Recurrence Relations for Computing Disciplines

The first few helpings for UMKC's CS291 and CS404 and a review for those in CS5592

Update Early Fall Semester 2016 - August 19, 2016 , rendered September 14, 2016

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

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 the main focus of this section. 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. In this hand-out, we attempt to first derive the explicit form of the solution, after which we find the asymptotic term. We try to find the term $g(n)$ such that $T(n) \sim g(n)$ and $g(n)$ is as simple as possible. 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 are largely missing from this hand-out. 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

$T(n) = T(n-1) + f(n)$
and

$T(n) = aT(n-1) + f(n)$
We start explaining recurrence relations of the these forms, where $f(n)$ is either linear, quadratic, logarithmic, or exponential.

There are several techniques to solve these 'One Term' recurrence relations. We will solve a variety of them.

2.1 $T(n) = T(n-1) + f(n)$

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

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

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 seen to be

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

Notice that the number of terms in this summation is n. Also, as $f(n)$ is generally an increasing function (or at least non-decreasing), the 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 $\sum_{i=1}^{n} f(i)$.

$T(n) = \begin{cases} t_0 & n = 0 \\ T(n-1) + f(n) & n > 1 \end{cases}$ 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 solution of the recurrence relations involving $T(n-1)$ and $T(0)$.

2.2 Examples and Practice Problems

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

$$\begin{cases} T(n) = 3 & n \leq 4 \\ T(n) = T(n-1) + 6 & 5 \leq n \end{cases}$$

Solution: $T(n) = 6n - 21 = \Theta(n)$

You can derive this answer by substitution:

$$T(n) = T(4) + 6 + 6 + 6 \cdots + 6 \quad = \quad T(4) + (n-4)6 \quad = \quad 6\,n - 21 \quad = \quad \Theta(n)$$

Note, there are as many 'sixes' in the expression as "$n - 4$".

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

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

You can derive this answer by back substitution again, but now:

$$\begin{aligned} T(n) &= T(4) + 5 + 6 + 7 + \cdots + n \\ &\quad \text{// now add and subtract the sum } (1+2+3+4) \\ &= T(4) + (1+2+3+4) + 5 + 6 + 7 + \cdots + n - (1+2+3+4) \\ &\quad \text{// the part } (1 + \cdots + n) \text{ simplifies} \\ &= T(4) + \frac{n(n+1)}{2} - (1+2+3+4) \\ &= 3 + \frac{n(n+1)}{2} - (10) \\ &= \frac{1}{2}\,n^2 + \frac{1}{2}\,n - 7 \\ &= \Theta(n^2) \end{aligned}$$

and finally, $\boxed{T(n) = \frac{1}{2}\,n^2 + \frac{1}{2}\,n - 7 = \Theta(n^2)}$

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

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

Solution: $T(n) = n^2 + n - 17 = \Theta(n^2)$

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

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

Solution: $T(n) = \frac{1}{3}\,n^3 + \frac{1}{2}\,n^2 + \frac{1}{6}\,n - 27 = \Theta(n^3)$

You can derive this answer again by back substitution, but now:

$$\begin{aligned} T(n) &= T(4) + 5^2 + 6^2 + 7^2 + \cdots + n^2 \\ &\quad \text{// now add and subtract the sum } (1^2 + 2^2 + 3^2 + 4^2) \\ &= T(4) + (1^2 + 2^2 + 3^2 + 4^2) + 5^2 + 6^2 + 7^2 + \cdots + n^2 - (1^2 + 2^2 + 3^2 + 4^2) \\ &\quad \text{// the part } (1^2 + \cdots + n^2) \text{ simplifies} \\ &= T(4) + \frac{2\,n^3 + 3\,n^2 + n}{6} - (30) \\ &= \frac{1}{3}\,n^3 + \frac{1}{2}\,n^2 + \frac{1}{6}\,n - 27 \\ &= \Theta(n^3) \end{aligned}$$

and finally, $\boxed{T(n) = \frac{1}{3}\,n^3 + \frac{1}{2}\,n^2 + \frac{1}{6}\,n - 27 = \Theta(n^3)}$

2.2.1 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]);
```

2.3 $T(n) = a\,T(n-1) + f(n),\ \ a > 1$

Next, we consider one-term recurrence relations 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, forward substitution can be used to get the explicit expression for $T(n)$

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

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 deceasing, 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, and $f(n)$. 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 $\sum_{i=1}^{n} a^{n-i} f(i)$.

$T(n) = \begin{cases} t_0 & n = 0 \\ a\,T(n-1) + f(n) & n > 1 \end{cases}$ with the inhomogeneous part $f(n)$ as specified	
Constant $f(n) = c_0$	$T(n) = t_0\ a^n + c_0 \left(\frac{a^n - 1}{a-1}\right) = \left(t_0 + \frac{a^n-1}{a-1}\right) a^n + o(a^n) = \Theta(\ a^n\ )$
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 solution of the recurrence relations involving $a\,T(n-1)$ and $T(0)$ for $a > 1$.

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

2.3.1 Examples and Practice Problems

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

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

You can derive this answer again by back substitution:

$$\begin{aligned}
T(n) &= 5\,T(n-1) + 6 \\
&= 5\ (5\,T(n-2) + 6\,) + 6\ = 5^2\,T(n-2) + (5+1)\ 6 \\
&= 5^2\ (5\,T(n-3) + 6\,) + (5+1)\ 6 = 5^3\,T(n-3) + \left(5^2 + 5 + 1\right)\ 6 \\
&= \ldots \\
&= 5^{n-4}\,T(4) + \left(5^{n-3} + 5^{n-4} + 5^{n-5} + \cdots 5^2 + 5^1 + 5^0\right)\ 6 \\
&= 5^{n-4}\,T(4) + \frac{5^{n-2}-1}{4}\,6 \\
&= 3\,5^{n-4} + \frac{3}{2}\,5^{n-2} - \frac{3}{2} \\
&= \frac{3}{625}\,5^n + \frac{3}{30}\,5^n - \frac{3}{2} \\
&= \frac{9}{1250}\,5^n - \frac{3}{2} \\
&= \Theta(5^n)
\end{aligned}$$

Check this with the MAPLE statement:

```
T1:=n -> rsolve({T(4) = 3, T(n) = 5T(n-1)+6}, T(n));
```

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

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

Solution: $T(n) = \dfrac{9}{16}\,2^n - n - 2 = \Theta(2^n)$

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

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

Solution: $T(n) = \dfrac{9}{16}\,2^n - n - 2 = \Theta(2^n)$

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:

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

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. $a_1^2 + 4\, a_2 > 0$:

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

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

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

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

$$\text{either} \quad T(n) = c_1(r_1)^n + c_2(r_2)^n \ ,$$

$$\text{where} \quad r_1 = \frac{a_1 + \sqrt{a_1^2 + 4a_2}}{2} \qquad r_2 = \frac{a_1 - \sqrt{a_1^2 + 4a_2}}{2}$$

$$c_1 = \frac{t_1 - t_0 r_2}{r_1 - r_2} \qquad c_2 = \frac{r_1 t_1 - t_1}{r_1 - r_2}$$

$$\text{or} \quad T(n) = (c_3 \sin(n\theta) + c_4 \cos(n\theta))^n \ r^n \ , c_3, c_4, r \text{ are all real}$$

$$\text{where} \quad r = \sqrt{a_1^2 + 2a_2} \text{ and } c_3, c_4, \theta \text{ can be determined as well.}$$

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:

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

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:

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

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:

$$\left(x^2 - a_1\ x - a_2\right) \times\ (x - s) = 0 \tag{1}$$

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 must be solved 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.	$f(n) = s^n$	$\left(x^2 - a_1\ x^1 - a_2\ \right)(x - s)$
2.	$f(n) = 2\ 3^n$	$\left(x^2 - a_1\ x^1 - a_2\ \right)(x - 3)$
3.	$f(n) = (1 + 2\ n + n^3)\ 3^n$	$\left(x^2 - a_1\ x^1 - a_2\ \right)(x - 3)^4$
4.	$f(n) = (1 + 2\ n + n^7)\ 3^n$	$\left(x^2 - a_1\ x^1 - a_2\ \right)(x - 3)^8$
5.	$f(n) = 1 + 2\ n + n^3$	$\left(x^2 - a_1\ x^1 - a_2\ \right)(x - 1)^4$
6.	$f(n) = (1 + 2\ n)\ 3^n \ + (2\ n + n^3)\ 4^n$	$\left(x^2 - a_1\ x^1 - a_2\ \right)(x - 3)^2\ (x - 4)^4$
7.	$f(n) = (1 + 2\ n)\ 3^n \ + (2\ n + n^3)$	$\left(x^2 - a_1\ x^1 - a_2\ \right)(x - 3)^2\ (x - 1)^4$

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	Asymptotics
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.	$-2^n + 3^n$	$(x-1)^3\,(x-2)\,(x-3)^2$	$-\frac{1}{4} + 4\,2^n + \frac{3}{4}\,(2n-5)3^n$	$\sim \frac{3}{4} n 3^n$

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

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

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:

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

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 :

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

Thus, in general, we can state:

Each distinct root r_i has a multiplicity m_i, and adds an additive term to the solution $T(n)$ of the form

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

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 $r_{\cdot}$ with multiplicity m, add a term $(c_{\cdot,1} + c_{\cdot,2}\ n + \cdots + c_{\cdot,m}\ n^m)(r_{\cdot})^n$ to the solution

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

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

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

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}:

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

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

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

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:

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

which can be factored in:

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

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:

	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.	$f(n) = s^n$	$\left(x^k - a_1\,x^{k-1} \cdots - a_k\right)(x-s)$
2.	$f(n) = 2\,3^n$	$\left(x^k - a_1\,x^{k-1} \cdots - a_k\right)(x-3)$
3.	$f(n) = (1 + 2\,n + n^3)\,3^n$	$\left(x^k - a_1\,x^{k-1} \cdots - a_k\right)(x-3)^4$
4.	$f(n) = (1 + 2\,n + n^7)\,3^n$	$\left(x^k - a_1\,x^{k-1} \cdots - a_k\right)(x-3)^8$
5.	$f(n) = (1 + 2\,n)\,3^n\ + (2\,n + n^3)\,4^n$	$\left(x^k - a_1\,x^{k-1} \cdots - a_k\right)(x-3)^2\ (x-4)^4$
6.	$f(n) = (1 + 2\,n)\,3^n\ + (2\,n + n^3)$	$\left(x^k - a_1\,x^{k-1} \cdots - a_k\right)(x-3)^2\ (x-1)^4$

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 $f(n)$.

$p_{d'}(s')\,s'^n$. 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 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. $\begin{cases} T(n) = n & n \leq 2 \\ T(n) = 2T(n-1) + 3\,T(n-2) + 4 & n > 2 \end{cases}$

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:

$$\boxed{\lg n\ \leq h(n) \leq\ 1.44\ \lg n} \tag{3}$$

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 $\Theta(n^\gamma)$:

$$\boxed{1.89\ \phi^h \leq n(h) \leq 2\ 2^h,\ \text{ where }\ \phi = 1.61\ \text{ is the golden ratio}} \tag{4}$$

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:

$$\boxed{\begin{cases} M(0) = 1 & h = 0 \\ M(h) = 1 + 2M(h-1) & h > 0 \end{cases}} \text{ and with solution } \boxed{M(h) = 2\,2^h - 1} \tag{5}$$

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	...	$2^h - 1$

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:

$$\boxed{\begin{cases} m(0) = 1 & h = 0 \\ m(h) = 1 + m(h-1) + m(h-2) & h > 0 \end{cases}} \tag{6}$$

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	133	...	$\approx 1.9\,(1.62)^h - 1$

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,

$$\begin{aligned} m(h)+1 &= \left(\frac{5+2\sqrt{5}}{5}\right)\left(\frac{1+\sqrt{5}}{2}\right)^h + \left(\frac{5-2\sqrt{5}}{5}\right)\left(\frac{1-\sqrt{5}}{2}\right)^h && (7)\\ &= \alpha\ \phi^h + \beta\ \phi_0^h && (8)\\ &\approx 1.8944\,(1.61803)^h + 0.1056\,(-0.61803)^h && (9) \end{aligned}$$

and

$$m(h) \approx 1.8944\,(1.61803)^h \qquad (10)$$

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:

$$\boxed{1.8944\times\ \phi^h \approx m(h) \le n(h) \le M(h) \le 2\times\ 2^h,} \qquad (11)$$

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

$$\boxed{h_m(n) = \lceil\ \log_2(n+1) - 1\ \rceil} \qquad (12)$$

Of course, we could eliminate the ceiling notation, and get

$$\boxed{\log_2(n+1) - 1 \ \leq h_m(n)} \tag{13}$$

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)

$$h_M(n) \approx \log_\phi\left(\frac{n}{\alpha}\right) = \log_\phi(n) - \log_\phi(\alpha) = \frac{\lg(n)}{\lg(\phi)} - \frac{\lg(\alpha)}{\lg(\phi)} \tag{14}$$

The result of the previous sections can now be combined.

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

$$\boxed{log_2(n+1) - 1 \leq h_m(n) \leq h(n) \leq h_M(n) \leq \frac{\lg(n)}{\lg(\phi)} \ < \ 1.44\ \lg n} \tag{15}$$

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:

$$\boxed{\lg n \ \leq h(n) \leq \ 1.44\ \lg n} \tag{16}$$

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 $T^W(n) = \Theta(\log n)$.

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:

$$\boxed{\lg(n+1) - 1 \ \leq h_m(n) \leq h(n) \leq h_M(n) \leq= \ 1.440420092\ \lg(n+2) - 1.327724} \tag{17}$$

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

$$\boxed{T^W(n) = \Theta(\log n)} \tag{18}$$

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 For now, the last thing to do is to present the information gained so far to help us in finding the potential heights of AVL trees with a given number of nodes n. In the above sections, we have determined both $m(h)$ and $M(h)$ as the lower and upper bound for the number of nodes that can be tolerated, or are needed, for an AVL tree with given height h. Indeed, the easiest way (i.e. the least confusing way) to quickly use iteration to generate the numbers for $m(h)$ and $M(h)$. Now you can turn this around: After this, a table with

We illustrate this in a table, where an 'x' is placed whenever at least one AVL exists with n nodes and height h. Each row represents and interval, and all elements in this interval have the same height-possibilities.

		AVL trees with n nodes and height h									
	$h=$	0	1	2	3	4	5	6	7	8	9
$n \in$	$[\,1\,]$	x									
$n \in$	$[\,2,\,3\,]$		x								
$n \in$	$[\,4,\,6\,]$			x							
$n \in$	$[\,7\,]$			x	x						
$n \in$	$[\,8,\,11\,]$				x						
$n \in$	$[\,12,\,15\,]$				x	x					
$n \in$	$[\,16,\,19\,]$					x					
$n \in$	$[\,20,\,31\,]$					x	x				
$n \in$	$[\,32\,]$						x				
$n \in$	$[\,33,\,53\,]$						x	x			
$n \in$	$[\,54,\,63\,]$						x	x	x		
$n \in$	$[\,64,\,87\,]$							x	x		
$n \in$	$[\,88,\,127\,]$							x	x	x	
$n \in$	$[\,128,\,132\,]$								x	x	
$n \in$	$[\,133,\,221\,]$								x	x	x
	$\vdots$									$\ddots$	$\ddots$
	$m(h)$	1	2	4	7	12	20	33	54	88	133
	$M(h)$	1	3	7	15	31	63	127	255	511	1023

Table 7: The possibilities of constructing an AVL BST with the given number of nodes n and given height h (or depth) is indicated by an 'x'. The number of possibilities is not shown.

This table only indicates the possibilities, and does not get into the number of different AVL trees that can be constructed with n nodes. Yet, certain lessons can be learned: For $n \in [54, 132]$ it shows that all AVL trees have heights between 5 and 8, so that the average certainly is also between 5 and 8. This can be juxtaposed with similar numbers for the (unrestricted) binary trees, for which the lower bound for the height is also 5. The upper bound is $n-1$ or 131, and the *average* height is $\Theta(\sqrt{n})$, or between 7 and 12. This simple demonstration shows that the performance of an AVL tree is superior to the performance on an 'averaged' binary search tree.

6 Divide & Conquer

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 reflects 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 $f(n)$ represents 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.

Substitution The subsequence that we consider are formed by $T(1)$, $T(2)$, $T(4)$, $T(8)$, $T(16), \cdots$, or $T(2^0)$, $T(2^1)$, $T(2^2)$, $T(2^3)$, $T(2^4), \cdots$, and the recurrence relation for $T(2^4)$ becomes $T(2^4) = T(2^3) + f(2^4)$. So rename $T(2^0)$ as $S(0)$, rename $T(2^1)$ as $S(1)$, $T(2^2)$ as $S(2)$, and so on. Such renaming results in the

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

which is a one-term relation we studied in the earlier subsection. Once the explicit closed form solution for $S(i)$ has been found, (in terms of i) and once an asymptotic $\Theta(\ \cdot\ )$ class for S has been found (again in terms of i), we also have a closed form solution for $T(2^i)$ in terms of i, or a closed form solution for $T(n)$ in terms of $i = \lg n$. So we now have an explicit closed form expression for n, when n is a power of 2.

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

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

The value of $T(n)$ is a^L times the initial value, t, plus the sum of products:
The terms in the sequence $\langle a^L, a^{L-1}, a^{L-2}, \cdots, a^0 \rangle$ are multiplied with the corresponding terms in the sequence $\langle f(b^1), f(b^2), f(b^3), \cdots, f(b^L) = f(n) \rangle$. The first sequence is usually decreasing, whereas the terms in the second are usually increasing These are multiplied term-by-term, and then added together: the sequence of products can be increasing, decreasing, or constant, and we will see examples for these. The explicit expression is known and reasonably simple in several cases, and the asymptotic growth class be determined. But in many more cases, the explicit form is either not known, or more complicated than needed. In all cases, we are interested in understanding the asymptotic behavior of $T(n)$, so the explicit expression for $T(n)$ is not always needed. We will consider special cases for which we can find the explicit solution; we consider the function $f(n)$ to be either constant, logarithmic, and monomial. Needless to say, there are many special cases to consider.

Before going into the details, we would like to find expressions for L, a^L and b^L in terms of n. Since it is defined as $L = \log_b\ n$, the term b^L is equal to n. Also, when $a = b$, $a = b^2$ and so on, then $a^L = b^L = n$, $a^L = \left(b^2\right)^L = n^2$ 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 $\alpha = \log_b\ a = \frac{\lg a}{\lg b}$, as well as $L = \log_b\ n = (\lg n)/(\lg b) = \beta \lg\ n$ and $a^L = n^{(\log_b\ a)} = n^\alpha$ where

$$\boxed{\alpha = \log_b\ a = \frac{\lg a}{\lg b}} \quad \text{and} \quad \boxed{\beta = \log_b\ 2 = \frac{1}{\lg b}}. \tag{19}$$

To get a feel for the value of α: If $a < b$, then $0 < \alpha < 1$, if $b < a < b^2$, then $1 < \alpha < 2$, if $b^2 < a < b^3$, then $2 < \alpha < 3$, and so on.

6.1 The Mini-Master Theorem for explicitly given functions

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

$$\boxed{\mathcal{F} = \left\{c, \quad c\ \log_b n, \quad c\ \lg n, \quad c\, n, \quad c\, n^k, \quad c\, n^k\ \log_b n\right\} \text{ and } k \in \mathbb{R}^+}$$

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.

The asymptotic form of the original recurrence relation where $f(n)$ has several terms from the allowable set, is dominated by the solution with the 'dominating' term. But each of the individual terms have a similar breakdown in the asymptotic behavior, and is summarized below. For each of the individual recurrence relations with the function $f(n)$

in our set, the asymptotic behavior is characterized according to $a < b^k$, or $a = b^k$, or $a > b^k$. The value for k is given for terms in the subset $\{c\,n, c\,n^k, c\,n^k\ \log_b n\}$, while we define $k = 0$ for terms in the subset $\{c, c\ \log_b n, c\ \lg n\}$.

MINI MASTER THEOREM FOR $f(n) \in \mathcal{F}$ Let a recurrence relation be given by

$$T(n) = \begin{cases} t & n = 1 \\ aT\left(\frac{n}{b}\right) + f(n) & n > 1 \end{cases}, \text{ and } f(n) \in \mathcal{F} \text{ such that the "highest power" of } f(n) \text{ is } k \tag{20}$$

Define α by $\alpha = \log_b\ a = \frac{\lg a}{\lg b}$ and β by $\beta = \log_b\ 2$ (so that $\log_b\ n = \beta \lg n$). The case that $b = 2$ is quite common, in which case $\alpha = \lg a$ and $\beta = 1$. The asymptotic expansion of the solution $T(n)$ is indicated in the table below, which shows the the coefficients of the dominant term in the asymptotic region:

Mini Master Theorem for $f(n) \in \mathcal{F}$ with dominant term $= c\,n^k$.			
IF	$1 \le a < b^k$	**THEN**	$T(n) \sim \dfrac{b^k}{b^k - a}\, c\, n^k$ (21a)
	$a = b^k$		$T(n) \sim c\ (\beta \lg\ n) n^k$ (21b)
	$a > b^k$		$T(n) \sim \left(t + \dfrac{b^k\, c}{a - b^k}\right) n^\alpha$ (21c)

Mini Master Theorem for $f(n) \in \mathcal{F}$ with dominant term $= c\,n^k\ \log_b n$			
IF	$1 \le a < b^k$	**THEN**	$T(n) \sim \dfrac{b^k\, c}{b^k - a}\, n^k\ \beta \lg\ n$ (22a)
	$a = b^k$		$T(n) \sim \frac{1}{2}\, c\, (\beta \lg\ n)^2\, n^k$ (22b)
	$a > b^k$		$T(n) \sim \left(t + \dfrac{a\, b^k\, c}{(a - b^k)^2}\right)\ n^\alpha$ (22c)

If only the asymptotic class is required, then both tables can be combined. Either way, you still have to separate the cases on the largest contribution to the sum, according to the relative sizes of a and b^k:

Mini Master Theorem for $f(n) \in \mathcal{F}$.			
IF	$1 \le a < b^k$	**THEN**	$T(n) = \Theta(\ f(n))$ (23a)
	$a = b^k$		$T(n) = \Theta\,(f(n) \lg\ n)$ (23b)
	$a > b^k$		$T(n) = \Theta(\ n^\alpha\ )$ (23c)

6.2 Examples

Use the Mini-Master Theorem to find the asymptotic expansion and asymptotic class of the following recurrence relations. That is, identify a, b, k, the applicable formula, $T(n) \sim \ldots$, $T(n) = \ldots + o(\cdot)$ and the $T(n) = \Theta(\ )$ class.

1.

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

Therefore, equations (29b) and (33b) apply: $\boxed{T(n) \sim \lg n}$, $\boxed{T(n) = \lg n + o(\lg n)}$ and $\boxed{T(n) = \Theta(\lg n)}$

2.

$$T(n) = \begin{cases} 3 & n \leq 1 \\ T(n/2) + n & 1 < n \end{cases} \qquad \text{Note } \begin{cases} a = 1 \\ b = 2 \ \text{ so that } a < b^k. \\ k = 1 \end{cases}$$

Therefore, equations (29a) and (33a) apply: $\boxed{T(n) \sim 2n}$, $\boxed{T(n) = 2\,n + o(n),}$ and $\boxed{T(n) = \Theta(n)}$

3.

$$T(n) = \begin{cases} 3 & n \leq 1 \\ T(n/2) + n^2 & 1 < n \end{cases} \qquad \text{Note } \begin{cases} a = 1 \\ b = 2 \ \text{ so that } a < b^k. \\ k = 2 \end{cases}$$

Therefore, equations (29a) and (33a) apply: $\boxed{T(n) \sim \frac{4}{3}\, n^2}$, $\boxed{T(n) = \frac{4}{3}\, n^2 + o(n^2)}$ and $\boxed{T(n) = \Theta(n^2)}$

4.

$$T(n) = \begin{cases} 3 & n \leq 1 \\ 2\,T(n/2) + 4 & 1 < n \end{cases} \qquad \text{Note } \begin{cases} a = 2 \\ b = 2 \ \text{ so that } a > b^k \text{ and we need } \alpha = \log_b\ a = 1. \\ k = 0 \end{cases}$$

Therefore, equations (29c) and (33c) apply: $\boxed{T(n) \sim 7\ n}$, $\boxed{T(n) = 7\ n + o(n)}$ and $\boxed{T(n) = \Theta(n)}$

5.

$$T(n) = \begin{cases} 3 & n \leq 1 \\ 2\,T(n/2) + n & 1 < n \end{cases} \qquad \text{Note } \begin{cases} a = 2 \\ b = 2 \ \text{ so that } a = b^k. \\ k = 1 \end{cases}$$

Therefore, equations (29b) and (33b) apply: $\boxed{T(n) \sim (3 + \lg n)n}$, $\boxed{T(n) = (3 + \lg n)n + o(n)}$ and $\boxed{T(n) = \Theta(n \lg n)}$

6.

$$T(n) = \begin{cases} 3 & n \leq 1 \\ 2\,T(n/2) + n^2 & 1 < n \end{cases} \qquad \text{Note } \begin{cases} a = 2 \\ b = 2 \ \text{ so that } a < b^k. \\ k = 2 \end{cases}$$

Therefore, equations (29a) and (33a) apply: $\boxed{T(n) \sim 2\ n^2}$

7.

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

Therefore, equations (30b) and (33b) apply: $\boxed{T(n) \sim \frac{1}{2}\ (\lg\ n)^2}$

8.

$$T(n) = \begin{cases} 3 & n \leq 1 \\ 2\,T(n/2) + \lg n & 1 < n \end{cases} \qquad \text{Note } \begin{cases} a = 2 \\ b = 2 \\ k = 0 \end{cases} \text{ so that } a > b^k \text{ and we need } \alpha = \log_b\, a = 1.$$

Therefore, equations (29c) and (33c) apply: $\boxed{T(n) \sim 4n}$

9.

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

$$\text{Note } \begin{cases} a = 2 \\ b = 2 \end{cases} \text{ and } \begin{cases} c = \frac{1}{2} \\ k = 1 \end{cases} \text{ thus } a = b^k \text{ and } \boxed{T(n) \sim (3 + \tfrac{1}{2} \lg n)n}$$

10.

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

$$\text{Note } \begin{cases} a = 2 \\ b = 3 \end{cases} \text{ and } \begin{cases} c = 2 \\ k = 0 \end{cases} \text{ thus } a > b^k \text{ so that } \alpha = 0.63 \;\&\; \boxed{T(n) \sim 5\,n^{\alpha}}$$

11.

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

$$\text{Note } \begin{cases} a = 2 \\ b = 3 \\ k = 1 \end{cases} \text{ and } \boxed{T(n) = \Theta(n^k) = \Theta(n)}$$

12.

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

$$\text{Note } \begin{cases} a = 2 \\ b = 3 \\ k = 2 \end{cases} \text{ and } \boxed{T(n) = \Theta(n^k) = \Theta(n^2)}$$

13.

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

14.

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

6.3 Proof of the Mini-Master Theorem.

First[1], 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. After that, we simply assume that $f(n)$ is only one term of a particular form, and we take every term separately.

6.3.1 Solution of sum is sum of solution

THEOREM The solution $T(n)$ of the recurrence relation

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

is the sum $T(n) = T_1(n) + T_2(n)$, where $T_1(n)$ and $T_2(n)$ are the solutions to the two individual recurrence relations

$$T_1(n) = \begin{cases} t_1 & n = 1 \\ aT_1\left(\frac{n}{b}\right) + f_1(n) & n > 1 \end{cases} \quad \text{and} \quad T_2(n) = \begin{cases} t - t_1 & n = 1 \\ aT_2\left(\frac{n}{b}\right) + f_2(n) & n > 1 \end{cases}$$

for any choice of t_1

The proof is based on unwinding the recursive definition,

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

So the solution of this recurrence relation is composed of summing the two solutions of two separate recurrence relations when the $f_(n)$ and $f_2(n)$ are taken individually, as long as the value of the initial condition is distributed as well.

It suffices therefore to solve the recurrence relation for each function $f(n)$ out of this set individually, as we will do below. 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.

Note that for all of these recurrence relations, 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, there are three cases:

1. If the last term, $f(n)$, dominates the first term, $a^L = n^\alpha$, in the asymptotic region, that is, if $n^\alpha = o(f)$, 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 last term, $f(n)$, is pretty much equal to the first term, that is, if $f(n) = \Theta(n^\alpha)$, 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 $\log_b(n)\ f(n) = \gamma \lg n\ f(n)$, where γ is the conversion factor between logarithms of different bases.

3. If the last term, $f(n)$, is dominated by the first term, $a^L = n^\alpha$, that is, if $f(n) = o(n^\alpha)$, then the largest contribution to $T(n)$ comes from the first few terms, $T(n) \approx a^L f(1) + a^{L-1} f(b^1) + a^{L-2} f(b^2) + a^{L-3} f(b^3) \cdots$.

[1] 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 $n \leq 4$.

6.3.2 Special case: constant function

The expression for $T(n)$ is:

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

which when $a = 1$ evaluates to

$$\begin{aligned} T(n) = T(b^L) &= t + c\ \log_b n = t + c\,\beta \lg n \\ &= c\,\beta \lg n + o(\lg n) \\ &= \Theta(\lg n) \end{aligned}$$

When $a \neq 1$, this becomes

$$\begin{aligned} T(n) = T(b^L) &= t\ n^\alpha + \frac{a^L - 1}{a - 1} c \\ &= \left(t + \tfrac{1}{a-1} \cdot \mathbf{c}\right) n^\alpha - \tfrac{1}{a-1}\ c \\ &= \left(t + \tfrac{1}{a-1} \cdot \mathbf{c}\right) n^\alpha - o(n^\alpha) \\ &\sim \left(t + \tfrac{1}{a-1} \cdot \mathbf{c}\right) n^\alpha \end{aligned}$$

6.3.3 Special case: logarithmic function and base is b, $f(n) = c\ \log_b n$

In this case, $f(b^i) = c\,i$ and the expression for $T(n)$ is:

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

which when $a = 1$ evaluates to

$$\begin{aligned} T(n) = T(b^L) &= t + \tfrac{1}{2}c\,(L^2 + L) = \tfrac{1}{2}c\,(\log_b\ n)^2 + \tfrac{1}{2}c\ \log_b\ n + t \\ &= \tfrac{1}{2}(\beta \lg\ n)^2 + \tfrac{1}{2}\beta \lg\ n + t \\ &= \tfrac{1}{2}c\,\beta^2\ (\lg\ n)^2 + o((\lg\ n)^2) \\ &= \Theta((\beta \lg\ n)^2) \end{aligned}$$

When $a \neq 1$, this becomes

$$\begin{aligned} T(n) = T(b^L) &= a^L t + a^{L-1}c\ + 2\,a^{L-2}c\ + 3\,a^{L-3}c\ \cdots + L\,a^0 c \\ &= a^L t + a^{L-1}c\left(1 + 2\,a^{-1} + 3\,a^{-2}\ \cdots + L\,a^{-(L-1)}\right) \\ &= a^L t + c\frac{a\,a^L - (a+1)L - a}{(a-1)^2} \\ &= \left(t + \frac{a\,c}{(a-1)^2}\right) a^L - c\frac{a+1}{(a-1)^2}L - \frac{ac}{(a-1)^2} \\ &= \left(t + \frac{a\,c}{(a-1)^2}\right) n^\alpha - c\frac{a+1}{(a-1)^2}\beta \lg\ n - \frac{ac}{(a-1)^2} \\ &= \left(t + \frac{a\,c}{(a-1)^2}\right) n^\alpha + o(n^\alpha) \\ &= \Theta(n^\alpha) \end{aligned}$$

Base is 2**,** $f(n) = c\ \lg n$

In this case, $f(b^i) = c\ \log_b 2\ i$, so that the previous case applies with a different constant.

6.3.4 Special case: linear function.

In this case, $f(n) = c\,n$ and $f(b^i) = c\,b^i$ and the expression for $T(n)$ is:

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

which when $a = b$ evaluates to

$$\begin{aligned} T(n) = T(b^L) &= b^L t + b^{L-1}\,c\,b^1\left(1 + (1)^1 + (1)^2\ \cdots + (1)^{L-1}\right) \\ &= (t + c\,L\,)\,b^L \\ &= (t + c\,\beta \lg\ n)\,n \\ &= c\,\beta\,n\ \lg\ n + o(n\ \lg\ n) \\ &= \Theta(n\ \lg\ n) \end{aligned}$$

When $a \neq b$, this becomes

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

When $a < b$, then also $\alpha < 1$ and

$$\begin{aligned} T(n) = T(b^L) &= \frac{b\,c}{b - a}\,n + o(n) \\ &= \Theta(n) \end{aligned}$$

Whereas when $a > b$, then also $\alpha > 1$ and

$$\begin{aligned} T(n) = T(b^L) &= \left(t + \frac{b\,c}{a - b}\right)n^\alpha + o(n^\alpha) \\ &= \Theta(n^\alpha) \end{aligned}$$

6.3.5 Special case: A monomial function.

In this case, $f(n) = c\,n^k$ so that $f(b^i) = c\,(b^k)^i$ and the expression for $T(n)$ is:

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

which when $a = b^k$ (and then also $\alpha = k$), evaluates to

$$\begin{aligned} T(n) = T(b^L) &= (t + c\,L\,)\,(b^k)^L \\ &= (t + c\,L\,)\,(b^L)^k \\ &= (t + c\,\beta \lg\ n)\,n^k \\ &= c\,\beta\,n^k\ \lg\ n + o(n^k\ \lg\ n) \\ &= \Theta(n^k\ \lg\ n) \end{aligned}$$

When $a \neq b$, this becomes

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

When $a < b^k$, then also $\alpha < k$ and

$$\begin{aligned} T(n) = T(b^L) &= \frac{b^k\, c}{b^k - a}\, n^k + o(n^k) \\ &= \Theta(n^k) \end{aligned}$$

Whereas when $a > b^k$, then also $\alpha > k$ and

$$\begin{aligned} T(n) = T(b^L) &= \left(t - \frac{b^k\, c}{b^k - a}\right) n^\alpha + o(n^\alpha) \\ &= \Theta(n^\alpha) \end{aligned}$$

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 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:

$$T(n) = \begin{cases} 3 & n \leq 1 \\ T(n/2) + \sqrt{n} & 1 < n \end{cases} \qquad \text{Note } \begin{cases} a = 1 \\ b = 2 \\ k = 1/2 \end{cases} \text{ so that } a < b^k.$$

Therefore, equations (29a) and (33a) apply: $\boxed{T(n) = \left(2 + \sqrt{2}\right)\ \sqrt{n} + o(\sqrt{n})}$ and $\boxed{T(n) = \Theta(\sqrt{n})}$

6.3.6 Special case: Product of logarithm and monomial.

In this case, $f(b^i) = c\, i\, (b^k)^i$ and the expression for $T(n)$ is:

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

which when $a = (b^k)$ evaluates to

$$\begin{aligned} T(n) = T(b^L) &= \left(t + c\, \tfrac{1}{2}\left(L + L^2\right)\right) (b^k)^L \\ &= \left(t + \tfrac{1}{2}\, c\, \left(\beta \lg\, n + (\beta \lg\, n)^2\right)\right) n^k \\ &= \left(t + \tfrac{1}{2}\, c\, \beta \lg\, n + \tfrac{1}{2}\, c\, \beta^2 (\lg\, n)^2\right) n^k \\ &= \tfrac{1}{2}\, c\, \beta^2 n^k\, (\lg\, n)^2 + o(n^k\, (\lg\, n)^2) \\ &= \Theta(n^k (\lg\, n)^2) \end{aligned}$$

When $a \neq b$, this becomes

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

When $a < b^k$, then also $\alpha < k$ and

$$T(n) = T(b^L) = \frac{b^k\, c\, \beta}{b^k - a}\, n^k\ \lg\ n\ + o(n^k\ \lg\ n\ ) = \Theta(n^k\ \lg\ n\ )$$

Whereas when $a > b^k$, then also $\alpha > k$ and

$$T(n) = T(b^L) = \left(t + \frac{a\, b^k\, c}{(b^k-a)^2} \right)\ n^\alpha\ + o(n^\alpha) = \Theta(n^\alpha)$$

6.4 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.

6.5 The full Master Theorem

Many additional results are available in this realm: How general can we make the recurrence relation, so that the conclusion in equations (33) holds? This is the master theorem, and will be discussed in the section for graduate students only. Wheres in the mini-master theorem we assumed explicit knowledge of the function $f(n)$, in the celebrated master theorem, only the asymptotic class of $f(n)$ is given, which makes a lot of difference.
The conclusion of the Mini-Master Theorem still stands for more general conditions, which will be made precise in graduate level courses. In particular, a, b, and k can be real numbers, rather than integers. The proofs must be adjusted however. Furthermore, the Mini-Master Theorem assumes that $f(n)$ is explicitly given, containing a finite number of polynomial terms, perhaps multiplied by a logarithm. The full Master Theorem, as it is traditionally presented, relaxes this requirement as well, and no longer requires that $f(n)$ is given in explicit form, but rather only in it's asymptotic form. The conclusion still stands, as long as there is an 'ϵ-band' that separates $f(n)$ from n^α. It requires that $f = O(n^{\alpha-\epsilon})$ or $f = \Omega(n^{\alpha+\epsilon})$ before the conclusion stands. If there is no such 'ϵ-band' separation, then there might be too much contribution from all the middle elements combined, even though one the ends ultimately dominates. The summation has no longer an exponential/geometric upper bound. Such functions are rare in complexity studies of algorithms, but such a function can be constructed: $f(n) = \frac{n}{\lg n}$ is an example of such a situation.

The full Master Theorem will be included here in the future, but not this semester.

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*.

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

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

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

$$T(n) = \begin{cases} t_1 & n = 1 \\ aT\left(\frac{n-1}{2}\right) + f(n) & n > 1 \end{cases}, \text{ and } f(n) \in \mathcal{F} \text{ such that the "highest power" of } f(n) \text{ is } k \tag{24}$$

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 $a = 2$:

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

Secondary Mini Master Theorem for $g(n) = \frac{n-1}{2}$ and $f(n) \in \mathcal{F}$ with dominant term $= c\, n^k\ \lg(n+1)$.				
IF	$a = 1,\ k = 0,$	$a = b^k$	**then** $T(n) \sim \frac{1}{2}\ c\ \lg^2(n+1)$	**and** $T(n) = \Theta\left(\lg^2 n\right)$
IF	$a = 1,\ k = 1,$	$a < b^k$	**then** $T(n) \sim 2\,c\,(n+1)\ \lg(n+1)$	**and** $T(n) = \Theta\,(n\ \lg n)$
IF	$a = 1,\ k = 2,$	$a < b^k$	**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	$a = 1,\ k \geq 2,$	$a < b^k$	**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	$a = 2,\ k = 0,$	$a > b^k$	**then** $T(n) \sim \frac{1}{2}\ (t_1 + 3c)(n+1)$	**and** $T(n) = \Theta\,(n)$
IF	$a = 2,\ k = 1,$	$a = b^k$	**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	$a = 2,\ k = 2,$	$a < b^k$	**then** $T(n) \sim 2\,c\,(n+1)^2\ \lg(n+1)$	**and** $T(n) = \Theta\left(n^2\ \lg n\right)$
IF	$a = 2,\ k \geq 2,$	$a < b^k$	**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 Mini Master Theorem for $g(n) = \frac{n-1}{2}$ and $f(n) \in \mathcal{F}$.			
IF $\begin{cases} 1 \le a < b^k \\ a = b^k \\ a > b^k \end{cases}$	**THEN**	$\begin{cases} T(n) = \Theta(\ f(n)) \\ T(n) = \Theta(f(n) \lg\ n) \\ T(n) = \Theta(\ n^\alpha\ ) \end{cases}$	(25a) (25b) (25c)

7.2 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 $1 \le g(n) < n$. 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 $T(n) \approx \sqrt{n}$.

Open question

Is there a generalization of the Master Theorem for more general $g(n)$-functions?

8 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.

8.1 Finding the best lower threshold

Consider for example the recurrence relation below.

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

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 $h(N) = T(N-1) + N^2$. In practice, (where the algorithm is actually implemented with an actual language on an actual platform), this best value for N must be found empirically: run many samples.

8.2 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

$$F_n = \frac{\phi_1^n - \phi_2^n}{\sqrt{5}}, \text{ where } \phi_1 = \tfrac{1}{2}\left(1 + \sqrt{5}\right), \phi_2 = \tfrac{1}{2}\left(1 - \sqrt{5}\right),$$

and this closed form involves radicals (i.e. $\sqrt{\cdot}$), and can not be represented in finite precision. Finding the exact value for ϕ is impossible, and finding an acceptable approximation is a numeric process, with its own complications and which this course does not address.

Representing the Fibonacci recurrence using matrix-vector notation, we get a recurrence relation between vectors,

$$\begin{bmatrix} F_{n+1}, & F_n \end{bmatrix} = \begin{bmatrix} F_n, & F_{n-1} \end{bmatrix} \times \begin{bmatrix} 1, & 1 \\ 1, & 0 \end{bmatrix} = \begin{bmatrix} F_1, & F_0 \end{bmatrix} \times \begin{bmatrix} 1, & 1 \\ 1, & 0 \end{bmatrix}^n . \tag{26}$$

So that the Fibonacci numbers are expressed as

$$F_{n+1} = \begin{bmatrix} F_1, & F_0 \end{bmatrix} \times \mathbf{M}^n \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} , \tag{27}$$

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. This approach will be elaborated upon in the future.

8.3 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: $T = O(T^+)$.

9 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.

9.1 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:

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

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:

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

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

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

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:

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

and

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

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

9.2 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,

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

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:

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

and

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

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

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

and finally,

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

9.3 A second order inhomogeneous equation

The inhomogeneous counter part follows immediately:

$$\begin{cases}T(0)=t_0,\ T(1)=t_1\\ T(n)=a_1T(n-1)+a_2T(n-2)+f(n) \qquad n>1\end{cases}$$

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

$$\begin{cases}T(0)x^0=t_0x^0\\ T(1)x^1=t_1x^1\\ T(2)x^2=a_1t_1x^2+a_2t_0x^2+f(2)x^2\\ T(3)x^3=a_1T(2)x^3+a_2t_1x^3+f(3)x^3\\ T(4)x^4=a_1T(3)x^4+a_2T(2)x^4+f(4)x^4\\ T(5)x^5=a_1T(4)x^5+a_2T(3)x^5+f(5)x^5\\ T(6)x^6=a_1T(5)x^6+a_2T(4)x^6+f(6)x^6\\ \quad\vdots\quad=\quad\vdots\quad+\quad\vdots\quad+\quad\vdots\end{cases}$$

Now define both the (ordinary) generating function of T, $G_T(x)=t_0x^0+t_1x^1+T(2)x^2+T(3)x^3+T(4)x^4+\cdots$, as well as the generating function for f: $G_f(x)=f_0x^0+f_1x^1+f(2)x^2+f(3)x^3+f(4)x^4+\cdots$. 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:

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

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.

9.4 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.

10 Case Study: Catalan Recursion and Combinatorics on binary trees.

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.

10.1 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:

$$C(n) = C(0)C(1)C(n-1) + C(1)C(1)C(n-2) + C(2)C(1)C(n-3) + \cdots + C(0)C(1)C(n-1). \quad (28)$$

or

$$\boxed{C(n) = \sum_{r=1}^{n} C(r-1)C(1)C(n-r) = \sum_{r=1}^{n} C(r-1)C(n-r)}. \quad (29)$$

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 $G_c^2(z)$,

$$G_c^2(z) = C(0)C(0) + (C(0)C(1) + C(1)C(0))\, z^1 + (C(0)C(2) + C(1)C(1) + C(2)C(0))\, z^2 + \quad (30)$$

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

$$G_c^2(z) = C(1) + (C(2))\, z^1 + (C(3))\ z^2 + \quad (31)$$

so that

$$G_c(z) = C(0) + z\ \big(C(1) + C(2)\, z^1 + C(3)\ z^2 + \cdots\big) = 1 + z\, G_c^2(z) \quad (32)$$

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:

$$G_c(z) = \frac{1 - \sqrt{1-4z}}{2z} = \frac{2}{1 + \sqrt{1-4z}} \quad (33)$$

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

$$\boxed{C(n) = \frac{1}{n+1}\begin{pmatrix} 2n \\ n \end{pmatrix}}. \tag{34}$$

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.

10.2 Advanced Topic: Beyond Catalan trees: The Ming An-Tu Numbers

Depth $d = n - 1$: If you have a binary tree with n nodes, then the depth of the tree is at most $d = n - 1$, which occurs when all internal nodes have exactly one child. Let us call $D(n, d)$ the number of binary trees with n nodes and depth d, then $D(n, n-1)$ is the number of 'skinny' trees. Counting the number of binary trees with this depth is another perfect example of both the addition and the product rule in combinatorial counting: Place the root first. There is only one way to place the root, but for all the following nodes, there are two choices:

$$D(n, n-1) = \overbrace{1}^{\text{root}} \times \overbrace{2 \times 2 \cdots \times 2}^{n-1 \text{ other nodes}} = 2^{n-1} \qquad n \geq 1$$

Taking the fraction of this number with the Catalan number $C(n)$ gives the probability that a random binary has depth $n - 1$:

$$\Pr[\text{ depth of random binary tree with } n \text{ nodes has depth } = n-1 \text{ }] = \frac{2^{n-1}}{C(n)} = \frac{(n+1)2^{n-1}}{\binom{2n}{n}} \approx \tfrac{1}{2} n^{3/2} \sqrt{\pi}\, (\tfrac{1}{2})^n,$$

where we used the asymptotic formula for the Catalan numbers. The numeric value of the probability is actually very small, but not zero. Some numbers are shown as part of the table below.

Depth $d = n - 2$: What about the number of binary trees with depth exactly $d = n - 2$? You can build these pretty much like the one above, except that we have to be careful with exact counting. First, place $n - 2$ nodes in a skinny binary tree with depth $n - 3$ as done above. This can be done in

$$\overbrace{1}^{\text{root}} \times \overbrace{2 \times 2 \cdots \times 2}^{n-3 \text{ other nodes}} = 2^{n-3}$$

We still have two nodes to add to this tree, and at least one must be used to increase the depth: We can add both nodes as children to the last node (the one that currently serves as leaf; there is only one way to do this), or we can add one of the remaining nodes (2 ways, either left or right), while we can add the last node to any of the nodes that are currently an internal node (there are $n - 3$ such nodes, a single choice at each node). Thus the number of binary tree that can be constructed form n nodes such that the tree has depth $d = n - 2$ is:

$$D(n, n-2) = \overbrace{2^{n-3}}^{n-2 \text{ nodes}} \times \left\{ \overbrace{1}^{\text{2 at leaf}} + \overbrace{2 \times (n-3) \times 1}^{\text{1 at leaf and 1 internal}} \right\} = (2n-5)\, 2^{n-3} \qquad n \geq 3$$

	Probability that a random binary tree has full depth	
n	$d = n-2$	$d = n-1$
1	-	1
2	-	1
3	0.200	0.800
4	0.429	0.571
5	0.476	0.381
6	0.424	0.242
7	0.336	0.149
8	0.246	0.090
9	0.171	0.053
10	0.114	0.030
11	0.074	0.017
12	0.046	0.010
13	0.029	0.006
27	1.1810^{-5}	9.6510^{-7}

Table 8: come up with a caption.

It would appear that you could have placed $n-1$ nodes in a skinny binary tree with depth $n-2$, after which you could add one last node, to any existing node, except the last one. So the answer would appear to be $(n-2)2^{n-2}$. But double counting could become a problem. Try this for $n = 3$ and you see what I mean.

You will note that the next few pages are not as well polished.

Such reasoning can be continued for depth $d = n-3$, although it becomes more complicated. It is perhaps best to refine the approach that gave us the Catalan numbers. It is still true that a mutual exclusive split exists for the left subtree and the right subtree, but we now have to be more careful with the depth. If the binary tree is to have a depth of d, then one (or both) subtrees must be of depth $d-1$. So there are three general categories, for each split $(i, n-i-i)$:

- left subtree has depth exactly $d-1$ and right subtree has depth strictly less than $d-1$
- left subtree has depth strictly less than $d-1$ and right subtree has depth exactly $d-1$
- both left subtree and right subtree have depths exactly $d-1$

The number of trees under the first two are identical, and putting these in an equation gives

$$D(n,d) = \sum_{i=0}^{n-1} \left\{ D(n-1-i, d-1) \times \left\{ D(i, d-1) + 2 * \sum_{j=0}^{d-2} D(i,j) \right\} \right\} \tag{35}$$

Although I do not believe that explicit expressions are available, except for $d = n-1$, $d = n-2$ and $d = n-3$, a few lines of programming allows you to generate a numeric table. David Callen mentions in `https://oeis.org/A073345` that 'This table was known to the Chinese mathematician Ming An-Tu, who gave the [...] recurrence in the 1730s'. Unfortunately, he does not give a citation. The same web-page seems to indicate that diagonals in this table follow a pattern of " Polynomial in n of degree k times an exponential", where k indicates the distance to the main diagonal, i.e. $k = n-d$. Note also that

$$C(n) = D(n,0) + D(n,1) + D(n,2) + \cdots D(n,n-2) + D(n,n-1) \tag{36}$$

as it should.

	Catalan	Binary trees with n nodes and depth d							
n	$C(n)$	$d=0$	1	2	3	4	5	6	7
0	1	1							
1	1	1							
2	2		$\boxed{2}$						
3	5		1	$\boxed{4}$					
4	14			6	$\boxed{8}$				
5	42			6	$\boxed{20}$	16			
6	132			4	40	$\boxed{56}$	32		
7	429			1	68	$\boxed{152}$	144	64	
8	1430				94	376	$\boxed{480}$	352	128
9	4862				114	844	$\boxed{1440}$	1376	832
10	16796				116	1744	4056	$\boxed{4736}$	3712
11	58786				94	3340	10856	$\boxed{15248}$	14272
12	208012				60	5976	27672	47104	$\boxed{50784}$
13	742900				28	10040	67616	140640	172640
14	2674440				8	15856	159184	407360	568832
15	9694845				1	23460	362280	1148560	1828832
16	35357670					32398	798616	3162016	5757088
⋮	⋮						⋱	⋱	⋱

Table 9: Catalan numbers $C(n)$ and (Ming An-Tu Tree-Depth numbers) $D(n,k)$ for selected n and k. The largest element in each row is boxed, as in $\boxed{56}$.

If you want to implement this formula, please be careful. First, it is a recursively equation, which you should *n*ot implement recursively, unless you use the 'option remember' or the 'memoize' option, if available. Instead, write an iterative program (yes, it is dynamic programming) and manage intermediate results in tables. A straightforward implementation will have time complexity of $\Theta(n^4)$, which can be reduced to $\Theta(n^3)$ after yet another table is introduced. Next, you will have to safe guard against numerical overflow (and perhaps underflow), as the numbers tend to get huge.

As time permits, (in other words, at a later time), I will present the table in different forms and add the empirical analysis of the average depth of a tree with given number n of nodes, as well as the average number of nodes of a random number of nodes in a tree with a given depth d. In the table, these are the row- and column averages. Finally, Flajolet and his collaborators studied properties of binary trees, and one of papers claims that the average depth has an asymptotic value of

$$E[D(n,d) \mid n] = 2\sqrt{\pi\, n} \qquad \text{for } n \to \infty$$

This answer is very plausible (the Catalan number has a square-root hidden inside), but their definitions are slightly different and I haven't had time yet to double check and validate. The curve for the expectations appear to follow a square root, and for $n = 80$, my program gives 25.35, whereas $2\sqrt{\pi\, n} = 31.707$. I will keep you posted.

Morale The morale of this story is that basic principles will go pretty far, even tough the resulting expressions may not simplify. Actually, most questions and problems you will face in practice are like this, so don't be discouraged. Serious questions may not have simple answers.

	Binary trees with n nodes and depth d								Catalan
n	$d=0$	1	2	3	4	5	6	7	$C(n)$
0	1								
1	1								1
2		$\boxed{2}$							2
3		1	$\boxed{4}$						5
4			6	$\boxed{8}$					14
5			6	$\boxed{20}$	16				42
6			4	40	$\boxed{56}$	32			132
7			1	68	$\boxed{152}$	144	64		429
8				94	376	$\boxed{480}$	352	128	1430
9				114	844	$\boxed{1440}$	1376	832	4862
10				116	1744	4056	$\boxed{4736}$	3712	16796
11				94	3340	10856	$\boxed{15248}$	14272	58786
12				60	5976	27672	47104	$\boxed{50784}$	208012
13				28	10040	67616	140640	172640	742900
14				8	15856	159184	407360	568832	2674440
15				1	23460	362280	1148560	1828832	9694845
16					32398	798616	3162016	5757088	35357670
⋮						⋱	⋱	⋱	⋮

Table 10: Catalan numbers $C(n)$ and (Ming An-Tu Tree-Depth numbers) $D(n,k)$ for selected n and k. The largest element in each row is boxed, as in $\boxed{56}$.

	Catalan	Probabilistic Anatomy of Binary trees with n nodes and depth d							
n	$C(n)$	$E[d]$	$\sigma[d]$	$cv[d]$	Mode	50-tile	90-tile	99-tile	99.9-tile
0	1	0							
1	1	0							
2	2	1.00							
3	5	1.80	0.40	0.22222					
4	14	2.57	0.49	0.19245					
5	42	3.23	0.68	0.21107					
6	132	3.87	0.80	0.20817					
7	429	4.47	0.93	0.20965					
8	1430	5.03	1.06	0.21122					
9	4862	5.56	1.18	0.21210					
10	16796	6.07	1.29	0.21263					
11	58786	6.59	1.39	0.21325					
12	208012	7.03	1.50	0.21383					
13	742900	7.48	1.60	0.21430					
14	2674440	7.91	1.70	0.21470					
15	9694845	8.33	1.79	0.21506					
20		10.29	2.23	0.21641					
25		12.02	2.61	0.21730					
30		13.61	2.97	0.21791					
35		15.08	3.29	0.21837					
40		16.45	3.60	0.21871					
45		17.75	3.89	0.21899					
50		18.97	4.15	0.21921					
55		20.14	4.42	0.21938					
60		21.26	4.66	0.21953					
65		22.34	4.91	0.21966					
70		23.38	5.14	0.21976					
75		24.38	5.36	0.21985					
80		25.35	5.57	0.21993					
85		26.27	5.79	0.21999					
90		27.21	5.99	0.22005					
95		28.11	6.18	0.22010					
100		28.97	6.37	0.22015					
125		33.04	7.28	0.22030					
150		36.72	8.09	0.22038					
175		40.12	8.84	0.22043					
200		43.29	9.54	0.22045					
⋮	⋮						⋱	⋱	⋱

Table 11: Catalan numbers $C(n)$ and (Ming An-Tu Tree-Depth numbers) $D(n,k)$ for selected n and k. The largest element in each row is boxed, as in $\boxed{56}$.

11 Case Study: Creating an Initial HEAP

There are two methods to construct an initial HEAP: a greedy method (one-by-one), and a divide-and-conquer method. Most textbooks analyze them by the aha-method: building tables and taking summations. However, both van be accomplished using recurrence relations:

11.1 greedy method

Maximal number of comparisons when initially constructing a MIN-heap:

$$T(n) = \begin{cases} 0 & n = 1 \\ T(n-1) + \lfloor \lg n \rfloor & n > 1 \end{cases} \tag{37}$$

11.2 divide and conquer

Maximal number of comparisons when initially constructing a MIN-heap:

$$T(n) = \begin{cases} 0 & n = 1 \\ 2T(\frac{n-1}{2}) + 2 \ (\lg(n+1) - 1) & n > 1 \end{cases} \tag{38}$$

11.3 An analysis without recursion:

Worst case time complexity for insertion **‘ one-by-one’** into a (binary) heap, assume $n = 2^L - 1$:

Consider a node at level	Worst case cost for inserting each node at this level	The total number of nodes at this level	total cost of this level
1	0	1	$0 \cdot 2^0$
2	1	2	$1 \cdot 2^1$
3	2	4	$2 \cdot 2^2$
4	3	8	$3 \cdot 2^3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
i	$i-1$	2^{i-1}	$(i-1) \cdot 2^{i-1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
L	$L-1$	2^{L-1}	$(L-1) \cdot 2^{L-1}$

So that in total

$$T^W(n) = \sum_{i=1}^{i=L} \left((i-1) \cdot 2^{i-1} \right) = (L-2)2^L + 2.$$

Remembering now that $n = 2^L - 1$, we have that $2^L = n + 1$, $L = \log_2(n+1)$ and $L - 2 = \log(\frac{n+1}{4})$, and thus

$$T^w(n) = (n+1)\log(\frac{n+1}{4}) + 2 = \boxed{\Theta(n \log n)}$$

Worst case time complexity for **‘Heapify’** to construct a new initial (binary) heap. We will develop this using H and h for heights, but still assume a full binary tree: $n = 2^H - 1$

Consider a node at height	Worst case cost to combine one new root with two existing heaps, each of height $j-1$, to form a new heap of height j	The total number of nodes at this height	total cost of this height	simplified total cost for this height. Let $a = \frac{1}{2}$
$H-1$	$2\times\ (H-1)$	1	$(H-1)\cdot 2^1$	$(H-1)\times\ a^{H-1}\times\ 2^H$
$H-2$	$2\times\ (H-2)$	2	$(H-2)\cdot 2^2$	$(H-2)\times\ a^{H-2}\times\ 2^H$
$H-3$	$2\times\ (H-3)$	4	$(H-3)\cdot 2^3$	$(H-3)\times\ a^{H-3}\times\ 2^H$
..	..	..	..	
j	$2\times\ j$	2^{H-j-1}	$(j)\cdot 2^{H-j}$	$j\times\ a^j\times\ 2^H$
..	..	..	..	
3	$2\times\ 3$	2^{H-4}	$(3)\cdot 2^{H-3}$	$3\times\ a^3\times\ 2^H$
2	$2\times\ 2$	2^{H-3}	$(2)\cdot 2^{H-2}$	$2\times\ a^2\times\ 2^H$
1	$2\times\ 1$	2^{H-2}	$(1)\cdot 2^{H-1}$	$1\times\ a^1\times\ 2^H$
0	$2\times\ 0$	2^{H-1}	$(0)\cdot 2^H$	$0\times\ a^0\times\ 2^H$

Factor out the term 2^H to get $T^W(n) = 2^H \sum_{j=0}^{H-1} \left(ja^j\right)$. This summation can be performed exactly, but it is easier to notice that the finite summation is smaller than the infinite summation, which converges to the value 2. Thus:

$$T^W(n) = 2^H \sum_{j=0}^{H-1} \left(ja^j\right) < 2^H \sum_{j=0}^{\infty} \left(ja^j\right) = 2^H \cdot 2 = (n+1)\cdot 2 = \boxed{\Theta(n)}$$

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 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 rank(pivot) $= i$, we have

$$T^A(n \mid \text{ rank(pivot)} = i\ ) = (n-1) + T^A(i-1) + T^A(1) + T^A(n-i) \tag{39}$$

Realizing that $T^A(1) = 0$ and by unconditioning[2], we get

$$\begin{aligned}
T^A(n) &= \sum_{i=1}^{n} \Pr[\text{ rank(pivot)} = i\ ]\ \underbrace{T^A(n \mid \text{ rank(pivot)} = i\ )} && (40)\\
&= \sum_{i=1}^{n} \left[\frac{1}{n} \quad \times \quad \left\{ \overbrace{(n-1) + T^A(i-1) + T^A(n-i)} \right\} \right] && (41)\\
&= (n-1) + \frac{1}{n} \sum_{i=1}^{n} \left\{ T^A(i-1) + T^A(n-i) \right\} && (42)\\
&= \frac{1}{n} \left\{ n(n-1) + 2 \left\{ T^A(0) + T^A(1) + \cdots + T^A(n-2) + T^A(n-1) \right\} \right\} && (43)
\end{aligned}$$

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

$$\begin{aligned}
n\, T^A(n) &= n(n-1) \qquad + 2 \left\{ T^A(0) + T^A(1) + \cdots + T^A(n-2) + T^A(n-1) \right\} && (44)\\
(n-1)\, T^A(n-1) &= (n-1)(n-2) + 2 \left\{ T^A(0) + T^A(1) + \cdots + T^A(n-2) \qquad \right\} && (45)
\end{aligned}$$

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

$$n\, T^A(n) - (n-1)\, T^A(n-1) = n(n-1) - (n-1)(n-2) + 2\, T^A(n-1) \tag{46}$$

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

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

$$n\,T^A(n) = 2\,(n-1) + (n+1)\,T^A(n-1) \tag{47}$$

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

$$\frac{T^A(n)}{n+1} = \frac{T^A(n-1)}{n} + 2\,\frac{n-1}{n(n+1)} = \frac{T^A(n-1)}{n} + 2\,\left\{\frac{2}{n+1} - \frac{1}{n}\right\} \tag{48}$$

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,

$$\begin{aligned}
\frac{T^A(n)}{n+1} &= 2\,\left\{\left(\frac{2}{n+1} - \frac{1}{n}\right) + \left(\frac{2}{n} - \frac{1}{n-1}\right) + \left(\frac{2}{n-1} - \frac{1}{n-2}\right) + \cdots + \left(\frac{2}{3} - \frac{1}{2}\right)\right\} && (49)\\
&= 2\,\left\{\frac{2}{n+1} \quad + \quad \frac{1}{n} \quad + \quad \frac{1}{n-1} \quad + \cdots + \quad \frac{1}{3} \quad - \frac{1}{2}\right\} && (50)\\
&= 2\,\left\{\frac{2}{n+1} + \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{3} + \frac{1}{2} + 1 \quad - \frac{1}{2} - 1 - \frac{1}{2}\right\} && (51)\\
&= 2\,\left\{\frac{2}{n+1} + H_n - 2\right\} && (52)
\end{aligned}$$

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

$$\boxed{T^A(n) = 2(n+1)H_n - 4n} \tag{53}$$

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

$$T^A(n) = 2(n+1)\ln n + (2\gamma - 4)n + (2\gamma + 1) + o(1) \tag{54}$$

or, less precise but more indicative of asymptotic behavior:

$$\begin{aligned}
T^A(n) &= 2\,n\ln n + o(n\ln n) = \Theta(\,n\ln n) && (55)\\
&= 1.386\,n\lg n + o(n\lg n) = \Theta(\,n\lg n) && (56)
\end{aligned}$$

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,

$$T^B(n) = (n-1) + T^B(\tfrac{n-1}{2}) + T^B(1) + T^B(\tfrac{n-1}{2}) \tag{57}$$

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

$$T^B(2^L - 1) = (2^L - 2) + 2T^B(2^{L-1} - 1) \tag{58}$$

which we can unwind:

$$\begin{aligned}
T^B(2^L-1) &= (2^L - 2^1) + 2\,T^B(2^{L-1}-1) && (59)\\
&= (2^L - 2^1) + (2^L - 2^2) + 2^2\,T^B(2^{L-2}-1) && (60)\\
&= (2^L - 2^1) + (2^L - 2^2) + (2^L - 2^3) + 2^3\,T^B(2^{L-3}-1) && (61)\\
&= (2^L - 2^1) + (2^L - 2^2) + \cdots + (2^L - 2^{L-1}) + 2^{L-1}\,T^B(1) && (62)\\
&= (L-1)2^L - (2^L - 2) && (63)\\
&= (\lg(n+1) - 1)(n+1) - (n-1) && (64)\\
&= (n+1)\lg(n+1) - 2n && (65)\\
&= n\lg(n+1) + o(n\lg n) && (66)\\
&= \Theta(n\lg n) && (67)
\end{aligned}$$

For the worst case, we have

$$T^W(n) = (n-1) + T^W(1) + T^W(n-1) \tag{68}$$

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

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

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

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

13 Case Study in Combinatorics: Drawing consecutive numbers in a Lottery

Let us consider a lottery game, where you typically draw 6 integers from 1 through 44 (without replacing them). What is the probability that there is no pair of consecutive integers if you draw 2 integers (from 1 through 44)?

Answer: There are several ways in which you could make a tree-diagram (that is, by conditioning). I will explain the one that stays closest to the experiment, and draw the first number. Now **condition on the value of the first draw**:

1. If the first number you draw is the '1'. Since this is an endpoint it has only one single neighbor and you need to remove both 1 and 2. There are 42 elements left for the second number you draw, and $R_2(44|[1]) = 42$.
2. If this first number you draw is either the 2 or the 3 or $\cdots$, or the 43, i.e. one of the 42 mid-points, then you also remove its two neighbors, and have 41 elements left for the second draw, thus $R_2(44|[\text{midpoint}]) = 41$.
3. If the first number you draw is 44, the other end-point, then you must remove both 44 and the single neighbor 43. There are 42 elements left for the second number you draw, $R_2(44|[44]) = 42$

We have used notation $R_2(44|A)$ to mean: The number of *Ordered Selection of 2 out of 33, given that A was drawn at the first selection*. These three options need to be combined to get the number of different ordered ways to draw (select) 2 'non-neighbors' from 44 numbers:

$$R_2(44) = 1 \times 42 + 42 \times 41 + 1 \times 42 = 43 \times 42$$

In order to get the number of *unordered* selection of 2 out of 44, which we will call $S_2(44)$, we need to divide by 2 because the order in which you drew the numbers is not important:

$$S_2(44) = \frac{42 + 42\times\ 41 + 42}{2} = \frac{43\times\ 42}{2}$$

Finally, to get the probability that there is no pair of consecutive integers if you draw 2 integers from 1 through 44, you need to divide $S_2(44)$ by the overall probability of getting two numbers from 44:

$$\Pr[\text{ 2 non-consecutive from 44 }] = \frac{S_2(44)}{\binom{44}{2}} = \frac{\frac{43\times\ 42}{2}}{\frac{44\times\ 43}{2}} = \frac{42}{44}$$

If you want to generalize this to taking two (2) non-neighboring elements from n integers, the reasoning is almost verbatim. There are 2 endpoints (which we group together), and $n-2$ midpoints:

$$\boxed{S_2(n) = \frac{(2)(n-2) + (n-2)(n-3)}{2} = \frac{(n-1)(n-2)}{2} = \binom{n-1}{2}}$$

and to get the probability, you again divide this by the overall probability of getting two numbers from n:

$$\boxed{\Pr[\text{ 2 non-consecutive from 44 }] = \frac{\binom{n-1}{2}}{\binom{n}{2}} = \frac{n-2}{n}}$$

What if you draw 3 integers (from 1 through 7)?

For this question, we also can make two tree different diagrams. The first we present is similar to the tree diagram we did for the "2 from 44" situation, by **conditioning on the value of first draw being an endpoint or a midpoint**, and subsequently by conditioning on the value of the second draw, as follows:

1. If this first number is the end-point 1, then you also remove the single neighbor 2 at the first choice, and you have 5 elements remaining (i.e. 3..7) to chose 2 elements from. We will expand this in a sub-tree by **conditioning on the second selection**:

   (a) If the second draw is 3, then remove 4 as well, and there are 3 elements left to draw the last element (namely $5, 6, 7$). So: $R_3(7|\ [1,3]) = \boxed{3}$

   (b) If the second draw is 4, then remove 3 and 5 as well, and there are 2 elements left to draw from (namely $6, 7$). So $R_3(7|\ [1,4]) = \boxed{2}$

   (c) If the second draw is 5, then remove 4 and 6 as well, and there are 2 elements left to select from (namely $3, 7$). So $R_3(7|\ [1,5]) = \boxed{2}$

   (d) If the second draw is 6, then remove 5 and 7 as well, and there are 2 elements left to select from (namely $3, 4$). So $R_3(7|\ [1,6]) = \boxed{2}$

   (e) If the second draw is 7, then remove 6 as well, and there are 3 elements left to select from (namely $3, 4, 5$). So $R_3(7|\ [1,7]) = \boxed{3}$

   This completes the sub-tree for all possibilities after the first draw was taken, and the value of the first draw was 1. The boxed numbers can be summed up to arrive at $R_3(7|\ [1]) = \boxed{12}$.

2. If this first number you draw is the mid-point 2, then you also remove the two neighbors 1 and 3, and you have 4 elements 4..7 to chose the last 2 elements from. We will expand this in a sub-tree by **conditioning on the second selection**:

   (a) If the second draw is 4, then remove 5 as well, and there are 2 elements left to select from (namely $6, 7$). So $R_3(7|\ [2,5]) = \boxed{2}$

   (b) If the second draw is 5, then remove 4 and 6 as well, and there is only a single element left to select the last element from (namely 7). So $R_3(7|\ [2,5]) = \boxed{1}$

   (c) If the second draw is 6, then remove 5 and 7 as well, and there is only a single element left to select the last element from (namely 4). So $R_3(7|\ [2,6]) = \boxed{1}$

   (d) If the second draw is 7, then remove 6 as well, and there are 2 elements left to select from (namely $4, 5$). So $R_3(7|\ [2,7]) = \boxed{2}$

   This completes the sub-tree for all possibilities after the first draw was taken, and the value of the first draw was 2. The boxed numbers can be summed up to arrive at $R_3(7|\ [2]) = \boxed{6}$.

3. If this first number is the mid-point 3, then you also remove the two neighbors 2 and 4, and you have 4 elements left to chose the last 2 elements from, namely 1 and $5, 6, 7$. But now, these numbers are not contiguous. We will expand this in a sub-tree by **conditioning on the second selection**:

   (a) If the second draw is 1, then there is no other elements to be removed. There are 3 elements left to select from (namely $5, 6, 7$). So $R_3(7|\ [3,1]) = \boxed{3}$

   (b) If the second draw is 5, then remove 6 as well, and there are 2 elements left to select from (namely $1, 7$). So $S_3(7|\ [3,5]) = \boxed{2}$

(c) If the second draw is 6, then remove 5 and 7 as well, and there is only 1 elements left to select from (namely 1). So $R_3(7|\,[3,6]) = \boxed{1}$

(d) If the second draw is 7, then remove 6 as well, and there are 2 elements left to select from (namely $1, 5$). So $R_3(7|\,[3,7]) = \boxed{2}$

This completes the sub-tree for all possibilities after the first draw was taken, and the value of the first draw was 3. The boxed numbers can be summed up to arrive at $R_3(7|\,[3]) = \boxed{8}$.

4. If this first number is the mid-point 4, then you also remove the two neighbors 3 and 5, and you have 4 elements left ($1, 2$ and $6, 7$) to chose the last 2 elements from. We will expand this in a sub-tree by **conditioning on the second selection**

   (a) If the second draw is 1, then remove 2 as well, and there are 2 elements left to select from (namely $6, 7$). So $R_3(7|\,[4,1]) = \boxed{2}$

   (b) If the second draw is 2, then remove 1 as well, and there are 2 elements left to select from (namely $6, 7$). So $R_3(7|\,[4,2]) = \boxed{2}$

   (c) If the second draw is 6, then remove 7 as well, and there are 2 elements left to select from (namely $1, 2$). So $R_3(7|\,[4,6]) = \boxed{2}$

   (d) If the second draw is 7, then remove 6 as well, and there are 2 elements left to select from (namely $1, 2$). So $R_3(7|\,[4,7]) = \boxed{2}$

   This completes the sub-tree for all possibilities after the first draw was taken, and the value of the first draw was 4. The boxed numbers can be summed up to arrive at $R_3(7|\,[4]) = \boxed{8}$.

5. If this first number is the mid-point 5, then you also remove the two neighbors 4 and 6, and you have 4 elements $1, 2, 3$ and 7 to chose the last 2 elements from. This is a mirror situation from midpoint 3: a total of 8 possibilities, and rather than showing the subtree, we can conclude that $R_3(7|\,[5]) = \boxed{8}$.

6. If this first number is the mid-point 6, then you also remove the two neighbors 5 and 7, and you have 4 elements $1, 2, 3, 4$ This is a mirror situation from midpoint 2: a total of 6 possibilities and rather than showing the subtree, we can conclude that $R_3(7|\,[6]) = \boxed{6}$.

7. If this first number is the end-point 7, then you also remove the single neighbor 6, and you have 5 elements $1..5$ to chose 2 elements from. This is the mirror situation and there are 12 ways to select two from five that are not neighbors and rather than showing the subtree, we can conclude that $R_3(7|\,[7]) = \boxed{12}$.

Adding all the branches together, you arrive at a total of $R_3(7) = 60$ possibilities, which needs to be divided by $3! = 6$ to arrive at $\boxed{S_3(7) = 10}$, and finally the probability that there is no pair of consecutive integers if you draw 3 integers from 1 through 7, you need to divide $S_3(7)$ by the overall probability of getting three numbers from 7:

$$\boxed{\Pr[\text{ 3 non-consecutive from 7 }] = \frac{S_3(7)}{\binom{7}{3}} = \frac{10}{35}}$$

As you can see, by conditioning on the value of the first draw will lead to a large counting tree, because all possible orders are encountered and accounted for. Below we will investigate a counting (conditioning) technique that generates only the 10 cases that are asked for.

The tree diagram presented above is very hard to generalize to any number of original numbers. We now present a tree diagram for which this is possible. We still **conditioning on the value of first draw**, but now recognize a pattern that repeats:

1. If this first number drawn is the end-point 1, then you also remove the single neighbor 2, and you have 5 elements 3..7 to chose 2 elements from. Note that these 5 elements are all different from the removed elements, so that we can just count the number of ways to do this: $R_2(5)$.

2. If this first number you draw is the mid-point 2, then you also remove the two neighbors 1 and 3, and you have 4 elements 4..7 to chose the last 2 elements from. The number of ways to do this is $R_2(4)$.

3. If this first number is the mid-point 3, then you also remove the two neighbors 2 and 4, and you have 4 elements left to chose the last 2 elements from, namely 1 and $5, 6, 7$. These numbers are not contiguous, but they are still distinct. So we take either 0 from one half, and 2 from the other, or we take one from each half, or we take 2 from the first half and none from the other. The number of ways to do this is $R_2(1).R_0(3) + R_1(1).R_1(3) + R_0(1).R_2(3)$.

4. If this first number is the mid-point 4, then you also remove the two neighbors 3 and 5, and you have 4 elements left ($1, 2$ and $6, 7$) to chose the last 2 elements from. These numbers are not contiguous, but they are still distinct. So we take either 0 from one half, and 2 from the other, or we take one from each half, or we take 2 from the first half and none from the other. The number of ways to do this is $R_2(2).R_0(2) + R_1(2).R_1(2) + R_0(2).R_2(2)$.

5. If this first number is the mid-point 5, then you also remove the two neighbors 4 and 6, and you have 4 elements $1, 2, 3$ and 7 to chose the last 2 elements from. This is a mirror situation from midpoint 3 and we can take either 0 from one half, and 2 from the other, or we take one from each half, or we take 2 from the first half and none from the other. The number of ways to do this is $R_2(3).R_0(1) + R_1(3).R_1(1) + R_0(3).R_2(1)$.

6. If this first number is the mid-point 6, then you also remove the two neighbors 5 and 7, and you have 4 elements $1, 2, 3, 4$ This is a mirror situation from midpoint 2 and the number of ways to do this is $R_2(4)$.

7. If this first number is the end-point 7, then you also remove the single neighbor 6, and you have 5 elements 1..5 to chose 2 elements from. This is the mirror situation of element 1 and and the number of ways to do this is $R_2(5)$.

These branches need to be added together to arrive at

$$\begin{aligned} R_3(7) &= R_2(5) \\ &+ R_2(4) \\ &+ R_2(1).R_0(3) + R_1(1).R_1(3) + R_0(1).R_2(3) \\ &+ R_2(2).R_0(2) + R_1(2).R_1(2) + R_0(2).R_2(2) \\ &+ R_2(3).R_0(1) + R_1(3).R_1(1) + R_0(3).R_2(1) \\ &+ R_2(4) \\ &+ R_2(5) \end{aligned}$$

This is an expression involving terms $R_i(j)$ where either $i < 3$, or $j < 7$, or both. In turn, this calls for computing these smaller values first in a table, and by subsequently solving $R_1(j)$ for $j =$ 1through 7, followed by $R_2(j)$ for $j =$ 1through 7, and finally $R_3(j)$ for $j =$ 1through 7. Much farther below, we show how this can be generalized.

n	$R_0(n)$	$R_1(n)$	$R_2(n)$	$R_3(n)$	$R_4(n)$	
1	1	1				
2	1	2				
3	1	3	2			
4	1	4	6			
5	1	5	12	6		
6	1	6	20	24		
7	1	7	30	60	24	
8	1	8	42	120	120	

Once the table has been generated, you can divide each $R_k(n)$-entry by $k!$ to arrive at $S_k(n)$, and finally the probability that there is no pair of consecutive integers if you draw k integers from 1 through n, you need to divide $S_k(n)$ by the overall probability of getting k numbers from n, i.e. divide by $\binom{n}{k}$. As you can see, by conditioning on the value of the first draw will lead to a large counting tree, because all possible orders are encountered and accounted for. Below we will investigate a counting (conditioning) technique that generates only the cases that are asked for.

An alternate tree

Before developing an alternate tree diagram, we can tabulate this situation to get some insight. Again, the question is: How many ways can 3 elements be selected from 7 elements, so that none of the 3 are consecutive?. In the table below, we indicate with an 'X' that an element (i.e. location) has been selected (drawn):

Count	1	2	3	4	5	6	7
1			X		X		X
2		X			X		X
3	X				X		X
4		X		X			X
5	X			X			X
6	X		X				X
7		X		X		X	
8	X			X		X	
9	X		X			X	
10	X		X		X		

Note that the table has 6 rows with location 7 (the last location) selected as the largest of the three X's, 3 rows with location 6 selected as the largest of the three, and 1 row with location 5 selected as the largest of the three, for a total of $S_3(7) = 6 + 3 + 1 = 10$. We will see this pattern again when we derive the formula for the general case, where we only generate these 10 possibilities.

When we derived the counting tree above for the "3 out of 7" counting experiment, we did this by staying close to the actual experiment and conditioned on the value of the first draw. Instead, we now imagine that the experiment is done and over with, and now look at which values (locations) have been included in the selection. There are still two ways to condition. We could condition on the whether or not the highest location is included, or is not included. Alternately, we could condition on the highest location that has been included. We will illustrate both for the "3 out of 7 non-consecutive numbers", and generalize both to the "k out of n non-consecutive numbers"

In the first method, we **condition on including, or not including, the last location**, 7.

1. If the last location 7 *is not* included, then you must find the number of ways to select the 3 integers from the first 6 locations. The number of ways that this can be done is what we called $S_3(6)$.
2. If the last location 7 *is* included, then location 6 will not be included and you must find the number of ways to select the additional 2 integers from the 5 integers that remain to be considered. The number of ways that this can be done is what we called $S_2(5)$

These two branches need to be combined:

$$\boxed{S_3(7) = S_3(6) + S_2(5)}$$

So we can calculate $S_3(7)$ as soon as $S_3(6)$ and $S_2(5)$ are available. This is a recursive relation is very similar the recursion of the binomial coefficients.

In the second method, we **condition on highest location that was included**:

1. If the last location 7 *is* included, then location 6 will not be included and you must find the number of ways to select the additional 2 integers from the 5 integers that remain. The number of ways that this can be done is what we called $S_2(5)$.
2. If the last location 7 *is not* included, and location 6 *is* included, then location 5 will not be included and you must find the number of ways to select the additional 2 integers from the 4 integers that remain. The number of ways that this can be done is what we called $S_2(4)$.
3. If both locations 7 and 6 *are not* included, then location 5 *must be* included, then location 4 will not be included and you must find the number of ways to select the additional 2 integers from the 3 integers that remain. The number of ways that this can be done is what we called $S_2(3)$.
4. It is not possible to select 3 non-consecutive numbers from 4 locations: $S_3(4) = 0$.

These three branches need to be combined:

$$\boxed{S_3(7) = S_2(5) + S_2(4) + S_2(3)}$$

So we can calculate $S_3(7)$ as soon as $S_2(5)$, $S_2(4)$ and $S_2(3)$ are available.

For both these methods, the values of $S_k(n)$ for small values of both k and n can be written in a table. We present the table twice: On the left side, you will find numerical values, whereas on the right, you will find the equivalent

values written as binomial coefficients, which will aid in finding the general expression below.

n	$S_0(n)$	$S_1(n)$	$S_2(n)$	$S_3(n)$	$S_4(n)$	
1	1	1				
2	1	2				
3	1	3	1			
4	1	4	3			
5	1	5	6	1		
6	1	6	10	4		
7	1	7	15	10	1	
8	1	8	21	20	5	

n	$S_0(n)$	$S_1(n)$	$S_2(n)$	$S_3(n)$	$S_4(n)$	
1	$\binom{2}{0}$	$\binom{1}{1}$				
2	$\binom{3}{0}$	$\binom{2}{1}$				
3	$\binom{4}{0}$	$\binom{3}{1}$	$\binom{2}{2}$			
4	$\binom{5}{0}$	$\binom{4}{1}$	$\binom{3}{2}$			
5	$\binom{6}{0}$	$\binom{5}{1}$	$\binom{4}{2}$	$\binom{3}{3}$		
6	$\binom{7}{0}$	$\binom{6}{1}$	$\binom{5}{2}$	$\binom{4}{3}$		
7	$\binom{8}{0}$	$\binom{7}{1}$	$\binom{6}{2}$	$\binom{5}{3}$	$\binom{4}{4}$	
8	$\binom{9}{0}$	$\binom{8}{1}$	$\binom{7}{2}$	$\binom{6}{3}$	$\binom{5}{4}$	

Note, that indeed $S_3(7)$ is indeed equal to either $S_3(6) + S_2(5)$ or $S_2(5) + S_2(4) + S_2(3)$.

Generalization to selecting k non-consecutive numbers from among the integers 1 through n :

Suppose you are interested in selecting k integers from among the integers 1 through n, such that you do not chose two consecutive numbers. So fix the value of k and let $S_k(n)$ represent the number of ways that you can do this. We will indicate three slightly different methods to arrive at

$$\boxed{S_k(n) = \binom{n+1-k}{k}}$$

.

In the first method, follow the example above and **condition on including, or not including, the last location**, n. The derivation is pretty similar, and we assume first that n is pretty large, as compared to k:

1. If the last location n *is not* included, then you must find the number of ways to select the k integers from the first $n-1$ locations. The number of ways that this can be done is what we called $S_k(n-1)$.

2. If the last location n *is* included, then location $n-1$ will not be included and you must find the number of ways to select the additional $k-1$ integers from the $n-2$ integers that remain to be considered. The number of ways that this can be done is what we called $S_{k-1}(n-2)$

These two branches need to be combined:

$$\boxed{S_k(n) = S_k(n-1) + S_{k-1}(n-2)}$$

So we can calculate $S_k(n)$ as soon as $S_k(n-1)$ and $S_{k-1}(n-2)$ are available. This is a recursive relation is very similar the recursion of the binomial coefficients. However, we do need to specify the boundary equations. Please note, that

$$S_k(n) = 0, \text{ for } n = 1, 2, 3, \cdots 2k-2$$

. Also

$$S_0(n) = 1, \text{ because there is one way to 'pick nothing'}$$

and

$$S_1(n) = n, \text{ for } n = 1, 2, \cdots \text{ because a single item has no neighbor}$$

. This gives the fully specified recurrence relations:

$$\boxed{S_k(n) = \begin{cases} n & k = 1 \\ 0 & k > 1 \text{ and } n < 2k-1 \\ S_k(n-1) + S_{k-1}(n-2) & k > 1 \text{ and } n \geq 2k \end{cases}}$$

This is a recursive relation that is identical to a recursion of the binomial coefficients, except for the boundary, and the solution is as given above.

In the second method, follow again the example above for the "3 out-of 7" and **condition on highest location that was included**:

1. If the last location n *is* included, then location $n-1$ will not be included and you must find the number of ways to select the additional $k-1$ integers from the $n-2$ integers that remain. The number of ways that this can be done is what we called $S_{k-1}(n-2)$.
2. If the last location n *is not* included, and location $n-1$ *is* included, then location $n-2$ will not be included and you must find the number of ways to select the additional $k-1$ integers from the $n-3$ integers that remain. The number of ways that this can be done is what we called $S_{k-1}(n-3)$.
3. If both locations n and $n-1$ *are not* included, and location $n-2$ *is* included, then location $n-3$ will not be included and you must find the number of ways to select the additional $k-1$ integers from the $n-4$ integers that remain. The number of ways that this can be done is what we called $S_{k-1}(n-4)$.
4. $\cdots$
5. If locations n through $n-i+1$ *are not* included, and location $n-i$ *is* included, then location $n-i-1$ will not be included and you must find the number of ways to select the additional $k-1$ integers from the $n-i-2$ integers that remain. The number of ways that this can be done is what we called $S_{k-1}(n-i-2)$.
6. $\cdots$
7. It is not possible to select k non-consecutive numbers from $2k-2$ locations, so the last location is for $n = 2k+1$: $S_k(2k+1) = 1$ and $S_{k-1}(2k-1) = 1$.

These branches need to be combined, and thus:

$$S_k(n) = S_{k-1}(n-2) + S_{k-1}(n-3) + S_{k-1}(n-4) + \cdots S_{k-1}(2k-1)$$

Incorporating the boundary equations gives the fully specified recurrence relations:

$$\boxed{S_k(n) = \begin{cases} n & k = 1 \\ S_{k-1}(n-2) + S_{k-1}(n-3) + \cdots S_{k-1}(2k-1) & k > 1 \text{ and } n \geq 2k \end{cases}}$$

This is a recursive relation that is identical to a recursion of the binomial coefficients, except for the boundary, and the solution is as given above.

Right at the beginning, we showed that by **conditioning on the value of the first draw** will lead so combinatorial summations, which we will generalize here. Condition on the location of the very first selection you make: If you select location i, then you also remove locations $i-1$ and $i+1$, and you have to select $k-1$ additional locations from two intervals: one interval with $i-2$ potential locations and one interval with $n-i-1$ potential locations. So you can chose j from one interval, and $k-1-j$ from the other, for various values of j. We notated these as $R_j(i-1)$ and $R_{k-1-j}(n-i-1)$, thus if you condition on i being picked first, then

$$R_k(n|i) = \{R_0(i-1)R_{k-1}(n-i-1)+R_1(i-1)R_{k-2}(n-i-1)+R_2(i-1)R_{k-3}(n-i-1)+\cdots R_{k-1}(i-1)R_0(n-i-1)\}$$

or

$$R_k(n|i) = \sum_{j=0}^{k-1}\{R_j(i-1)R_{k-1-j}(n-i-1)\}$$

and finally

$$R_k(n) = \sum_{i=1}^{n}\{R_k(n|i)\} = \sum_{i=1}^{n}\{\sum_{j=0}^{k-1}\{R_j(i-1)R_{k-1-j}(n-i-1)\}\}$$

The last formula gives a formula that is easily implemented and numerical values are readily obtained, although finding closed form expressions in terms of binomials are harder to recognize.

Epilogue

The first two methods use the greedy method to solve the problem, whereas the third method uses the divide-and-conquer approach. The resulting recurrence relations for the last methods are quite common (unfortunately) when you analyze algorithms on (binary) trees or graphs, when the greedy method does not apply.

14 Future sections

The current hand-out does not yet include some topics that are 'close to being done' and will be added shortly, such as additional case studies and additional advanced techniques for average case studies, different boundaries, additional recurrence relations that are solved with generating functions.