Recurrence Relations for Computing Disciplines

The first few helpings**

Version September, 2018, rendered September 5, 2018

©
Appie van de Liefvoort
School of Computing and Engineering
University of Missouri – Kansas City

for an undergraduate course on CS-oriented discrete math (e.g. UMKC's CS291)
for an undergraduate course on Analysis of Algorithms (e.g. UMKC's CS404)
and a deepening review for a graduate course in the Analysis of Algorithms (e.g. UMKC's CS5592)

This write-up serves to give students a working knowledge of some of the common solution methods for recurrence relations as needed for a first and second course in a computing curriculum. It also serves as a foundation for the graduate level presentation, although several modules still need to be added for graduate students. The notation used is particularly suited for complexity studies of algorithms in an algorithms course that analyses the time complexity of algorithms that are designed according the greedy, divide-and-conquer, and dynamic programming) methods. This write-up is a living document and started many years ago as notes for a senior level or first year graduate course. I have since added material that is appropriate for a first year exposure to recurrence relation. For UMKC, this means:

CS291 Discrete Math II The course covers the introduction, and the sections on First Order, Second Order, and Higher Order recurrence relations. At the end of the course, students should master most of the exercises in these sections.

CS404 Introduction to Algorithms and Complexity Since CS291 is a prerequisite course for CS404, students in the CS404 class should master the First and Second Order recurrence relations, and know when and where to look for Higher Order recurrences, should these be needed. The CS404 course covers mostly the sections on Divide-and-Conquer and Secondary recurrence Relations, but the section on additional considerations should be given considerable attention. The applications however are very important: some case studies are presented here, and many more will be presented in class. Currently, case studies on the Catalan recursion, AVL tree analysis, and the Analyzing the average time for Quicksort, and the initial construction of a Heap (as priority queue) are included. There is also a case study on combinatorics, which goes beyond the course. After the CS404 course, the student should master most of the exercises in these sections. This material is also presented in the Appendix of various algorithm books.

CS5592 Design and Analysis of Algorithms Since CS291 and CS404 are among the prerequisite courses for CS5592, students in the CS5592 class should be prepared to deepen and extend their knowledge as presented in the sections written for CS291 and CS404. The graduate course deepens both results and techniques, e.g. the full version of the Master Theorem (still to come), and several 'bounding' recurrences. Some of these (such as such as matrix methods, generating functions, induced bounding recurrences, and optimizing thresholds) are not yet in this hand-out, so please take notes carefully.

I have tried to minimize the typo's in the theory side and in the formulas, but the exercises are not yet organized as well as they should. I used 'copy-and-paste' quite a bit, and it shows. Some problems in the various 'examples and practice problems' are still in the wrong place or are duplicated, please accept my apologies.

I very much appreciate any feedback. thanks, appie

Prerequisites

Before starting this section, you should

- have a working knowledge of algebra, including exponential and logarithmic numbers and notation, and including solving linear equations
- have a working knowledge of sequences and series, in particular geometric and arithmetic
- have a working knowledge of proof by induction
- have a working knowledge of the asymptotic notation, including $T(n) \sim g(n), T(n) = O(\,\cdot\,)$ and $T(n) = \Theta(\,\cdot\,)$
- have a working knowledge of some elementary probability

Learning Outcomes

After studying this section, including the completion of a majority of the problems, you should be able to:

- find both explicit closed form expressions and determine asymptotic classification for first and second order linear recurrences,
- apply a simplified version of the Master Theorem

1 Introduction

Many algorithms are studied by setting up an equation that reflects how the underlying problem gets smaller as first steps are taken. The resulting expression equates the ’big problem’ in terms of ’smaller problems’, and the resulting expression is called a recurrence relation, or difference equation. This is similar to many problems in engineering, the physical sciences and the social sciences which are studied by setting up an equation that reflects how the system changes over time. As time is continuous, you have a differential equation. In the algorithms we study, time changes one-step-at-a-time, that is, time is taken to be discrete, then you have a difference equation. Either way, differential equation and difference equations play an important role in analyzing the time behavior of systems in these disciplines. The physical systems studied in computer science are often of discrete nature: you take only one step in solving a problem or creating a structure. How many ways can a particular data structure present itself? How long does it take to execute a program, or run an algorithm knowing that they are executed step-by-step, data structures are changed one element at a time, problems are broken down in a collection of sub-problems, and solutions to larger problems are constructed by enlarging the solution to smaller problems. Studying combinatorial structures and studying the time and space complexity of algorithms requires the determination and subsequent solution of an equation containing the differences of the unknown function. Some concrete examples are given in Table 1.

Situation	Recurrence Relation
Number of comparisons in Bubble sort	$T(n) = \begin{cases} 0 & n = 1 \\ T(n-1) + n - 1 & n > 1 \end{cases}$
Minimal number of comparisons in Insertion sort	$T(n) = \begin{cases} 0 & n = 1 \\ T(n-1) + 1 & n > 1 \end{cases}$
Maximal number of comparisons in Insertion sort	$T(n) = \begin{cases} 0 & n = 1 \\ T(n-1) + n - 1 & n > 1 \end{cases}$
Average number of comparisons in Insertion sort	$T(n) = \begin{cases} 0 & n = 1 \\ T(n-1) + \frac{n-1}{2} & n > 1 \end{cases}$
Maximal number of comparisons in a variation of binary search	$T(n) = \begin{cases} 0 & n = 1 \\ T(\frac{n}{2}) + 1 & n > 1 \end{cases}$
Maximal number of comparisons in another variation of binary search	$T(n) = \begin{cases} 0 & n = 1 \\ T(\frac{n-1}{2}) + 2 & n > 1 \end{cases}$
Minimal number of comparisons in Merge sort	$T(n) = \begin{cases} 0 & n = 1 \\ 2T(\frac{n}{2}) + \frac{n}{2} & n > 1 \end{cases}$
Maximal number of comparisons in Merge sort	$T(n) = \begin{cases} 0 & n = 1 \\ 2T(\frac{n}{2}) + n - 1 & n > 1 \end{cases}$
Minimal number of nodes in an AVL tree of height h	$\begin{cases} n(0) = 1;\ n(1) = 2 \\ n(h) = 1 + n(h-1) + n(h-2) & n > 1 \end{cases}$
Maximal number of nodes in an AVL tree of height h	$M(h) = \begin{cases} 1 & h = 0 \\ 1 + 2M(h-1) & h > 1 \end{cases}$
Number of nodes in an n-cube	$N(n) = \begin{cases} 1 & n = 0 \\ 2N(n-1) & n > 1 \end{cases}$
Number of links in an n-cube	$L(n) = \begin{cases} 1 & h = 0 \\ 2L(n-1) + 2^{n-1} & n > 1 \end{cases}$
Number of moves in Tower of Hanoi with 3 pegs and n discs	$M_3(n) = \begin{cases} 1 & n = 1 \\ 2M_3(n-1) + 1 & n > 1 \end{cases}$
Number of moves made by the smallest disc in the Tower of Hanoi with 3 pegs and n discs	$M_3(1, n) = \begin{cases} 1 & n = 1 \\ 2M_3(1, n-1) & n > 1 \end{cases}$
Number of moves in Tower of Hanoi with 4 pegs and n discs, and any k with $1 \leq k \leq n$	$M_4(n) = \begin{cases} M_3(k) & n \leq k \\ 2M_4(n-k) + M_3(k) & n > k \end{cases}$
Number of binary trees with n nodes The Catalan Recursion	$\begin{cases} C(0) = 1;\ C(1) = 1 \\ C(n) = \sum_{i=0}^{n-1} C(i-1) \times C(n-i) & n > 1 \end{cases}$

Table 1: A sampling of recurrence relations encountered in a first algorithm course

There are several separate aspects of recurrence relations:

Construction

This very much depends on the application area, and is not the primary focus of this write up. The analysis of recursive algorithms, (reduce and conquer, divide and conquer, dynamic programming) often give rise to recurrence relations. They also arise when counting combinatorial objects. Furthermore, the formulation also occurs in First-Step-Analysis as a paradigm, (a precursor to such Markov processes as random walks and birth death processes), where certain states are recurring in time, are thus formulated recursively, and so on. Although the actual construction and derivation of recurrence relations is not the primary focus of this write-up, we will present some case studies where some of the classical situations are presented. In particular, we present the Catalan recursion, the QuickSort recursion, the AVL tree analysis, and show one or two non-trivial combinatorial recursions.

Evaluate $T(n)$ for particular values of n

The recurrence relation, once it is constructed, gives rise immediately to either an iterative or recursive algorithm to find particular values. These can be implemented on just about any computing device and in just about any language or spreadsheet. But these must be done with great care: Optimization problems, as studied in algorithm courses and as solved using algorithms using the Dynamic programming paradigm, are best represented as a recursively formulated idea, and the object is then to find the actual value. This is not the focus of this hand-out. Neither are the potentially computational challenges since either the intermediate values, the final values, or both, might cause overflow or underflow, which are not always detected. Evaluating particular values for $T(n)$ is not the focus of this section.

Find a closed form expressions for $T(n)$

The recurrence relation itself is an expression for $T(n)$ that completely specifies T as a function, once the initial values are taken into account. An explicit closed form for $T(n)$ is easier and more desirable for a study on the behavior of $T(n)$, and for comparing with other complexity functions. Finding explicit forms of recurrence relations is addressed in this hand out. Note that simple recurrence relations can be solved with a computer algebra system such as MAPLE.

Find the asymptotic form for T

In most of the computing application for recurrence relations, $T(n)$ is an increasing function in n, so that $T(n)$ grows without bound as n grows without bound. We like to understand how fast $T(n)$ grows, and thus try to classify T according its asymptotic behavior in the 'neighborhood' of infinity. We try to find the term $g(n)$ such that $T(n) \sim g(n)$ and $g(n)$ is as simple as possible. This is also addressed in this hand out. Firstly, we derive the explicit form of the solution, after which we find the asymptotic term. There are situations where the explicit form is very difficult to determine in which case simplifying steps can be taken, as long as these simplifying steps keep the asymptotic behavior intact. There are techniques that bypass the explicit solution, and directly identify the asymptotic behavior of the function $T(n)$. These techniques are needed for a graduate level presentation, and several of these techniques are incorporated in this hand-out, but for their proofs you need to check the literature (e.g. CLR)

2 First Order Recurrence Relations

As shown in the table above, many recurrence relations are 'One Term' recurrence relations, such as

$$T(n) = T(n-1) + f(n) \quad \text{and} \quad T(n) = aT(n-1) + f(n) \tag{1}$$

First, consider the one-term recurrence relations of the form

$$\begin{cases} T(0) = t_0 \\ T(n) = T(n-1) + f(n) & n > 0 \end{cases}$$

In the above equation, the term "$f(n)$"in the recursive part is called the *inhomogeneous part*, and by unwinding the recursion, the explicit expression for $T(n)$ is obtained. This process is called "back substitution," which we illustrated as follows: Write the equations underneath one another, and add all equations together: all terms to the left of the equality sign remain to the left in the result, and all the terms to the right remain in the right. But notice that most terms with a "$T(.)$" cancel, by construction.

$$\begin{array}{rcl c r c l l}
T(n) & = & T(n-1) + f(n) & & T(n) & & \cancel{T(n-1)}^{a} & +f(n) \\
T(n-1) & = & T(n-2) + f(n-1) & & +\cancel{T(n-1)}^{a} & & +\cancel{T(n-2)}^{b} & +f(n-1) \\
T(n-2) & = & T(n-3) + f(n-2) & & +\cancel{T(n-2)}^{b} & & +\cancel{T(n-3)}^{c} & +f(n-2) \\
T(n-3) & = & T(n-4) + f(n-3) & & +\cancel{T(n-3)}^{c} & & +\cancel{T(n-4)}^{d} & +f(n-3) \\
T(n-4) & = & T(n-5) + f(n-4) & & +\cancel{T(n-4)}^{d} & & +\cancel{T(n-5)}^{e} & +f(n-4) \\
\vdots & & \vdots & \text{add vertical} & \vdots & = & \vdots & \\
T(2) & = & T(1) + f(2) & & +\cancel{T(2)} & & +\cancel{T(1)} & +f(2) \\
T(1) & = & T(0) + f(1) & & +\cancel{T(1)} & & +\cancel{T(0)} & +f(1) \\
T(0) & = & t_0 & & +\cancel{T(0)} & & & +t_0 \\
\hline
 & & & \text{result:} & T(n) & = & t_0 & +\sum_{i=1}^{n} f(i)
\end{array}$$

The result is thus

$$T(n) = t_0 + f(1) + f(2) + \cdots f(n) = t_0 + \sum_{i=1}^{n} f(i) \tag{2}$$

Notice that the number of terms in this summation is n (or $n+1$ if the boundary value is counted). Also, as $f(n)$ is generally an increasing function (or at least non-decreasing), the individual terms in this summation become larger with n. In the table below we specialize the function $f(n)$ to be a linear, quadratic, and k-degree monomial in n, as well as a logarithmic and exponential function in n. The explicit expressions are all based of finding a closed form expression for the summations of the terms $\sum_{i=1}^{n} f(i)$.

2.1 Examples and Practice Problems

The following problems will give you practice in applying this technique, but also gives you practice in adjusting the technique to accommodate slightly different (initial) conditions. For all subproblems, unless instructed otherwise, find both the explicit closed form for $T(n)$, and it's asymptotic dominant term.

1. $\begin{cases} T(n) = 3 & n \le 0 \\ T(n) = T(n-1) + 6 & 1 \le n \end{cases}$ Solution: $T(n) = 6n + 3 \sim 6n$

$T(n) = \begin{cases} t_0 & n = 0 \\ T(n-1) + f(n) & n > 1 \end{cases}$ with the inhomogeneous part $f(n)$ as specified			
Constant	$f(n) = c_0$	$T(n) = t_0 + c_0\, n$	$\sim c_0\, n$
Linear	$f(n) = c_1\, n$	$T(n) = t_0 + \left(\frac{1}{2}\, n^2 + \frac{1}{2}\, n\right)\, c_1$	$\sim \frac{1}{2}\, c_1 n^2$
Quadratic Monomial	$f(n) = c_2\, n^2$	$T(n) = t_0 + \left(\frac{1}{3}\, n^3 + \frac{1}{2}\, n^2 + \frac{1}{6}\, n\right)\, c_2$	$\sim \frac{1}{3}\, c_2 n^3$
Monomial of degree k	$f(n) = c_k\, n^k$	$T(n) = t_0 + \left(\frac{1}{k+1}\, n^{k+1} + \frac{1}{2}\, n^k + o(n^k)\right)\, c_k$	$\sim \frac{1}{k+1}\, c_k n^{k+1}$
Logarithmic	$f(n) = c\, \lg n,$	$T(n) = t_0 + c\, \lg(n!)$	$\sim cn\, \lg n$
Exponential	$f(n) = c\, 2^n,$	$T(n) = t_0 + c\, 2^{n+1} - 2c$	$\sim 2c\, 2^n$

Table 2: The explicit solution and it's asymptotic dominant term of recurrence relations involving $T(n-1)$ and $T(0)$.

2. $\begin{cases} T(n) = 3 & n \le 4 \\ T(n) = T(n-1) + 6 & 5 \le n \end{cases}$ Solution: $T(n) = 6n - 21 \sim 6n$

3. $\begin{cases} T(n) = 3 & n \le 0 \\ T(n) = T(n-1) + 2\, n & 1 \le n \end{cases}$ Solution: $T(n) = n^2 + n + 3 \sim n^2$

4. $\begin{cases} T(n) = 3 & n \le 4 \\ T(n) = T(n-1) + 2\, n & 5 \le n \end{cases}$ Solution: $T(n) = n^2 + n - 17 \sim n^2$

5. $\begin{cases} T(n) = 3 & n \le 0 \\ T(n) = T(n-1) + 4\, n\, + 5 & 1 \le n \end{cases}$ Solution: $T(n) = 2\, n^2 + 7n + 3 \sim 2\, n^2$

6. $\begin{cases} T(n) = 3 & n \le 0 \\ T(n) = T(n-1) + 6\, n^2 & 1 \le n \end{cases}$ Solution: $T(n) = 2\, n^3 + 3\, n^2 + n + 3 \sim 2\, n^3$

7. $\begin{cases} T(n) = 3 & n \le 4 \\ T(n) = T(n-1) + 6\, n^2 & 5 \le n \end{cases}$ Solution: $T(n) = 2\, n^3 + 3\, n^2 + n - 177 \sim 2\, n^3$

2.1.1 Using MAPLE

I strongly recommend that you experiment with some of the recurrence relations on a computer algebra system (CAS) that allows for symbolic computation, such as MAPLE, Mathematica, Matlab, SimPy, and others. For instance, type the following into a MAPLE session:

```
rsolve({T(0) = 3, T(n) = T(n-1)+n^2}, T(n)); simplify(%)
```

Which immediately gives the answer to one of the recurrence relations above. You should experiment with some of these, and plot (or otherwise compare) the solutions to closely related recurrence equations to study the impact of the various values of constants. Please take advantage of the availability of such softwares. Consider for example the following for a MAPLE session:

```
T1:=n -> rsolve({T(0) = 3, T(n) = T(n-1)+n^2}, T(n));
T2:=n -> rsolve({T(0) = 3, T(n) = T(n-1)+n+ n^2}, T(n));
T3:=n -> rsolve({T(0) = 3, T(n) = T(n-1)+2*n^2}, T(n));
plot([T1(n),T2(n),T3(n)], n=10..20, color=[red,blue,green]);
```

2.2 $T(n) = a\,T(n-1) + f(n),\;\; a > 1$

Next, we consider one-term recurrence relations where the recurring term is multiplied by a constant and is of the form

$$\begin{cases} T(n) = t_0 & n = 0 \\ T(n) = a\,T(n-1) + f(n), \quad a > 1 & n > 0 \end{cases}$$

Again, back substitution can be used to get the explicit expression for $T(n)$. Again, we write each equation separately, but now multiply the entire equation to make sure the coefficients match so that they still cancel when adding:

$$\begin{array}{rcl}
T(n) &=& a\,T(n-1) + f(n) \\
a\,\Big(T(n-1) &=& a\,T(n-2) + f(n-1)\Big) \\
a^2\,\Big(T(n-2) &=& a\,T(n-3) + f(n-2)\Big) \\
a^3\,\Big(T(n-3) &=& a\,T(n-4) + f(n-3)\Big) \\
a^4\,\Big(T(n-4) &=& a\,T(n-5) + f(n-4)\Big) \\
\vdots & & \vdots \\
a^{n-2}\,\Big(T(2) &=& a\,T(1) + f(2)\Big) \\
a^{n-1}\,\Big(T(1) &=& a\,T(0) + f(1)\Big) \\
a^{n}\,\Big(T(0) &=& t_0\Big)
\end{array}$$

$$\begin{array}{lcccc}
 & T(n) & \cancelto{a}{a\,T(n-1)} & + & f(n) \\
 & +\cancelto{a}{a\,T(n-1)} & +\cancelto{b}{a^2\,T(n-2)} & & +\,a\,f(n-1) \\
 & +\cancelto{b}{a^2\,T(n-2)} & +\cancelto{c}{a^3\,T(n-3)} & & +\,a^2\,f(n-2) \\
 & +\cancelto{c}{a^3\,T(n-3)} & +\cancelto{d}{a^4\,T(n-4)} & & +\,a^3\,f(n-3) \\
 & +\cancelto{d}{a^4\,T(n-4)} & +\cancelto{e}{a^5\,T(n-5)} & & +\,a^4\,f(n-4) \\
\text{add} & \vdots & = \;\vdots & & \\
 & +\cancel{a^{n-2}\,T(2)} & +\cancel{a^{n-1}\,T(1)} & & +\,a^{n-2}\,f(2) \\
 & +\cancel{a^{n-1}\,T(1)} & +\cancel{a^{n}\,T(0)} & & +\,a^{n-1}\,f(1) \\
 & +\cancel{a^{n}\,T(0)} & & & +\,a^{n}\,t_0 \\
\hline
\text{result:} & T(n) & = \;\; t_0 & & +\sum_{i=1}^{n} a^{n-i}\,f(i)
\end{array}$$

The final equation is worth repeating explicitly:

$$T(n) = a^n\,t_0 + a^{n-1}f(1) + a^{n-2}f(2) + \cdots a^2 f(n-2) + a^1 f(n-1) + f(n) = t_0 + \sum_{i=1}^{n} a^{n-i} f(i),$$

There are still a linear number of terms in this summation, and the terms may or may not be increasing. There are two somewhat competing forces at work: because $a > 1$, the sequence $a^n, a^{n-1}, a^{n-2}, \cdots, a, 1$ is decreasing, whereas the sequence $f(1), f(2), f(3), \cdots f(n-1), f(n)$ is usually increasing. The sequence of their products can be decreasing, more or less flat, or increasing. It just depends on the specific function f for the sequence $a^{n-1}f(1), a^{n-2}f(2), \cdots a^1 f(n-1), f(n)$ to be increasing, decreasing, or constant. In the table below we specialize $f(n)$ to be a linear, quadratic, and k-degree polynomial in n, as well as a logarithmic and exponential function in n. The explicit form of $T(n)$ is available for these cases, although the form is rather involved and not informative. Should it be needed, one could use a symbolic system like MAPLE to generate the form. As n grows, compare the two extremes: a^n is exponential in n, which is then compared with $f(n)$. So if $f = O(a^n)$, which is the case when f is constant, logarithmic, linear, quadratic or polynomial, then the exponential term dominates. The explicit expressions are all based on finding a closed form expression for the summations of the terms $\sum_{i=1}^{n} a^{n-i} f(i)$.

Whenever $f(n)$ is a monomial of degree k, $f(n) = c_k\,n^k$, then the explicit expression is still possible, but rather complicated. If additionally $a > 1$, then the asymptotic complexity is $T = \Theta(a^n)$, whereas if $a < 1$, then $T(n) \sim \Theta((n+1)(n+2)(n+3)...(n+k))$, so that $T(n) = \Theta(n^k)$. The full expression involves Eulerian polynomials and is beyond the current exposition.

$T(n) = \begin{cases} t_0 & n = 0 \\ a\,T(n-1) + f(n) & n > 1 \end{cases}$ with the inhomogeneous part $f(n)$ as specified		
Constant	$f(n) = c_0$	$T(n) = t_0\,a^n + c_0 \left(\frac{a^n - 1}{a-1}\right) \quad = \left(t_0 + \frac{c_0}{a-1}\right) a^n + o(a^n) = \Theta(\ a^n\ )$
Linear	$f(n) = c_1\,n$	$T(n) = t_0\,a^n + c_1\,a\,\frac{a^n-1}{(a-1)^2} - n\,\frac{1}{a-1} = \left(t_0 + \frac{c_1\,a}{(a-1)^2}\right) a^n + o(a^n) = \Theta(\ a^n\ )$

Table 3: The explicit solution and it's asymptotic dominant term of recurrence relations involving $a\,T(n-1)$ and $T(0)$ for $a > 1$.

2.2.1 Examples and Practice Problems

For all subproblems, unless instructed otherwise, find both the explicit closed form for $T(n)$, and it's asymptotic dominant term.

1. $\begin{cases} T(n) = 3 & n \le 0 \\ T(n) = T(n-1) + 6 & 1 \le n \end{cases}$ Solution: $T(n) = 6n + 3 \sim 6n$

2. $\begin{cases} T(n) = 3 & n \le 0 \\ T(n) = 5\,T(n-1) + 6 & 1 \le n \end{cases}$ Solution: $T(n) = \frac{9}{2}\,5^n - \frac{3}{2} \sim \frac{9}{2}\,5^n$

3. $\begin{cases} T(n) = 3 & n \le 4 \\ T(n) = 5\,T(n-1) + 6 & 5 \le n \end{cases}$ Solution: $T(n) = \frac{9}{1250}\,5^n - \frac{3}{2} \sim \frac{9}{1250}\,5^n$

Check this with the MAPLE statement: `T1:=n -> rsolve({T(4) = 3, T(n) = 5T(n-1)+6}, T(n));`

4. $\begin{cases} T(n) = 3 & n \le 1 \\ T(n) = 2\,T(n-1) + n & 1 < n \end{cases}$ Solution: $T(n) = 3\,2^n - n - 2 \sim 3\,2^n$

5. $\begin{cases} T(n) = 3 & n \le 2 \\ T(n) = 2\,T(n-1) + n & 2 < n \end{cases}$ Solution: $T(n) = \frac{7}{4}\,2^n - n - 2 \sim \frac{7}{4}\,2^n$

3 Second Order Recurrence Relations

3.1 Homogeneous relations

In this section, we consider two-term recurrence relations of the form

$$\begin{cases} T(0) = t_0,\ T(1) = t_1 \\ T(n) = a_1\, T(n-1) + a_2\, T(n-2) \qquad n > 1 \end{cases}$$

These occur often in combinatorial situations. The form of the solution is given by either $T(n) = c_1 r_1^n + c_2 r_2^n$ or $T(n) = (c_{11} + c_{12}\ n) \times r_1^n$, which is explained below. The formal proof that these forms are the only possible solutions relies on the transformation of the recurrence relation to generating functions, and this discussion is postponed for now. You can reason as follows: Since the closed form solution to the one-term recurrence relation $T(n) = a_1\ T(n-1),\ n > 1$ is $T(n) = t_0\ a_1^n$, we are inspired *to try* a solution of the similar form, $T(n) = c\ x^n$, and will find restricting conditions on the values of x for which this may satisfy the recurrence relation. So substitute the trial solution in the defining equation:

$$\begin{aligned} T(n) &= a_1\, T(n-1)\ + a_2\, T(n-2) \\ x^n &= \quad a_1\, x^{n-1} \quad + \quad a_2\, x^{n-2} \end{aligned}$$

Dividing both sides by x^{n-2}, and the two term recurrence induces a quadratic equation

$$x^2 - a_1 x - a_2 = 0.$$

So, if the solution is to be of the form $T(n) = c\ x^n$, then x *must* satisfy this quadratic equation. Generally speaking, there are two roots of this quadratic equation (or one single root with multiplicity 2), we must keep both roots open as a possibility. Thus the roots of this quadratic equation determine the exact form of the solution. The roots can be either distinct and real, distinct and complex valued, or single root with multiplicity two. We consider each case separately. Note that each case will look a little different, but all have two constants that are as yet unknown, but their values are determined by setting up two equations (two boundary equations $T(2)$ and $T(3)$) with two unknowns.

Two distinct real roots, i.e. $a_1^2 + 4\, a_2 > 0$:

$$\begin{aligned} T(n) &= c_1 (r_1)^n + c_2 (r_2)^n \ , \\ &\quad \text{where} \quad r_1 = \frac{a_1 + \sqrt{a_1^2 + 4a_2}}{2} \qquad r_2 = \frac{a_1 - \sqrt{a_1^2 + 4a_2}}{2} \\ &\quad \phantom{\text{where}} \quad c_1 = \frac{t_1 - t_0 r_2}{r_1 - r_2} \qquad c_2 = \frac{r_1 t_1 - t_1}{r_1 - r_2} \end{aligned}$$

One root with multiplicity 2*, i.e.* $a_1^2 + 4\, a_2 = 0$:

$$\begin{aligned} T(n) &= (c_{11} + c_{12}\, n)(r_1)^n \\ &\quad \text{where} \quad r_1 = \frac{a_1}{2} \\ &\quad c_{11} = t_0 \qquad c_{12} = \frac{2t_1}{a_1} - t_0 \end{aligned}$$

Two distinct complex roots, i.e. $a_1^2 + 4\, a_2 < 0$:

$$\begin{aligned}
\text{either}\quad T(n) &= c_1(r_1)^n + c_2(r_2)^n\ , \\
&\quad \text{where}\quad r_1 = \frac{a_1 + \sqrt{a_1^2 + 4a_2}}{2} \qquad r_2 = \frac{a_1 - \sqrt{a_1^2 + 4a_2}}{2} \\
&\quad c_1 = \frac{t_1 - t_0 r_2}{r_1 - r_2} \qquad c_2 = \frac{r_1 t_1 - t_1}{r_1 - r_2} \\
\text{or}\quad T(n) &= (c_3 \sin(n\theta) + c_4 \cos(n\theta))^n\ r^n\ \ , c_3, c_4, r \text{ are all real} \\
&\quad \text{where}\quad r = \sqrt{a_1^2 + 2a_2}\ \text{ and } c_3, c_4, \theta \text{ can be determined as well.}
\end{aligned}$$

This last situation (i.e. complex roots) will not be needed in this course.

3.1.1 Examples and Practice Problems

1. Find both the explicit closed form for $T(n)$, and the asymptotic class of T:

$$\begin{cases} T(0) = 4,\ T(1) = 4 \\ T(n) = 2T(n-1) + 3T(n-2) \quad n \geq 2 \end{cases}$$

The equation induces the associated characteristic equation $x^2 - 2x - 3 = (x+1)(x-3)$ which has two distinct real roots, so that the solution is given by $T(n) = c_1(-1)^n + c_2 3^n$, where c_1 and c_2 can be found form the initial conditions: $T(n) = 2(-1)^n + 2\,3^n = \Theta(\ 3^n\ )$.

2. Find both the explicit closed form for $T(n)$, and the asymptotic class of T:

$$\begin{cases} T(0) = 1,\ T(1) = 6 \\ T(n) = 4\,T(n-1) - 4\,T(n-2) \quad n \geq 2 \end{cases}$$

The equation induces the associated characteristic equation $x^2 - 4x + 2 = (x-2)^2$ which has a single root with multiplicity two, so that the solution is given by $T(n) = (c_{11} + c_{12}\ n) \times 2^n$, where c_{11} and c_{12} can be found form the initial conditions: $T(n) = (1+2n)2^n = \Theta(\ 2^n\ )$.

3. Use MAPLE to plot solutions to the above recurrence relations with different initial values, or with different coefficients.

3.2 Second Order, Inhomogeneous Recurrence Relations

A recurrence relation is homogeneous if the all the terms on the right-hand side involve terms like $T(n-i)$. This is rarely the case for time complexity studies. The inhomogeneous part is simply the expression that is left after all the terms with $T(n-i)$ have been removed, and we will use $f(n)$ to indicate the inhomogeneous part. Generally speaking, inhomogeneous recurrence relations are hard to solve, except if the both the homogeneous and the inhomogeneous parts are of a particular form. These are discussed in this section, and we start with the inhomogeneous part being an exponential term, that is, n appears in the exponent, $f(n) = a_s\, s^n$ for some constants a_s and s.

3.2.1 Inhomogeneous function $f(n) = a_s s^n$ for some known constants a_s and s

Consider now

$$\begin{cases} T(n) = t_n & n \leq 1 \\ T(n) = a_1\, T(n-1) + a_2\, T(n-2) + a_s s^n & n > 1 \end{cases}$$

Notice that we studied the homogeneous part in the previous subsection, whose solution could be found by finding the roots of the associated polynomial. It is not hard to show that the current situation, two consecutive recurrence relations can be used to solve for the inhomogeneous part, which can then be eliminated, essentially removing the homogeneous part at the cost of increasing the order. The recurrent definition for $T(n-1)$ can be used to solve for s^{n-1}:

$$\begin{aligned} T(n-1) &= a_1\, T(n-2) + a_2\, T(n-3) + a_s s^{n-1} \\ &\quad \text{thus, isolating the inhomogeneous part,} \\ a_s s^{n-1} &= T(n-1) - a_1\, T(n-2) - a_2\, T(n-3) \\ &\quad \text{and multiply both sides by } s \\ a_s s^n &= s\ (T(n-1) - a_1\, T(n-2) - a_2\, T(n-3)\ ) \end{aligned}$$

Substitute this part into the defining recurrent definition for $T(n)$:

$$T(n) = a_1\, T(n-1) + a_2\, T(n-2) + s\ (T(n-1) - a_1\, T(n-2) - a_2\, T(n-3)\ )$$

So we have changed this non-homogeneous recurrence relation into a homogeneous recurrence relation. The order of this new recurrence relation is three, one higher than the one we started with. The solution technique for third order recurrence relations is similar to those we have already seen (and will be covered fully in the next section). They also have a characteristic equation, which is identified by first substituting $T(n) = x^n$, and then dividing by x^{n-3}. This gives:

$$x^3 = a_1\, x^2 + a_2\, x + s\ \left(x^2 - a_1\, x^1 - a_2\ \right)$$

which can be factored as:

$$\left(x^2 - a_1\, x - a_2\right) \times\ (x - s) = 0 \tag{3}$$

So, not only have we identified another recurrence relation that is homogeneous, of one degree higher than the inhomogeneous one we started with, and whose solution is identical to the original $T(n)$, we have also shown that the characteristic equation of this extended recurrence relation is the product of the original characteristic equation, and the term $(x - s)$. Depending on how s relates to the roots of the characteristic equation, we have

The solution form for $T(n)$:

$$\begin{array}{lrcll} & T(n) & = & c_1(r_1)^n + c_2(r_2)^n + c_3(r_3)^n & \text{if all roots are distinct} \\ \text{or} & T(n) & = & (c_{11} + c_{12}\,n)(r_1)^n + c_3(r_3)^n & \text{if two roots are distinct} \\ \text{or} & T(n) & = & (c_{11} + c_{12}\,n + c_{13}\,n^2)(r_1)^n & \text{if one root with multiplicity 3} \end{array}$$

The constants can be solved for (in terms of known quantities) by setting up three equations from the boundary values $T(2)$, $T(3)$ and $T(4)$ to solve for them.

3.2.2 Other inhomogeneous functions, including linear functions

This results can be extended quite a bit. In fact, the inhomogeneous part above was $f(n) = a_s s^n$ and this can be extended to $f(n) = a_s(n)s^n$ where $a_s(n)$ is now a polynomial in n. The characteristic polynomial must now be adjusted and the term $(x - s)$ now has a higher degree, depending on the degree of $a_s(n)$: it is one degree higher. Notice, that this opens the door for inhomogeneous functions that are polynomials, by taking $s = 1$. Furthermore, there could be more of such terms in the inhomogeneous part; they are best shown by example:

	The extended characteristic equation for $T(n)$ when the inhomogeneous form $f(n)$ is as indicated. $T(n) = a_1\,T(n-1) + a_2\,T(n-2) + f(n)$	
	Inhomogeneous function $f(n)$	Extended characteristic polynomial
1.	$f(n) = s^n$	$(x^2 - a_1\,x^1 - a_2\,)\,(x - s)$
2.	$f(n) = 2\,3^n$	$(x^2 - a_1\,x^1 - a_2\,)\,(x - 3)$
3.	$f(n) = (1 + 2\,n + n^3)\,3^n$	$(x^2 - a_1\,x^1 - a_2\,)\,(x - 3)^4$
4.	$f(n) = (1 + 2\,n + n^7)\,3^n$	$(x^2 - a_1\,x^1 - a_2\,)\,(x - 3)^8$
5.	$f(n) = 1 + 2\,n + n^3$	$(x^2 - a_1\,x^1 - a_2\,)\,(x - 1)^4$
6.	$f(n) = (1 + 2\,n)\,3^n + (2\,n + n^3)\,4^n$	$(x^2 - a_1\,x^1 - a_2\,)\,(x - 3)^2\;(x - 4)^4$
7.	$f(n) = (1 + 2\,n)\,3^n + (2\,n + n^3)$	$(x^2 - a_1\,x^1 - a_2\,)\,(x - 3)^2\;(x - 1)^4$

Table 4: The extended characteristic polynomial for $T(n) = a_1\,T(n-1) + a_2\,T(n-2) + f(n)$ and various forms of $f(n)$.

So we started with an inhomogeneous recurrence relation, where both the homogenous part and the inhomogeneous part had a particular (but still fairly common) form. From this, we identified another recurrence relation that is homogeneous, and had a higher degree than the inhomogeneous one we started with. Finally, we extended the characteristic equation. and the solution is described in terms of the roots of this extended characteristic equation. First we must make a note of caution: We extended the characteristic equation by (a power of) $\times(x - s)$ and the general solution can be described.

3.2.3 Describing the solution

The solution is described in terms of the roots of this extended characteristic equation. First we must make a note of caution: If s is a also root of the original characteristic polynomial, then their multiplicities must be added, before drawing up the general solution. Thus, as before, each distinct root r_i of the extended polynomial has a multiplicity m_i, and adds an additive term to the solution $T(n)$ of the form:

$$T(n) = \quad \cdots \quad + (c_{i,1} + c_{i,2}\,n + \cdots + c_{i,m_i}\,n^{m_i})(r_i)^n \; + \quad \cdots$$

and there are as many terms in the solution as there are distinct roots in the extended characteristic equation. After finding the roots (usually a numeric process, unless you use the matrix-formalism discussed later), you still need to find values for all the constants. These can be found by setting up the boundary equations. Fortunately, these are all linear equations and the values could be found, perhaps with the aid of an computer algebra system, such as MAPLE. This will not be pursued here, but we will show a number of examples. The examples have been carefully constructed to have zero's that are natural numbers. This can not be expected in everyday situations.

	From $T(n) = 4\,T(n-1) - 3\,T(n-2) + f(n)$ to closed form, $T(0) = 0, T(1) = 1$ for all examples.			
	$f(n) =$	Factored Extended polynomial	Solution	Asymptotic DT
1.	0	$(x-1)\,(x-3)$	$-\frac{1}{2} + \frac{1}{2}\,3^n$	$\sim \frac{1}{2}\,3^n$
2.	2^n	$(x-1)\,(x-3)\,(x-2)$	$\frac{3}{2} - 4\,2^n + \frac{5}{2}\,3^n$	$\sim \frac{5}{2}\,3^n$
3.	$n\,2^n$	$(x-1)\,(x-3)\,(x-2)^2$	$-\frac{1}{2} - 4\,(n+2)\,2^n + \frac{17}{2}\,3^n$	$\sim \frac{17}{2}\,3^n$
4.	$n^2\,2^n$	$(x-1)\,(x-3)\,(x-2)^3$	$\frac{7}{2} - 4\,(n^2+4n+12)\,2^n + \frac{89}{2}\,3^n$	$\sim \frac{89}{2}\,3^n$
5.	2	$(x-1)^2\,(x-3)$	$-(n+1) + 3^n$	$\sim 3^n$
6.	$2\,n$	$(x-1)^3\,(x-3)$	$-\frac{1}{4}(2\,n^2 + 8\,n + 7)\; + \frac{7}{4}\,3^n$	$\sim \frac{7}{4}\,3^n$
7.	$-2^n + 3^n$	$(x-1)^3\,(x-2)\,(x-3)^2$	$-\frac{1}{4} + 4\,2^n + \frac{3}{4}\,(2n-5)3^n$	$\sim \frac{3}{2}n3^n$

Table 5: Finding the explicit closed form for $T(n)$.

3.2.4 Asymptotics

The asymptotic classification of $T(n)$ is related to the largest root, which is always positive in our application area. If r is the largest root, and if m is it's multiplicity, then $T = \Theta(\, n^{m-1} r^n\,)$.

3.2.5 Practice Problems

Solve the following recurrence relations that can be solved with the techniques presented in this section. In particular, find the associated and the extended polynomials, factorize the latter polynomial, and completely solve the recurrence relations, where you identify the asymptotic classification. I would encourage you to use computational devices for intermediate steps.

1. $\begin{cases} T(n) = n & n \le 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) + 4 & n > 2 \end{cases}$

2. $\begin{cases} T(n) = n & n \le 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) - 4 & n > 2 \end{cases}$

3. $\begin{cases} T(n) = n & n \le 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) + 4\,n & n > 2 \end{cases}$

4. $\begin{cases} T(n) = n & n \le 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) + 4^n & n > 2 \end{cases}$

5. $\begin{cases} T(n) = n & n \le 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) - 4\,(-1)^n & n > 2 \end{cases}$

6. $\begin{cases} T(n) = n & n \le 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) - 4\,n\,(-1)^n & n > 2 \end{cases}$

The next section is pretty much a repeat of this section, except that higher order recurrence relations are addressed. Both techniques and results are similar, but notation is much more meticulous.

4 Higher order linear recurrence relations

In this section, we extend the results for the two-term recurrence relations to a higher order. The approach is still pretty much the same as in the second order, except that we need to introduce yet another variable, k.

$$\begin{cases} T(n) = t_n & n \leq k \\ T(n) = a_1\, T(n-1) + a_2\, T(n-2) + \cdots + a_k\, T(n-k) & n > k \end{cases}$$

Again, these occur often in combinatorial situations, in an occasional algorithm, and as an intermediate step in solving inhomogeneous recurrence relations of a certain kind. The form of the solution is given by terms like $T(n) = \sum c_i r_i^n$ or $T(n) = (c_{i,1} + c_{i,2}\, n) \times r_i^n$. Again, the formal proof that these forms are the only possible solutions relies on generating functions, and will not be shown. For now, we can reason exactly as before. Since the closed form solution to the one-term and two-term recurrence relations $T(n) = a_1\, T(n-1),\ n > 1$ and $T(n) = a_1\, T(n-1) + a_2\, T(n-2),\ n > 2$ is $T(n) = t_0\, r_1^n$ and $T(n) = c_1\, r_1^n + c_2\, r_2^n$ we are inspired *to try* again a solution of the similar form, $T(n) = c\, x^n$, and will find limiting conditions on the values of x for which this may satisfy the equation. So substitute the trial solution into the defining equation:

$$\begin{aligned} T(n) &= a_1\, T(n-1) + a_2\, T(n-2) + \cdots + a_k\, T(n-k) \\ x^n &= a_1\, x^{n-1} + \quad a_2\, x^{n-2} \quad + \cdots + \quad a_k\, x^{n-k} \end{aligned}$$

Divide both sides of the equation by x^{n-k}, and the k- term recurrence induces a polynomial equation of degree k:

$$x^k - a_1\, x^{k-1} - a_2\, x^{k-2} - \cdots - a_k\ = 0.$$

So, if the solution is to be of the form $T(n) = c\, x^n$, then x must satisfy this polynomial quadratic equation. There are k roots of this equation, counting multiplicities, and we must keep all roots open as a possibility. Thus the roots, with their multiplicities, determine the exact form of the solution. The roots can be either real or complex valued, and each root as a multiplicity of at least one. Furthermore, for each root, we introduce as many constants as the multiplicity of the root. In total, there are k constants to be introduced that are as yet unknown, and their values an be determined by setting up k equations with k unknowns. We will show several examples of the solution instead. Let r_1 be a root with multiplicity 1, 2 or 3, then the solution is of the corresponding form:

Examples of solution form :

$$\begin{aligned} T(n) &= c_1(r_1)^n + c_2(r_2)^n + c_3(r_3)^n + c_4(r_4)^n + \cdots + c_k(r_k)^n \\ \text{or}\quad T(n) &= (c_{11} + c_{12}\, n)(r_1)^n + c_3(r_3)^n + c_4(r_4)^n + \cdots + c_k(r_k)^n \\ \text{or}\quad T(n) &= (c_{11} + c_{12}\, n + c_{13}\, n^2)(r_1)^n + c_4(r_4)^n + \cdots + c_k(r_k)^n \end{aligned}$$

Thus, in general, we can state:

Each distinct root r_i of the characteristic polynomial and with a multiplicity m_i adds an additive term to the solution $T(n)$:

$$T(n) = \quad \cdots \quad + (c_{i,1} + c_{i,2}\, n + \cdots + c_{i,m_i}\, n^{m_i - 1})(r_i)^n \ + \quad \cdots$$

If the coefficients $a_0 \cdots a_k$ are all real, then the complex roots, if any, come in complex conjugate pairs, which can then be combined again to write the solution with real components only. This will not be pursued here. Concluding, the solution is straightforward once the roots have been identified, yet the notation is cumbersome. Generally, you can find the roots algebraically for $k = 1, 2, 3, 4$, while for $k = 5$ or higher the roots can be found only numerically (Galois theory). Unfortunately, finding the exact roots from the coefficients a_i is generally an ill-conditioned problem and, this topic falls outside the current course.

The asymptotic classification is also easy identified, once all the roots are known: $T = \Theta(\ n^m\ r^n\ )$, where r is the root with largest absolute value, and m it's multiplicty. This largest root is easily found, once all roots have been found. But think frugally: why go through the trouble, both time- wise and numerically wise, to find *all* the roots if you are interested in *only one* of them, even though it should have the property of being the largest? Again, this interesting topic goes beyond the current exposé, and might be included sometime as an appendix. (See also the linear algebra viewpoint, far below).

4.1 Steps for Solutions

The problem of finding the closed form expressions for $T(n)$ can thus be broken down in a few steps:

Step 1. Make sure it is a linear homogeneous recurrence relation with constant coefficients

Step 2. Identify the associated polynomial

Step 3. Factor the associated polynomial, or otherwise find the roots with their multiplicities. This is not covered in this class. The quadratic formula is usually presented in high school, factoring cubic equations is presented both in high school and in college algebra, where also special higher order equations are covered. Feel free to use a CAS system for this part, such as MAPLE.

Step 4. For each root $r_.$ with multiplicity m, add a term $(c_{.,1} + c_{.,2}\, n + \cdots + c_{.,m}\, n^m)(r_.)^n$ to the solution

Step 5. Set up a system of k equations in the k unknown values of the constants c_i or $c_{.,-}$

Step 6. Solve this system of equations. Again, this is not covered in this course. You should be able to solve systems with two or three equations by hand. For larger systems, or for coefficients and root values that are not easily manipulated by hand, and you should use a CAS system such as (MAPLE), or use a numerical package with care.

Step 7. Verify your work, by computing the value of $T(n)$ for one or more values of n. (Yes, we all make mistakes, so double check your answers yourself)

4.2 Examples

Notice that all examples shown in this table are of order 4, and that there are 4 constants, whose values need to be determined from the initial values of $T(0)..T(3)$ (or from any 4 consecutive initial values that have incorporated the initial, or boundary values). Pay attention to how we go from recurrence relation to associated polynomial, and then to factored polynomial, then to identifying roots with their multiplicity, and finally to the closed form for the solution $T(n)$.

	Examples to handle fourth-order recurrence relations.	
1.	Recursive Def.	$T(n) = 14\,T(n-1) - 71\,T(n-2) + 154\,T(n-3) - 120\,T(n-4)$
	Assoc. Polynomial	$x^4 = 14\,x^3 - 71\,x^2 + 154\,x - 120$
	Factored Polynomial	$(x-2)\,(x-3)\,(x-4)\,(x-5) = 0$
	Closed Form	$T(n) = c_1\,(2)^n + c_2\,(3)^n + c_3\,(4)^n + c_4\,(5)^n$
	Asymptotic	$T(n) = c_4\,5^n + o(5^n) = \Theta(5^n)$
2.	Recursive Def.	$T(n) = 10\,T(n-1) - 5\,T(n-2) - 160\,T(n-3) + 336\,T(n-4)$
	Assoc. Polynomial	$x^4 = 10\,x^3 - 5\,x^2 - 160\,x + 336$
	Factored Polynomial	$(x-3)\,(x-4)\,(x+4)\,(x-7) = 0$
	Closed Form	$T(n) = c_1\,(3)^n + c_2\,(4)^n + c_3\,(-4)^n + c_4\,(7)^n$
	Asymptotic	$T(n) = c_7\,7^n + o(7^n) = \Theta(7^n)$
3.	Recursive Def.	$T(n) = 8\,T(n-1) - 24\,T(n-2) + 32\,T(n-3) - 16\,T(n-4)$
	Assoc. Polynomial	$x^4 = 8\,x^3 - 24\,x^2 + 32\,x - 16$
	Factored Polynomial	$(x-2)^4 = 0$
	Closed Form	$T(n) = (c_{11} + c_{12}\,n + c_{13}\,n^2 + c_{14}\,n^3)\,(2)^n$
	Asymptotic	$T(n) = c_{14} n^3\,2^n + o(n^3\,2^n) = \Theta(n^3\,2^n)$
4.	Recursive Def.	$T(n) = 4\,T(n-1) + 18\,T(n-2) + 20\,T(n-3) + 7\,T(n-4)$
	Assoc. Polynomial	$x^4 = 4\,x^3 + 18\,x^2 + 20\,x + 7$
	Factored Polynomial	$(x+1)^3\,(x-7) = 0$
	Closed Form	$T(n) = (c_{11} + c_{12}\,n + c_{13}\,n^2)\,(-1)^n + c_4\,(7)^n$
	Asymptotic	$T(n) = c_4\,7^n + o(7^n) = \Theta(7^n)$
5.	Recursive Def.	$T(n) = 10\,T(n-1) - 37\,T(n-2) + 60\,T(n-3) - 36\,T(n-4)$
	Assoc. Polynomial	$x^4 = 10\,x^3 - 37\,x^2 + 60\,x - 36$
	Factored Polynomial	$(x-2)^2\,(x-3)^2 = 0$
	Closed Form	$T(n) = (c_{11} + c_{12}\,n)\,(2)^n + (c_{21} + c_{22}\,n)\,(3)^n$
	Asymptotic	$T(n) = c_{22} n\,3^n + o(n\,3^n) = \Theta(n\,3^n)$

4.3 Practice Problems

Identify the following recurrence relations as either a *linear homogeneous recurrence relations of kth order with constant coefficients*, or not of this type. Also, for those that are, find the associated polynomial and its degree.

1. $T(n) = 4\,T(n-1)$
2. $T(n) = 3\,T(n-1) + 4\,T(n-2)$
3. $T(n) = 2\,T(n-1) + 8\,T(n-2)$
4. $T(n) = T(n-1) + 12\,T(n-2)$
5. $T(n) = 13\,T(n-1) + 12\,T(n-2)$
6. $T(n) = 12\,T(n-1) + 16\,T(n-2)$
7. $T(n) = 2\,T(n-1) + 7\,T(n-2) + 4\,T(n-3)$
8. $T(n) = T(n-1) + 10\,T(n-2) + 8\,T(n-3)$
9. $T(n) = 13\,T(n-2) + 12\,T(n-3)$
10. $T(n) = 6\,T(n-2) + 8\,T(n-3) + 3\,T(n-4)$
11. $T(n) = T(n-1) + 9\,T(n-2) + 11\,T(n-3) + 4\,T(n-4)$

Identify the form of the explicit closed form, if the factorization of the associated polynomial is given below.

12. $(x-3)\,(x-4)\,(x+4)\,(x-7)\,(x-8) = 0$
13. $(x-3)\,(x-4)^2\,(x+4)^3\,(x-7)^3 = 0$
14. $(x-\sqrt{3}\,)\,(x-\sqrt{4}\,)^2\,(x+\sqrt{4}\,) = 0$
15. $(x-333)\,(x-444)\; = 0$

For the next group of questions, completely solve the recurrence relations, and identify the asymptotic classification, that is, find the simplest f, so that $T = \Theta(\;f\;)$. You should use a computational tool to help you with these. The first recurrence relation, for instance, can be solved with MAPLE with the statement as shown. Note the difference between the smooth and curly brackets.

```
rsolve({T(0) = 0, T(1) = 1, T(n) = 2*T(n-1)+3*T(n-2)}, {T});
```

16. $\begin{cases} T(n) = n & n \le 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) & n > 2 \end{cases}$

17. $\begin{cases} T(n) = n-1 & n \le 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) & n > 2 \end{cases}$

18. $\begin{cases} T(n) = n+1 & n \le 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) & n > 2 \end{cases}$

19. $\begin{cases} T(n) = n^2 & n \le 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) & n > 2 \end{cases}$

20. $\begin{cases} T(n) = n & n \le 3 \\ T(n) = 2T(n-1) + 3\,T(n-2) + 4\,T(n-3) & n > 3 \end{cases}$

4.4 Inhomogeneous versions

A recurrence relation is homogeneous if the all the terms on the right-hand side involve terms like $T(n-i)$. This is rarely the case for time complexity studies. The inhomogeneous part is simply the expression that is left after all the terms with $T(n-i)$ have been removed, and we will use $f(n)$ to indicate the inhomogeneous part. Generally speaking, inhomogeneous recurrence relations are hard to solve, except if the both the homogeneous and the inhomogeneous parts are of a particular form. These are discussed in this section, and we start with the inhomogeneous part being an exponential term, that is, n appears in the exponent, $f(n) = a_s\, s^n$ for some constants a_s and s.

4.4.1 Inhomogeneous function $f(n) = a_s\, s^n$ for some constants a_s and s

Consider now

$$\begin{cases} T(n) = t_n & n \le k \\ T(n) = a_1\, T(n-1) + a_2\, T(n-2) + \cdots + a_k\, T(n-k) + a_s s^n & n > k \end{cases}$$

Notice that we studied the homogeneous part in the previous section, whose solution could be found by finding the roots of the associated polynomial. It is not hard to show that the current situation, two consecutive recurrence relations can be used to solve for the inhomogeneous part, which can then be eliminated, essentially removing the homogeneous part at the cost of increasing the order. The recurrent definition for $T(n-1)$ can be used to solve for s^{n-1}:

$$\begin{aligned} T(n-1) &= a_1\, T(n-2) + a_2\, T(n-3) + \cdots + a_k\, T(n-k-1) + a_s s^{n-1} \\ &\ \text{thus, isolating the inhomogeneous part,} \\ a_s s^{n-1} &= T(n-1) - a_1\, T(n-2) - a_2\, T(n-3) - \cdots - a_k\, T(n-k-1) \\ &\ \text{and multiply both sides by } s \\ a_s s^n &= s\ (T(n-1) - a_1\, T(n-2) - a_2\, T(n-3) - \cdots - a_k\, T(n-k-1)\ ) \end{aligned}$$

Substitute this part into the recurrent definition for $T(n)$:

$$\begin{aligned} T(n) &= \quad a_1\, T(n-1) + a_2\, T(n-2) + \cdots + a_k\, T(n-k) \\ &\quad + \ s\ (T(n-1) - a_1\, T(n-2) - a_2\, T(n-3) - \cdots - a_k\, T(n-k-1)\ ) \end{aligned}$$

So we have identified another recurrence relation that is homogeneous, of one degree higher than the one we started with, and whose solution is identical to the $T(n)$ we started with. To find the associated characteristic equation, we first substitute $T(n) = c\, x^n$, and then divide by x^{n-k-1}. This gives:

$$x^{k+1} - a_1\, x^k - a_2\, x^{k-1} - a_3\, x^{k-2} \cdots - a_k\, x \quad + s\, \Big(x^k - a_1\, x^{k-1} - a_2\, x^{k-2} - a_3\, x^{k-3} \cdots - a_k\Big) = 0$$

which can be factored in:

$$\Big(x^k - a_1\, x^{k-1} - a_2\, x^{k-2} - a_3\, x^{k-3} \cdots - a_k\Big)\, (x-s) = 0 \tag{4}$$

So, not only have we identified another recurrence relation that is homogeneous, of one degree higher than the inhomogeneous one we started with, and whose solution is identical to the original $T(n)$, we have also shown that the characteristic equation of this extended recurrence relation is the product of the original characteristic equation, and the term $(x-s)$.

This results can be extended quite a bit, but any proofs are omitted. In fact, the inhomogeneous part can be of the form $p_d(s)\, s^n$, where $p_d(s)$ is a polynomial of order d in s, or even sums of such terms with differing values of s: $p_{d'}(s')\, s'^n$. For each term $p_d(s)\, s^n$ in the inhomogeneous part, the characteristic polynomial of the homogeneous part is multiplied by $(x-s)^{d+1}$. We will show these forms by example only, in the hope to avoid the introduction of many more variables and meticulous manipulations.

So we started with an inhomogeneous recurrence relation, where both the homogenous part and the inhomogeneous part had a

	The extended characteristic equation for $T(n)$ when the inhomogeneous form $f(n)$ is as indicated. $T(n) = a_1\,T(n-1) + \cdots + a_k\,T(n-k) + f(n)$	
	Inhomogeneous function $f(n)$	Extended characteristic polynomial
1.	$f(n) = s^n$	$\big(x^k - a_1\,x^{k-1} \cdots - a_k\,\big)\,(x-s)$
2.	$f(n) = 2\,3^n$	$\big(x^k - a_1\,x^{k-1} \cdots - a_k\,\big)\,(x-3)$
3.	$f(n) = (1+2\,n+n^3)\,3^n$	$\big(x^k - a_1\,x^{k-1} \cdots - a_k\,\big)\,(x-3)^4$
4.	$f(n) = (1+2\,n+n^7)\,3^n$	$\big(x^k - a_1\,x^{k-1} \cdots - a_k\,\big)\,(x-3)^8$
5.	$f(n) = (1+2\,n)\,3^n\ + (2\,n+n^3)\,4^n$	$\big(x^k - a_1\,x^{k-1} \cdots - a_k\,\big)\,(x-3)^2\,\ (x-4)^4$
6.	$f(n) = (1+2\,n)\,3^n\ + (2\,n+n^3)$	$\big(x^k - a_1\,x^{k-1} \cdots - a_k\,\big)\,(x-3)^2\,\ (x-1)^4$

Table 6: The extended characteristic polynomial for $T(n) = a_1\,T(n-1) + a_2\,T(n-2) + \cdots + a_k\,T(n-k) + f(n)$ and various forms of $f(n)$.

particular (but still fairly common) form. For these we identified another recurrence relation that is homogeneous, and had a higher degree than the inhomogeneous one we started with. These are two recurrence relations, describing the same sequence $T(n)$. The solution is thus described in terms of the roots of the extended characteristic equation. First we must make a note of caution: We extended the characteristic equation by (a power of) $\times(x-s)$, and if s already is a root of the original characteristic polynomial, then their multiplicities must be added, before drawing up the general solution.
Thus, as before, each distinct root r_i of the extended polynomial has a multiplicity m_i, and adds an additive term to the solution $T(n)$ of the form:

$$T(n) = \quad \cdots \quad + (c_{i,1} + c_{i,2}\,n + \cdots + c_{i,m_i}\,n^{m_i})(r_i)^n\ + \quad \cdots$$

Finding the closed form of the solution

Although we now have found the values of the roots, we still need the values for the coefficients c_i or $c_{-,i}$, and there are now a total of $k\ + d + 1$ or $k\ + d + 1\ + d' + 1$, and so on. These can be found by setting up the boundary equations. Fortunately, these are all linear equations and the values could be found, perhaps with the aid of an CAS, such as MAPLE. Note, that we need at least k new pieces of information. In fact, $d+1$ (or $d+1\ + d'+1$) of the constants could be determined by substituting the solution form into the original defining recurrence. This will not be pursued here.

Finding the asymptotic form of the solution

Finding the explicit closed form is numerically fairly straightforward for the cases covered so far, but also a lot of work. If only the asymptotic behavior is desired, then much can be shortened.

4.5 Practice Problems

Solve the following recurrence relations that can be solved with the techniques presented in this section. In particular, find the associated and the extended polynomials, factorize the latter polynomial, and completely solve the recurrence relations, where you identify the asymptotic classification. I would encourage you to use computational devices for intermediate steps.

1. $\begin{cases} T(n) = n & n \le 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) + 4 & n > 2 \end{cases}$

2. $\begin{cases} T(n) = n & n \leq 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) - 4 & n > 2 \end{cases}$

3. $\begin{cases} T(n) = n & n \leq 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) + 4\,n & n > 2 \end{cases}$

4. $\begin{cases} T(n) = n & n \leq 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) + 4^n & n > 2 \end{cases}$

5. $\begin{cases} T(n) = n & n \leq 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) - 4\,(-1)^n & n > 2 \end{cases}$

6. $\begin{cases} T(n) = n & n \leq 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) - 4\,n\,(-1)^n & n > 2 \end{cases}$

5 Case Study: Analysis of AVL trees

If you have a regular binary search tree, then the standard operations (find-insert-delete) have a worst case time complexity that is linear, $T^W(n) = \Theta(n)$. This is because there is no limitation on the structure of a regular binary tree, and a binary search tree could be envisioned where all internal nodes have exactly one child. Early attempts to balance the binary search tree had a high cost to maintain this balance. An AVL tree is a binary search tree that is structurally constrained so that height of the left subtree and the height of the right sub tree differ by no more than one. Furthermore, the supporting algorithms for inserting and deleting elements from the AVL tree that maintain this AVL-property have acceptable overhead. In other words, the left and right subtree are balanced in some way, but not perfectly. We will show that the depth of an AVL is $\Theta(\ \lg n\ )$, so that the worst-case time complexity of the standard operations is $\Theta(\ \lg n\ )$ as well. Historically, the AVL tree is the first binary search tree structure with logarithmic performance. The implementation relies on maintaining, at each node, information on height-balance which is used to decide when and how to rotate nodes. But this additional information, to be stored at each node, which makes it less popular as compared to other binary search tree structures with logarithmic performance that were introduced later. This note does not explain the supporting algorithms, and we refer to text books for this. The worst case analysis depends on the height (or depth) of an AVL tree, and this note analyses the structure of an AVL tree to conclude that the height of an AVL-tree with n nodes is $\Theta(\log n)$. In fact, we show below that

THEOREM Let the number of nodes in an AVL tree be given as n. Then the height of this tree, $h(n)$, has upper and lower bounds that are both in the asymptotic logarithmic class $\Theta(\log n)$, and thus also $h(n) = \Theta(\log n)$. In fact we show the dominant coefficient is between 1 and 1.44:

$$\boxed{\lg n\ \leq h(n) \leq\ 1.44\ \lg n} \tag{5}$$

Since both lower— and upper bounds have a logarithmic growth, we can also conclude that for all standard operations on an AVL tree, we have $T^W(n) = \Theta(\log n)$.

Sketch of the proof For a given n, however, it is not easy to come up with explicit expressions for a its height, or even lower– and upper bounds. In stead, we will identify the boundaries of the area in the positive half-plane containing all valid pairs of n and corresponding h. We find this boundary by asking: How many elements (i.e. nodes) are needed to construct an AVL tree with given height h? How many nodes do you need minimally, and how many nodes can you tolerate maximally?

THEOREM Let the height of an AVL tree be given (and fixed) at h. Then the number of nodes in this tree, $n(h)$, has upper and lower bounds that are both in the asymptotic exponential class, and thus also $n(h) = \Theta(c^h)$. In fact, the coefficients for lower and upper bounds are known as well :

$$\boxed{1.89\ \phi^h \leq n(h) \leq 2\ 2^h,\ \text{ where }\ \phi = 1.61\ \text{ is the golden ratio}} \tag{6}$$

We then turn this inequality around and find the minimal and maximal height h for given n, and thereby prove the theorem.

Thus, we reason as follows: If you have n nodes, and want the smallest height tree, you will construct a tree where as many nodes as possible have both children (the thickest tree). On the other hand, if you want an upper bound on the height, you will try to construct an AVL tree where the nodes have the fewest children allowed without violating the AVL property (the skinniest tree). We will perform these constructions, find correct relationships between n and h, and then turn these relationships around to find lower– and upper bounds for the height.

5.1 Bounds for the number of nodes in an AVL tree with given height h

5.1.1 Upper-bound, $M(h)$ for given height h

Let $M(h)$ be the maximal number of nodes that can be used to create an AVL-tree with height h. In fact, we do not need to make special use of the AVL property, and these results are valid for regular binary trees as well. By looking at some small values, we get $M(0) = 1$, $M(1) = 3$, $M(2) = 7$, $M(3) = 15$, $M(4) = 31$, and so on. More generally, the tree with the next higher height can be constructed from one root and two subtrees that each have height $h-1$, and each has a maximal number of nodes, so the expression becomes a recurrence relation:

$$\boxed{\begin{cases} M(0) = 1 & h = 0 \\ M(h) = 1 + 2M(h-1) & h > 0 \end{cases}} \text{ and with solution } \boxed{M(h) = 2\,2^h - 1} \tag{7}$$

A table showing some of the values is readily generated,

height	0	1	2	3	4	5	6	7	8	9	...	h
$M(h)$	1	3	7	15	31	63	127	255	511	1023	...	$2^{h+1} - 1$

This table tells you that, given a value of h, the largest number of nodes you can accommodate in an AVL tree of height $h = 3$ is $M(3) = 15$, and so on. But you can also turn it around a little: If you want to construct an AVL tree with $n = 15$ nodes in the AVL tree then you can construct an AVL tree with height $h = 3$, but perhaps you can also construct AVL trees with height 4 or higher. Similarly, if you have $n = 16$ nodes, then you must construct an AVL tree with height at least $h = 4$. The same conclusion stands if you have $n = 25$ nodes, and if you have $n = 40$ nodes, then the AVL tree you construct has height at least 5. This indicates that the explicit expression for $M(h)$ can be used to find the minimal height of an AVL tree with a given number of n of nodes.

We will now use a similar process to find a lower bound for the height.

5.1.2 Lower-bound, $m(h)$ for given height h

Let $m(h)$ be the minimal number of nodes that can be used to create an AVL-tree with height h. We now need to make use of the special structural property of an AVL tree, and use that the height from the left and right subtrees differ by no more than one. In fact, for the minimal number of nodes with a given h, we construct AVL trees where the height from the left and right subtrees always differ by exactly one. We now get $m(0) = 1$, $m(1) = 2$, $m(2) = 4$, $m(3) = 7$, $m(4) = 12$, and so on. More generally, the tree with the next higher height can be constructed from one root and two subtrees that have heights $h-1$ and $h-2$, and each has a minimal number of nodes, so the expression becomes a recurrence relation:

$$\boxed{\begin{cases} m(0) = 1 & h = 0 \\ m(h) = 1 + m(h-1) + m(h-2) & h > 0 \end{cases}} \tag{8}$$

We can iterate the recurrence relation and generate a table of values:

height	0	1	2	3	4	5	6	7	8	9	...	h
$m(h)$	1	2	4	7	12	20	33	54	88	143	...	$\approx 1.9\,(1.62)^h - 1$

This table tells us e.g. that the minimal number of nodes you must have to construct an AVL tree of height $h = 3$ is $m(3) = 7$, and that the minimal number of nodes for a tree of height $h = 5$ is $m(5) = 20$. If you have $n = 20$ nodes in the AVL tree then you can construct an AVL tree with height $h = 5$, but if you have $n = 19$ nodes, then you do not have enough nodes to construct tree with $h = 5$, although you can construct an AVL tree with height $h = 4$. Finally, if you have $n = 25$ nodes in the AVL tree, then you have more than enough nodes to construct an AVL tree of height $h = 5$ (for which you need only $m(5) = 20$ nodes), but you do not have enough nodes to construct such a tree of height $h = 6$, because you would need $m(6) = 33$ nodes.

For our purposes we need to find an asymptotic equality. We can solve the recurrence relation explicitly, using the methods from the previous section/chapter, although this one is somewhat more involved. Note that the recursive part, $m(h) = 1 + m(h-1) + m(h-2)$, is similar to that of the Fibonacci numbers. In fact, by adding a 1 to both sides, we get $\underline{m(h) + 1} =$

$\underline{m(h-1)+1} + \underline{m(h-2)+1}$, and the underlined terms satisfy the Fibonacci recurrence. Thus,

$$m(h)+1 = \left(\frac{5+2\sqrt{5}}{5}\right)\left(\frac{1+\sqrt{5}}{2}\right)^h + \left(\frac{5-2\sqrt{5}}{5}\right)\left(\frac{1-\sqrt{5}}{2}\right)^h \quad (9)$$

$$= \alpha \quad \phi^h \quad + \quad \beta \quad \phi_0^h \quad (10)$$

$$\approx 1.8944\,(1.61803)^h \quad + 0.1056\,(-0.61803)^h \quad (11)$$

and

$$m(h) \approx 1.8944\,(1.61803)^h \quad (12)$$

where we introduced the constants α, β, ϕ (*the golden ratio*) and ϕ_0 for notational convenience. Notice that ϕ_0 is less than 1 (in absolute value) so that $0.1056\,(-0.61803)^h \to 0$, as $h \to \infty$, which justifies the last approximation.

5.1.3 Combined bound for number of nodes in AVL tree with given height h

The result of the previous sections can now be combined in a theorem.

THEOREM Let the height of an AVL tree be given (and fixed) at h. Then the number of nodes in this tree, $n(h)$, has upper and lower bounds as follows:

$$\boxed{1.8944\times\, \phi^h \approx m(h) \leq n(h) \leq M(h) \leq 2\times\, 2^h,} \quad (13)$$

5.2 Bounds for the height of an AVL tree with given number of nodes, n

5.2.1 Lower-bound, $h_m(n)$ for given n

As explained above, the expression 'For a given height h, you can have at most $M(h)$ nodes, where $M(h) = 2^h - 1$' can be used to deduce an expression for 'For a given number of nodes n, what is the minimal height that must be constructed?' Let $h_m(n)$ indicate this minimal height for a given n. These two expressions share the same relation ship $n = 2^{h_m(n)} - 1$, and can be solved for $h_m(n)$ as $h_m(n) = \lg(n+1)$. This equation is only valid for "special" values of n, such that $\lg(n+1)$ is an integer. To extend it for all n, we can use the table again to reason: we get for $n = 1, 3, 7, 15, \cdots$ that $h_m(1) = 0$, $h_m(3) = 1$, $h_m(7) = 2$, $h_m(15) = 3$, and so on. Also, for any number 'in between' you need to get the next higher value. This leads to the wanted result: Given n nodes in an AVL tree, then the actual height of such an AVL is at least $h_m(n) = \lceil \log_2(n+1) - 1 \rceil$. Upon eliminate the ceiling notation, this is

$$\boxed{\log_2(n+1) - 1 \leq h_m(n)} \quad (14)$$

5.2.2 Upper-bound, $h_M(n)$ for given n

Quite similarly, the expression 'For a given height h, you need at least $m(h)$ nodes, where $m(h) \approx 1.8944\times\, \phi^h$' can be used to find an expression for 'Given a certain number of nodes n, what is the maximal height of the tree that can be constructed?' Let $h_M(n)$ indicate this maximal height for a given n. These two expressions share the same relation ship $n = 1.8944\times\, \phi^h_M(n)$, and can be solved for $h_M(n)$ as $h_M(n) = \log_\phi(n/1.8944)$. This equation is only valid for "special" values of n that result in an integer result. However, it can be shown that (we skip the derivation)

$$h_M(n) \approx \log_\phi\left(\frac{n}{\alpha}\right) = \log_\phi(n) - \log_\phi(\alpha) = \frac{\lg(n)}{\lg(\phi)} - \frac{\lg(\alpha)}{\lg(\phi)} \quad (15)$$

5.2.3 Combined bound on the height of an AVL tree with n nodes

Given an AVL tree with n nodes, then the height of this tree, $h(n)$, is bound below and above by

$$\boxed{log_2\,(n+1) - 1 \le h_m(n) \le h(n) \le h_M(n) \le \frac{\lg(n)}{\lg(\phi)} \;<\; 1.44\; \lg n} \tag{16}$$

THEOREM Let the number of nodes in an AVL tree be given as n. Then the height of this tree, $h(n)$, is bound between two constant multiples of $\lg n$, and thus also $h(n) = \Theta(\log n)$. In fact we show the dominant coefficient is between 1 and 1.44:

$$\boxed{\lg n \;\le h(n) \le\; 1.44\; \lg n} \tag{17}$$

Since both lower— and upper bounds have a logarithmic growth, we can also conclude that for all standard operations on an AVL tree, we have $T^W(n) = \Theta(\log n)$.

The proof is fairly straightforward for $n = 1, 2, 4, 7, 12, 20 \cdots$ that $h_M(1) = 0$, $h_M(2) = 1$, $h_M(4) = 2$, $h_M(7) = 3$, and so on. For a number 'in between' you need to get the previous, lower value. This leads to the wanted result: Given n nodes in an AVL tree, then the maximal height of such an AVL is approximately

$$h_M(n) \approx \log_\phi\left(\frac{m(h)+1}{\alpha}\right) \tag{18}$$

This is almost the bound we seek and we need to be just a little more careful. It is left as an exercise to show that you can subtract a fixed value 1 to get a lower bound, and this -1 disappears when taking the limit. The result is that, for all n,

$$\left\lfloor \log_\phi\left(\frac{n}{\alpha}\right) \right\rfloor \le h_M(n) \le \left\lfloor \log_\phi\left(\frac{n+2}{\alpha}\right) \right\rfloor \tag{19}$$

For our purpose, we only need the upper bound, and we can eliminate the floor notation, convert to $log-$ base 2, and convert to numerical values:

$$\boxed{h_M(n) \le\; \log_\phi\left(\frac{n+2}{\alpha}\right) \;=\; \log_\phi(n+2) - \log_\phi(\alpha) =\; 1.440420092\; \lg(n+2) - 1.327724} \tag{20}$$

5.3 Conclusion and closing remarks

Given n nodes in an AVL tree, then the height of this tree is bound below by $h_m(n)$ and bound above by $h_M(n)$, for which we have upper and lower bounds:

$$\boxed{\lg(n+1) - 1 \;\le h_m(n) \le h(n) \le h_M(n) \le 1.440420092\; \lg(n+2) - 1.327724} \tag{21}$$

Since both lower— and upper bounds have a logarithmic growth, we can also conclude that for all standard operations on an AVL tree, we have

$$\boxed{T^W(n) = \Theta(\log n)} \qquad \text{or more precisely} \qquad \boxed{T^W(n) \sim c\log n,\; \text{with } 1 \le c \le 1.44} \tag{22}$$

We should note, that the ratio of upper bound to lower bound is only about 1.44, which indicates a fairly narrow range for the height.

5.4 Advanced Topic: Drilling down the numbers

This section can be skipped for CS404 and CS5592. I am using it as a placeholder some ongoing questions.

5.4.1 Advanced Topic: Counting Skinny AVL trees

In the above, we computed the minimal number of nodes needed to construct an AVL tree with given height h. It was constructed by combining a new root-node with two such AVL-trees, one of height $h-1$, the other of height $h-2$. It did not matter which of these was left or right, which begs the question: How many skinny AVL trees can be constructed, with height h? Let $S(h)$ represent this number, then this is

$$\boxed{\begin{cases} S(0) = 1 & h = 0 \\ S(1) = 2 & h = 1 \\ S(h) = S(h-1)S(h-2) + S(h-2)S(h-1) = 2S(h-1)S(h-2) & h > 1 \end{cases}} \tag{23}$$

The solution is interesting: $S(h) = \boxed{2^{\text{FIB'}(h)}}$, where $\text{FIB'}(h) = 0, 1, 2, 4, 7\ \ldots$ are one number away from the regular Fibonacci numbers. This sequence has been studied and is known as A174677 in the Integer Sequence webpage.

5.4.2 Advanced Topic: Counting ALL AVL trees

We can continue this line of reasoning, and go after the total number of AVL trees that can be constructed. Let $A(h)$ represent this number, then

$$\boxed{\begin{cases} A(0) = 1 & h = 0 \\ A(1) = 3 & h = 1 \\ A(h) = A(h-1)A(h-2) + A(h-2)A(h-1) + A(h-1)A(h-1) & h > 1 \end{cases}} \tag{24}$$

This recurrence relation does not appear to have a closed form solution, and the numbers become very large, very quickly: $A(h) = 1, 3, 15, 315, 108675, 11878720875\ldots$. This sequence has been studied and is known as A029758 in the Integer Sequence webpage. It was studied by Knuth and others.

5.4.3 Advanced Topic: Counting Nodes at different levels

Given a minimal AVL tree with height h, then the next question is: How many nodes are there are level ℓ, for $\ell = 1\ \ldots\ h$? Let $L(\ell, h)$ be this number, then this should be $2^0, 2^1, 2^2\ \ldots$ for a while, but then no more, and the sequence ultimately becomes decreasing again. The recurrence relation is

$$\boxed{\begin{cases} L(h, 1) = 1 & h \geq 0 \\ L(h, \ell) = L(h-1, \ell-1) + L(h-2, \ell-1) & \ell < h, h \geq 1 \end{cases}} \tag{25}$$

This is readily implemented, even with a simple spreadsheet. Some observations are readily made, and then easily proven: (Assume that the root has $\ell = 1$, and a single (root-)node has height $h = 1$, for notational convenience.) So with this:

$$\boxed{\begin{cases} L(h, 1) = 1 & h \geq 0 \\ L(h, \ell) = 2^\ell & \ell < h/2, h \geq 1 \text{ and } h \text{ even} \\ L(h, \ell) = 2^\ell & \ell < (h-1)/2, h \geq 1 \text{ and } h \text{ odd} \\ L(h, h-1) = h & h \geq 1 \\ L(h, h) = 1 & h \geq 1 \end{cases}} \tag{26}$$

The recurrence relation itself strongly resembles that of the binomial coefficients, and so does the answer:

$$\boxed{L(h,\ell) = \sum_{i=0}^{\ell} \binom{h}{\ell} \quad h \geq 1} \tag{27}$$

Thus this is essentially the sum of the first $\ell + 1$ entries of the standard binomial (Pascal) triangle. It has also obvious connections with the binomial transform of a finite sequence of 1's: $L(h,\ell)$ is the binomial transform of $\langle 1, 1, \ldots 1, 0, 0, \ldots \rangle$ (there are ℓ entries equal to 1).

h= 1	1									
h= 2	1	1								
h= 3	1	2	1							
h= 4	1	3	3	1						
h= 5	1	4	6	4	1					
h= 6	1	5	10	10	5	1				
h= 7	1	6	15	20	15	6	1			
h= 8	1	7	21	35	35	21	7	1		
h= 9	1	8	28	56	70	56	28	8	1	
h=10	1	9	36	84	126	126	84	36	9	1
	...									

Figure 1: The Pascal triangle.

h= 1	1									
h= 2	1	1								
h= 3	1	2	1							
h= 4	1	2	3	1						
h= 5	1	2	4	4	1					
h= 6	1	2	4	7	5	1				
h= 7	1	2	4	8	11	6	1			
h= 8	1	2	4	8	15	16	7	1		
h= 9	1	2	4	8	16	26	22	8	1	
h=10	1	2	4	8	16	31	42	29	9	1
	...									

Figure 2: The Minimal AVL triangle.

6 Divide & Conquer Recurrences

Divide and conquer algorithms, as well as dynamic programming algorithms are based on finding solutions to smaller sized problems, which are then incorporated in the solution of the original problem. Take merge sort for example: Split the problem in two halves, sort each half separately (in parallel perhaps), and combine the sorted halves. The recurrence relation that represents the time complexity is

$$\begin{cases} T(1) = 0,\ T(2) = 1 \\ T(n) = T\left(\lfloor \frac{n}{2} \rfloor\right) + T\left(\lceil \frac{n}{2} \rceil\right) + f(n) \quad n > 2 \end{cases}$$

where $f(n)$ is the time complexity to merge the two halves. This section deals with the various variations of such recurrence relations. One of the difficulties is the appearance of the floor- and the ceiling functions, which are needed because the size of the two sub-problems are whole numbers. We could not use simply $\frac{n}{2}$, unless n is even. But if we assume that n is even, and that n is even at every level of recursion, then n is a power of 2. This is the recurring theme of this section. So, if we assume that n is a power of 2, that is $n = \{1, 2, 4, 8, 16, \cdots\}$, then the recurrence relation does not involve floor- and the ceiling functions, and we can perhaps find an explicit expression for $T(n)$ when n is limited to this set. We now have a recurrence relation where $T(n)$ is expressed into one recursive term on the right, in addition to an inhomogeneous term. We can bring this 'one term' recurrence relation (with the argument being halved) into another 'one term' recurrence relation (with the argument being reduced by one), by using substitution. This is what we show next.

General D&C recurrences This idea must be generalized to perhaps more than two parts, since Divide and conquer algorithms, as well as dynamic programming algorithms are based on finding solutions to multiple, but smaller sized problems, which are then incorporated in the solution of the original problem. Merge sort and (the best case of) Quick sort are examples, but we will encounter many more: Split the problem in several parts (often two halves), sort each part separately (in parallel perhaps), and combine the solutions of these parts. This is a recurring theme and this section treats recurrence relations of this basic form:

$$T(n) = \begin{cases} t & n = 1 \\ aT\left(\frac{n}{b}\right) + f(n) & n > 1. \end{cases}$$

We make a number of assumptions and to simplify the insight.

1. We assume that a and b are integers each larger than 1, with a possibly equal to 1. Although this is not a critical assumption, it is typical in computer science applications,
2. We assume that n is a power of b, that is $n \in \{1, b, b^2, b^3, b^4, \cdots\}$. This avoids invoking the floor and ceiling functions, and explicit expression for $T(n)$ can often be derived when n is limited to this set. In other words, $n = b^L$ for some $L = 0, 1, 2, \ldots$ and $L = \log_b n$. This again allows a substitution: replace $T(n) = aT\left(\frac{n}{b}\right) + f(n)$ by $S(m) = aS(m-1) + f(2^m)$ and use the known results for one-term recurrence relations.
3. If n is *not* a power of b, then n is between two consecutive powers of b. The only functions $f(n)$ we consider are boxed: if $b^i < n < b^{I+1}$ then $f(b^i) < f(n) < f(b^{I+1})$. This means that the function $f(n)$ is bound below by $f(b^i)$ and bound above by $f(b^{i+1})$, and thus the Θ– classification of $T(n)$ is equal to the Θ– classification of $T(n^L)$.

In this particular section, we aim to give intuition into the back substitution approach, which we introduced in the section on linear homogeneous recurrence relations, and we repeat the illustration: Write the equations underneath one another, and add all equations together: all terms to the left of the equality sign remain to the left in the result, and all the terms to the right remain in the right. We would like most of the T-terms to cancel, and this can happen by multiplying the equations by an appropriate power of a. After this, most terms cancel, by construction and the expression for $T(n)$ becomes: (Remember that $n = b^L$, so that $n/b^i \equiv b^{L-i}$),

$$\begin{array}{rcl}
T(n) &=& a\,T(\,n/b\,) + f(n)\\
a\Big(T(\,n/b\,) &=& a\,T(\,n/b^2\,) + f(\,n/b\,)\Big)\\
a^2\Big(T(\,n/b^2\,) &=& a\,T(\,n/b^3\,) + f(\,n/b^2\,)\Big)\\
a^3\Big(T(\,n/b^3\,) &=& a\,T(\,n/b^4\,) + f(\,n/b^3\,)\Big)\\
a^4\Big(T(\,n/b^4\,) &=& a\,T(\,n/b^5\,) + f(\,n/b^4\,)\Big)\\
\vdots && \vdots\\
a^{L-1}\Big(T(2) &=& a\,T(1) + f(2)\Big)\\
a^{L}\Big(T(1) &=& t\quad\Big)
\end{array}$$

$$\begin{array}{llllr}
 & T(n) & & \cancelto{a}{a\,T(\,n/b\,)} & +\quad f(n)\\
 & +\cancelto{a}{a\,T(\,n/b\,)} & & +\cancelto{b}{a^2\,T(\,n/b^2\,)} & +\,a\,f(\,n/b\,)\\
 & +\cancelto{b}{a^2\,T(\,n/b^2\,)} & & +\cancelto{c}{a^3\,T(\,n/b^3\,)} & +\,a^2\,f(\,n/b^2\,)\\
 & +\cancelto{c}{a^3\,T(\,n/b^3\,)} & & +\cancelto{d}{a^4\,T(\,n/b^4\,)} & +\,a^3\,f(\,n/b^3\,)\\
 & +\cancelto{d}{a^4\,T(\,n/b^4\,)} & & +\cancelto{e}{a^5\,T(\,n/b^5\,)} & +\,a^4\,f(\,n/b^4\,)\\
\text{add} & & \vdots\ =\ \vdots & & \\
 & +\cancel{a^{L-1}\,T(2)} & & +\cancel{a^L\,T(1)} & +\,a^{L-1}\,f(2)\\
 & +\cancel{a^L\,T(1)} & & & +\,a^L\,t\\
\hline
\text{result:} & T(n) & = & a^L\,t & +\sum_{i=1}^{L} a^{L-i}\,f(b^i)
\end{array}$$

The result is worth it's own equation (with number):

$$\boxed{T(n) = T(b^L) = a^L\,t + a^{L-1}f(b^1) + a^{L-2}f(b^2) + a^{L-3}\,f(b^3)\cdots + a\,f(b^{L-1}) + a^0 f(b^L)} \tag{28}$$

Notice that the number of terms in this summation is now $L = \log_b n$ (or $L+1$ is the boundary equation is counted). The exact solution is obtained for a fair number of functions, and these are all based on carefully manipulating the appropriate summations. To simplify matters, we first prove that if the function $f(n)$ is the sum of constituent functions, then the explicit expression for $T(n)$ can be written as the sum of explicit expressions.

Before going into the details, we would like to find expressions for L, a^L and b^L in terms of n. Since $L = \log_b n$, by definition, the term b^L is equal to n. Also, whenever a is an integer power of b, such as $a = b$, $a = b^2$ and so on, then $a^L = n$, or $a^L = n^2$ and so on. Whenever $a \neq b$ (or $a \neq b^2$), then the term a^L is more involved and can be written in more familiar notation using $x = b^{\log_b\ x}$ for all x. Generally, we have $a^L = (b^{\log_b\ a})^L = b^{(\log_b\ a)\times L} = b^{L\times(\log_b\ a)} = n^{(\log_b a)} = n^\alpha$ where $\alpha = \log_b a = \lg a/\lg b$.

6.1 Weak Master Theorem: Asymptotic Class of the Solution

We often need only the asymptotic behavior of the solution. Let us consider the full expression again, equation (28), but let us assume that $t = 0$, for ease of discussion (we will add it back at the appropriate time). Notice that the number of terms in this summation is $L = \log_b\ n$ and that there are two competing forces at work: Whenever $a > 1$, then the sequence $a^L, a^{L-1}, a^{L-2}, \cdots, a, 1$ is a deceasing sequence, whereas the sequence $f(b), f(b^2), \cdots f(b^L)$ is normally speaking an increasing sequence. The sequence of their individual products (this is called the Hadamard product) can be decreasing, more or less flat, or increasing. It just depends on the specific function $f(n)$: The sequence of products $a^{L-1}f(b)$, $a^{L-2}f(b^2), \cdots\ a^0 f(b^L)$ is normally one of three possibilities: increasing, constant, or decreasing. Thus the asymptotic character of their sum is similarly depending on this. The largest contribution to the sum $T(n)$ comes from either the first few terms, from all terms more or less equally, or from the last few terms. Intuitively, these three cases are characterized as:

1. If the first term is dominated by the last term in the sequence, or $n^\alpha = o(f)$, or equivalently $f = \omega(n^\alpha)$, then the largest contribution to $T(n)$ comes from the last few terms, $T(n) \approx \cdots + a^2 f(b^{L-2}) + af(b^{L-1}) + f(b^L)$.
2. If the first and last terms are pretty much equal, that is, if $n^\alpha = \Theta(f(n))$ or $f(n) = \Theta(n^\alpha)$, then all terms make a (more or less equal) contribution to $T(n)$. The number of terms is $\log_b n$ and the asymptotic complexity is $\log_b(n)\ f(n)$.
3. If the last term is dominated by the first term in the sequence, or if $f(n) = o(n^\alpha)$, then the largest contribution to $T(n)$ comes from the first few terms, $T(n) \approx a^L f(1) + a^{L-1}f(b^1) + a^{L-2}f(b^2) + a^{L-3}f(b^3)\cdots$.

So the asymptotic behavior is determined by the relative asymptotic behavior of $f(n)$ relative to $n^{\log_b a}$. If $f(n) = \Theta(n^\alpha)$, then all $\log n$ terms contribute just about equally. If either $f(n) = o(n^\alpha)$ or $f(n) = \omega(n^\alpha)$ then either end contributes most, and for technical reasons, we need $f(n)$ to be *polynomially separated* from n^α: we want either $f(n) = \mathcal{O}(n^{\alpha-\epsilon})$ or $f(n) = \Omega(n^{\alpha+\epsilon})$ for some ϵ, which must be a non-zero and positive number, even though it can be very small. Some other technicalities on regular behavior is stated in the theorem, which we now state without further proof.

WEAK MASTER THEOREM

The asymptotic behavior of $T(n)$, defined by the recurrence relation $T(n) = aT\left(\frac{n}{b}\right) + f(n)$, is (with $\alpha = \log_b a$):

$T(n) = \Theta(n^\alpha)$	if $f(n) = \mathcal{O}(n^{\alpha-\epsilon})$ for a positive ϵ	(case WMT-a)
$T(n) = \Theta(f(n)\log_b n)$	if $f(n) = \Theta(n^\alpha)$	(case WMT-b)
$T(n) = \Theta(f(n))$	if $f(n) = \Omega(n^{\alpha+\epsilon})$ for a positive ϵ and $af(n/b) \leq c' f(n)$ for some positive c'	(case WMT-c)
No conclusion	if $f(n)$ does not fall into above categories	(case WMT-n)

When to use the Weak Master Theorem

So you can use the WMT to classify the asymptotic growth class of a single algorithm whose complexity function is given by a covered recurrence relation. You can also use it to decide between two different algorithms if their asymptotic classes are different. If however $T_X(n) = \Theta(T_Y(n))$, then the WMT does not provide enough separation or distinction between the two. Below, we show that the Strong Master Theorem might be useful in these cases.

Polynomial Separation

The condition that $f(n)$ is polynomially separated from n^α is needed because no additional assumptions about $f(n)$ are made. For most functions, it is true that both "all terms in the summation are more-or-less equal" and the "case WMT-b" applies. Yet, there are also functions $f(n)$ so that "all terms in the summation are more-or-less equal" BUT their summation pushes the solution to outside "case WMT-b". Consider e.g. a specific recurrence relation that falls in this gap of the Master Theorem: $T(n) = 2\,T(n/2) + n\ln n$ (where we use the natural logarithm for ease of exposition). In this example, $a = b = 2$ so that $\alpha = 1$. Now it is clear that $f(n) = n\ \ln n$ and n^α are asymptotically different, so that "case WMT-b" does NOT apply. "Case WMT-a" would apply if $f(n) = \mathcal{O}(n^{\alpha-\epsilon})$ for a positive ϵ, but the reverse is true: $n^{\alpha-\epsilon} = \mathcal{O}(f(n))$ for ever positive ϵ, so that "case WMT-a" does NOT apply. Finally, "case WMT-c" would apply if $f(n) = \Omega(n^{\alpha+\epsilon})$ for a positive ϵ. But consider

$$\lim_{n\to\infty} \frac{f(n)}{n^{\alpha+\epsilon}} = \lim_{n\to\infty} \frac{n\ \ln n}{n^{1+\epsilon}} = \lim_{n\to\infty} \frac{\ln n}{n^\epsilon}$$

Consider using L'Hospital, and remembering that $1/n$ and $\epsilon n^{\epsilon-1}$ are the derivatives of $\ln n$ and n^ϵ

$$\lim_{n\to\infty} \frac{1/n}{\epsilon\, n^{\epsilon-1}} = \lim_{n\to\infty} \frac{1}{\epsilon\, n^\epsilon} = 0, \qquad \text{for every positive value of } \epsilon.$$

We can thus conclude that $f(n) = o(n^{\alpha+\epsilon})$ for every positive ϵ, and thus NOT that $f(n) = \Omega(n^{\alpha+\epsilon})$ for any positive value of ϵ and thus the "case WMT-c" does NOT apply either. The relation $T(n) = 2\,T(n/2) + n\ln n$ falls thus into the gap since it is not covered by cases "WMT-a", "WMT-b" or "WMT-c", and the Master Theorem cannot be used to reach a conclusion on the asymptotic type of $T(n)$. If you want to find the asymptotic behavior for this recurrence relation, you would either have to find the explicit form of $f(n)$, or use the Strong Master Theorem, which is covered in the next sub-section, can be used to show that the asymptotic behavior of $T(n)$, defined by $T(n) = 2\,T(n/2) + n\lg n$ (yes, we use the base-2 for the logarithm again), is $T(n) \approx \frac{1}{2} n\,(\lg n)^2$ and thus $T(n) = \Theta(f(n)\lg n)$.

6.2 Examples

1. $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = T(n/2) + 4 & 1 < n \end{cases}$

Note that $a = 1, b = 2$ and $\alpha = 0$, so that $f(n) = \Theta(n^{\alpha})$. Case WMT-b applies and $\boxed{T(n) = \Theta(\lg n)}$

2. $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = T(n/2) + n & 1 < n \end{cases}$

Note that $a = 1, b = 2$ and $\alpha = 0$, so that $f(n) = \Omega(n^{\alpha+\epsilon})$. Case WMT-c applies and $\boxed{T(n) = \Theta(n)}$

3. $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = T(n/2) + n^2 & 1 < n \end{cases}$

Note that $a = 1, b = 2$ and $\alpha = 0$, so that $f(n) = \Omega(n^{\alpha+\epsilon})$. Case WMT-c applies and $\boxed{T(n) = \Theta(n^2)}$

4. $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = 2\,T(n/2) + 4 & 1 < n \end{cases}$

Note that $a = 2, b = 2$ and $\alpha = 1$, so that $f(n) = \mathcal{O}(n^{\alpha-\epsilon})$. Case WMT-a applies and $\boxed{T(n) = \Theta(n)}$

5. $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = 2\,T(n/2) + n & 1 < n \end{cases}$

Note that $a = 2, b = 2$ and $\alpha = 1$, so that $f(n) = \Theta(n^{\alpha})$. Case WMT-b applies and $\boxed{T(n) = \Theta(n\ \lg n)}$

6. $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = 2\,T(n/2) + n^2 & 1 < n \end{cases}$

Note that $a = 2, b = 2$ and $\alpha = 1$, so that $f(n) = \Omega(n^{\alpha+\epsilon})$. Case WMT-c applies and $\boxed{T(n) = \Theta(n^2)}$

6.3 Practice Problems

The following problems will give you practice in applying this technique, but also gives you practice in adjusting the technique to accommodate slightly different conditions.

7. $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = 2\,T(n/2) + 2 & 1 < n \end{cases}$

8. $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = 2\,T(n/2) + n/2 & 1 < n \end{cases}$

9. $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = 2\,T(n/2) + 2\,n & 1 < n \end{cases}$

10. $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = 2\,T(n/3) + n^2 & 1 < n \end{cases}$

11. $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = 3\,T(n/2) + 2 & 1 < n \end{cases}$

12. $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = 3\,T(n/2) + 2\,n & 1 < n \end{cases}$

13. (grad students) $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = T(n/2) + \lg n & 1 < n \end{cases}$

14. (grad students) $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = 2\,T(n/2) + n\lg n & 1 < n \end{cases}$

15. (grad students) $\begin{cases} T(n) = 3 & n \leq 1 \\ T(n) = 2\,T(n/2) + n^2\lg n & 1 < n \end{cases}$

6.4 Exercises

1. The CDE (Chief Development Engineer) asked several different teams to design an algorithmic solution for a problem. The algorithmic performance for these solutions are given by recurrence relations.

part 1): Rank the algorithms from fastest to slowest when the problem size is 128
part 2): Rank the algorithms from fastest to slowest when the problem size is in the asymptotic region. For this part, to use the *Weak Master Theorem*.

$$T_X(n) = \begin{cases} 1 & n = 0 \\ 1000 & n = 1 \\ T_X(n/2) + 10\,n & 1 < n \end{cases} \qquad T_Y(n) = \begin{cases} 1 & n = 0 \\ 100 & n = 1 \\ T_Y(n/2) + 100\,n & 1 < n \end{cases} \qquad T_Z(n) = \begin{cases} 1 & n = 0 \\ 10 & n = 1 \\ T_Z(n/2) + 1000\,n & 1 < n \end{cases}$$

2. Repeat the exercise for

$$T_X(n) = \begin{cases} 1 & n = 0 \\ 5000 & n = 1 \\ T_X(n/2) + 10\,n^2 & 1 < n \end{cases} \qquad T_Y(n) = \begin{cases} 1 & n = 0 \\ 6000 & n = 1 \\ T_Y(n/2) + 100\,n & 1 < n \end{cases} \qquad T_Z(n) = \begin{cases} 1 & n = 0 \\ 7000 & n = 1 \\ T_Z(n/2) + 1000 & 1 < n \end{cases}$$

3. Repeat the exercise for

$$T_X(n) = \begin{cases} 1 & n = 0 \\ 5000 & n = 1 \\ 2\,T_X(n/2) + 10\,n^2 & 1 < n \end{cases} \qquad T_Y(n) = \begin{cases} 1 & n = 0 \\ 6000 & n = 1 \\ 2\,T_Y(n/2) + 100\,n & 1 < n \end{cases} \qquad T_Z(n) = \begin{cases} 1 & n = 0 \\ 7000 & n = 1 \\ 2\,T_Z(n/2) + 1000 & 1 < n \end{cases}$$

4. Repeat the exercise for

$$T_X(n) = \begin{cases} 1 & n=0 \\ 300 & n=1 \\ T_X(n/2) + 9\,n^2 & 1<n \end{cases} \quad T_Y(n) = \begin{cases} 1 & n=0 \\ 600 & n=1 \\ T_Y(n/2) + 6\,n^2 & 1<n \end{cases} \quad T_Z(n) = \begin{cases} 1 & n=0 \\ 900 & n=1 \\ T_Z(n/2) + 3\,n^2 & 1<n \end{cases}$$

---

5. Repeat the exercise for

$$T_X(n) = \begin{cases} 1 & n=0 \\ 3 & n=1 \\ 4\,T_X(n/2) + 7\,n & 1<n \end{cases} \quad T_Y(n) = \begin{cases} 1 & n=0 \\ 60 & n=1 \\ 3\,T_Y(n/2) + 10\,n^2 & 1<n \end{cases} \quad T_Z(n) = \begin{cases} 1 & n=0 \\ 9 & n=1 \\ 2\,T_Z(n/2) + 5\,n^2 & 1<n \end{cases}$$

---

6. Repeat the exercise for

$$T_X(n) = \begin{cases} 0 & n=0 \\ 1 & n=1 \\ 4T_X(n/2) + 2\,n & 2\le n \end{cases} \quad T_Y(n) = \begin{cases} 0 & n=0 \\ 2 & n=1 \\ 3T_Y(n/2) + 10\,n & 2\le n \end{cases} \quad T_Z(n) = \begin{cases} 1 & n=0 \\ 3 & n=1 \\ 2\,T_Z(n/2) + 5\,n^2 & 1<n \end{cases}$$

---

7. Repeat the exercise for

$$T_X(n) = \begin{cases} 0 & n=0 \\ 9 & n=1 \\ 4T_X(n/2) + 3\,n & 2\le n \end{cases} \quad T_Y(n) = \begin{cases} 0 & n=0 \\ 9 & n=1 \\ 2T_Y(n/2) + 6\,n^2 + n & 2\le n \end{cases}$$

---

8. Repeat the exercise for (watch out, different starting conditions)

$$T_X(n) = \begin{cases} 1 & n<4 \\ 50 & n=4 \\ T_X(n/2) + 10\,n^2 & 4<n \end{cases} \quad T_Y(n) = \begin{cases} 1 & n<8 \\ 60 & n=8 \\ T_Y(n/2) + 100\,n & 8<n \end{cases} \quad T_Z(n) = \begin{cases} 1 & n<16 \\ 70 & n=16 \\ T_Z(n/2) + 1000 & 16<n \end{cases}$$

---

9. Repeat the exercise for

$$T_X(n) = \begin{cases} 1 & n=0 \\ 50 & n=1 \\ 2\,T_X(n/2) + 10\,n^2 & 1<n \end{cases} \quad T_Y(n) = \begin{cases} 1 & n=0 \\ 200 & n=1 \\ 2\,T_Y(n/2) + 100\,n & 1<n \end{cases} \quad T_Z(n) = \begin{cases} 1 & n=0 \\ 800 & n=1 \\ 2\,T_Z(n/2) + 1000 & 1<n \end{cases}$$

---

10. Repeat the exercise for (watch out, different starting conditions)

$$T_X(n) = \begin{cases} 1 & n<4 \\ 50 & n=4 \\ 2\,T_X(n/2) + 10\,n^2 & 4<n \end{cases} \quad T_Y(n) = \begin{cases} 1 & n<8 \\ 200 & n=8 \\ 2\,T_Y(n/2) + 100\,n & 8<n \end{cases} \quad T_Z(n) = \begin{cases} 1 & n<16 \\ 800 & n=16 \\ 2\,T_Z(n/2) + 1000 & 16<n \end{cases}$$

---

11. (grad students) Is $f(n) = \log n$ polynomially smaller than $g(n) = n$? Prove your answer.

---

12. (grad students) Is $f(n) = \log n$ polynomially smaller than $g(n) = (\log n)^2$? Prove your answer.

---

13. (grad students) Is $f(n) = \log n$ polynomially smaller than $g(n) = n - \log n$? Prove your answer.

6.5 Strong Master Theorem: Asymptotic Equivalence of the Solution

Having the asymptotic $\Theta(\ )$- classification is usually sufficient, but sometimes it is simply not enough and we need the asymptotic equivalence class. In this case, we need to make an additional assumption on $f(n)$: we assume that there is one dominating term that comes form a certain collection of basis functions $\mathcal{F}$, defined as

$$\boxed{\mathcal{F} = \left\{c, \quad c\,\log_b n, \quad c\,(\log_b n)^\ell, \quad c\,\lg n, \quad c\,n, \quad c\,n^k, \quad c\,n^k\,\log_b n, \quad c\,n^k\,(\log_b n)^\ell\right\}} \tag{29}$$

where $c > 0$, $k \in \mathbb{R}^+$ and $\ell \in \mathbb{N}$. Note, that these are all equal to the last element for different values of k and ℓ. Thus let $f(n)$ have a dominant term $f_d(n)$ that is a member of this basis set and define $f_r(n) = f(n) - f_d(n)$ as the remainder; the remainder could be zero. If $f_r(n) = o(f_d(n)$ then perhaps $f_r(n)$ plays no role in the asymptotic equivalence of $T(n)$ and $f_r(n)$ can perhaps be ignored. This is the case when $f_d(n)$ and $f_r(n)$ are *polynomially separated* from each other, which will be defined below only for the most basis function:

$$f_r(n) = \mathcal{O}(n^{k-\epsilon}(\log_b n)^\ell) \quad \text{for} \quad f_d(n) = c\,n^k\,(\log_b n)^\ell \quad \text{and} \quad k > 0 \tag{30}$$

for a positive non-zero value of ϵ, no matter how small. We now present the *Strong Master Theorem*:

STRONG MASTER THEOREM

(Please note that earlier versions of this hand out referred to this as the *mini-Master Theorem*).

The functions $T(n)$ and $T_d(n)$, defined by the two recurrence relations below, are asymptotically equal, i.e. $T(n) \sim T_d(n)$.

$$T(n) = \begin{cases} t & n = 1 \\ aT\left(\frac{n}{b}\right) + f(n) & n > 1 \end{cases} \quad \text{and} \quad T_d(n) = \begin{cases} t & n = 1 \\ aT_d\left(\frac{n}{b}\right) + f_d(n) & n > 1 \end{cases}, \tag{31}$$

and where $f_d(n) \in \mathcal{F}$ is the dominant term of $f(n)$ such that equation (30) is satisfied.

The proof follows from the knowing the fully explicit solution that will be presented in the next subsection. We will present here the resulting asymptotic equivalences for these in a table form i table 7. Notice that knowledge of $f_d(n)$ implies knowing the values of c, k and ℓ, and also that the initial boundary equation might become important, so we also have a value of t that might appear in the solution.

6.6 Examples and Practice Problems – SMT

The following problems will give you practice in applying this technique, but also gives you practice in adjusting the technique to accommodate slightly different conditions. That is, in addition to identifying a, b, and α, also identify t, c, k and ℓ and decide on the applicable formulas in a column of the tables above. Draw conclusions as appropriate.

1.

$$T(n) = \begin{cases} 3 & n \le 1 \\ T(n/2) + 4 & 1 < n \end{cases} \qquad \text{Note} \begin{cases} a = 1 \\ b = 2 \\ \alpha = 0 \end{cases} \text{and} \begin{cases} t = 3 \\ c = 4 \\ k = 0 \\ \ell = 0 \end{cases} \text{so that } a = b^k.$$

Therefore, the SMT-b case applies and $\boxed{T(n) \sim \lg n}$

2.

$$T(n) = \begin{cases} 3 & n \le 1 \\ T(n/2) + n & 1 < n \end{cases} \qquad \text{Note} \begin{cases} a = 1 \\ b = 2 \\ \alpha = 0 \end{cases} \text{and} \begin{cases} t = 3 \\ c = 1 \\ k = 1 \\ \ell = 0 \end{cases} \text{so that } a < b^k.$$

The dominant term of $T_d(n)$, $T_d(n) = \begin{cases} t & n = 1 \\ aT_d\left(\frac{n}{b}\right) + f_d(n) & n > 1 \end{cases}$ when $f_d(n)$ a basis function in $\mathcal{F}$ as indicated.

$f_d(n) =$	$a > b^k, \alpha = \log_b a$ "a^L" dominates "$f(n)$" **SMT-a**	$a = b^k$ "a^L" is about equal to "$f(n)$" **SMT-b**	$1 \le a < b^k$ "a^L" is dominated by "$f(n)$" **SMT-c**
c	$\sim \left(t + \frac{1}{a-1}\, c\right) n^\alpha$	$\sim c\, \log_b n$	na
$c\, \log_b n$	$\sim \left(t + \frac{a}{(a-1)^2} c\right) n^\alpha$	$\sim \frac{1}{2} c\, (\log_b n)^2$	na
$c\, (\log_b n)^\ell$	$\sim \left(t + \frac{a A_\ell(a)}{(a-1)^{\ell+1}} c\right)\, n^\alpha$	$\sim c\, \frac{1}{(\ell+1)}\, (\log_b n)^{\ell+1}$	na
$c\, n$	$\sim \left(t + \frac{b}{a-b}\, c\right) n^\alpha$	$\sim (c\, n\,) \log_b n$	$\sim \frac{b}{b-a}\, (c\, n)$
$c\, n^k$	$\sim \left(t + \frac{b^k}{a-b^k} c\right) n^\alpha$	$\sim \left(c\, n^k\right) \log_b n$	$\sim \frac{b^k}{b^k-a} \left(c\, n^k\right)$
$c\, n^k \log_b n$	$\sim \left(t + \frac{a\, b^k}{(a-b^k)^2}\, c\right) n^\alpha$	$\sim \frac{1}{2} \left(c\, n^k \log_b n\right) \log_b n$	$\sim \frac{b^k}{b^k-a} \left(c\, n^k \log_b n\right)$
$c\, n^k (\log_b n)^\ell$	$\sim \left(t + \frac{y}{(1-y)^{\ell+1}}\, A_\ell(y)\, c\right)\, n^\alpha,$	$\sim \frac{1}{(\ell+1)}\, \left(cn^k (\log_b n)^\ell\right) \log_b n$	$\sim \frac{b^k}{b^k-a}\, \left(c\, n^k (\log_b(n))^\ell\right)$

Table 7: The dominant terms for selected Divide & Conquer type recurrence relations. The $A_\ell(a)$ refers to Eulerian numbers, see text, and $y = b^k/a$.

Therefore, the SMT-a case applies and $\boxed{T(n) \sim 2n}$

3.

$$T(n) = \begin{cases} 3 & n \le 1 \\ T(n/2) + n^2 & 1 < n \end{cases} \quad \text{Note} \begin{cases} a = 1 \\ b = 2 \\ \alpha = 0 \end{cases} \text{and} \begin{cases} t = 3 \\ c = 1 \\ k = 2 \\ \ell = 0 \end{cases} \text{so that } a < b^k.$$

Therefore, the SMT-a case applies and $\boxed{T(n) \sim \frac{4}{3}\, n^2}$

4.

$$T(n) = \begin{cases} 3 & n \le 1 \\ 2\, T(n/2) + 4 & 1 < n \end{cases} \quad \text{Note} \begin{cases} a = 2 \\ b = 2 \\ \alpha = 1 \end{cases} \text{and} \begin{cases} t = 3 \\ c = 4 \\ k = 0 \\ \ell = 0 \end{cases} \text{so that } a > b^k$$

Therefore, $\boxed{T(n) \sim 7\, n}$

5.

$$T(n) = \begin{cases} 3 & n \le 1 \\ 2\, T(n/2) + n & 1 < n \end{cases} \quad \text{Note} \begin{cases} a = 2 \\ b = 2 \\ \alpha = 1 \end{cases} \text{and} \begin{cases} t = 3 \\ c = 1 \\ k = 1 \\ \ell = 0 \end{cases} \text{so that } a = b^k.$$

Therefore, the SMT-b case applies $\boxed{T(n) \sim n \lg n}$

6.

$$T(n) = \begin{cases} 3 & n \le 1 \\ 2\, T(n/2) + n^2 & 1 < n \end{cases} \quad \text{Note} \begin{cases} a = 2 \\ b = 2 \\ \alpha = 1 \end{cases} \text{and} \begin{cases} t = 3 \\ c = 1 \\ k = 2 \\ \ell = 0 \end{cases} \text{so that } a < b^k.$$

Therefore, $\boxed{T(n) \sim 2\, n^2}$

7. (grad students only)

$$T(n) = \begin{cases} 3 & n \le 1 \\ T(n/2) + \lg n & 1 < n \end{cases} \qquad \text{Note } \begin{cases} a = 1 \\ b = 2 \\ \alpha = 0 \end{cases} \text{ and } \begin{cases} t = 3 \\ c = 1 \\ k = 0 \\ \ell = 1 \end{cases} \text{ so that } a = b^k.$$

8. (grad students only)

$$T(n) = \begin{cases} 3 & n \le 1 \\ 2\,T(n/2) + \lg n & 1 < n \end{cases}$$

9.

$$T(n) = \begin{cases} 3 & n \le 1 \\ 2\,T(n/2) + n/2 & 1 < n \end{cases}$$

10.

$$T(n) = \begin{cases} 3 & n \le 1 \\ 2\,T(n/3) + 2 & 1 < n \end{cases}$$

11.

$$T(n) = \begin{cases} 3 & n \le 1 \\ 2\,T(n/3) + 2n & 1 < n \end{cases}$$

12.

$$T(n) = \begin{cases} 3 & n \le 1 \\ 2\,T(n/3) + n^2 & 1 < n \end{cases}$$

13.

$$T(n) = \begin{cases} 3 & n \le 1 \\ 3\,T(n/2) + 2 & 1 < n \end{cases}$$

14.

$$T(n) = \begin{cases} 3 & n \le 1 \\ 3\,T(n/2) + 2n & 1 < n \end{cases}$$

6.7 Case Study

Suppose an important algorithm X is the industry standard. It has a performance of $T(n) = \Theta(n^2 \log n)$ with recurrence relation

$$T(n) = \begin{cases} 10 & n < 2 \\ 128 & n = 2 \\ 4\,T(n/2) + 8\,n^2 & 2 < n \end{cases}$$

You desperately want to get an edge on the competition, and you wonder where you should concentrate your efforts. What is the performance impact (asymptotic region) of the following possibilities? Do they improve the performance, and by how much? Which one has the biggest "bang for the buck"? And finally, which part of the algorithm is represented by the improvement?
It should be noted that the full expression for $T(n)$ is $T(n) = 8n^2 \lg n + 24n^2$

1. Find a way to reduce $T(2)$ from 128 down to 64.
   Answer: the solution would be $T(n) = 8n^2 \lg n + 8n^2$, so effects only the second (non-dominant) term.
2. Find a way to reduce $T(4)$ from 640 down to 320.
   Answer: the solution would be $T(n) = 8n^2 \lg n + 4n^2$, so effects only the second (non-dominant) term.
3. Find a way to reduce $f(n)$ from $8n^2$ down to $6n^2$.
   Answer: the solution would be $T(n) = 6n^2 \lg n + 26n^2$, so effects both terms, most importantly the coefficient for the dominant term goes from 8 to 6, representing a 25% savings.
4. Find a way to reduce $f(n)$ from $8n^2$ down to $100n\,(\log_2 n)^2$.
   Answer: the solution would be $T(n) = 582n^2$, which represents a reduction in magnitude for the dominant term. It is with a much higher coefficient, so it is practically important only for n such that $8 \lg n > 582$, or $n \approx 2^{72} \approx 8\;10^{21}$. This is derived by using case SMT-a, and thus $\left(t + \frac{y}{(1-y)^{\ell+1}}\,A_\ell(y)\;c\right)\;n^\alpha$, where $\alpha = 2$, $\ell = 2$, $y = 1/2$, $A_\ell(y) = 1 + y$, and $t = -18$ as it is the solution of $T(2) = 4\,t + 100 \cdot 2\;(\log_2 2)^2$.
5. Explore a change in $T(n)$ from $4\,T(n/2) + 10\,n^2$ to $3\,T(n/2) + \;n^3$.
   Answer: the solution would be $T(n) = 8/5n^3 + 192/5n^{1.58}$, which is more expensive in the asymptotic region, and which has a cross-over at $n = 35.22$. So this is only beneficial for $n \leq 35$.
6. Explore a change in $T(n)$ from $4\,T(n/2) + 10\,n^2$ to $3\,T(n/2) + 16\,n^2$.
   Answer: the solution would be $T(n) = 64n^2 - 128/3n^{1.58}$, which represents a reduction in magnitude for the dominant term. It is with a much higher coefficient, so it is practically important for n such that $8 \lg n > 64$, or $n > 256$.
7. Explore a change from $T(n) = 4\,T(n/2) + 10\,n^2$ to $T(n) = 16\,T(n/4) + 10\,n^2$.
   Answer: the solution would be $T(n) = 5n^2 \lg n + 22n^2$, so effects both terms, most importantly the coefficient for the dominant term goes from 8 to 5, representing a 37.5% savings.
8. Explore a change from $T(n) = 4\,T(n/2) + 10\,n^2$ to $T(n) = 15\,T(n/4) + n^3$.
   Answer: the solution would be $T(n) \sim 64/49n^3$, which is more expensive in the asymptotic region, and has a cross-over at $n = 30.695$. So this is only beneficial for $n \leq 30$.

6.8 Exercises

The CDE (Chief Development Engineer) asked several different teams to design an algorithmic solution for a problem. The algorithmic performance for these solutions are given by recurrence relations. Rank the algorithms from fastest to slowest when the problem size is in the asymptotic region. Use either the *Weak Master Theorem*, or the *Strong Master Theorem*, whichever one allows you to draw your conclusion. If the SMT still does not provide enough information, then state so.

1. $T_X(n) = \begin{cases} 1 & n=0 \\ 50 & n=1 \\ T_X(n/2) + 10\,n^2 & 1<n \end{cases}$ $T_Y(n) = \begin{cases} 1 & n=0 \\ 60 & n=1 \\ T_Y(n/2) + 100\,n & 1<n \end{cases}$ $T_Z(n) = \begin{cases} 1 & n=0 \\ 70 & n=1 \\ T_Z(n/2) + 1000 & 1<n \end{cases}$

2. $T_X(n) = \begin{cases} 1 & n=0 \\ 5000 & n=1 \\ T_X(n/2) + 10\,n^2 & 1<n \end{cases}$ $T_Y(n) = \begin{cases} 1 & n=0 \\ 6000 & n=1 \\ T_Y(n/2) + 100\,n & 1<n \end{cases}$ $T_Z(n) = \begin{cases} 1 & n=0 \\ 7000 & n=1 \\ T_Z(n/2) + 1000 & 1<n \end{cases}$

3. $T_X(n) = \begin{cases} 1 & n=0 \\ 300 & n=1 \\ T_X(n/2) + 9\,n^2 & 1<n \end{cases}$ $T_Y(n) = \begin{cases} 1 & n=0 \\ 600 & n=1 \\ T_Y(n/2) + 6\,n^2 & 1<n \end{cases}$ $T_Z(n) = \begin{cases} 1 & n=0 \\ 900 & n=1 \\ T_Z(n/2) + 3\,n^2 & 1<n \end{cases}$

4. $T_X(n) = \begin{cases} 1 & n=0 \\ 3 & n=1 \\ 4\,T_X(n/2) + 7\,n & 1<n \end{cases}$ $T_Y(n) = \begin{cases} 1 & n=0 \\ 60 & n=1 \\ 3\,T_Y(n/2) + 10\,n^2 & 1<n \end{cases}$ $T_Z(n) = \begin{cases} 1 & n=0 \\ 9 & n=1 \\ 2\,T_Z(n/2) + 5\,n^2 & 1<n \end{cases}$

5. $T_X(n) = \begin{cases} 0 & n=0 \\ 1 & n=1 \\ 4T_X(n/2) + 2\,n & 2\le n \end{cases}$ $T_Y(n) = \begin{cases} 0 & n=0 \\ 2 & n=1 \\ 3T_Y(n/2) + 10\,n & 2\le n \end{cases}$ $T_Z(n) = \begin{cases} 1 & n=0 \\ 3 & n=1 \\ 2\,T_Z(n/2) + 5\,n^2 & 1<n \end{cases}$

6. $T_X(n) = \begin{cases} 0 & n=0 \\ 9 & n=1 \\ 4T_X(n/2) + 3\,n & 2\le n \end{cases}$ $T_Y(n) = \begin{cases} 0 & n=0 \\ 9 & n=1 \\ 2T_Y(n/2) + 6\,n^2 + n & 2\le n \end{cases}$

7. $T_X(n) = \begin{cases} 1 & n<4 \\ 50 & n=4 \\ T_X(n/2) + 10\,n^2 & 4<n \end{cases}$ $T_Y(n) = \begin{cases} 1 & n<8 \\ 60 & n=8 \\ T_Y(n/2) + 100\,n & 8<n \end{cases}$ $T_Z(n) = \begin{cases} 1 & n<16 \\ 70 & n=16 \\ T_Z(n/2) + 1000 & 16<n \end{cases}$

8. $T_X(n) = \begin{cases} 1 & n=0 \\ 50 & n=1 \\ 2\,T_X(n/2) + 10\,n^2 & 1<n \end{cases}$ $T_Y(n) = \begin{cases} 1 & n=0 \\ 200 & n=1 \\ 2\,T_Y(n/2) + 100\,n & 1<n \end{cases}$ $T_Z(n) = \begin{cases} 1 & n=0 \\ 800 & n=1 \\ 2\,T_Z(n/2) + 1000 & 1<n \end{cases}$

9. $T_X(n) = \begin{cases} 1 & n<4 \\ 50 & n=4 \\ 2\,T_X(n/2) + 10\,n^2 & 4<n \end{cases}$ $T_Y(n) = \begin{cases} 1 & n<8 \\ 200 & n=8 \\ 2\,T_Y(n/2) + 100\,n & 8<n \end{cases}$ $T_Z(n) = \begin{cases} 1 & n<16 \\ 800 & n=16 \\ 2\,T_Z(n/2) + 1000 & 16<n \end{cases}$

6.9 Explicit Solution, $f(n)$ is a single term

In this section, we assume that $f(n)$ is a single term coming from the set

$$\boxed{\mathcal{F} = \Big\{c, \quad c\,\log_b n, \quad c\,(\log_b n)^\ell, \quad c\,\lg n, \quad c\,n, \quad c\,n^k, \quad c\,n^k\,\log_b n, \quad c\,n^k\,(\log_b n)^\ell\Big\}} \tag{32}$$

where $c > 0$, $\ell \in \mathbb{N}$ and $k \in \mathbb{R}^+$. We will refer to these single term functions as the *basis functions*. If the function $f(n)$ is in fact a summation of these terms (with possibly different values for k), then the theorem of the next section can be used to compose full and explicit solutions. Included in the set of functions that can be constructed from these basis functions and for which explicit solutions can be derived, are all (finite) polynomials, a (finite) polynomial multiplied by a logarithmic term, and non-integer powers of n (like square root). These are the most common forms for an inhomogeneous term in the analysis of some of the easy and some mid-level advanced algorithms. The results are summarized below in the table, before showing the individual proofs; these proofs are all based on taking (finite) summations.

The explicit form of $T(n)$, $T(n) = \begin{cases} 0 & n = 1 \\ aT\left(\frac{n}{b}\right) + f(n) & n > 1 \end{cases}$ when $f(n)$ a basis function in $\mathcal{F}$ as indicated.

$f(n) =$	$a = b^k$	$a \neq b^k, \qquad \alpha = \log_b\, a$
c	$c\,\log_b n$	$\frac{1}{a-1}\,c\,(n^\alpha - 1)$
$c\,\log_b n$	$\frac{1}{2}c\,\log_b n + \frac{1}{2}c\,(\log_b n)^2$	$\frac{a}{(a-1)^2}\,c\,(n^\alpha - 1) - \frac{a+1}{(a-1)^2}\,c\,\log_b n$
$c\,n$	$c\,n\,\log_b n$	$\frac{b}{b-a}\,c\,(n - n^\alpha)$
$c\,n^k$	$c\,n^k\log_b n$	$\frac{b^k}{b^k-a}\,c\,(n - n^\alpha)$
$c\,n^k\log_b n$	$\frac{1}{2}c\,n^k\log_b n + \frac{1}{2}c\,n^k\,(\log_b n)^2$	$\frac{a\,b^k}{(b^k-a)^2}\,c\,\left(n^\alpha - n^k\right) + \frac{b^k}{b^k-a}\,c\,n^k\,\log_b n$

Table 8: The explicit solution for selected Divide & Conquer type of recurrence relations

6.9.1 A constant function

The recursive definition is $T(n) = aT\left(\frac{n}{b}\right) + c$ and the expression for $T(n)$ is:

$$\begin{aligned} T(n) &= a^L t + a^{L-1}c + a^{L-2}c + a^{L-3}c \cdots + a^0 c \\ &= \begin{cases} t + c\,\log_b n & a = 1 \\ \left(t + \frac{1}{a-1}\cdot c\right) n^\alpha - \frac{1}{a-1}\,c & a \geq 1 \end{cases} \end{aligned}$$

6.9.2 A logarithmic function

The recursive definition is $T(n) = aT\left(\frac{n}{b}\right) + c\log_b n$ so that $f(n) = f(b^i) = c\,i$. The expression for $T(n)$ is:

$$\begin{aligned} T(n) &= a^L t + a^{L-1}c\left(1 + 2\,a^{-1} + 3\,a^{-2}\,\cdots + L\,a^{-(L-1)}\right) \\ &= \begin{cases} \frac{1}{2}c(\log_b\,n)^2 + \frac{1}{2}c\log_b\,n + t & \text{for } a = 1 \\ \left(t + \dfrac{a\,c}{(a-1)^2}\right) n^\alpha - c\dfrac{a+1}{(a-1)^2}\,\log_b n - \dfrac{ac}{(a-1)^2} & \text{for } a \neq 1 \end{cases} \end{aligned}$$

6.9.3 Linear function.

The recursive definition is $T(n) = aT\left(\frac{n}{b}\right) + c\,n$ so that $f(n) =$ ¸n. The expression for $T(n)$ is:

$$\begin{aligned}
T(n) &= a^L t + a^{L-1} c\, b^1 + a^{L-2}\, c\, b^2 + a^{L-3}\, c\, b^3 \cdots + a^0\, c\, b^L \\
&= a^L t + a^{L-1}\, c\, b^1 \left(1 + (b/a)^1 + (b/a)^2\, \cdots + (b/a)^{L-1}\right) \\
&= \begin{cases} t\,n + c\,n\log_b n & \text{for } a = b \\ \left(t + \dfrac{b\,c}{a-b}\right) n^\alpha + \dfrac{b\,c}{b-a}\, n & \text{for } a \neq b \end{cases}
\end{aligned}$$

6.9.4 Special case: A monomial function.

The recursive definition is $T(n) = aT\left(\frac{n}{b}\right) + c\,n^k$ so that $f(n) = c\,n^k$. The expression for $T(n)$ is:

$$\begin{aligned}
T(n) &= a^L t + a^{L-1} c\,(b^k)^1 + a^{L-2}\, c\,(b^k)^2 + a^{L-3}\, c\,(b^k)^3 \cdots + a^0\, c\,(b^k)^L \\
&= a^L t + a^{L-1}\, c\,(b^k) \left(1 + ((b^k)/a)^1 + ((b^k)/a)^2\, \cdots + ((b^k)/a)^{L-1}\right) \\
&= \begin{cases} t\,n^k + c\,n^k\, \log_b n & \text{for } a = b^k \\ \left(t + \dfrac{b^k\,c}{a-b^k}\right) n^\alpha + \dfrac{b^k\,c}{b^k-a}\, n^k & \text{for } a \neq b^k \end{cases}
\end{aligned}$$

6.9.5 Special case: Another monomial function.

Note Although we used k as the exponent of n in the function, which made us think that k should be a natural number, we did not make use of that fact, and the conclusion remains valid even if k is any fixed real number. A recurrence relation with $f(n) = \sqrt{n}$ is thus included. Technically, the number (k) could be negative, but then the function $f(n)$ would be decreasing which is not happening in algorithmic algorithms. However, we show the result by an example:

$$T(n) = \begin{cases} 3 & n \le 1 \\ T(n/2) + \sqrt{n} & 1 < n \end{cases} \qquad \text{Note } a = 1, b = 2, \text{ and } k = 1/2$$

Therefore, $T(n) = \left(2 + \sqrt{2}\,\right)\, \sqrt{n} + 1 - \sqrt{2}$

6.9.6 Product of a monomial and a logarithm.

The recursive definition is $T(n) = aT\left(\frac{n}{b}\right) + c\,n^k \log_b n$ so that $f(n) = c\,n^k \log_b n$. The expression for $T(n)$ is:

$$\begin{aligned}
T(n) &= a^L t + a^{L-1} c\,(b^k)^1 + 2\,a^{L-2}\, c\,(b^k)^2 + 3\,a^{L-3}\, c\,(b^k)^3 \cdots + L\,a^0\, c\,(b^k)^L \\
&= a^L t + a^{L-1}\, c\,b^k \left\{1 + 2(b^k/a)^1 + 3(b^k/a)^2\, \cdots + L(b^k/a)^{L-1}\right\}
\end{aligned}$$

which when $a = b^k$ evaluates to

$$T(n) = \left(t + c\,\tfrac{1}{2}\left(L + L^2\right)\right)(b^k)^L = t\,n^k + \tfrac{1}{2}\,c\,n^k \log_b n + \tfrac{1}{2}\,c\,n^k (\log_b n)^2$$

When $a \neq b^k$, this becomes

$$\begin{aligned}
T(n) &= a^L t + a^{L-1}\, c\,b^k \left\{\frac{L\,(b^k/a)^L}{(b^k/a)-1} - \frac{(b^k/a)^L - 1}{((b^k/a)-1)^2}\right\} \\
&= n^\alpha\, t + \qquad c\,b^k \left\{\frac{\log_b n\, n^k}{b^k - a} - a\,\frac{n^k - n^\alpha}{(b^k-a)^2}\right\} \\
&= \left(t + \frac{a\,b^k}{(b^k-a)^2}\,c\right) n^\alpha + \frac{b^k\,c}{b^k-a}\, n^k\, \log_b n - \frac{a\,b^k\,c}{(b^k-a)^2}\, n^k
\end{aligned}$$

6.9.7 Power of logarithmic function

The recursive definition is $T(n) = aT\left(\frac{n}{b}\right) + c(\log_b n)^\ell$ so that $f(n) = f(b^i) = c\, i^\ell$. The expression for $T(n)$ is:

$$T(n) = a^L t + \left(a^{L-1}\ + 2^\ell\, a^{L-2}\ + 3^\ell\, a^{L-3}\ \cdots + L^\ell\, a^0\right) c$$

Even when $a = 1$, this expression does not have an easily expressible form. The solution can be written as

$$\begin{aligned} T(n) &= t + c\sum_{i=1}^{L} i^\ell = t + c\,\left(1^\ell + 2^\ell + 3^\ell \ldots L^\ell\right) \\ &= t + c\left\{A(\ell,0)\binom{L+1}{\ell+1} + A(\ell,1)\binom{L+2}{\ell+1} + A(\ell,2)\binom{L+3}{\ell+1} + \ldots + A(\ell,\ell-1)\binom{L+\ell}{\ell+1}\right\} \\ &= t + c\sum_{j=0}^{\ell-1} A(\ell,j)\binom{L+1+j}{\ell+1}, \end{aligned}$$

where $A(\ell, j)$ are Eulerian numbers. These Eulerian numbers are also denoted as $\left\langle {\ell \atop j} \right\rangle$, and are somewhat similar to binomial numbers. One of the properties needed is that $\sum_{j=0}^{\ell-1} A(\ell,j) = \ell!$. The explicit solution cannot be simplified, unfortunately. Below we are able to derive the asymptotic form, which has a simpler form. The situation that $a > 1$ (since $a < 1$ does not occur in algorithm analysis) is not any better. The explicit form of the solution is even more complicated and involves Eulerian polynomials, but we are able to discover the asymptotic dominant term, which has a simpler form. Consider what happens as $n \to \infty$, and thus also $L \to \infty$. Then, for all finite j, the asymptotic expression is indiependent of j:

$$\binom{L+1+j}{\ell+1} \sim \frac{L^{\ell+1}}{(\ell+1)!}, \quad \text{for all } j$$

and thus

$$T(n) \sim t + c\,\frac{L^{\ell+1}}{(\ell+1)!}\sum_{j=0}^{\ell-1} A(\ell,j) \sim t + c\,\frac{L^{\ell+1}}{(\ell+1)!}\,\ell! \boxed{\sim t + c\,\frac{1}{(\ell+1)}\,(\log_b n)^{\ell+1}}$$

Consider now the situation that $a > 1$ (since $a < 1$ does not occur in algorithm analysis), then the explicit form of the solution is rather complicated, and even the asymptotic form is far from straightforward. Consider however that, whenever $|x| < 1$, and as $L \to \infty$,

$$x^1\ + 2^\ell\, x^2\ + 3^\ell\, x^3\ \cdots + L^\ell\, x^L\ \to \frac{x\, A_\ell(x)}{(1-x)^{\ell+1}}. \tag{33}$$

where $A_\ell(x)$ is the ℓ^{th} Eulerian polynomial. The coefficients are the Eulerian numbers. For notational convenience, define y as $y = 1/a$. Then,

$$\begin{aligned} T(n) &= a^L\, t + a^L\, c\,\left((\tfrac{1}{a})^1\ + 2^\ell\,(\tfrac{1}{a})^2\ + 3^\ell\,(\tfrac{1}{a})^3\ \cdots + L^\ell\,(\tfrac{1}{a})^L\right) \\ &= a^L\, t + a^L\, c\,\left(y^1\ + 2^\ell\, y^2\ + 3^\ell\, y^3\ \cdots + L^\ell\, y^L\right) \\ &\sim a^L\, t + a^L\, c\,\frac{y}{(1-y)^{\ell+1}}\,A_\ell(y) = t\, n^{\log_b a} + \frac{y}{(1-y)^{\ell+1}}\,A_\ell(y)\, c\, n^{\log_b a} \end{aligned}$$

Finally the asymptotic dominant term can be written in two equivalent forms: (using the fact that coefficients of the Eulerian polynomial form a palindrome), so that $A_\ell(x) = x^{\ell-1}A_\ell(1/x)$)

$$\boxed{T(n) \sim t\, n^{\log_b a} + \frac{a}{(a-1)^{\ell+1}}A_\ell(a)\, c\, n^{\log_b a}} \qquad \text{or equivalently,}$$

$$\boxed{T(n) \sim t\, n^{\log_b a} + \frac{a^\ell}{(a-1)^{\ell+1}}\,A_\ell(1/a)\, c\, n^{\log_b a}}$$

These two expressions are easily confused. The first one is more attractive to study analytical properties, the latter one is computationally more attractive. We show some examples in the accompanying table.Note also that $A_\ell(1)$ is the sum of the Eulerian numbers, and thus $A_\ell(1) = \ell!$. The first few polynomials are given in the table below. Others polynomials, and the methods to generate them, can be found on-line.[1]

The dominant term of the solution for $T(n) = \begin{cases} t & n=1 \\ aT\left(\frac{n}{b}\right) + c\,(\log_b n)^\ell & a \geq 1, n > 1 \end{cases}$

ℓ	$A_\ell(a)$	$a=1$	$a=2$	$a=3$	$a=4$
		$\alpha=0$	$\alpha=\log_b 2$	$\alpha=\log_b 3$	$\alpha=\log_b 4$
$\ell=0$	1	$\sim c\log_b n$	$\sim (t+c)n^\alpha$	$\sim (t+\frac{1}{2}c)n^\alpha$	$\sim (t+\frac{1}{3}c)n^\alpha$
$\ell=1$	1	$\sim \frac{1}{2}c(\log_b n)^2$	$\sim (t+2c)n^\alpha$	$\sim (t+\frac{3}{4}c)n^\alpha$	$\sim (t+\frac{4}{9}c)n^\alpha$
$\ell=2$	$1+a$	$\sim \frac{1}{3}c(\log_b n)^3$	$\sim (t+6c)n^\alpha$	$\sim (t+\frac{3}{2}c)n^\alpha$	$\sim (t+\frac{20}{27}c)n^\alpha$
$\ell=3$	$1+4a+a^2$	$\sim \frac{1}{4}(\log_b n)^4$	$\sim (t+26c)n^\alpha$	$\sim (t+\frac{33}{8}c)n^\alpha$	$\sim (t+\frac{44}{27}c)n^\alpha$
$\ell=4$	$1+11a+11a^2+a^3$	$\sim \frac{1}{5}c(\log_b n)^5$	$\sim (t+150c)n^\alpha$	$\sim (t+15c)n^\alpha$	$\sim (t+\frac{380}{81}c)n^\alpha$
$\ell=5$	$1+26a+66a^2+26a^3+a^4$				

Table 9: The dominant term of the solution for various values of ℓ and a.

6.9.8 Product of a monomial and the power of a logarithm.

The recursive definition is $T(n) = aT\left(\frac{n}{b}\right) + c\,n^k(\log_b n)^\ell$ so that $f(n) = c\,n^k(\log_b n)^\ell$ and $f(b^i) = c\,b^{i.k}i^\ell = c\,i^\ell\,(b^k)^i$. The expression for $T(n)$ is:

$$\begin{aligned} T(n) &= a^L\,t + a^{L-1}f(b^1) + a^{L-2}f(b^2) + a^{L-3}\,f(b^3)\cdots + a\,f(b^{L-1}) + a^0 f(b^L) \\ &= a^L\,t + a^{L-1}c\,1^\ell\,(b^k)^1 + a^{L-2}c\,2^\ell\,(b^k)^2 + \cdots + a\,c\,(L-1)^\ell\,(b^k)^{L-1} + a^0\,c\,L^\ell\,(b^k)^L \\ &= a^L\,t + a^L c\,\left\{1^\ell\,(b^k/a)^1 + 2^\ell\,(b^k/a)^2 + 3^\ell\,(b^k/a)^3 \cdots + (L-1)^\ell\,(b^k/a)^{L-1} + L^\ell\,(b^k/a)^L\right\} \end{aligned} \tag{34}$$

The situation is similar to the case we treated above, when $f(n)$ is a power of a logarithm only. The only difference is that the ratio $1/a$ is replaced by b^k/a. We start with $a = b^k$. Remembering that $a^L = b^{kL} = n^k$, we get,

$$\begin{aligned} T(n) &= t\,n^k + c\sum_{i=1}^{L} i^\ell \quad n^k = t\,n^k + \left(1^\ell + 2^\ell + 3^\ell \ldots L^\ell\right)\,c\,n^k \\ &= t\,n^k + \sum_{j=0}^{\ell-1} A(\ell,j)\binom{L+1+j}{\ell+1} \quad c\;n^k\;, \end{aligned}$$

where $A(\ell,j)$ are the Eulerian numbers again. The explicit solution cannot be simplified easily, but the asymptotic dominant term is equal to that for $k=0$, where the asymptote is now multiplied by n^k.

$$T(n) \quad\sim\quad t\,n^k + \frac{1}{(\ell+1)}\,c\,n^k(\log_b n)^{\ell+1}$$

[1] I found the wiki pages helpful, but also the paper by Dominique Foata, the paper by Amrik Singh Nimbran, and the book by Kyle Petersen on Eulerian Numbers

Consider now the situation that $a \neq b^k$. The explicit form of $T(n)$ is

$$T(n) \quad = \quad a^L\, t + a^L c\, \left\{ 1^\ell\, (b^k/a)^1 + 2^\ell\, (b^k/a)^2 + 3^\ell\, (b^k/a)^3 \cdots + (L-1)^\ell\, (b^k/a)^{L-1} + L^\ell\, (b^k/a)^L \right\}$$

First, assume that $a > b^k$ and define y as $y = b^k/a$ for notational convenience. Then $y < 1$ and the derivation is identical,

$$\boxed{T(n) \sim t\, n^{\log_b a} + \frac{y}{(1-y)^{\ell+1}}\, A_\ell(y)\, c\, n^{\log_b a}, \qquad \text{where } y = b^k/a}$$

Second, assume that $a < b^k$ and define z as $z = a/b^k$ for notational convenience. Then $z < 1$ and we write the form of $T(n)$, equation (34) in reverse order and factor the term $c\, L^\ell\, (b^k)^L$:

$$\begin{aligned} T(n) \quad &= \quad a^L\, t + c\, \left(L^\ell\, a^0 (b^k)^L + (L-1)^\ell\, a^1 (b^k)^{L-1} + (L-2)^\ell\, a^2 (b^k)^{L-2} + \cdots + 2^\ell\, a^{L-2} (b^k)^2 + 1^\ell\, a^L (b^k)^1 \right) \\ &= \quad a^L\, t + c\, (b^k)^L\, L^\ell\, \left(z^0 \ + (1 - \tfrac{1}{L})^\ell\, z^1 + (1 - \tfrac{2}{L})^\ell\, z^2 + \cdots + (1 - \tfrac{L-2}{L})^\ell\, z^{L-2} \ + (1 - \tfrac{L-1}{L})^\ell\, z^L \right) \qquad (35) \end{aligned}$$

Consider what happens as $n \to \infty$, and thus also $L \to \infty$. The summation $\left(1 + (1 - \frac{1}{L})^\ell\, z + \ldots + (1 - \frac{L-1}{L})^\ell\, z^L\right)$ approaches $\frac{1}{1-z}$ and thus

$$T(n) \quad \sim \quad a^L\, t + c\, (b^k)^L\, L^\ell\, \frac{1}{1-z} \qquad \boxed{\sim \frac{b^k}{b^k - a}\, c\, n^k (\log_b(n))^\ell} \qquad (36)$$

6.10 Explicit Solution, $f(n)$ is a sum of terms

Next[2], we show that if $f(n)$ is indeed the sum of two different terms, then the solution $T(n)$ is the sum of two different solutions for each term.

THEOREM Let the inhomogeneous part $f(n)$ be the sum of m individual terms, then the solution $T(n)$ of the recurrence relation is composed as the sum of the solutions for each of the $m+1$ terms as below. To be specific, let

$$T(n) = \begin{cases} t & n = 1 \\ aT\left(\frac{n}{b}\right) + f_1(n) + f_2(n) \dots f_m(n) & n > 1 \end{cases} \tag{37}$$

then $T(n) = T_0(n) + T_1(n) + T_2(n) + \dots T_m(n)$, where $T_0(n)$, and $T_1(n)$ through $T_m(n)$ are the solutions to the $m+1$ individual recurrence relations

$$T_0(n) = \begin{cases} t & n = 1 \\ aT_1\left(\frac{n}{b}\right) & n > 1 \end{cases} \quad \textit{and} \quad T_j(n) = \begin{cases} 0 & n = 1 \\ aT_\ell\left(\frac{n}{b}\right) + f_j(n) & n > 1 \end{cases} \quad \text{for } j = 1 \dots m \tag{38}$$

The proof is based on unwinding the recursive definition, which we show for two terms:

$$\begin{aligned} T(n) = T(b^L) &= a^L t \\ &\quad + a^{L-1} f_1(b^1) + a^{L-2} f_1(b^2) + a^{L-3} f_1(b^3) \dots + a^0 f_1(b^L) \\ &\quad + a^{L-1} f_2(b^1) + a^{L-2} f_2(b^2) + a^{L-3} f_2(b^3) \dots + a^0 f_2(b^L) \end{aligned}$$

So the solution of this recurrence relation is composed of summing the three solutions of the three separate recurrence relations when $f_0(n) = 0$, $f_1(n)$ and $f_2(n)$ are taken individually.

It suffices therefore to solve the recurrence relation for each function $f(n)$ out of this set individually, for which the solution is given above. For all these individual cases, we find both the explicit form, as well as asymptotic forms and we are able to carry the leading coefficient as well. The only thing to keep in mind is that we require $f(n)$ to be a *finite* sum of terms, so that we do not need to think about an infinite sum of $f_i(n)$ (and thus we do not need to think about the convergence of an infinite sum of $T_i(n)$).

6.11 Examples and Practice Problems

1. Find both the explicit closed form for $T(n)$, and the asymptotic dominant term of T for values of n that are even powers of b:

$$\begin{cases} T(n) = 2 & n \le 1 \\ T(n) = T(n/2) + 2n + 3 & 2 \le n \end{cases}$$

The solution is the sum of the solution of the three recurrence relations, $T(n) = T_0(n) + T_1(n) + T_2(n)$, where

$$\begin{cases} T_0(n) = 2 & n \le 1 \\ T_0(n) = T_0(n/2) & 2 \le n \end{cases} \qquad \begin{cases} T_1(n) = 0 & n \le 1 \\ T_1(n) = T_1(n/2) + 2n & 2 \le n \end{cases} \qquad \begin{cases} T_2(n) = 0 & n \le 1 \\ T_2(n) = T_2(n/2) + 3 & 2 \le n \end{cases}$$

All three recurrence relations have $a = 1$ and $b = 2$, and $T_0(n) = 2$, $T_1(n) = 2n \lg n$, and $T_2(n) = 3 \lg n$. Thus

$$T(n) = T_0(n) + T_1(n) + T_2(n) = 2n \lg n + 3 \lg n + 2 \sim 2n \lg n$$

2. Find both the explicit closed form for $T(n)$, and the asymptotic dominant term of T for values of n that are even powers of b:

$$\begin{cases} T(n) = 3 & n \le 1 \\ T(n) = 2T(n/2) + 4n + 5 & 2 \le n \end{cases}$$

[2] CS404 students should be able to reproduce these when a and b are constants, (such as 1 or 2) and when $f(n)$ is a single term. You may want to try the examples where the boundary is changed to $n \le 4$.

The solution is the sum of the solution of the three recurrence relations, $T(n) = T_0(n) + T_1(n) + T_2(n)$, where

$$\begin{cases} T_0(n) = 3 & n \le 1 \\ T_0(n) = 2T_0(n/2) & 2 \le n \end{cases} \qquad \begin{cases} T_1(n) = 0 & n \le 1 \\ T_1(n) = 2T_1(n/2) + 4n & 2 \le n \end{cases} \qquad \begin{cases} T_2(n) = 0 & n \le 1 \\ T_2(n) = 2T_2(n/2) + 5 & 2 \le n \end{cases}$$

All three recurrence relations have $a = 2$ and $b = 2$, and $T_0(n) = 3n$, $T_1(n) = 4n \lg n$, and $T_2(n) = 5n - 5$. Thus

$$T(n) = T_0(n) + T_1(n) + T_2(n) = 4n \lg n + 8n - 5 \sim 4n \lg n$$

3. Find both the explicit closed form for $T(n)$, and the asymptotic dominant term of T for values of n that are even powers of b:

$$\begin{cases} T(n) = 5 & n \le 1 \\ T(n) = 4T(n/2) + 3n + 2 & 2 \le n \end{cases}$$

Solution: $T(n) = \frac{26}{3}\, n^2 - 3n - \frac{2}{3} \sim \frac{26}{3}\, n^2$

4. Find both the explicit closed form for $T(n)$, and the asymptotic dominant term of T for values of n that are even powers of b:

$$\begin{cases} T(n) = 1 & n \le 1 \\ T(n) = T(n/2) + n + \lg n & 2 \le n \end{cases}$$

Solution: $T(n) = 2n + \frac{1}{2}\,(1 + \lg n)\, \lg n \sim 2n$

6.11.1 Exercise

Complete the table for $\ell = 5$ and $a = 1, 2, 3$, and 4.

6.12 The Weak, Strong, and other variants

In mathematics, the terms *strong* and *weak* are often associated with theorems, and we have taken the liberty to use them here as well. Generally speaking, the assumptions made with a *weak* version are a strict subset of the assumptions made with a *strong* version. Thus a *strong* version is less widely applicable. Consequently, more detailed conclusions can be drawn with a *strong* version. In terms of the Master Theorem, it is the *weak* version that is well known in the algorithms literature, and several other versions have appeared.

The *strong* version assumes that $f(n)$ has a particular form, that the dominant term $f_d(n)$ has a specific form (i.e. is a member of $\mathcal{F}$) and that remainder $f_r(n)$ is polynomially smaller. It then compares $f_d(n)$ with n^α. The *weak* version does not make any assumptions on $f(n)$ (i.e. need not be member of $\mathcal{F}$), it compares $f(n)$ directly with n^α, and insists that these two are polynomially separated. Thus, if we have a recurrence relation and the conditions for the *strong* version are satisfied, then the conditions for the *weak* version are automatically satisfied, but not vice versa. The big advantage at this point is that the SMT also gives the asymptotic equivalence, i.e. the exact value for the constant coefficient c, whereas the WMT gives an asymptotic tight bound. So when comparing the complexity of two algorithms, they could perhaps not be ranked for the efficiency using the WMT, but could perhaps still be separated with the SMT.
The situation is similar to a class room setting: The highest and lowest grade of a student in this class is determined by the grades of all students. The highest and lowest grade of a student who does the homework is a stronger statement and is usually quite different.

6.12.1 Example

Consider the following example. The CDE (Chief Development Engineer) of company asked several different teams to design an algorithmic solution for a problem. Which algorithm should be adopted and implemented if the asymptotic performance is the only consideration and if their algorithmic performance is given by recurrence relations $T_X(n)$ and $T_Y(n)$.

$$T_X(n) = \begin{cases} 30 & n = 1 \\ 2\,T_X(n/2) + n^2 + 3(\lg n)^2 + 77 & 1 < n \end{cases} \qquad T_Y(n) = \begin{cases} 5 & n = 1 \\ 4T_Y(n/2) + 20 & 1 < n \end{cases}$$

First, let us check that we can use the *Weak Master Theorem* for $T_X(n)$: The boundary is indeed at $n = 1$, and $t = 30$, $a = 2$, and $b = 2$. Furthermore, $f(n) = n^2 + 3(\lg n)^2 + 77$. The relative growth of this function must be compared with the function n^α, where $\alpha = \log_b a = 1$. Thus: $f(n) = \Omega(n^\alpha)$. In fact, we also have that $f(n) = \Omega(n^{\alpha+\epsilon})$ for any ϵ between 0 and 1, so that $f(n)$ and n^α are "polynomially separated". Next, we need to make sure that the regularity condition is satisfied. Note that $a\ f(\frac{n}{b}) = a\ \left(n^2/b^2 + 3(\lg n/b)^2 + 77\right) = a\ \left(1/4\,n^2 + 3\,(\lg n - 1)^2 + 77\right) < a\ \left(n^2 + 3\,(\lg n)^2 + 77\right) = 2\ f(n)$, so that: yes, the regularity condition is satisfied. Finally, the *WMT* can be applied, and the conclusion drawn that $T_X(n) = \Theta(n^2)$, so that the asymptotic growth of $T_X(n)$ is determined as $\Theta(n^2)$.

The *Weak Master Theorem* can also be applied for $T_Y(n)$ (check for yourself) which also results in: $T_Y(n) = \Theta(n^2)$, so that the asymptotic growth of $T_Y(n)$ is also determined as $\Theta(n^2)$. In particular, the *Weak Master Theorem* is an excellent tool to characterize the growth rate of a single function, but can not always be used to decide between two different algorithms.

We will now use the *Strong Master Theorem* on $T_X(n)$ and split the function $f(n)$ into a dominating part and a remainder part: $f(n) = f_d(n) + f_r(n)$, where $f_d(n) = n^2$ and $f_r(n) = 3(\lg n)^2 + 77$. Note that $f_d(n)$ and $f_r(n)$ are "polynomially separated", since $f_r(n) = 3(\lg n)^2 + 77 = o(n)$. The dominating term gives both $c = 1$ and $k = 2$, so that *will be completed next semester*

6.13 To Balance or not, that is the first question

Upon reflection, we have a situation where either the left-side polynomially dominates the summation (resulting in $T(n) = \Theta(n^\alpha)$), or the right-side (resulting in $T(n) = \Theta(f(n))$). When all elements are in balance, however, the complexity is $T(n) = \Theta(f(n)\log n)$, *and is not* $\Theta(\ \max\{n^\alpha,\ f(n)\ \}\ ) = \Theta(\ f(n)\ )$, which we had expected perhaps. So in a way, the master theorem is somewhat counter intuitive and it appears that we pay some kind of penalty of a multiplicative $\log n$. Note that a "log" penalty is not a polynomial penalty, so it is not all that bad, but still. This situation occurs more often in analysis of functions, when two asymptotes align[3], and the underlying message for designers might very well be: Keep looking for an additional improvement, no matter how small, which is feasible in a continuous setting (as in the paper cited), and may not be feasible in discrete algorithms.

6.14 To Balance or not, that is the second question as well

6.15 Validating asymptotic class

Although we mentioned this in the introduction, it is worth to re-iterate that we have proven it only for values of n, where n *is* a power of b. If n is *not* a power of b, then n is between two consecutive powers of b. The only functions $f(n)$ we consider are boxed: if $b^i < n < b^{I+1}$ then $f(b^i) < f(n) < f(b^{I+1})$. This means that the function $f(n)$ is bound below by $f(b^i)$ and bound above by $f(b^{i+1})$, and thus the Θ– classification of $T(n)$ is equal to the Θ– classification of $T(n^L)$. A more formal way is by considering a class of slowly varying functions, which accomplishes about the same.

6.16 Lessons for practitioners

1. If if $f(n) = \mathcal{O}(n^{\alpha-\epsilon})$ for any positive ϵ, and thus $T(n) = \Theta(n^{\alpha})$, then the complexity is still improved by a certain percentage if an algorithm can be found that is cheaper in solving (or implementing) small problem instances. That is, the initial value is made smaller. It also means that highly efficient programs should be written for smaller sized problems.

6.17 Finding the best lower threshold

Consider for example the recurrence relation below.

$$\begin{cases} T(n) = h(n) & n \le N^* \\ T(n) = T(n-1) + n^2 & N^* \le n \end{cases}$$

Then the solution depends on the cut-off value for N^*. How can we find the best value for N^*? In theory, and how do we find it in practice? Generally speaking, and in theory, the value for N^* should be such that $h(N^*) = T(N^*)$, where the defining recurrence is substituted for $T(N^*)$. Thus, in the above example, find N^* such that $h(N^*) = T(N^*-1) + (N^*)^2 = h(N^*-1) + (N^*)^2$. In practice, (where the algorithm is actually implemented with an actual language on an actual platform and with actual memory retrieval delays, and so on), this best value for N^* must be found empirically: run many samples.

6.18 Finding upper bounds

THEOREM Let the inhomogeneous part $f(n)$ be upper bounded by $g(n)$, $f(n) = \mathcal{O}(g(n))$, and let the $T_g(n)$ be the solution of a recurrence relation where $f(n)$ is replaced by $g(n)$, then $T(n) = \mathcal{O}(T_g(n))$.

[3]see e.g. the paper by Lester Lipsky, Chee-Min Henry Lieu, Abolfazl Tehranipour, and Appie van de Liefvoort. 1982. *On the asymptotic behavior of time-sharing systems*. **Commun. ACM 25**, 10 (October 1982), 707-714. DOI=http://dx.doi.org/10.1145/358656.358666

7 Secondary Recurrence Relations

We have studied recurrence relations like

$$T(n) = \begin{cases} t_1 & n = 1 \\ aT\left(\frac{n}{b}\right) + f(n) & n > 1 \end{cases}$$

and it sufficed to consider the solution only for the subsequence $n \in \{b^0, b^1, b^2, \cdots\ \}$. However, we need to adjust the method for recurrences like

$$T(n) = \begin{cases} t_0 & n = 0 \\ aT\left(\frac{n-1}{2}\right) + f(n) & n > 0 \end{cases}$$

which are recurrence relations that are very close in appearance to the $T\left(\frac{n}{b}\right)$ situation, but not identical. This is an example of what are called: Recurrence relations with secondary recurrences: The recurring part is

$$T(n_i) = aT(n_{i-1}) + f(n_i) \qquad \text{or} \qquad T(n) = aT\left(g(n)\right) + f(n)$$

where g is a function that is strictly contracting the argument: $1 \leq g(n) < n$. Frequently encountered functions are $g(n) = n - 1$, $g(n) = n - 2$, $g(n) = \frac{n}{2}$, and we have treated these functions already in earlier sections, or functions like $g(n) = \frac{n-1}{2}$ and $g(n) = \sqrt{n}$. More general functions are possible as well. Generally speaking: first solve the recurrence relation for $g(n)$, and then solve the recurrence relation for $T(n)$ only for the subset of n-values that are induced by the solution of $g(n)$, and the method is sometimes called *by substitution*, which is not to be confused with back-substitution.

7.1 Secondary Recurrence Relations with $g(n) = \frac{n-1}{2}$

Let us only consider $g(n) = \frac{n-1}{2}$ in this section. So instead of the subsequence $n \in \{2^0, 2^1, 2^2, \cdots\ \}$, we should now look into either the the subsequence $n \in \{0, 1, 3, 7, 15 \cdots\ \} = \{2^0 - 1, 2^1 - 1, 2^2 - 1, 2^3 - 1 \cdots\ \}$ or the subsequence that starts with 1 in stead of starting with 0, depending on the situation. This again allows a substitution: replace $T(n) = aT(g(n)) + f(n_i)$ by $S(m) = aS(m-1) + f(2^m - 1)$ and use the known results for one-term recurrence relations.

After unwinding the recursive definition, the expression for $T(n)$ becomes:

$$\boxed{T(n) = a^L t_0 + a^{L-1} f(1) + a^{L-2} f(3) + a^{L-3} f(7) \cdots + a^0 f(2^L - 1)}$$

or

$$\boxed{T(n) = a^{L-1} t_1 + a^{L-2} f(3) + a^{L-3} f(7) + a^{L-4} f(15) \cdots + a^0 f(2^L - 1)}$$

Again, we assume that $f(n)$ is the sum of a finite number of terms, with each term coming from the set

$$\boxed{\mathcal{F} = \Big\{c, \quad c\ \lg(n+1), \quad c\ \lg(n+1), \quad c\,n, \quad c\,n^k, \quad c\,n^k\ \lg(n+1)\Big\} \text{ and } k \in \mathbb{R}^+}$$

where k is any positive real and where each such term could occur multiple times with differing values of k. Included in this set are all (finite) polynomials, and a (finite) polynomial multiplied by a logarithmic term. These are the most common forms for an inhomogeneous term in the analysis of some of the easy and some mid-level advanced algorithms.

SECONDARY STRONG MASTER THEOREM FOR $g(n) = \frac{n-1}{2}$ AND $f(n) \in \mathcal{F}$ Let a recurrence relation be given by

$$T(n) = \begin{cases} t_1 & n = 1 \\ aT\left(\frac{n-1}{2}\right) + f(n) & n > 1 \end{cases} \text{, and } f(n) \in \mathcal{F} \text{ such that the "highest power" of } f(n) \text{ is } k \tag{39}$$

Define again α as before, and since $b = 2$, we have $\alpha = \lg\ a$. Notice also that β is not needed, since $\beta = \log_b\ 2 = 1$. The asymptotic expansion of the solution $T(n)$ is indicated in the table below, which shows the coefficients of the dominant term in the asymptotic region. We only show the specific expression only for $a = 1$ and $a = 2$:

Strong Master Theorem for secondary RecRel with $g(n) = \frac{n-1}{2}$ and $f(n) \in \mathcal{F}$ with dominant term $= c\, n^k$.			
IF	$a = 1,\ k = 0,\quad a = b^k$	**then** $T(n) \sim c\ \lg(n+1)$	**and** $T(n) = \Theta(\lg n)$
IF	$a = 1,\ k = 1,\quad a < b^k$	**then** $T(n) \sim 2\,c\,(n+1)$	**and** $T(n) = \Theta(\ n\ )$
IF	$a = 1,\ k = 2,\quad a < b^k$	**then** $T(n) \sim \frac{4}{3}c\,(n+1)^2$	**and** $T(n) = \Theta(n^2)$
IF	$a = 1,\ k \geq 2,\quad a < b^k$	**then** $T(n) \sim \frac{2^k}{2^k-1}\,c\,(n+1)^k$	**and** $T(n) = \Theta(n^k)$
IF	$a = 2,\ k = 0,\quad a > b^k$	**then** $T(n) \sim \frac{1}{2}\,(t_1 + c)(n+1)$	**and** $T(n) = \Theta(\ n\ )$
IF	$a = 2,\ k = 1,\quad a = b^k$	**then** $T(n) \sim c \cdot (n+1)\ \lg(n+1)$	**and** $T(n) = \Theta(n \lg n\ )$
IF	$a = 2,\ k = 2,\quad a < b^k$	**then** $T(n) \sim 2\,c\,(n+1)^2$	**and** $T(n) = \Theta(n^2)$
IF	$a = 2,\ k \geq 2,\quad a < b^k$	**then** $T(n) \sim \frac{2^k}{2^k-2}\,c\,(n+1)^k$	**and** $T(n) = \Theta(n^k)$

Strong Master Theorem for secondary RecRel with $g(n) = \frac{n-1}{2}$ and $f(n) \in \mathcal{F}$ with dominant term $= c\, n^k\ \lg(n+1)$.			
IF	$a = 1,\ k = 0,\quad a = b^k$	**then** $T(n) \sim \frac{1}{2}\ c\ \lg^2(n+1)$	**and** $T(n) = \Theta\left(\lg^2 n\right)$
IF	$a = 1,\ k = 1,\quad a < b^k$	**then** $T(n) \sim 2\,c\,(n+1)\ \lg(n+1)$	**and** $T(n) = \Theta\,(n\ \lg n)$
IF	$a = 1,\ k = 2,\quad a < b^k$	**then** $T(n) \sim \frac{4}{3}\ c\,(n+1)^2\ \lg(n+1)$	**and** $T(n) = \Theta\left(n^2\ \lg n\right)$
IF	$a = 1,\ k \geq 2,\quad a < b^k$	**then** $T(n) \sim \frac{2^k}{2^k-1}\ c\,(n+1)^k\ \lg(n+1)$	**and** $T(n) = \Theta\left(n^k\ \lg n\right)$
IF	$a = 2,\ k = 0,\quad a > b^k$	**then** $T(n) \sim \frac{1}{2}\ (t_1 + 3c)(n+1)$	**and** $T(n) = \Theta\,(n)$
IF	$a = 2,\ k = 1,\quad a = b^k$	**then** $T(n) \sim \frac{1}{2}c(n+1)\lg^2(n+1)$	**and** $T(n) = \Theta\left(\ n\ \lg^2 n\right)$
IF	$a = 2,\ k = 2,\quad a < b^k$	**then** $T(n) \sim 2\,c\,(n+1)^2\ \lg(n+1)$	**and** $T(n) = \Theta\left(n^2\ \lg n\right)$
IF	$a = 2,\ k \geq 2,\quad a < b^k$	**then** $T(n) \sim \frac{2^k}{2^k-2}\ c\,(n+1)^k\ \lg(n+1)$	**and** $T(n) = \Theta\left(n^k\ \lg n\right)$

If only the asymptotic class is required, then the conclusions are actually identical to the asymptotic class for standard Divide-and-Conquer recurrence relations. Separate the cases on the largest contribution to the sum, according to the relative sizes of a and b^k:

Secondary Weak Master Theorem for $g(n) = \frac{n-1}{2}$ and $f(n) \in \mathcal{F}$.			
IF $\begin{cases} 1 \leq a < b^k \\ a = b^k \\ a > b^k \end{cases}$	**THEN**	$\begin{cases} T(n) = \Theta(f(n)) \\ T(n) = \Theta(f(n) \lg n) \\ T(n) = \Theta(n^\alpha) \end{cases}$	(40a) (40b) (40c)

7.2 Exercises

1. The time complexity for two algorithms, known to correctly solve the same problem, is given by the recurrence relations:

$$T_X(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ T_X(n/2) + 1 & 2 \leq n \end{cases} \qquad T_Y(n) = \begin{cases} 0 & n \leq 1 \\ 1 & n = 2 \\ T_Y((n-1)/2) + 2 & 2 < n \end{cases}$$

Find the explicit forms of their best- and worst- case time complexity functions. Feel free to limit the domain of your functions to powers of 2, or almost powers of 2, but the answers should be in terms of n. Note: this corresponds to the two versions of binary search on an array; it also corresponds to the performance of a find on a balanced binary tree (internal nodes have informational information) and on a (2-3)-tree, where internal nodes only have directional information). In short: Is it better to make one additional comparison, and reduce the sub-problem?

7.3 Secondary Recurrence Relations with general $g(n)$

So a secondary recurrence relation with $g(n) = \frac{n-1}{2}$ was fairly reasonable. For more general functions of $g(n)$, the process is similar. Look again at the original recurrence relation,

$$T(n) = \begin{cases} t_1 & n = 1 \\ aT(g(n)) + f(n) & n > 1, \end{cases}$$

The only requirement is that $g(n)$ is again an integer and that $1 \leq g(n) < n$. Assume that $g(n)$ is invertible and that $h(n)$ is its inverse: $h(g(n)) = g(h(n)) = n$, $h(n)$ is again an integer and $n < h(n)$. So if $g(n) = n - 1$, then $h(n) = n + 1$. Similarly, if $g(n) = \frac{n}{b}$, then $h(n) = bn$, if $g(n) = \frac{n-1}{2}$ then $h(n) = 2n + 1$, and so on. Now define a sequence of numbers by $n_0 = 1$ (or whatever the boundary is in the original recurrence relation), and then $n_1 = h(n_0)$, $n_2 = h(n_1)$, and so on. Thus we have a sequence of numbers so that the $h(\cdot)$-function provides the next higher, and the $g(\cdot)$-function provides the next smaller. Now substitute the values n_m and $n_{m-1} = g(n_m)$ in the original recurrence relation, then,

$$T(n_m) = \begin{cases} t_1 & n = 1 \\ aT(n_{m-1}) + f(n_m) & n > 1, \end{cases}$$

Finally, change notation and use shorthand notation by writing $S(m)$ for $T(n_m)$, and the recurrence relation becomes much friendlier:

$$S(m) = \begin{cases} t_1 & m = 1 \\ aS(m-1) + f(n_m) & m > 1, \end{cases}$$

This is a first order recurrence relation that can usually be solved with the methods from that section. After which, you need to write the answers again in terms of n.

EXAMPLE

$$T(n) = \begin{cases} t_2 & n = 2 \\ 2\,T\left(\sqrt{n}\right) + f(n) & n > 1, \end{cases}$$

In this case, $g(n) = \sqrt{n}$ and $h(n) = n^2$. Define the sequence $n_0 = 2$, $n_1 = h(n_0) = 2^2$, $n_2 = h(n_1) = 2^4$, and so on: $n_m = h(n_{m-1}) = 2^{2^m}$.

$$S(m) = \begin{cases} t_2 & m = 0 \\ 2\,S\,(m-1) + f(2^{2^m}) & m > 1, \end{cases}$$

If the function is $f(n) = \lg n$, then $f(2^{2^m}) = 2^m$ and the recurrence relation becomes “manageable”, with solution

$$S(m) = t_2 2^m + m\,2^m$$

which needs to be converted to n, using $n = 2^{2^m}$, and thus $\lg n = 2^m$ and $m = \lg\lg n$:

$$\boxed{T(n) = t_2 \lg n + (\lg\lg n)\ \lg n}$$

If the function is $f(n) = \sqrt{n}$, then $f(2^{2^m}) = 2^{2^{m-1}}$, but unfortunately, no closed form for $S(m) = \sum_i 2^{m-i} 2^{2^i}$ appears available, although the last term dominates, by far: $S(m) \approx 2^{2^{m-1}}$ and thus $T(n) \approx \sqrt{n}$.

8 Case Study: HEAPIFY

The HEAP is an interesting and important data structure, and we ask you to refer back to your data structures course, or to the appropriate wiki-pages. In these references, it is explained how an insert and how a delete is implemented. Typically, an insertion is accomplished by placing the new element in the only location that is structurally allowed, after which the element is *percolated up*, or PERC-UP, by comparing values with it's virtual parent, until all values from new location to root location are in the appropriate order. Similarly, a deletion is accomplished by putting the root-value aside for a moment, after which the last element is removed (the only location that can be removed structurally), and it's value placed in the root location. At this point, the left- and right-subtrees are correct heaps, and the value in the root-location is likely to violate the heap property. This element is *percolated down*, or PERC-DOWN, by comparing values with it's two virtual children, swapping with the appropriate child, and continue this PERC-DOWN until done. The worst case time complexity for either a PERC-UP or a PERC-Down is logarithmic.

Almost as an afterthought, there are typically two algorithms presented to create an initial heap, given just n elements without any additional structure: a greedy method (one-by-one), and a divide-and-conquer method. Most textbooks analyze them by the carefully setting up summations. However, both van be accomplished using recurrence relations:

8.1 Greedy method for HEAPIFY, an $O(n\ \lg n)$-time algorithm

This algorithm/program is essentially the one-by-one, one-after-the-other method.

```
algorithm ConstructHeap.V1($MyA$)
    from $i = 1$ to $i = n$ do   PERC-UP$(A, i)$   end for
end algorithm ConstructHeap
```

The maximal number of comparisons when initially constructing such a heap is given by a recurrence relation:

$$T(n) = \begin{cases} 0 & n = 1 \\ T(n-1) + \lfloor \lg n \rfloor & n > 1 \end{cases} \tag{41}$$

Most textbooks do not present a recurrence relation, but rather offer a summation directly, which for the worst case analysis is set up as follows: Assume that $n = 1, 3, 7, 15 \ldots\ldots$ or $n = 2^L - 1$ for some L. Then:

The root level, level 0, has 1 element, and no comparisons are needed.
The next (virtual) level, level 1, has 2 elements. Processing each element takes 1 comparison in the worst case.
The next (virtual) level, level 2, has 4 elements. Processing each element takes 2 comparisons in the worst case.
The next (virtual) level, level 3, has 8 elements. Processing each element takes 3 comparisons in the worst case.
And so on. The virtual level ℓ, has 2^ℓ elements. Processing each element takes ℓ comparisons in the worst case.
The last virtual level, $L - 1$, has 2^{L-1} elements. Processing each element takes $L - 1$ comparisons.

Combined, this results in

$$T^W_{\text{greedy}}(n) = \sum_{\ell=0}^{L-1} \ell\, 2^\ell = L\, 2^L - 2.2^L + 2 = n\ \lg n - 2.n + 2, \qquad \text{for } n = 2^L - 1 \text{ for some } L. \tag{42}$$

8.2 Divide and Conquer for HEAPIFY, a linear time algorithmm

There is an other method, often called HEAPIFY, but some text books refer to it by a different name. It is often presented as an "aha" algorithm/program/idea because the *programs* for these two different algorithms are so much similar.

```
**algorithm** Heapify.v1($MyA$)
        **from** $k = n/2$ **down to** $k = 1$ **do**  PERC-DOWN($A, k$) **end for**
**end algorithm** Heapify
```

The time complexity is then often an impressive derivation and results in a summation, which is then bounded. From algorithmic design viewpoint, the presented algorithm/program is regarded as a very clever implementation of a divide and conquer algorithm, an implementation that avoids recursion and uses iterative programming. Thus, from this viewpoint, it is equivalent to :

```
**algorithm** HEAPIFY.v2 ($n$ elements)
**if** $size == 2$ or 3
    **then** Construct right heap from use 1 or 2 comparisons construct a heap
    **else** Isolate the root element
        Construct the left-heap, HEAPIFY.v2($(n-1)/2$ elements)
        Construct the right-heap, HEAPIFY.v2($(n-1)/2$ elements)
        Call  PERC-DOWN($A, root$)
**end algorithm**
```

The maximal number of comparisons when initially constructing a heap in this manner is, remember that $n = 2^L - 1$ for some L:

$$T(n) = \begin{cases} 0 & n = 1 \\ 2T\left(\dfrac{n-1}{2}\right) + 2\ (\lg(n+1) - 2) & n > 1 \end{cases} \tag{43}$$

and solution

$$T(n) = 2(n+1) - 2\ \lg(n+1) - 2 = 2n - 2\lg(n+1) \sim 2n, \qquad \text{for } n = 2^L - 1 \text{ for some } L. \tag{44}$$

8.3 Rejoinder and Conclusion

An initial heap can be constructed in linear time. In fact, in less than $2n$ time.

9 Some other considerations

The closed form expression for $T(n)$ is often not as informative as its asymptotic behavior. And if the ultimate desire is to find this asymptotic classification, then there are opportunities to pay less attention to details that ultimately disappear in this asymptote. Prime examples are the Master Theorem, and bounding recurrence relations.

9.1 Intermediate Techniques – Linear Algebra notation

It was already indicated that finding the roots of the characteristic equation for higher order linear recurrence relations may be ill-conditioned. Rewriting the equations in matrix form could bypass this difficulty. But the results are even deeper. Many of the sequences we study in this course are integer sequences, and both iterative and recursive algorithms compute the members of the sequence while working in the integers. We have also seen that explicit closed forms can be found sometimes, that involve real numbers, such as the Fibonacci numbers. But real numbers may not have a finite representation, so we have found representations that are mathematically equivalent, but are not numerically equivalent. Consider the Fibonacci numbers; they are integers, are defined by the recursion $F_{n+1} = F_n + F_{n-1}$ and both the recursion and the Fibonacci numbers themselves can be represented in finite precision. Yet, an explicit closed form is

$$F_n = \frac{\phi_1^n - \phi_2^n}{\sqrt{5}}, \text{ where } \begin{cases} \phi_1 = \frac{1}{2}\left(1+\sqrt{5}\right) \approx 1.618033988 & \text{(the golden ratio)} \\ \phi_2 = \frac{1}{2}\left(1-\sqrt{5}\right) \approx -.6180339880 \end{cases}$$

Since the absolute value of ϕ_2 is less than 1, $F_n \approx \frac{1}{2}\,\phi_1^n$, and the numerical value can be obtained as the nearest integer to $\frac{1}{2}\,\phi_1^n = \frac{1}{2}\,exp(n\,\ln(\phi_1)) \approx \frac{1}{2}\,exp(0.4812118246\,n)$.

These closed forms involves radicals (i.e. $\sqrt{\cdot}$) and their approximated real numbers. Unfortunately, these cannot be represented in finite word length and finding values has become a numeric process, with its own complications (approximation error, error propagation, convergence, . . .). This course does not address these issues, but it does show that the Fibonacci numbers grow exponentially. It should also be noted that these expression are frowned upon and avoided by many algorithm theoreticians as only finite representations are acceptable. In the next paragraph, we explore a method to determine the exact value of the n-th Fibonacci number in $\lg n$ time, while still using exact representation and exact operations.
Representing the Fibonacci recurrence using matrix-vector notation, we get a recurrence relation between vectors,

$$\begin{bmatrix} F_{n+1}, & F_n \end{bmatrix} = \begin{bmatrix} F_n, & F_{n-1} \end{bmatrix} \times \begin{bmatrix} 1, & 1 \\ 1, & 0 \end{bmatrix} = \begin{bmatrix} F_1, & F_0 \end{bmatrix} \times \begin{bmatrix} 1, & 1 \\ 1, & 0 \end{bmatrix}^n . \tag{45}$$

So that the Fibonacci numbers are expressed as

$$F_{n+1} = \begin{bmatrix} F_1, & F_0 \end{bmatrix} \times \mathbf{M}^n \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} , \tag{46}$$

where $\mathbf{M} = \left(\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, so that F_{n+1} can be obtained fairly efficient without loss of numerical error by powering the matrix: $\mathbf{M} \to \mathbf{M}^2 \to \mathbf{M}^4 \to \mathbf{M}^8 \cdots$, i.e. a logarithmic number of matrix multiplications. Of course, the eigenvalues of the matrix $\mathbf{M}$ are the ϕ_1 and ϕ_2 from above. Below, we extend this idea to higher order difference equations.

Encouraged by this, consider now the k^{th} order recurrence relations of the form, which we only show for a fifth-order ($k = 5$), to avoid notational difficulties.

$$T(n) = \begin{cases} T(1) = t_1 = 11 & n = 1 \\ T(2) = t_2 = 12 & n = 2 \\ T(3) = t_3 = 13 & n = 3 \\ T(4) = t_4 = 14 & n = 4 \\ T(5) = t_5 = 15 & n = 5 \\ a_1\,T(n-1) + a_2\,T(n-2) + a_3\,T(n-3) + a_4\,T(n-4) + a_5\,T(n-5)\ + f(n) & n > 5 \end{cases} \tag{47}$$

Introduce the (row-) vectors

$$\mathbf{u} = \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \end{array}\right] \end{array} \tag{48}$$

$$\mathbf{T}(5) = \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc} t_5 & t_4 & t_3 & t_2 & t_1 \end{array}\right] \end{array} \tag{49}$$

and for $n > 5$

$$\mathbf{T}(n) = \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc} T(n) & T(n-1) & T(n-2) & T(n-3) & T(n-4) \end{array}\right] \end{array} \tag{50}$$

Also, define the matrix (matrices like this are often called *companion* matrices, and the characteristic polynomial of the matrix is conveniently identical to the characteristic polynomial of our recurrence relation)

$$\mathbf{M} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & a_1 & 1 & 0 & 0 & 0 \\ 2 & a_2 & 0 & 1 & 0 & 0 \\ 3 & a_3 & 0 & 0 & 1 & 0 \\ 4 & a_4 & 0 & 0 & 0 & 1 \\ 5 & a_5 & 0 & 0 & 0 & 0 \end{array} \tag{51}$$

Please note that the recurrence relation can be written with this new notation:

$$\mathbf{T}(n) = \begin{cases} \mathbf{T}(5) & n = 5 \\ \mathbf{T}(n-1)\mathbf{M} + f(n)\mathbf{u} & n > 5 \end{cases} \tag{52}$$

If you need convincing, just write it out (please verify every column for correctness)

$$\mathbf{T}(n) = \left[\begin{array}{ccccc} T(n) & T(n-1) & T(n-2) & T(n-3) & T(n-4) \end{array}\right] \tag{53}$$

$$= \left[\begin{array}{ccccc} T(n-1) & T(n-2) & T(n-3) & T(n-4) & T(n-5) \end{array}\right] \left[\begin{array}{ccccc} a_1 & 1 & 0 & 0 & 0 \\ a_2 & 0 & 1 & 0 & 0 \\ a_3 & 0 & 0 & 1 & 0 \\ a_4 & 0 & 0 & 0 & 1 \\ a_5 & 0 & 0 & 0 & 0 \end{array}\right] \tag{54}$$

$$+ \quad f(n) \left[\begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \end{array}\right] \tag{55}$$

We can now use standard back-substitution, but with matrices

$$\begin{aligned}
\mathbf{T}(n) &= \mathbf{T}(n-1)\,\mathbf{M} + f(n)\,\mathbf{u} && (56)\\
\mathbf{T}(n-1)\,\mathbf{M} &= \mathbf{T}(n-2)\,\mathbf{M}^2 + f(n-1)\,\mathbf{u}\,\mathbf{M} && (57)\\
\mathbf{T}(n-2)\,\mathbf{M}^2 &= \mathbf{T}(n-3)\,\mathbf{M}^3 + f(n-2)\,\mathbf{u}\,\mathbf{M}^2 && (58)\\
\mathbf{T}(n-3)\,\mathbf{M}^3 &= \mathbf{T}(n-4)\,\mathbf{M}^4 + f(n-3)\,\mathbf{u}\,\mathbf{M}^3 && (59)\\
\vdots \quad & \quad \vdots \\
\mathbf{T}(6)\,\mathbf{M}^{n-6} &= \mathbf{T}(5)\,\mathbf{M}^{n-5} + f(6)\,\mathbf{u}\,\mathbf{M}^{n-6} && (60)
\end{aligned}$$

and thus

$$\mathbf{T}(n) = \mathbf{T}(5)\,\mathbf{M}^{n-5} + f(6)\,\mathbf{u}\,\mathbf{M}^{n-6} + f(7)\,\mathbf{u}\,\mathbf{M}^{n-7} + \cdots + f(n-2)\,\mathbf{u}\,\mathbf{M}^2 + f(n-1)\,\mathbf{u}\,\mathbf{M} + f(n)\,\mathbf{u} \tag{61}$$

Isolating the first component (the actual term we were looking for in the first place)

$$T(n) = \mathbf{T}(5)\,\mathbf{M}^{n-5}\,\mathbf{u}^T + f(6)\,\mathbf{u}\,\mathbf{M}^{n-6}\,\mathbf{u}^T + f(7)\,\mathbf{u}\,\mathbf{M}^{n-7}\,\mathbf{u}^T + \cdots + f(n-2)\,\mathbf{u}\,\mathbf{M}^2\,\mathbf{u}^T + f(n-1)\,\mathbf{u}\,\mathbf{M}\,\mathbf{u}^T + f(n)\,\mathbf{u}\,\mathbf{u}^T \tag{62}$$

So this is an alternate expression for the solution of $T(n)$, which can be used to study the asymptotic behavior of the solution: another tug-of-war between $\mathbf{T}(5)\ \mathbf{M}^{n-5}\ \mathbf{u}^T$ and $f(n)\ \mathbf{u}\ \mathbf{u}^T$. Now as n becomes large, then $\mathbf{M}^n \sim \sigma^n \mathbf{x}\ \mathbf{y}^T$, where σ is the largest eigenvalue (in absolute value) of the matrix, which we assume to be unique for now, and where $\mathbf{x}$ and $\mathbf{y}$ are the left- and right eigenvectors. The numeric values can be obtained fairly efficient without loss of numerical error by powering the matrix: $\mathbf{M} \rightarrow \mathbf{M}^2 \rightarrow \mathbf{M}^4 \rightarrow \mathbf{M}^8 \cdots \sim \sigma^n \mathbf{x}\ \mathbf{y}^T$.

Thus compare $f(n)$ with σ^n, and arrive at a master theorem for these recurrence relations, which we show in full generality.

WEAK MASTER THEOREM FOR mTH ORDER REC. RELATIONS

The asymptotic behavior of $T(n)$, defined by the recurrence relation $T(n) = a_1\,T(n-1) + a_2\,T(n-2) + \cdots + a_k\,T(n-k) + f(n)$, is, with σ equal to the largest eigenvalue of $\mathbf{M}$ of multiplicity m (which is also the largest root of the characteristic polynomial)

$T(n) = \Theta(n^m\ \sigma^n)$	if $f(n) = o(\sigma^n)$	(case WMTm-a)
$T(n) = \Theta(n\ f(n))$	if $f(n) = \Theta(\sigma^n)$	(case WMTm-b)
$T(n) = \Theta(f(n))$	if $f(n) = \omega(\sigma^n)$	(case WMTm-c)

The proofs are just a little trickier as compared to the previous chapter, but it should be noted that $\mathbf{u}\ \mathbf{M}^k\ \mathbf{u}^T$ is just the $(1,1)$ element of the matrix $\mathbf{M}^k$. Proofs are becoming easier by isolating the largest eigenvalue and by writing $\mathbf{M} = \sigma\ \mathbf{M}'$, or by using a similarity transformation to a diagonal matrix, and turning the proof into an exercise in summations.

9.2 Bounding recurrence relations

Suppose you have a very generic looking recurrence relation

$$\begin{cases} T(i) = t_i, & i = 0, 1, \cdots d \\ T(n) = g(T(n-1), T(n-2)) + f(n) & n > d \end{cases}$$

where either the function g or the function f is rather complicated, or their arguments involve floor and ceiling functions, you may be able to define two bounding recurrence relations T^- and T^+ such that $T^-(n) \leq T(n) \leq T^+(n)$ for all n under consideration. If both T^- and T^+ are in the same asymptotic class, then so is T. If you can find only T^+ with $T(n) \leq T^+(n)$, then you still have an upper bound for T: $T = O(T^+)$.

10 Intermediate Techniques – Generating functions

The methods we have introduced so far are sufficient for the 'first encounter' with recurrence relations, but do not cover the full range needed for combinatorial and algorithm analysis. This section sketches some additional techniques, although there are many interesting recurrence relations that are go beyond these methods as well.

This method is the primary method for inhomogeneous recurrence relations and certain non-linear recurrence relations. These are best demonstrated with examples, and future handouts will incorporate full-history equations such as the quick-sort recurrence, and the Catalan recurrence.

10.1 A second order homogeneous equation

Consider the two-term recurrence relations we have studied before,

$$\begin{cases} T(0) = t_0,\ T(1) = t_1 \\ T(n) = a_1\, T(n-1) + a_2\, T(n-2) \qquad n > 1 \end{cases}$$

We expressed the solution in terms of the roots r_1 and r_2 (of just r_1 if it is a double root) of the associated quadratic equation $x^2 - a_1 x - a_2 = 0$, but deferred a full development to the current section, where we show how generating functions can be used.

Write the recurrence relation for several values of n, and multiply each such equation by x^n:

$$\begin{cases} T(0)\, x^0 = t_0\, x^0 \\ T(1)\, x^1 = t_1\, x^1 \\ T(2)\, x^2 = a_1\, t_1\, x^2 + a_2\, t_0\, x^2 \\ T(3)\, x^3 = a_1\, T(2)\, x^3 + a_2\, t_1\, x^3 \\ T(4)\, x^4 = a_1\, T(3)\, x^4 + a_2\, T(2)\, x^4 \\ T(5)\, x^5 = a_1\, T(4)\, x^5 + a_2\, T(3)\, x^5 \\ T(6)\, x^6 = a_1\, T(5)\, x^6 + a_2\, T(4)\, x^6 \\ \quad\vdots \quad = \quad \vdots \quad + \quad \vdots \end{cases}$$

Now define the (ordinary) generating function $G_T(x) = t_0\, x^0 + t_1\, x^1 + T(2)\, x^2 + T(3)\, x^3 + T(4)\, x^4 + \cdots$, and add these equations. The left side of these equations add to $G_T(x)$, while the terms on the right side can also be combined:

$$\begin{aligned} G_T(x) \quad &= \quad t_0\, x^0 + t_1\, x^1 \\ &\qquad\qquad + a_1\ \left(\qquad\quad t_1\, x^1 + T(2)\, x^2 + T(3)\, x^3 + \cdots \right)\ x^1 \\ &\qquad\qquad + a_2\ \left(t_0\, x^0 + t_1\, x^1 + T(2)\, x^2 + T(3)\, x^3 + \cdots \right)\ x^2 \\ &= \quad t_0\, x^0 + t_1\, x^1 + a_1\ \left(-t_0\, x^0 + G_T(x) \right)\ x^1 + a_2\ \left(\ G_T(x) \right)\ x^2 \end{aligned}$$

This gives a single equation, which can be solved for $G_T(x)$:

$$G_T(x)\ \left(1 - a_1\, x - a_2\, x^2\right) = t_0\, x^0 + t_1\, x^1 - a_1\, t_0\, x^1 = t_0 + (t_1 - a_1\, t_0)x$$

and

$$G_T(x) = \frac{t_0 + (t_1 - a_1\, t_0)x}{1 - a_1\, x - a_2\, x^2} = (t_0 + (t_1 - a_1\, t_0)x\ )\ \frac{1}{1 - a_1\, x - a_2\, x^2}$$

Note that $G_T(x)$ is a rational function in x (i.e. a ratio of polynomials), and that $T(i)$ is the coefficient of x^i in the Taylor series of $G_T(x)$, so we will convert the rational function back into a power series so that the coefficients can be 'picked off'. The common method for a rational function is to write it first as partial fractions, and then take each term in this partial fraction separately. The method relies on root-finding of the denominator polynomial and most calculus books explain this process well. We use the explanation in [CS404appendixBender], and assume that the quadratic equation has two separate roots, $(\ 1 - a_1\, x - a_2\, x^2) =$

$(1 - u_1 x)(1 - u_2 x)$. In that case, straight algebraic manipulation leads to the following expressions:

$$\begin{aligned}\frac{1}{(1-u_1x)(1-u_2x)} &= \frac{\frac{u_1}{u_1-u_2}}{1-u_1x} + \frac{\frac{u_2}{u_2-u_1}}{1-u_2x} \\ &= \tfrac{u_1}{u_1-u_2}\left(1+u_1^1x^1+u_1^2x^2+u_1^3x^3+\cdots\right) + \tfrac{u_2}{u_2-u_1}\left(1+u_2^1x^1+u_2^2x^2+u_2^3x^3+\cdots\right)\end{aligned}$$

and

$$\begin{aligned}\frac{x}{(1-u_1x)(1-u_2x)} &= \frac{\frac{1}{u_1-u_2}}{1-u_1x} + \frac{\frac{1}{u_2-u_1}}{1-u_2x} \\ &= \tfrac{1}{u_1-u_2}\left(1+u_1^1x^1+u_1^2x^2+u_1^3x^3+\cdots\right) + \tfrac{1}{u_2-u_1}\left(1+u_2^1x^1+u_2^2x^2+u_2^3x^3+\cdots\right)\end{aligned}$$

These can be combined to get the coefficients for the powers of x^i.

10.2 Example

Consider the same example that we have covered before:

$$\begin{cases} T(0)=4,\ T(1)=4 \\ T(n)=2T(n-1)+3T(n-2) \quad n>2 \end{cases}$$

Write the recurrence relation for several values of n, and multiply each such equation by x^n:

$$\begin{cases} T(0)\,x^0 = 4\,x^0 \\ T(1)\,x^1 = 4\,x^1 \\ T(2)\,x^2 = 2\,t_1\,x^2 + 3\,t_0\,x^2 \\ T(3)\,x^3 = 2\,T(2)\,x^3 + 3\,t_1\,x^3 \\ T(4)\,x^4 = 2\,T(3)\,x^4 + 3\,T(2)\,x^4 \\ \quad\vdots \quad = \quad \vdots \quad + \quad \vdots \end{cases}$$

Now define the (ordinary) generating function $G_T(x) = 4\,x^0 + 4\,x^1 + T(2)\,x^2 + T(3)\,x^3 + T(4)\,x^4 + \cdots$, and add these equations,

$$\begin{aligned} G_T(x) &= t_0\,x^0 + t_1\,x^1 \\ &\quad + 2\,\left(4\,x^1 + T(2)\,x^2 + T(3)\,x^3 + \cdots\right)\,x^1 \\ &\quad + 3\,\left(4\,x^0 + 4\,x^1 + T(2)\,x^2 + \cdots\right)\,x^2 \\ &= 4\,x^0 + 4\,x^1 + 2\,\left(-4\,x^0 + G_T(x)\right)\,x^1 + 3\,\left(G_T(x)\right)\,x^2 \end{aligned}$$

This gives a single equation, which can be solved for $G_T(x)$:

$$G_T(x)\,\left(1 - 2\,x - 3\,x^2\right) = 4\,x^0 + 4\,x^1 - 2\,.4\,x^1 = 4 + (4 - 2.\,4)x = 4 - 4\,x$$

and

$$G_T(x) = \frac{4-4\,x}{1-2\,x-3\,x^2} = \frac{4-4\,x}{1-2\,x-3\,x^2} = \frac{4-4\,x}{(1+x)(1-3\,x)}$$

This case leads to the following expressions:

$$\begin{aligned}
\frac{1}{(\,1+x\,)(\,1-3x\,)} &= \frac{1}{4}\,\frac{1}{1+x}+\frac{3}{4}\,\frac{1}{1-3\,x} \\
&= \frac{1}{4}\,\left(1-x^1+x^2-x^3+\cdots\right)+\frac{3}{4}\,\left(1+3\,x^1+3^2\,x^2+3^3\,x^3+\cdots\right) \\
&= \quad \cdots + \quad (\frac{1}{4}\,(-1)^i+\frac{3}{4}\,(3)^i\,)\,x^i \quad + \quad \cdots
\end{aligned}$$

and

$$\begin{aligned}
\frac{x}{(\,1+x\,)(\,1-3x\,)} &= -\frac{1}{4}\,\frac{1}{1+x}+\frac{1}{4}\,\frac{1}{1-3\,x} \\
&= -\frac{1}{4}\,\left(1-x^1+x^2-x^3+\cdots\right)+\frac{1}{4}\,\left(1+3\,x^1+3^2\,x^2+3^3\,x^3+\cdots\right) \\
&= \quad \cdots + \quad (-\frac{1}{4}\,(-1)^i+\frac{1}{4}\,(3)^i)\,x^i + \quad \cdots
\end{aligned}$$

These are then combined to get the coefficients of $G_T(x)$ for the powers of x^i:

$$\begin{aligned}
G_T(x) &= \quad \cdots + \quad (4\,\frac{1}{4}\,(-1)^i+4\,\frac{3}{4}\,(3)^i\,+4\,\frac{1}{4}\,(-1)^i-4\,\frac{1}{4}\,(3)^i)\,x^i + \quad \cdots \\
&= \quad \cdots + \quad (2\,(-1)^i+2\,(3)^i\,)\,x^i + \quad \cdots
\end{aligned}$$

and finally,

$$T(n)=2\,(-1)^i+2\,(3)^i\,.$$

10.3 A second order inhomogeneous equation

The inhomogeneous counter part follows immediately:

$$\begin{cases} T(0)=t_0,\;T(1)=t_1 \\ T(n)=a_1\,T(n-1)+a_2\,T(n-2)+f(n) \qquad n>1 \end{cases}$$

As before, write the recurrence relation for several values of n, and multiply each such equation by x^n:

$$\begin{cases}
T(0)\,x^0=t_0\,x^0 \\
T(1)\,x^1=t_1\,x^1 \\
T(2)\,x^2=a_1\,t_1\,x^2+a_2\,t_0\,x^2+ \quad f(2)\,x^2 \\
T(3)\,x^3=a_1\,T(2)\,x^3+a_2\,t_1\,x^3+ \quad f(3)\,x^3 \\
T(4)\,x^4=a_1\,T(3)\,x^4+a_2\,T(2)\,x^4+ \quad f(4)\,x^4 \\
T(5)\,x^5=a_1\,T(4)\,x^5+a_2\,T(3)\,x^5+ \quad f(5)\,x^5 \\
T(6)\,x^6=a_1\,T(5)\,x^6+a_2\,T(4)\,x^6+ \quad f(6)\,x^6 \\
\quad\vdots \quad = \quad \vdots \quad + \quad \vdots \quad + \quad \vdots
\end{cases}$$

Now define both the (ordinary) generating function of T, $G_T(x)=t_0\,x^0+t_1\,x^1+T(2)\,x^2+T(3)\,x^3+T(4)\,x^4+\cdots$, as well as the generating function for f: $G_f(x)=f_0\,x^0+f_1\,x^1+f(2)\,x^2+f(3)\,x^3+f(4)\,x^4+\cdots$. We will now assume that $f_0=0$

and $f_1 = 0$, but this can be changed to accommodate 'easily recognizable' functions. Again, the equations are added:

$$\begin{aligned} G_T(x) &= t_0\,x^0 + t_1\,x^1 \\ &\qquad + a_1\,\left(\qquad t_1\,x^1 + T(2)\,x^2 + T(3)\,x^3 + \cdots \right)\,x^1 \\ &\qquad + a_2\,\left(t_0\,x^0 + t_1\,x^1 + T(2)\,x^2 + T(3)\,x^3 + \cdots \right)\,x^2 \\ &\qquad + \quad \left(\qquad\qquad f(2)\,x^2 + f(3)\,x^3 + \cdots \right) \\ &= t_0\,x^0 + t_1\,x^1 + a_1\,\left(-t_0\,x^0 + G_T(x) \right)\,x^1 + a_2\,\left(G_T(x) \right)\,x^2 + G_f(x) \end{aligned}$$

This gives a single equation, which can be solved for $G_T(x)$:

$$G_T(x)\,\left(1 - a_1\,x - a_2\,x^2\right) = t_0\,x^0 + t_1\,x^1 - a_1\,t_0\,x^1 + G_f(x) = t_0 + (t_1 - a_1\,t_0)x + G_f(x)$$

and

$$G_T(x) = \frac{t_0 + (t_1 - a_1\,t_0)x + G_f(x)}{1 - a_1\,x - a_2\,x^2}$$

Note that $T(i)$ is the coefficient of x^i in the power series of $G_T(x)$, and the denominator polynomial can still be converted to a power series as before. But the appearance of $G_f(x)$ makes it less obvious how to 'pick off' the proper coefficients. The underlying principle remains the same, only the amount of manipulation has been increased. If $G_f(x)$ is also a rational function, then you are probably better off to simplify $G_T(x)$ first, but is very much a case-by-case scenario.

10.4 Epilogue

We have skipped a lot of detail that is needed to prove that this method works. We have taken a sequence $T(n)$ of numbers and transformed it into a function $G_T(x)$. We have not indicated the domain of either. In our case, $T(n)$ are integers since they counts combinatorial objects or they count timing functions. We have not indicated anything about x (we take it to be real) and we have ignored the radius of convergence of $G_T(x)$, which may very well be zero, since the rapid growth of the $T(n)$ as coefficients. Future additions to this hand-out will go into details here.

Generating functions themselves form a large area of study, and various special situation generating functions exist: probability generating functions, moment generating functions, ordinary generating functions, exponential generating functions, and Z-transforms. They all differ from one another, sometimes in subtle ways. We have applied them to counting functions (timing and combinatorial counts) which are positive integers that may grow so fast that $G_T(x)$ may not converge, even for small x. In other cases, the coefficients (like our $T(n)$ could be real, complex, or even functions themselves. Also, the domain of x could be real, or complex. These issues go beyond the current application.

11 Case Study: Catalan Recursion

Let us look at a non-trivial example where we use only the addition- and the multiplication principle to find the number of binary trees that can be constructed with n nodes, and we then extend the result (or refine) to finding the number of binary trees that can be constructed with n nodes that have depth (or height) d.

11.1 The Catalan Number

Suppose we want the find the number of binary trees we can construct from n nodes, we call this $C(n)$. The answer is fairly easy for $n = 1, n = 2$ and $n = 3$: $C(1) = 1, C(2) = 2$ and $C(3) = 5$. The binary tree is defined recursively, and our counting example follows this as well: Every binary with n nodes, will have one node as root, a number of nodes in the left-subtree, and the remaining nodes in the right-subtree. The number of nodes is distributed over the left-and right subtree and is either $\langle 0 \mid 1 \mid n-1\rangle$, or $\langle 1 \mid 1 \mid n-2\rangle$, or $\langle 2 \mid 1 \mid n-3\rangle$, through $\langle n-2 \mid 1 \mid 1\rangle$ and $\langle n-1 \mid 1 \mid 0\rangle$, where we used the $|$-symbol to delimit the nodes. Furthermore, these splits are mutually exclusive and collectively exhaustive: A particular binary tree falls into exactly one such split. Thus the desired count of $C(n)$ is the sum over all possible splits,

$$C(n) = C(\langle 0 \mid 1 \mid n-1\rangle) + C(\langle 1 \mid 1 \mid n-2\rangle) + C(\langle 2 \mid 1 \mid n-3\rangle) \cdots + \cdots C(\langle n-2 \mid 1 \mid 1\rangle) + C(\langle n-1 \mid 1 \mid 0\rangle)$$

where we used suggestive notation $C(\langle i \mid 1 \mid n-1-i\rangle)$ to indicate the number of binary trees we can construct with the particular split. But notice that the particular choice we make for the left- subtree is independent of the particular choice for the right- subtree, so these two choices are to be multiplied, using the multiplication rule: $C(\langle i \mid 1 \mid n-1-i\rangle) = C(i) \times C(1) \times C(n-1-i)$. So putting it all together:

$$C(n) = C(0)C(1)C(n-1) \;+\; C(1)C(1)C(n-2) \;+\; C(2)C(1)C(n-3) \;+\; \cdots \;+\; C(0)C(1)C(n-1). \tag{63}$$

or

$$\boxed{C(n) = \sum_{r=1}^{n} C(r-1)C(1)C(n-r) = \sum_{r=1}^{n} C(r-1)C(n-r)}. \tag{64}$$

This is a recurrence relation which can best be solved using the generating function techniques discussed in an earlier section and gives the Catalan number. Define the (ordinary) generating function for the Catalan numbers as $G_c(z) = C(0)\ z^0 + C(1)\ z^1 + C(2)\ z^2 + C(3)\ z^3 + C(4)\ z^4 + \cdots$. Now notice that the square of this generating function is $G_c^2(z)$,

$$G_c^2(z) = C(0)C(0) + (C(0)C(1) + C(1)C(0))\ z^1 + (C(0)C(2) + C(1)C(1) + C(2)C(0))\ z^2 + \tag{65}$$

where the expression that forms the coefficient of z^n is recognized as the right-hand side of the recursion for $C(n)$, thus

$$G_c^2(z) = C(1) + (C(2))\ z^1 + (C(3))\ \ z^2 + \tag{66}$$

so that

$$G_c(z) = C(0) + z\ \left(C(1) + C(2)\ z^1 + C(3)\ \ z^2 + \cdots\right) = 1 + z\, G_c^2(z) \tag{67}$$

This is a quadratic equation, where the unknown is a function, and the solution (need to pick the $+$-sign because $G_c(0)$ must be 1) can be written as either of the two fractions:

$$G_c(z) = \frac{1-\sqrt{1-4z}}{2z} = \frac{2}{1+\sqrt{1-4z}} \tag{68}$$

The Catalan number is then the coefficient in the Taylor expansion of this function,

$$\boxed{C(n) = \frac{1}{n+1}\binom{2n}{n}}. \tag{69}$$

The numeric value of the Catalan numbers are easily generated: $1, 1, 2, 5, 14, 42, 132, \ldots$. For n growing large, the Catalan number is approximately $C(n) \approx \frac{4^n}{n^{3/2}\sqrt{\pi}}$.

These Catalan numbers occur quite often when counting discrete objects: number of stack-permutations, number of queue-permutations, number of ways to parenthesize an expression, Dyck paths, Motzkin paths, non-crossing partitions of a polygon, arrival and departure events in a queue, and so on. (More than 150 situations have been identified where the Catalan numbers are used to express the answer, see Stanley's book-addition). Warning: The Catalan number is also associated with full binary trees with n leaves, and the two are similar, but not equal.

12 Case Study: Quick sort recursion

Quick sort and it's analysis s rather fundamental in the computing sciences. First, and perhaps foremost, it is one of the most commonly used sorting algorithms, and furthermore, it's analysis is a fore bearer of things to come; many other algorithms have a rather similar set-up. Because its popularity there are many variations and improvements either implemented or assigned as home work and projects in academia.

Some assumptions: The input of n elements is assumed to be a random permutation, that is, every permutation is equally likely (uniformly distributed over all permutations of $n!$ different permutations of the data elements). This is sometimes referred to as a random permutation. Let us assume that the pivot element is at a predetermined position, e.g. the first element. After selecting the pivot element, the quick sort algorithm compares all other elements against the pivot (at a cost of $T_{split}(n) = n - 1$) and splits the array in three parts: 1. "Elements less, or equal to the pivot", 2. "The pivot itself" and 3. "Elements larger, or equal to the pivot". Parts 1 and 3 need to be sorted, and this is done recursively.

So the question is, where did the pivot end up? We assumed a random permutation, and the location of the pivot is now random; the element will end up at location i, with equal probability. The relative position is called *the rank* of an element, and the event where the rank of the pivot is i, has a probability

$$\Pr[\text{ rank(pivot)} = i\,] = \frac{1}{n}, \quad \text{for all } i = 1 \cdots n$$

So assuming that the $\text{rank(pivot)} = i$, and assuming that all elements are unique, there are three sub-arrays, of sizes $i - 1$ (of elements $\leq$ the pivot), 1 (the pivot), and $n - i$ (of elements $\geq$ the pivot).

So conditioned on you knowing that the $\text{rank(pivot)} = i$, we have

$$T^A(n \mid \text{ rank(pivot)} = i\ ) = (n-1) + T^A(i-1) + T^A(1) + T^A(n-i) \tag{70}$$

Realizing that $T^A(1) = 0$ and by unconditioning[4], we get

$$\begin{aligned}
T^A(n) &= \sum_{i=1}^{n} \Pr[\text{ rank(pivot)} = i\,]\ \underbrace{T^A(n \mid \text{ rank(pivot)} = i\ )} && (71)\\
&= \sum_{i=1}^{n} \left[\frac{1}{n} \quad \times \quad \left\{\overbrace{(n-1) + T^A(i-1) + T^A(n-i)}\right\}\right] && (72)\\
&= (n-1) + \frac{1}{n}\sum_{i=1}^{n}\left\{T^A(i-1) + T^A(n-i)\right\} && (73)\\
&= \frac{1}{n}\left\{n(n-1) + 2\left\{T^A(0) + T^A(1) + \cdots + T^A(n-2) + T^A(n-1)\right\}\right\} && (74)
\end{aligned}$$

This recurrence relation is affectionally known as the 'Quick sort recurrence relation', and needs to be manipulated quite a bit before we can solve it with any of the earlier methods described. First multiply this equation by n, and write the resulting equation

[4]you may recognize the 'law of total expectation from your probability class

again, but now for argument $(n-1)$:

$$\begin{aligned} n\,T^A(n) &= n(n-1) \qquad\quad +2\,\left\{T^A(0)+T^A(1)+\cdots+T^A(n-2)+T^A(n-1)\right\} && (75)\\ (n-1)\,T^A(n-1) &= (n-1)(n-2)+2\,\left\{T^A(0)+T^A(1)+\cdots+T^A(n-2) \qquad\qquad\quad \right\} && (76)\end{aligned}$$

Now subtract the second equation from the first, to get another recurrence relation that looks somewhat friendlier:

$$n\,T^A(n)-(n-1)\,T^A(n-1)=n(n-1)-(n-1)(n-2)+2\,T^A(n-1) \tag{77}$$

This can be simplified, (using algebraic adding/subtracting):

$$n\,T^A(n)=2\,(n-1)+(n+1)\,T^A(n-1) \tag{78}$$

Now divide both sides by $n(n+1)$:

$$\frac{T^A(n)}{n+1}=\frac{T^A(n-1)}{n}+2\,\frac{n-1}{n(n+1)}=\frac{T^A(n-1)}{n}+2\,\left\{\frac{2}{n+1}-\frac{1}{n}\right\} \tag{79}$$

This is finally a form that can be solved. Either use a substitution of variables, or unwind the recurrence relation with back substitution. We will do the latter,

$$\begin{aligned}\frac{T^A(n)}{n+1} &= 2\left\{\left(\frac{2}{n+1}-\frac{1}{n}\right)+\left(\frac{2}{n}-\frac{1}{n-1}\right)+\left(\frac{2}{n-1}-\frac{1}{n-2}\right)+\cdots+\left(\frac{2}{3}-\frac{1}{2}\right)\right\} && (80)\\ &= 2\left\{\frac{2}{n+1}+\frac{1}{n}+\frac{1}{n-1}+\cdots+\frac{1}{3}-\frac{1}{2}\right\} && (81)\\ &= 2\left\{\frac{2}{n+1}+\frac{1}{n}+\frac{1}{n-1}+\cdots+\frac{1}{3}+\frac{1}{2}+1 \quad -\frac{1}{2}-1-\frac{1}{2}\right\} && (82)\\ &= 2\left\{\frac{2}{n+1}+H_n-2\right\} && (83)\end{aligned}$$

where the sum reciprocals is written as the harmonic number, H_n.

$$\boxed{T^A(n)=2(n+1)H_n-4n} \tag{84}$$

This Harmonic number is approximately (see Wiki) $H_n=\ln n+\gamma+\frac{1}{2n}-\frac{1}{12n^2}+o(n^{-3})=0.6931471806\lg n+\gamma+\frac{1}{2n}-\frac{1}{12n^2}+o(n^{-3})$, where γ is known as the Euler-Mascheroni constant, $\gamma=0.5772156649$. Putting this all back in the equation, gives

$$T^A(n)=2(n+1)\ln n+(2\gamma-4)n+(2\gamma+1)+o(1) \tag{85}$$

or, less precise but more indicative of asymptotic behavior:

$$\boxed{\begin{aligned} T^A(n) &= 2\,n\ln n+o(n\ln n)=\Theta(\,n\ln n) && (86)\\ &= 1.386\,n\lg n+o(n\lg n)=\Theta(\,n\lg n) && (87)\end{aligned}}$$

This completes the average case analysis of the basic quick sort algorithm. Please note, that the best case for quick sort occurs when parts 1 and 3 are always qual in size,

$$T^B(n)=(n-1)+T^B(\tfrac{n-1}{2})+T^B(1)+T^B(\tfrac{n-1}{2}) \tag{88}$$

Assuming that $n=2^L-1$, for some L, (and thus $2^L=n+1$, and $L=\lg(n+1)$), this is

$$T^B(2^L-1)=(2^L-2)+2T^B(2^{L-1}-1) \tag{89}$$

which we can unwind:

$$\begin{aligned}
T^B(2^L-1) &= (2^L-2^1)+2\,T^B(2^{L-1}-1) && (90)\\
&= (2^L-2^1)+(2^L-2^2)+2^2\,T^B(2^{L-2}-1) && (91)\\
&= (2^L-2^1)+(2^L-2^2)+(2^L-2^3)+2^3\,T^B(2^{L-3}-1) && (92)\\
&= (2^L-2^1)+(2^L-2^2)+\cdots+(2^L-2^{L-1})+2^{L-1}\,T^B(1) && (93)\\
&= (L-1)2^L-(2^L-2) && (94)\\
&= (\lg(n+1)-1)(n+1)-(n-1) && (95)\\
&= (n+1)\lg(n+1)-2n && (96)\\
&= n\lg(n+1)+o(n\lg n) && (97)\\
&= \Theta(n\lg n) && (98)
\end{aligned}$$

For the worst case, we have

$$T^W(n)=(n-1)+T^W(1)+T^W(n-1) \qquad (99)$$

with solution $T^W(n)=\frac{1}{2}n(n-1)=\frac{1}{2}n^2+o(n^2)=\Theta(n^2)$

12.1 Other variations

Let us now look at the median-of-3 variation. The best- and worst case will be left as an exercise. The average case is however significantly more difficult, since the

$$\Pr[\,\text{rank(pivot)}=i\,]=\frac{\binom{i-1}{1}\binom{1}{1}\binom{n-i}{1}}{\binom{n}{3}},\quad \text{for all } i=2,\cdots,n-1$$

and this now depends on i and the probability density is no longer uniform. Similarly, the variation where the pivot is picked as the median of $2k+1$ elements. The best- and worst case is again an exercise, while the average incorporates

$$\Pr[\,\text{rank(pivot)}=i\,]=\frac{\binom{i-1}{k}\binom{1}{1}\binom{n-i}{k}}{\binom{n}{2k+1}},\quad \text{for all } i=k+1,\ \cdots\ n-k$$

and this now depends on both i and k and the shape of the probability density becomes more of a bell -curve (i.e. normally distributed).

13 Missing and Future sections

The current hand-out (Fall 2018) does not include some case studies from the combinatorics, that are not covered in algorithm courses. Also, sections are planned on probabilistic applications, stochastic analysis, additional algorithmic case studies and additional advanced techniques for average tie complexity studies, different boundaries, additional recurrence relations that are solved with generating functions.