Time Efficiency of Recursive Algorithms

Steps in mathematical analysis of recursive algorithms:

1. Decide on parameter $n$ indicating input size
2. Identify algorithm’s basic operation
3. Determine worst, average, and best case for input of size $n$
4. Set up a recurrence relation and initial condition(s) for $C(n)$ - the number of times the basic operation will be executed for an input size $n$.
5. Solve the recurrence to obtain a closed form or estimate the order of growth of the solution
Last Class: Recursive evaluation of $n!$

Recursive algorithm for $n!$

Input size: $n$

Basic operation: multiplication “*”

Let $C(n)$ be the number of multiplications needed to compute $n!$, then

$\begin{align*}
C(0) &= 0 \\
C(n) &= C(n-1) + 1 \quad \text{for } n > 0
\end{align*}$

ALGORITHM $Factorial(n)$

if $n = 0$
    return 1
else
    return $Factorial(n-1) \times n$

Initial condition

Recurrence relation

Solve the recurrence
A sequence: an ordered list of numbers.

For example: 0, 2, 4, 6, … (even integers)

How to represent a sequence: \( x(n) \) -- General term of the sequence

The index in the sequence

- **Explicit mathematic formula**: e.g., \( x(n) = n + 1 \) for \( n \geq 0 \)
- **Recurrence relation**: e.g., \( x(n) = x(n-1) + 1 \) and \( x(0) = 1 \)

Solving the recurrence \( \leftrightarrow \) finding the explicit formula
Last Class: Solutions of Recurrence Relations

\[ C(n) = C(n - 1) + 1 \text{ for } n > 0 \]

**General solution**

\[ C(n) = C(0) + n \text{ for } n > 0 \]

- A class of solutions **ignoring initial condition**
- Satisfying the recurrence relation with an arbitrary constant

**Particular solution**

\[ C(n) = n \text{ for } n > 0, C(0) = 0 \]

- Satisfying the recurrence relation and the particular initial condition
Example 4 – Solving Recurrence Relations Using Backward Substitutions

\[ T(n) = T(n/2) + 2n \text{ for } n > 1, \quad T(1) = 2 \]

Let \( n = 2^k \), \( k \) is an integer and \( k > 0 \)

\[ T(n) = T(n/2) + 2n \rightarrow T(2^k) = T(2^{k-1}) + 2 \times 2^k \]
Example 4 – Solving Recurrence Relations Using Backward Substitutions

\[ T(n) = T(n/2) + 2n \text{ for } n > 1, \quad T(1) = 2 \]

\[ T(n/2) = T(2^{k-1}) \]

\[ T(2^k) = T(2^{k-1}) + 2 \times 2^k = T(2^{k-2}) + 2 \times 2^{k-1} + 2 \times 2^k \]

\[ = T(2^{k-3}) + 2 \times 2^{k-2} + 2 \times 2^{k-1} + 2 \times 2^k \]

\[ = T(2^{k-k}) + 2 \times \left[ 2^{k-(k-1)} + \cdots + 2^{k-1} + 2^k \right] \]

\[ = T(1) + 2 \times \sum_{i=1}^{k} 2^i = T(1) + 2 \times (2^{k+1} - 1 - 1) \]

\[ n = 2^k \]

\[ = 2 + 2 \times (2n - 2) = 4n - 2 \]
Solution to Important Recurrence Types

One (constant) operation reduces problem size by one.

\[ T(n) = T(n-1) + c \quad \text{for } n > 1 \]
\[ T(1) = d \]
Solution: \[ T(n) = (n-1)c + d \] linear, e.g., factorial

A pass through input reduces problem size by one.

\[ T(n) = T(n-1) + cn \quad \text{for } n > 1 \]
\[ T(1) = d \]
Solution: \[ T(n) = \left[ \frac{n(n+1)}{2} - 1 \right] c + d \] quadratic, e.g., insertion sort

One (constant) operation reduces problem size by half.

\[ T(n) = T(n/2) + c \quad \text{for } n > 1 \]
\[ T(1) = d \]
Solution: \[ T(n) = c \log_2 n + d \] logarithmic, e.g., binary search

Note: you can have similar solution with an arbitrary base \( b \)

A pass through input reduces problem size by half.

\[ T(n) = 2T(n/2) + cn \quad \text{for } n > 1 \]
\[ T(1) = d \]
Solution: \[ T(n) = cn \log_2 n + d n \] \( n \log_2 n \), e.g., mergesort
Example 1 – Solving Recurrence Relations Using the Solutions of Important Recurrence Types

One (constant) operation reduces problem size by one.

\[ T(n) = T(n - 1) + c \quad \text{for } n > 1 \quad T(1) = d \]

Solution: \[ T(n) = (n - 1)c + d \]

Example:

\[ T(n) = T(n - 1) + 2 \quad \text{for } n > 1, \quad T(1) = 2 \]

\[ c =? \quad \text{and} \quad d =? \]

\[ c = 2 \quad \text{and} \quad d = 2 \]

\[ \rightarrow T(n) = 2(n-1) + 2 = 2n \]
Example 2 – Solving Recurrence Relations Using the Solutions of Important Recurrence Types

A pass through input reduces problem size by one.

\[ T(n) = T(n - 1) + cn \quad \text{for } n > 1 \quad T(1) = d \]

Solution: \[ T(n) = \left[ \frac{n(n + 1)}{2} - 1 \right] c + d \]

Example:

\[ T(n) = T(n - 1) + 2n \quad \text{for } n > 0, \quad T(0) = 2 \]

\[ c = \text{?} \quad \text{and} \quad d = \text{?} \]

\[ c = 2 \quad \text{and} \quad d = 2 \rightarrow T(n) = \left[ \frac{n(n+1)}{2} - 1 \right] * 2 + 2 = n^2 + n \]

\[ \neq n^2 + n + 2 \]

What’s wrong? \[ T(1) = d \]
Example 2 – Solving Recurrence Relations Using the Solutions of Important Recurrence Types

A pass through input reduces problem size by one.

\[ T(n) = T(n-1) + cn \quad \text{for } n > 1 \quad T(1) = d \]

**Solution:** \[ T(n) = [n(n + 1)/2 - 1] c + d \]

Example:

\[ T(n) = T(n - 1) + 2n \quad \text{for } n > 0, \quad T(0) = 2 \]

\[ c =? \quad \text{and} \quad d =? \]

\[ T(1) = T(0) + 2 = 2 + 2 = 4 \rightarrow d = 4 \]

\[ c = 2 \quad \text{and} \quad d = 4 \rightarrow T(n) = \left\lceil \frac{n(n+1)}{2} - 1 \right\rceil \times 2 + 4 = n^2 + n + 2 \]
Example 3 – Solving Recurrence Relations Using the Solutions of Important Recurrence Types

One (constant) operation reduces problem size by half.

\[ T(n) = T(n/2) + c \quad \text{for } n > 1 \quad T(1) = d \]

Solution: \( T(n) = c \log_2 n + d \)

Example:

\[ T(n) = T(n/2) + 1 \quad \text{for } n > 1, \quad T(1) = 2 \]

\( c = ? \) and \( d = ? \)

\( c = 1 \) and \( d = 2 \) \( \rightarrow T(n) = \log_2 n + 2 \)
Example 4 – Solving Recurrence Relations Using the Solutions of Important Recurrence Types

A pass through input reduces problem size by half.

\[ T(n) = 2T(n/2) + cn \quad \text{for } n > 1 \quad T(1) = d \]

Solution: \[ T(n) = cn \log_2 n + d n \]

Example:

\[ T(n) = 2T(n/2) + 3n \quad \text{for } n > 1, \quad T(1) = 2 \]

\[ c =? \quad \text{and} \quad d =? \]

\[ c = 3 \quad \text{and} \quad d = 2 \rightarrow T(n) = 3n \log_2 n + 2n \]
Linear second-order recurrences with constant coefficients

\[ ax(n) + bx(n - 1) + cx(n - 2) = f(n) \quad a \neq 0 \]

Second-order term A function of \( n \)

\[ a, b, \text{ and } c \text{ are constant coefficients.} \]

\[ f(n) = 0 \quad \text{homogeneous} \]

\[ f(n) \neq 0 \quad \text{inhomogeneous} \]
Linear second-order recurrences with constant coefficients - Homogeneous case

Homogeneous case:

\[ ax(n) + bx(n-1) + cx(n-2) = 0 \quad a \neq 0 \]

Characteristic equation:

\[ ar^2 + br + c = 0 \]

Roots of the characteristic equation:

\[ r_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

Case 1: Two real number solutions

Case 2: One real number solution

Case 3: Two complex number solutions
Linear second-order recurrences with constant coefficients - Homogeneous case

Homogeneous case:

\[ ax(n) + bx(n-1) + cx(n-2) = 0 \quad a \neq 0 \]

Characteristic equation:

\[ ar^2 + br + c = 0 \]

Roots of the characteristic equation determine the general solution:

- **case1** \( x(n) = \alpha r_1^n + \beta r_2^n \quad r_1 \neq r_2 \quad r_1, r_2 \in \mathbb{R} \)
- **case2** \( x(n) = \alpha r^n + \beta nr^n \)
- **case3** \( x(n) = \gamma^n [\alpha \cos n\theta + \beta \sin n\theta] \)

\[ r_{1,2} = u \pm jv \quad \gamma = \sqrt{u^2 + v^2} \quad \theta = \arctan \frac{v}{u} \]

\( \alpha \) and \( \beta \) are constants for the general solution
Example - Homogeneous case

Homogeneous case:

\[ x(n) - 10x(n-1) + 25x(n-2) = 0 \]

Characteristic equation:

\[ r^2 - 10r + 25 = 0 \]

Roots of the characteristic equation determine the general solution:

\[ r = 5 \]

\[ x(n) = \alpha r^n + \beta n r^n \]

General solution

\[ x(n) = \alpha (5)^n + \beta n (5)^n \]

\( \alpha \) and \( \beta \) are arbitrary constants
Example - Homogeneous case

General solution:

\[ x(n) = \alpha (5)^n + \beta n (5)^n \]

How to get the particular solution?

Given the initial condition

\[ x(0) = 0 \quad x(1) = 5 \]

\[ x(0) = x(n = 0) = \alpha (5)^0 + \beta * 0 * (5)^0 = \alpha \quad \Rightarrow \quad \alpha = 0 \]

\[ x(1) = x(n = 1) = \alpha (5)^1 + \beta * 1 * (5)^1 = 5 \beta \quad \Rightarrow \quad \beta = 1 \]

\[ x(n) = n (5)^n \]
Example 2 – Computing Fibonacci Number

\[ F(n) = F(n - 1) + F(n - 2) \]

**Initial condition:** \( F(0) = 0 \quad F(1) = 1 \)

The Fibonacci sequence:

0, 1, 1, 2, 3, 5, 8, 13, 21, …

How to compute Fibonacci number?

1. Nonrecursive definition-based algorithm \( \Theta(n) \)
2. Recursive definition-based algorithm \( \Theta(n) \)

Can we give an explicit math function for \( F(n) \)?
Example 2 – Computing Fibonacci Number

\[ F(n) = F(n - 1) + F(n - 2) \]

\[ F(n) - F(n - 1) - F(n - 2) = 0 \]
Example 2 – Computing Fibonacci Number

\[ F(n) - F(n-1) - F(n-2) = 0 \]

Characteristic function: \( r^2 - r - 1 = 0 \)

Roots: \( r_{1,2} = \frac{1 \pm \sqrt{5}}{2} \)

case 1 \( x(n) = \alpha r_1^n + \beta r_2^n \quad r_1 \neq r_2 \quad r_1, r_2 \in \mathbb{R} \)

General solution: \( F(n) = \alpha \left( \frac{1 + \sqrt{5}}{2} \right)^n + \beta \left( \frac{1 - \sqrt{5}}{2} \right)^n \)
Example 2 – Computing Fibonacci Number

\[ F(n) - F(n-1) - F(n-2) = 0 \]

Characteristic function: \[ r^2 - r - 1 = 0 \]

General solution: \[ F(n) = \alpha \left( \frac{1 + \sqrt{5}}{2} \right)^n + \beta \left( \frac{1 - \sqrt{5}}{2} \right)^n \]

Particular solution:

Since \( F(0) = 0 \) and \( F(1) = 1 \)

\[ \alpha = \frac{1}{\sqrt{5}} \quad \text{and} \quad \beta = -\frac{1}{\sqrt{5}} \]

\[ F(n) = \frac{1}{\sqrt{5}} \left( \frac{1 + \sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left( \frac{1 - \sqrt{5}}{2} \right)^n \]
Inhomogeneous case:

\[ ax(n) + bx(n - 1) + cx(n - 2) = f(n) \quad a \neq 0 \]

Its general solution is the summation of one of its particular solution and the general solution of

\[ ax(n) + bx(n - 1) + cx(n - 2) = 0 \]

• Nontrivial problem for an arbitrary \( f(n) \)
• Can be solved for special \( f(n) \), e.g., a constant
Example

\[ x(n) - 10x(n-1) + 25x(n-2) = 16 \]

The homogeneous case: \[ x(n) - 10x(n-1) + 25x(n-2) = 0 \]

Step 1: find a particular solution of the inhomogeneous function

\[ x(n) = c \quad \Rightarrow \quad c = 1 \]

Step 2: find the general solution of the homogeneous function

\[ x(n) = \alpha(5)^n + \beta n(5)^n \]

The general solution of inhomogeneous function

\[ x(n) = \alpha(5)^n + \beta n(5)^n + 1 \]

The particular solution can be obtained given the initial condition!