# Algorithm analysis: Solving the recursion problem

**William Hendrix**

*Lecture 7*

# Today

- Review
- Modelling recursive functions
- Solving linear recurrences with constant coefficients (LRCCs)
  - Characteristic equation
  - Homogeneous LRCCs
  - Multiple roots
  - Nonhomogeneous LRCCs

# Review: analysis

- Identify loops and function calls
  - Everything else is *O(1)*
- *For loops*:
  - Estimate loop body running time
    - Might depend on iteration #
  - Estimate number of iterations
  - Total time: sum of all iterations
    - Estimate: number of iterations * longest iteration
    - Works well if all iterations are same complexity
  - *If statements:* decide how many times they execute
- *For functions:*
  - Analyze other functions separately
  - Recursive functions: set up a recurrence and solve
- Overall complexity: largest loop or function call complexity

# Yesterday's example

- **Algorithm:** Selection Sort

*Input:*

```
data:  an array of integers to sort
n:     the number of values in data
```

*Output:* permutation of `data` such that `data[1] ≤ ... ≤ data[n]`

*Pseudocode:*

```
for i = 1 to n
  Let m be the location of the min value in the array data[i..n]
  Swap data[i] and data[m]
end
return data
```

**Proof idea: for** loop takes *n* iterations of *n*-*i* time.

$$\sum_{i=1}^{n} n - i = \underbrace{(n-1) + (n-2) + \ldots}_{n/2 \text{ of size } \geq n/2} + 2 + 1 + 0$$

$$\Omega(\frac{n^2}{4}) = \Omega(n^2)$$

$$\Theta(n^2) \text{ total time}$$

# Recursion example

- **Algorithm:** Insertion Sort (recursive)

*Input:*

`data:   an array of integers to sort`

`n:      the number of values in data`

*Output:* permutation of `data` such that `data[1] ≤ ... ≤ data[n]`

*Pseudocode:*

```
if n > 1
  Call Insertion Sort on data[1..n-1]
  Let ins = data[n]
  Let j = last index of data[1..n-1] ≤ ins
  Shift data[j+1..n-1] to the right one space
  data[j+1] = ins
end
return data
```

*Hint*: Let $T(n)$ be the time complexity for Insertion Sort on an array of size $n$.
Write a function for $T(n)$ in terms of $T(n\text{-}1)$ and $n$

# Recursion example solution, step 1

- $T(1) = O(1)$
- $T(n) = T(n-1) + O(n)$, for $n > 1$

# Recursion example solution, step 1

- *T(1) = O(1)*
- *T(n) = T(n-1) + O(n)*, for *n > 1*

*Proof.* If $n = 1$, only the **if** condition and **return** statement are executed, for a total of $O(1)$ time, so $T(1) = O(1)$.
If $n > 1$, line 2 calls Insertion Sort on $data[1..n-1]$, which takes $T(n-1)$ time. Lines 4 and 5 can be accomplished by using a loop that starts on the right side of the array and shifts elements right until it finds an element less than $ins$. In the worst case, this loop may shift all $n-1$ elements left, so they takes $O(n-1) = O(n)$ time. All other statements (lines 1, 3, 6, and 8) take $O(1)$ time, which is dominated by the $O(n)$ time taken by lines 4 and 5. Thus, $T(n) = T(n-1) + O(n)$ for $n > 1$. □

# Linear recurrences with constant coefficients

- Recurrence: function or sequence defined in terms of previous values
  - Fibonacci sequence: *F(n) = F(n-1) + F(n-2)*
- LRCC: recurrence of the form

$$T(n) = c_1 T(n-1) + c_2 T(n-2) + \ldots + c_k T(n-k) + f(n)$$

  - Each recursive term must be a multiple of a previous value in the sequence
    - No quadratic terms, products of previous values, coefficients that depend on *n*, etc.
  - May include additional terms that depend on *n*
- Linear homogeneous recurrences with constant coefficients
  - All terms are multiples of previous values
  - $f(n) = 0$
  - **Examples:** $F(n) = F(n-1) + F(n-2), \;\; g(n) = 2g(n-1)$
- Nonhomogeneous recurrences
  - $f(n) \neq 0$
  - **Examples:** $T(n) = T(n-1) + 2, H(n) = 2H(n-1) + 1$

# Characteristic polynomial

- A.k.a. "characteristic equation"
- Polynomial whose solution relates to the closed form solution of a linear recurrence with constant coefficients
- Degree is the distance between *T(n)* and the earliest value that appears in the recurrence equation
- Coefficient for $x^k$ is negation of coefficient of *T(n-k-1)*, or 1 for highest power
  - Nonhomogeneous recurrences: ignore non-recursive terms
- **Examples:** $g(n) = 2g(n-1)$

Degree 1: $c(x) = x - 2$

Coefficient 1

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

Degree 2: $c(x) = x^2 - x - 1$

Coefficient 1

$$a(n) = a(n-1) - 6a(n-2)$$
$$c(x) = x^2 - x + 6$$

$$T(n) = \overset{0T(n-1)+}{\vee}\,4T(n-2)$$
$$c(x) = x^2 - 4$$

# Solving linear homogeneous recurrences

**Theorem.** *If the characteristic equation of the recurrence $a_n$ has distinct roots $r_1, r_2, \ldots, r_k$, then $a_n = d_1{r_1}^n + d_2{r_2}^n + \ldots + d_k{r_k}^n$, for some constants $d_1, \ldots, d_k$.*

*Proof.* Complicated!

**Example.** $T(n) = 3T(n-1) - 2T(n-2)$, where $T(0) = 1$ and $T(1) = 2$

*Char. eqn:* $c(x) = x^2 - 3x + 2 = 0$

$$(x-1)(x-2) = 0$$

$$x_1 = 1 \quad x_2 = 2$$

$$T(n) = d_1 1^n + d_2 2^n$$

$$T(n) = d_1 + d_2 2^n$$

$$T(0) = 1 = d_1 + d_2$$

$$T(1) = 2 = d_1 + 2d_2$$

$$d_2 = 1$$

$$d_1 = 0$$

$$T(n) = O(2^n)$$

$\boxed{\textit{Insight}\text{: Use } T(0) \text{ and } T(1) \text{ to solve for } d_1 \text{ and } d_2!}$

# Roots with multiplicity

- Add an extra power of $n$ to $r^n$ for every additional root

More formally:
If the characteristic equation of the recurrence $T(n)$ has $k$ total roots $r_1, r_2, \ldots, r_j$ with multiplicities $m_1, \ldots, m_j$, then:

$$T(n) = O(n^{m_1 - 1}){r_1}^n + \ldots + O(n^{m_j - 1}){r_j}^n.$$

**Example.** $T(n) = 4T(n-1) - 4T(n-2)$

*Char. poly:*

$$\begin{aligned} x^2 - 4x + 4 &= 0 \\ (x-2)^2 &= 0 \\ x_1 = x_2 &= 2 \\ T(n) &= O(n)2^n \\ &= O(n2^n) \end{aligned}$$

# Recurrence exercises

**Find the asymptotic growth of the following recurrences.**

1. $A(n) = 9A(n-1)$

2. $B(n) = 9B(n-2)$

3. $C(n) = 7C(n-1) - 12C(n-2)$

4. $D(n) = 8D(n-1) - 16D(n-2)$

5. $E(n) = 4E(n-1) - 2E(n-2)$

# Recurrence exercises

**Find the asymptotic growth of the following recurrences.**

1. $A(n) = 9A(n-1)$

   $A(n) = O(9^n)$

2. $B(n) = 9B(n-2)$

   $B(n) = O(3^n)$

3. $C(n) = 7C(n-1) - 12C(n-2)$

   $C(n) = O(4^n)$

4. $D(n) = 8D(n-1) - 16D(n-2)$

   $D(n) = O(n4^n)$

5. $E(n) = 4E(n-1) - 2E(n-2)$

   $E(n) = O((2+\sqrt{2})^n)$

# Nonhomogeneous recurrences

- In general, *T(n)* may have terms that aren't just recursive calls

$$T(n) = c_1 T(n-1) + c_2 T(n-2) + \ldots + c_k T(n-k) + f(n)$$

- If *f(n)* is polynomial:
  - Add $O(n^m f(n))$ to the complexity of homogeneous solution
  - $m$ = multiplicity of $r=1$ as root of char. eqn.
    - Increases degree of *f(n)* by *m*

- **Example:** $T(n) = 4T(n-1) - 3T(n-2) + 2n - 1$

*Char. eqn:* $c(x) = x^2 - 4x + 3 = 0$

$$(x-1)(x-3) = 0$$

$$x_1 = 1 \quad x_2 = 3$$

$$T(n) = d_1 + d_2 2^n + O(n^m f(n))$$

$$T(n) = O(2^n) + O(n^2)$$

$$T(n) = O(2^n)$$

# Summary: recurrences with Big-Oh

- *Recall*: coefficients don't matter, and only the largest term counts
- **Summary:** $T(n) = O(n^{m_1 - 1}{r_1}^n + \ldots + n^{m_k - 1}{r_k}^n + n^m f(n))$
  - $r_i$: roots of characteristic polynomial
  - $m_i$: multiplicity of root $r_i$
  - $m$: multiplicity of 1
  - $f(n)$: non-recursive terms in recurrence (polynomial only)
- **Exercise:** $T(n) = 2T(n-1) - 2n^2$

$$c(x) = x - 2$$

$$\text{Root: } x = 2$$

$$\text{General form: } T(n) = O(2^n + n^m(n^2)),$$

$$1 \text{ is not a root} \rightarrow m = 0$$

$$T(n) = O(2^n + n^2)$$

$$\boxed{= O(2^n)}$$

# Recursion example, revisited

- **Algorithm:** Insertion Sort (recursive)

*Input:*

`data:  an array of integers to sort`

`n:     the number of values in data`

*Output:* permutation of `data` such that `data[1] ≤ ... ≤ data[n]`

*Pseudocode:*

```
if n > 1
  Call Insertion Sort on data[1..n-1]
  Let ins = data[n]
  Let j = last index of data[1..n-1] ≤ ins
  Shift data[j+1..n-1] to the right one space
  data[j+1] = ins
end
return data
```

- $T(n) = T(n\text{-}1) + O(n)$, for $n > 1$, and $T(1) = O(1)$
- Prove $T(n) = O(n^2)$

# Recursion example solution, step 2

- $T(n) = T(n\text{-}1) + O(n)$, for $n > 1$, and T(1) = O(1)
- Prove $T(n) = O(n^2)$

$c(x) = x - 1$
Roots: $x_1 = 1$

Nonhomogeneous function:
$f(n)$ has degree 1, so $F(n) = O(n)$
$s = 1$, so $m = 1$

General form:

$$\begin{aligned} T(n) &= d_1({x_1}^n) + O(n^m f(n)) \\ &= d_1(1^n) + O(n^1(n)) \\ &= O(1) + O(n^2) \\ &= O(n^2) \end{aligned}$$

Induction is possible but unpleasant.

# Coming up

- **Exam 1** will be next Tuesday
  - Practice exam posted tonight or tomorrow
  - Thursday will be practice
- Data structures after Exam 1

- **Homework 5** is due Thursday

- **Recommended readings:** Chapters 1 and 2
- **Practice problems:** Any from section 1.10 (p. 27) or 2.10 (p. 57)