Recurrence Relations & Recursion

Computer Science has recursion.
Mathematics has recurrence relations.

Example(s):
Recurrence Relations

**Definition:** Recurrence Relation

**Example(s):**

Solving Recurrence Relations
Definition: Linear Homogeneous Recurrence Relation
With Constant Coefficients (LHRRWCC) Of Degree $k$

Example(s):

Solving LHRRWCCs Of Degree 2 (1/2)
Theorem: Assume a characteristic equation
\[ w^2 - c_1w - c_2 = 0 \] with \( c_1, c_2 \in \mathbb{R} \) and roots \( r_1 \) and \( r_2 \) such that \( r_1 \neq r_2 \). The sequence \( \{R(n)\} \) is a solution to \( R(n) = c_1R(n-1) + c_2R(n-2) \) iff
\[ R(n) = \alpha_1 r_1^n + \alpha_2 r_2^n \] where \( n \in \mathbb{Z}^* \) and \( \alpha_1, \alpha_2 \in \mathbb{R} \).

Solution Procedure: LHRRWCCs of Degree 2

1. Identify \( c_1 \) & \( c_2 \) and create the characteristic equation
   \[ w^2 - c_1w - c_2 = 0 \]

2. Insert the roots of the characteristic equation \( (r_1 \ & r_2) \) into the closed-form expression \( R(n) = \alpha_1 r_1^n + \alpha_2 r_2^n \)

3. Using the initial conditions, create two equations in two unknowns \( (\alpha_1 \ and \ \alpha_2) \)

4. Solve for \( \alpha_1 \) and \( \alpha_2 \) to complete the solution
Example: Solving a LHRRWCC of Degree 2

Solve: \[ R(n) = 3R(n - 1) - 2R(n - 2) \]

where \( R(0) = 200 \) and \( R(1) = 220 \).

“Divide & Conquer” Recurrence Relations (1 / 2)

From the Latin *Divide Et Impera* (“divide and rule”)

Background:
“Divide & Conquer” Recurrence Relations (2 / 2)

Example(s):

Solving Divide & Conquer Rec. Relations (1 / 6)

“Find The Pattern” (a.k.a. Iterative (or Backward) Substitutions)

Example(s):
Conjecture: \( S(n) = k \cdot \log_2 n + 1 \)
Conjecture: \( Q(n) = \frac{n(n+1)}{2} \)

Extra Slides

The remaining slides in this topic are some that I no longer cover in class. I won’t ask about them on a quiz or an exam, but they could be referenced on a homework or in section.
Theorem: (The Master Theorem) Given a recursive function of the form $T(n) = a \cdot T(n/b) + c \cdot n^d$, where:

- $T(n)$ is an increasing function,
- $n = b^k$,
- $k$ is an integer $> 0$,
- $a$ is a real $\geq 1$,
- $b$ is an integer $> 1$,
- $c$ is a real $> 0$, and
- $d$ is a real $\geq 0$,

then:

$$f(n) = \begin{cases} 
O(n^d) & \text{if } a < b^d \\
O(n^d \cdot \log_2 n) & \text{if } a = b^d \\
O(n^{\log_b a}) & \text{if } a > b^d 
\end{cases}$$


Example(s):

Binary Search's recurrence: $S(n) = S(n/2) + k$

Recall: We determined $S(n) = k \cdot \log_2 n + 1 \Rightarrow O(\log_2 n)$

From the Master Theorem: $T(n) = a \cdot T(n/b) + c \cdot n^d$

For Bin. Search, $a = 1$, $b = 2$, $c = k$, and $d = 0$

The 2nd case applies: $a = b^d$ ($1 = 2^0$)

Therefore, $S(n)$ is $O(n^d \cdot \log_2 n)$, or $O(\log_2 n)$.

$\Rightarrow$ We got it right!

*Note:* Master Theorem doesn't fit Quicksort's worst case.