s_i(i-1)x^{i-1} - \cdots - a_k s_i(0)x^0 \quad \text{for } i = 0, \ldots, r. \end{aligned}$$

With $s_0(x) = 1$, the polynomials $s_0, \ldots, s_r$ form a basis for the set of all polynomials of degree r, so for each polynomial t of degree at most r there are coefficients $b_0, \ldots, b_r$ such that $t(n) = b_0 s_0(n) + \cdots + b_r s_r(n)$ for all n. So we have

$$\begin{aligned} & t(k)c^k - a_1 t(k-1)c^{k-1} - a_2 t(k-2)c^{k-2} - \ldots - a_k t(0)c^0 \\ &= (\sum_{i=0}^r b_i s_i(k))c^k - a_1(\sum_{i=0}^r b_i s_i(k-1))c^{k-1} - a_2(\sum_{i=0}^r b_i s_i(k-2))c^{k-2} \\ &\qquad - \cdots - a_k(\sum_{i=0}^r b_i s_i(0))c^0 \\ &= \sum_{i=0}^r b_i c^i p^{(i)}(c) \quad = \quad 0 \end{aligned}$$

So for every polynomial t of degree at most r, the sequence $t(n)c^n$ satisfies at least the first case of the recursion: $t(k)c^k = a_1 t(k-1)c^{k-1} + \cdots + a_k t(0)c^0$. But then it satisfies the recursion for all n, since for each n the function $u(i) = t(i + (n-k))$ is also a polynomial of degree at most k, and

$$\begin{aligned} t(n)c^n = u(k)c^k c^{n-k} &= \left(a_1 u(k-1)c^{k-1} + \cdots + a_k u(0)c^0\right) c^{n-k} \\ &= a_1 t(n-1)c^{n-1} + \cdots + a_k t(n-k)c^{n-k} \end{aligned}$$

. This proves the second claim. The last claim follows, since the solution space of the recursion is k-dimensional, the characteristic polynomial p has k zeros, if we count multiplicities, and we have found as many linear independent solutions as there are zeros of p.

A disadvantage of this classic proof is that it produces complex solutions, even if we are only interested in real ones. The characteristic polynomial is real, so the zeros, if they are not real, occur in conjugate pairs $(x+iy), (x-iy)$, each of which generates a complex basis function $(x+iy)^n$, $(x-iy)^n$. Since the solution of the recursion is real, the coefficients of the functions must be chosen such that the imaginary parts cancel.

Example: The recursion $f(n) = -f(n-2)$ has the characteristic polynomial x^2+1, which has the zeros $+i$ and $-i$, so the general complex solution is $(a+ib)i^n + (c+id)(-i)^n$. This is real if $a-c=0$ and $b+d=0$, so the general real solution is $(a+ib)i^n + (a-ib)(-i)^n$, which is, however, not as useful description of the function as $a(-1)^{n/2}$ for n even and $b(-1)^{(n+1)/2}$ for n odd.

With the polar coordinate representation of complex multiplication, which corresponds to addition of angles and multiplication of radii, it is easy to see that the functions $(x+iy)^n$, $(x-iy)^n$ correspond to rotations in opposite directions, so they are symmetric to the real axis. The coefficients in any complex linear combination that gives a real value for all n then must also form a conjugate pair.

Theorem 13.2: Let $p(x)$ be the characteristic polynomial of the recursion $f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_k f(n-k)$. If a complex number $x+iy$ is a zero of p, then $f_1(n) = ((x+iy)^n + (x-iy)^n)$ and $f_2(n) = i((x+iy)^n - (x-iy)^n)$ are real solutions of the recursion. If $x+iy$ is a r-fold zero, then also $n((x+iy)^n + (x-iy)^n)$, $ni((x+iy)^n - (x-iy)^n)$,..., $n^{r-1}((x+iy)^n + (x-iy)^n)$, $n^{r-1}i((x+iy)^n - (x-iy)^n)$ are real solutions of the recursion. All real solutions of the recursion are real linear combinations of these basic solutions.

A natural variant of this type of recursion is to add another function, like the overhead terms we discussed before. The recursions of the form $f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_k f(n-k) + g(n)$ are known as inhomogeneous linear recursions with constant coefficients. The set of all solutions of such a recursion does not form a linear subspace of functions anymore, but still forms an affine subspace, a translate of the solution set of the homogeneous linear recursion, that is, without $g(n)$-term. Finding explicit solutions is usually not possible, but the asymptotic behavior of the solutions is simple.

13.2 Exercise Problems

14. Multi-variable recursions

In this chapter we discuss multi-variable recursions, like those for binomial coefficients, or those defining the Ackermann function

14.1 Results

some text

14.2 Exercise Problems

15. Number Theoretic Functions

In this chapter the asymptotic behavior of some number theoretic functions is discussed

15.1 Results

some text

15.2 Exercise Problems

16. Hints

In this chapter we give hints for selected exercises.

17. Solutions

In this chapter we give solutions for many exercises.

17.1 Solutions to Chapter 1.

17.2 Solutions to Chapter 2.

17.3 Solutions to Chapter 3.

17.4 Solutions to Chapter 4.

17.5 Solutions to Chapter 5.

17.6 Solutions to Chapter 6.

17.7 Solutions to Chapter 7.

17.8 Solutions to Chapter 8.

17.9 Solutions to Chapter 9.

17.10 Solutions to Chapter 10.

17.11 Solutions to Chapter 11.

17.11 Solutions to Chapter 12.

17.11 Solutions to Chapter 13.

17.11 Solutions to Chapter 14.

17.11 Solutions to Chapter 15.