Recursion

Slides by Christopher M. Bourke
Instructor: Berthe Y. Choueiry

Spring 2006

Computer Science & Engineering 235
Introduction to Discrete Mathematics
Sections 6.1 - 6.2 of Rosen
cse235@cse.unl.edu

Recursive Algorithms

A *recursive algorithm* is one in which objects are defined in terms of other objects of the same type.

Advantages:

- Simplicity of code
- Easy to understand

Disadvantages:

- Memory
- Speed
- Possibly redundant work

Tail recursion offers a solution to the memory problem, but really, do we *need* recursion?

Recursive Algorithms

Analysis

We've already seen how to analyze the running time of algorithms. However, to analyze recursive algorithms, we require more sophisticated techniques.

Specifically, we study how to define & solve *recurrence relations*.

Motivating Example

Factorial

Recall the factorial function.

$$n! = \begin{cases} 1 & \text{if } n = 1 \\ n \cdot (n-1)! & \text{if } n > 1 \end{cases}$$

Consider the following (recursive) algorithm for computing $n!$:

Algorithm (Factorial)

```
INPUT   : n ∈ ℕ
OUTPUT  : n!
if n = 1 then
    return 1
end
else
    return FACTORIAL(n − 1) × n
end
```

Motivating Example

Factorial - Analysis?

How many multiplications $M(n)$ does Factorial perform?

- When $n = 1$ we don't perform any.
- Otherwise we perform 1.
- *Plus* how ever many multiplications we perform in the recursive call, Factorial$(n-1)$.
- This can be expressed as a formula (similar to the definition of $n!$.

$$\begin{aligned} M(0) &= 0 \\ M(n) &= 1 + M(n-1) \end{aligned}$$

- This is known as a *recurrence relation*.

Recurrence Relations I

Definition

Definition

A *recurrence relation* for a sequence $\{a_n\}$ is an equation that expresses a_n in terms of one or more of the previous terms in the sequence,

$$a_0, a_1, \ldots, a_{n-1}$$

for all integers $n \geq n_0$ where n_0 is a nonnegative integer.

A sequence is called a *solution* of a recurrence relation if its terms satisfy the recurrence relation.

Recurrence Relations II
Definition

Consider the recurrence relation: $a_n = 2a_{n-1} - a_{n-2}$.
It has the following sequences a_n as solutions:

1. $a_n = 3n$,
2. $a_n = 2^n$, and
3. $a_n = 5$.

Initial conditions + recurrence relation uniquely determine the sequence.

Recurrence Relations III
Definition

Example

The Fibonacci numbers are defined by the recurrence,

$$\begin{aligned} F(n) &= F(n-1) + F(n-2) \\ F(1) &= 1 \\ F(0) &= 1 \end{aligned}$$

The solution to the Fibonacci recurrence is

$$f_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

(your book derives this solution).

Recurrence Relations IV
Definition

More generally, recurrences can have the form

$$T(n) = \alpha T(n-\beta) + f(n), \quad T(\delta) = c$$

or

$$T(n) = \alpha T\left(\frac{n}{\beta}\right) + f(n), \quad T(\delta) = c$$

Note that it may be necessary to define *several* $T(\delta)$, *initial conditions*.

Recurrence Relations V
Definition

The *initial conditions* specify the value of the first few necessary terms in the sequence. In the Fibonacci numbers we needed *two* initial conditions, $F(0) = F(1) = 1$ since $F(n)$ was defined by the two previous terms in the sequence.

Initial conditions are also known as *boundary conditions* (as opposed to the *general conditions*).

From now on, we will use the subscript notation, so the Fibonacci numbers are

$$\begin{aligned} f_n &= f_{n-1} + f_{n-2} \\ f_1 &= 1 \\ f_0 &= 1 \end{aligned}$$

Recurrence Relations VI
Definition

Recurrence relations have two parts: recursive terms and non-recursive terms.

$$T(n) = \underbrace{2T(n-2)}_{\text{recursive}} + \underbrace{n^2 - 10}_{\text{non-recrusive}}$$

Recursive terms come from when an algorithm calls itself.

Non-recursive terms correspond to the "non-recursive" cost of the algorithm—work the algorithm performs *within* a function.

We'll see some examples later. First, we need to know how to *solve* recurrences.

Solving Recurrences

There are several methods for solving recurrences.

- Characteristic Equations
- Forward Substitution
- Backward Substitution
- Recurrence Trees
- Maple!

Linear Homogeneous Recurrences

Definition

A *linear homogeneous recurrence relation* of degree k with constant coefficients is a recurrence relation of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

with $c_1, \ldots, c_k \in \mathbb{R}$, $c_k \neq 0$.

- Linear: RHS is a sum of multiples of previous terms of the sequence (linear combination of previous terms). The coefficients are all constants (not functions depending on n).
- Homogeneous: no terms occur that are not multiples of the a_j's.
- Degree k: a_n is expressed in terms of k terms of the sequence.

Linear Homogeneous Recurrences

Examples

Examples

The Fibonacci sequence is a linear homogeneous recurrence relation. As are the following.

$$a_n = 4a_{n-1} + 5a_{n-2} + 7a_{n-3}$$

$$a_n = 2a_{n-2} + 4a_{n-4} + 8a_{n-8}$$

How many initial conditions do we need to specify for these? As many as the degree, $k = 3, 8$ respectively.

So, how do we *solve* linear homogeneous recurrences?

Solving Linear Homogeneous Recurrences I

We want a solution of the form $a_n = r^n$ where r is some (real) constant.

We observe that $a_n = r^n$ is a solution to a linear homogeneous recurrence if and only if

$$r^n = c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k r^{n-k}$$

We can now divide both sides by r^{n-k}, collect terms, and we get a k-degree polynomial.

$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_{k-1} r - c_k = 0$$

Solving Linear Homogeneous Recurrences II

$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_{k-1} r - c_k = 0$$

This is called the *characteristic equation* of the recurrence relation.

The roots of this polynomial are called the *characteristic roots* of the recurrence relation. They can be used to find solutions (if they exist) to the recurrence relation. We will consider several cases.

Second Order Linear Homogeneous Recurrences

A *second order* linear homogeneous recurrence is a recurrence of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

Theorem (Theorem 1, p414)

Let $c_1, c_2 \in \mathbb{R}$ and suppose that $r^2 - c_1 r - c_2 = 0$ is the characteristic polynomial of a 2nd order linear homogeneous recurrence which has two distinct[1] roots, r_1, r_2.

Then $\{a_n\}$ is a solution if and only if

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$$

for $n = 0, 1, 2, \ldots$ where α_1, α_2 are constants dependent upon the initial conditions.

[1] we discuss how to handle this situation later.

Second Order Linear Homogeneous Recurrences

Example

Example

Find a solution to

$$a_n = 5a_{n-1} - 6a_{n-2}$$

with initial conditions $a_0 = 1, a_1 = 4$

- The characteristic polynomial is

$$r^2 - 5r + 6$$

- Using the quadratic formula (or common sense), the root can be found;

$$r^2 - 5r + 6 = (r-2)(r-3)$$

so $r_1 = 2, r_2 = 3$

Second Order Linear Homogeneous Recurrences

Example Continued

- Using the 2nd-order theorem, we have a solution,

$$a_n = \alpha_1(2^n) + \alpha_2(3^n)$$

- Now we can plug in the two initial conditions to get a system of linear equations.

$$\begin{aligned} a_0 &= \alpha_1(2)^0 + \alpha_2(3)^0 \\ a_1 &= \alpha_1(2)^1 + \alpha_2(3)^1 \end{aligned}$$

$$\begin{aligned} 1 &= \alpha_1 + \alpha_2 \qquad (1) \\ 4 &= 2\alpha_1 + 3\alpha_2 \qquad (2) \end{aligned}$$

Second Order Linear Homogeneous Recurrences

Example Continued

- Solving for $\alpha_1 = (1 - \alpha_2)$ in (1), we can plug it into the second.

$$\begin{aligned} 4 &= 2\alpha_1 + 3\alpha_2 \\ 4 &= 2(1 - \alpha_2) + 3\alpha_2 \\ 4 &= 2 - 2\alpha_2 + 3\alpha_2 \\ 2 &= \alpha_2 \end{aligned}$$

- Substituting back into (1), we get

$$\alpha_1 = -1$$

- Putting it all back together, we have

$$\begin{aligned} a_n &= \alpha_1(2^n) + \alpha_2(3^n) \\ &= -1 \cdot 2^n + 2 \cdot 3^n \end{aligned}$$

Second Order Linear Homogeneous Recurrences

Another Example

Example

Solve the recurrence

$$a_n = -2a_{n-1} + 15a_{n-2}$$

with initial conditions $a_0 = 0$, $a_1 = 1$.

If we did it right, we have

$$a_n = \frac{1}{8}(3)^n - \frac{1}{8}(-5)^n$$

How can we check ourselves?

Single Root Case

Recall that we can only apply the first theorem if the roots are *distinct*, i.e. $r_1 \neq r_2$.

If the roots are not distinct ($r_1 = r_2$), we say that one characteristic root has *multiplicity two*. In this case we have to apply a different theorem.

Theorem (Theorem 2, p416)

Let $c_1, c_2 \in \mathbb{R}$ with $c_2 \neq 0$. Suppose that $r^2 - c_1r - c_2 = 0$ has only one distinct root, r_0. Then $\{a_n\}$ is a solution to $a_n = c_1a_{n-1} + c_2a_{n-2}$ if and only if

$$a_n = \alpha_1 r_0^n + \alpha_2 n r_0^n$$

for $n = 0, 1, 2, \ldots$ where α_1, α_2 are constants depending upon the initial conditions.

Single Root Case

Example

Example

What is the solution to the recurrence relation

$$a_n = 8a_{n-1} - 16a_{n-2}$$

with initial conditions $a_0 = 1, a_1 = 7$?

- The characteristic polynomial is

$$r^2 - 8r + 16$$

- Factoring gives us

$$r^2 - 8r + 16 = (r - 4)(r - 4)$$

so $r_0 = 4$

Single Root Case

Example

- By Theorem 2, we have that the solution is of the form

$$a_n = \alpha_1 4^n + \alpha_2 n 4^n$$

- Using the initial conditions, we get a system of equations;

$$\begin{aligned} a_0 = 1 &= \alpha_1 \\ a_1 = 7 &= 4\alpha_1 + 4\alpha_2 \end{aligned}$$

- Solving the second, we get that $\alpha_2 = \frac{3}{4}$
- And so the solution is

$$a_n = 4^n + \frac{3}{4}n4^n$$

- We should check ourselves...

General Linear Homogeneous Recurrences

There is a straightforward generalization of these cases to higher order linear homogeneous recurrences.

Essentially, we simply define higher degree polynomials.

The roots of these polynomials lead to a general solution.

The general solution contains coefficients that depend only on the initial conditions.

In the general case, however, the coefficients form a *system* of linear inequalities.

General Linear Homogeneous Recurrences I

Distinct Roots

Theorem (Theorem 3, p417)

Let $c_1, \ldots, c_k \in \mathbb{R}$. Suppose that the characteristic equation

$$r^k - c_1 r^{k-1} - \cdots - c_{k-1} r - c_k = 0$$

has k distinct roots, $r_1, \ldots, r_k$. Then a sequence $\{a_n\}$ is a solution of the recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

if and only if

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \cdots + \alpha_k r_k^n$$

for $n = 0, 1, 2, \ldots$, where $\alpha_1, \alpha_2, \ldots, \alpha_k$ are constants.

General Linear Homogeneous Recurrences

Any Multiplicity

Theorem (Theorem 4, p418)

Let $c_1, \ldots, c_k \in \mathbb{R}$. Suppose that the characteristic equation

$$r^k - c_1 r^{k-1} - \cdots - c_{k-1} r - c_k = 0$$

has t distinct roots, $r_1, \ldots, r_t$ with multiplicities $m_1, \ldots, m_t$.

General Linear Homogeneous Recurrences

Any Multiplicity

Theorem (Continued)

Then a sequence $\{a_n\}$ is a solution of the recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

if and only if

$$\begin{aligned} a_n \quad = \quad & (\alpha_{1,0} + \alpha_{1,1} n + \cdots + \alpha_{1,m_1-1} n^{m_1-1}) r_1^n + \\ & (\alpha_{2,0} + \alpha_{2,1} n + \cdots + \alpha_{2,m_2-1} n^{m_2-1}) r_2^n + \\ & \quad \vdots \\ & (\alpha_{t,0} + \alpha_{t,1} n + \cdots + \alpha_{t,m_t-1} n^{m_t-1}) r_t^n + \end{aligned}$$

for $n = 0, 1, 2, \ldots$, where $\alpha_{i,j}$ are constants for $1 \leq i \leq t$ and $0 \leq j \leq m_i - 1$.

Linear Nonhomogeneous Recurrences

For recursive algorithms, cost functions are often *not* homogenous because there is usually a non-recursive cost depending on the input size.

Such a recurrence relation is called a *linear nonhomogeneous recurrence relation*.

Such functions are of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + f(n)$$

Linear Nonhomogeneous Recurrences

Here, $f(n)$ represents a non-recursive cost. If we chop it off, we are left with

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

which is the *associated homogenous recurrence relation*.

Every solution of a linear nonhomogeneous recurrence relation is the sum of a particular solution and a solution to the associated linear homogeneous recurrence relation.

Linear Nonhomogeneous Recurrences

Theorem (Theorem 5, p420)

If $\{a_n^{(p)}\}$ is a particular solution of the nonhomogeneous linear recurrence relation with constant coefficients

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + f(n)$$

then every solution is of the form $\{a_n^{(p)} + a_n^{(h)}\}$, where $\{a_n^{(h)}\}$ is a solution of the associated homogenous recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

Linear Nonhomogeneous Recurrences

There is no *general* method for solving such relations. However, we can solve them for special cases.

In particular, if $f(n)$ is a polynomial or exponential function (or more precisely, when $f(n)$ is the product of a polynomial and exponential function), then there is a general solution.

Linear Nonhomogeneous Recurrences

Theorem (Theorem 6, p421)

Suppose that $\{a_n\}$ satisfies the linear nonhomogeneous recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + f(n)$$

where $c_1, \ldots, c_k \in \mathbb{R}$ and

$$f(n) = (b_t n^t + b_{t-1} n^{t-1} + \cdots + b_1 n + b_0) \cdot s^n$$

where $b_0, \ldots, b_n, s \in \mathbb{R}$.

Linear Nonhomogeneous Recurrences

Theorem (Continued)

When s is not a root of the characteristic equation of the associated linear homogeneous recurrence relation, there is a particular solution of the form

$$(p_t n^t + p_{t-1} n^{t-1} + \cdots + p_1 n + p_0) \cdot s^n$$

When s is a root of this characteristic equation and its multiplicity is m, there is a particular solution of the form

$$n^m (p_t n^t + p_{t-1} n^{t-1} + \cdots + p_1 n + p_0) \cdot s^n$$

Linear Nonhomogeneous Recurrences

The examples in the text are quite good (see pp420–422) and illustrate how to solve simple nonhomogeneous relations.

We may go over more examples if you wish.

Also read up on *generating functions* in section 6.4 (though we may return to this subject).

However, there are alternate, more intuitive methods.

Other Methods

When analyzing algorithms, linear homogenous recurrences of order greater than 2 hardly ever arise in practice.

We briefly describe two "unfolding" methods that work for a lot of cases.

Backward substitution – this works exactly as its name implies: starting from the equation itself, work backwards, substituting values of the function for previous ones.

Recurrence Trees – just as powerful but perhaps more intuitive, this method involves mapping out the recurrence tree for an equation. Starting from the equation, you unfold each recursive call to the function and calculate the non-recursive cost at each level of the tree. You then find a general formula for each level and take a summation over all such levels.

Backward Substitution

Example

Example

Give a solution to

$$T(n) = T(n-1) + 2n$$

where $T(1) = 5$.

We begin by *unfolding* the recursion by a simple substitution of the function values.

Observe that

$$T(n-1) = T((n-1)-1) + 2(n-1) = T(n-2) + 2(n-1)$$

Substituting this into the original equation gives us

$$T(n) = T(n-2) + 2(n-1) + 2n$$

Backward Substitution

Example – Continued

If we continue to do this, we get the following.

$$\begin{aligned} T(n) &= T(n-2) + 2(n-1) + 2n \\ &= T(n-3) + 2(n-2) + 2(n-1) + 2n \\ &= T(n-4) + 2(n-3) + 2(n-2) + 2(n-1) + 2n \\ &\vdots \\ &= T(n-i) + \textstyle\sum_{j=0}^{i-1} 2(n-j) \end{aligned}$$

I.e. this is the function's value at the i-th iteration. Solving the sum, we get

$$T(n) = T(n-i) + 2n(i-1) + 2\frac{(i-1)(i-1+1)}{2} + 2n$$

Backward Substitution

Example – Continued

We want to get rid of the recursive term. To do this, we need to know at what iteration we reach our base case; i.e. for what value of i can we use the initial condition, $T(1) = 5$?

We can easily see that when $i = n-1$, we get the base case.

Substituting this into the equation above, we get

$$\begin{aligned} T(n) &= T(n-i) + 2n(i-1) - i^2 + i + 2n \\ &= T(1) + 2n(n-1-1) - (n-1)^2 + (n-1) + 2n \\ &= 5 + 2n(n-2) - (n^2 - 2n + 1) + (n-1) + 2n \\ &= n^2 + n + 3 \end{aligned}$$

Recurrence Trees

When using recurrence trees, we graphically represent the recursion.

Each node in the tree is an instance of the function. As we progress downward, the size of the input decreases.

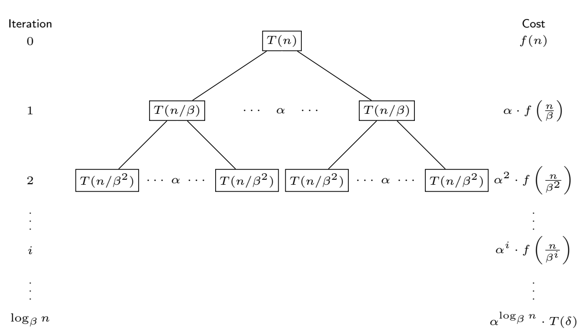
The contribution of each level to the function is equivalent to the number of nodes at that level times the non-recursive cost on the size of the input at that level.

The tree ends at the depth at which we reach the base case.

As an example, we consider a recursive function of the form

$$T(n) = \alpha T\left(\frac{n}{\beta}\right) + f(n), \quad T(\delta) = c$$

Recurrence Trees



Recurrence Trees

Example

The total value of the function is the summation over all levels of the tree:

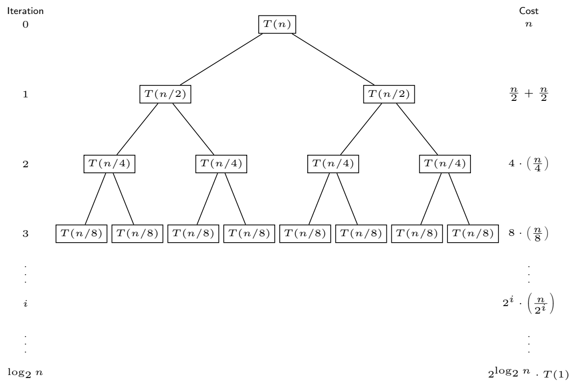
$$T(n) = \sum_{i=0}^{\log_\beta n} \alpha^i \cdot f\left(\frac{n}{\beta^i}\right)$$

We consider the following concrete example.

Example

$$T(n) = 2T\left(\frac{n}{2}\right) + n, \quad T(1) = 4$$

Recurrence Trees
Example – Continued



Recurrence Trees
Example – Continued

The value of the function then, is the summation of the value of all levels. We treat the last level as a special case since its non-recursive cost is different.

$$T(n) = 4n + \sum_{i=0}^{(\log_2 n)-1} 2^i \frac{n}{2^i} = n(\log n) + 4n$$

Smoothness Rule I

In the previous example we make the following assumption: that n was a power of two; $n = 2^k$. This was necessary to get a nice depth of $\log n$ and a *full* tree.

We can restrict consideration to certain powers because of the *smoothness rule*.

Definition

A function $f : \mathbb{N} \mapsto \mathbb{R}$ is called *smooth* if it is monotonically nondecreasing and

$$f(2n) \in \Theta(f(n))$$

Most "slow" growing functions (logarithmic, polylogarithmic, polynomial) are smooth while exponential functions are not.

Smoothness Rule II

Theorem

For a smooth function $f(n)$ and a fixed constant $b \in \mathbb{Z}$ such that $b \geq 2$,

$$f(bn) \in \Theta(f(n))$$

Thus the order of growth is preserved.

How To Cheat With Maple I

Maple and other math tools are great resources. However, they are not substitutes for knowing how to solve recurrences yourself.

As such, you should *only* use Maple to *check* your answers. Recurrence relations can be solved using the `rsolve` command and giving Maple the proper parameters.

The arguments are essentially a comma-delimited list of equations: general and boundary conditions, followed by the "name" and variable of the function.

How To Cheat With Maple II

```
> rsolve({T(n) = T(n-1) + 2*n, T(1) = 5}, T(n));
```

$$1 + 2(n+1)\left(\frac{1}{2}n + 1\right) - 2n$$

You can clean up Maple's answer a bit by encapsulating it in the `simplify` command:

```
> simplify(rsolve({T(n) = T(n-1) + 2*n, T(1) = 5},
T(n)));
```

$$3 + n^2 + n$$