Fundamental Algorithmic Techniques III

January 28, 2026

Outline

Divide & Conquer

Recurrence Relations

Master Theorem

Multiplying Square Matrices

$$\mathbf{C} = \mathbf{A} \cdot \mathbf{B},$$

n operations $\forall i, j$: $\qquad c_{ij} = \sum_{k=0}^{n} a_{ik} \cdot b_{kj}.$

Divide and conquer:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \quad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix},$$

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}. \tag{4.4}$$

Decomposing with 8 sub-operations: $T(n) = 8T(n/2) + \Theta(1)$,
so $\mathbf{T(n)} = \ (\mathbf{n^3})$ (master theorem $c = 3 = log_2(8)$).

Strassen Algorithm

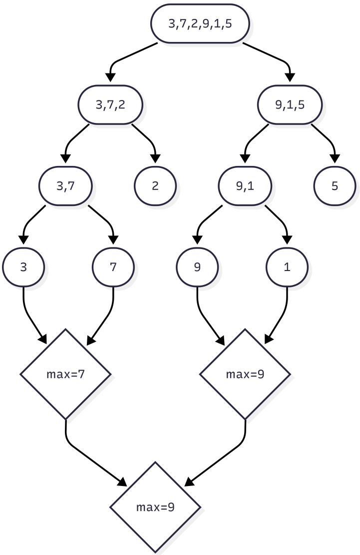
$T(n) = 7T(n/2) + \Theta(1)$, so $\mathbf{T(n)} = \ (\mathbf{n}^{\mathbf{2.81}})$

$$
\begin{aligned}
M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22})B_{11} \\
M_3 &= A_{11}(B_{12} - B_{22}) \\
M_4 &= A_{22}(B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12})B_{22} \\
M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22})(B_{21} + B_{22})
\end{aligned}
$$

Result Blocks:

$$
\begin{aligned}
C_{11} &= M_1 + M_4 - M_5 + M_7 \\
C_{12} &= M_3 + M_5 \\
C_{21} &= M_2 + M_4 \\
C_{22} &= M_1 - M_2 + M_3 + M_6
\end{aligned}
$$

Find Maximum: Divide and Conquer



Problem: Find the maximum element in an array of n numbers.

Approach:

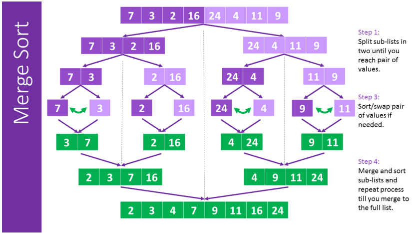
- **Divide**: Split into two halves
- **Conquer**: Recursively find max
- **Combine**: $\max(\text{left}, \text{right})$

Complexity:
$T(n) = 2T(n/2) + cn$
$\Longrightarrow \mathcal{O}(n)$

Sorting

Insertion sort: $T(n) = an^2 + bn + c,\ a,\ b\,c, \in \mathbb{N}$
So $T(N) = \mathcal{O}(n^2)$.

But can we do better?
Yes, actually $\mathcal{O}(nlog(n))$ with MergeSort and QuickSort.



Merge Sort

Recurrence Relation: Mathematical Description

Let $T(n)$ be a recurrence relation defined for $n \geq 1$ by:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where:

- $a \geq 1$ is the number of subproblems in the recursion,
- $b > 1$ is the factor by which the input size is reduced in each subproblem,
- $f(n)$ is the cost of dividing the problem and combining the results.

Recursive: Not always good!

Iterative vs Recursive Factorial: Complexity Comparison

	Iterative	**Recursive**
Time Complexity	$O(n)$	$O(n)$
Space Complexity	$O(1)$	$O(n)$
Stack Overflow?	No	Yes

```julia

function factorial_iter(n::Int)
    result = 1
    for i in 2:n
        result *= i
    end
    return result
end

function factorial_recu(n::Int)
    n <= 1 ? 1 : n * factorial(n - 1)
end
```

Factorial in Julia

Master Theorem

Asymptotic behavior of $T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$:
critical exponent: $c_{\text{crit}} = \log_b a$

1. **Case 1 (Subproblem Dominated):**
   If $f(n) = O(n^c)$ where $c < c_{\text{crit}}$, then:
   $$T(n) = \Theta(n^{\log_b a})$$
2. **Case 2 (Balanced):**
   If $f(n) = \Theta(n^{c_{\text{crit}}})$, then:
   $$T(n) = \Theta(n^{c_{\text{crit}}} \log n) = \Theta(n^{\log_b a} \log n)$$
3. **Case 3 (Work Dominated):**
   If $f(n) = \Omega(n^c)$ where $c > c_{\text{crit}}$, and if the **regularity condition** holds:
   $$af\left(\frac{n}{b}\right) \leq kf(n) \quad \text{for constant } k < 1 \text{ and all sufficiently large } n,$$
   then:
   $$T(n) = \Theta(f(n))$$

Master Theorem : Limitations & Examples

Limitations:

- $T(n)$ not monotone, e.g. $T(n) = sin(n)$
- $f(n)$ not polynomial, e.g. $f(n) = 2^n$
- a not a constant, e.g. $a = 2n$

Examples: verify!

1. $T(n) = 4T(\frac{n}{2}) + n$,
   $\Rightarrow \text{Case 1}$, Subproblem dominated, $T(n) = \Theta(n^2)$
2. $T(n) = 2T(\frac{n}{2}) + n$,
   $\Rightarrow \text{Case 2}$, Balanced, $T(n) = \Theta(nlog(n))$
3. $T(n) = 3T(\frac{n}{2}) + n^2$,
   $\Rightarrow \text{Case 3}$, Work dominated, $T(n) = \Theta(n^2)$
4. $T(n) = 2T(2n) + nlog(n)$,
   $\Rightarrow$ trivially not applicable!
5. $T(n) = T(n-1) + 1$,
   $\Rightarrow$ Not applicable! $n - 1 \neq n/b$, actually $T(n) = \Theta(n!)$

Master Theorem : Proof with Tree Approach

$$T(n) = aT(\frac{n}{b}) + f(n)$$

- level 0: work $f(n)$, a subproblems of size n/b
- ...
- level i: work $a^i \cdot f(n/b^i)$, a^i subproblems of size n/b^i

stops when $i = log_b(n)$, and using $a \cdot log_b(n) = n \cdot log_b(a)$:

$$T(n) = \sum_{i=1}^{\log_b n} a^i \cdot f(n/b^i) + \Theta(n^{\log_b a}),$$

with work and leaf decomposition contributions.

...Study each 3 cases separately for the proof...