CS3383 Lecture 1.1: The Master Theorem with applications

David Bremner

January 16, 2024

CS3383 Lecture 1.1: The Master Theorem with applications

David Bremner

January 16, 2024

Divide and Conquer Continued

- The Master Theorem
- Matrix Multiplication

Generic divide and conquer algorithm

function SOLVE(P)
    if $|P|$ is small **then**
        SolveDirectly(P)
    else
        $P_1 \ldots P_k =$ Partition(P)
        for $i = 1 \ldots k$ **do**
            $S_i =$ Solve(P_i)
        end for
        Combine($S_1 \ldots S_k$)
    end if
end function

- How many recursive calls?
- how big are subproblems?
- cost to combine?

/Subtype /Text/F 1/T
(Video)/Contents
(video/11.2-generi[redacted]

Common recursive structure

A typical Divide and Conquer algorithm

- b the branch factor, number of recursive calls
- s the split, how many parts is input split
- d the degree of the polynomial for overhead

/Subtype /Text/F 1/T (Video)/Contents (video/11.3-parameters.mkv)

The Master Theorem

If $\exists$ constants $b > 0$, $s > 1$ and $d \geq 0$ such that
$T(n) = b \cdot T(\lceil \frac{n}{s} \rceil) + \Theta(n^d)$, then

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } d > log_s b \quad \text{(equiv. to } b < s^d\text{)} \\ \Theta(n^d \log n) & \text{if } d = log_s b \quad \text{(equiv. to } b = s^d\text{)} \\ \Theta(n^{\log_s b}) & \text{if } d < log_s b \quad \text{(equiv. to } b > s^d\text{)} \end{cases}$$

/Subtype /Text/F 1/T (Video)/Contents (video/11.4-master [redacted]

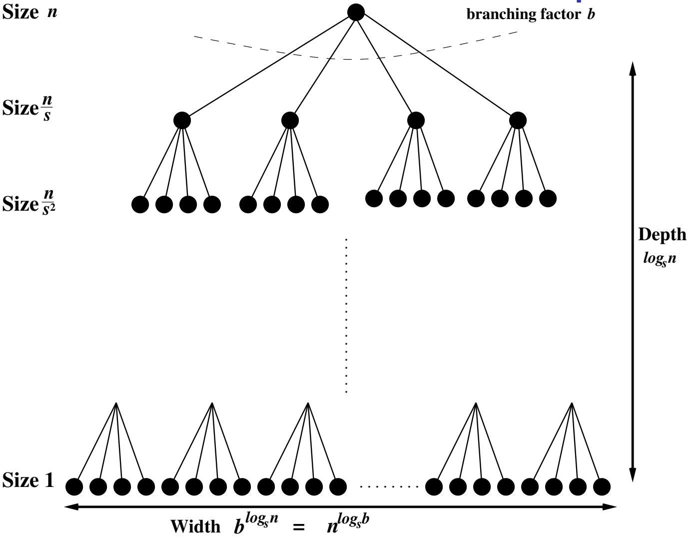
The Master Theorem

If $\exists$ constants $b > 0$, $s > 1$ and $d \geq 0$ such that
$T(n) = b \cdot T(\lceil \frac{n}{s} \rceil) + \Theta(n^d)$, then

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } d > log_s b \quad (\text{equiv. to } b < s^d) \\ \Theta(n^d \log n) & \text{if } d = log_s b \quad (\text{equiv. to } b = s^d) \\ \Theta(n^{\log_s b}) & \text{if } d < log_s b \quad (\text{equiv. to } b > s^d) \end{cases}$$

/Subtype /Text/F 1/T (Video)/Contents
(video/11.5-master-2.mkv)

Proof of Master theorem, in pictures



/Subtype /Text[redacted]
1/T (Video)/Contents (video/11.6-picture.mkv)

Sanity check: Merge sort

Master Theorem

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } d > log_s b \\ \Theta(n^d \log n) & \text{if } d = log_s b \\ \Theta(n^{\log_s b}) & \text{if } d < log_s b \end{cases}$$

Merge Sort

- $T(n) = bT(n/s) + \theta(n^d)$
- b how many recursive calls?
- s what is the the split (denominator of size)
- d degree

Sanity check: Merge sort

Master Theorem

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } d > log_s b \\ \Theta(n^d \log n) & \text{if } d = log_s b \\ \Theta(n^{\log_s b}) & \text{if } d < log_s b \end{cases}$$

Merge Sort

- $T(n) = bT(n/s) + \theta(n^d)$
- b how many recursive calls?
- s what is the the split (denominator of size)
- d degree

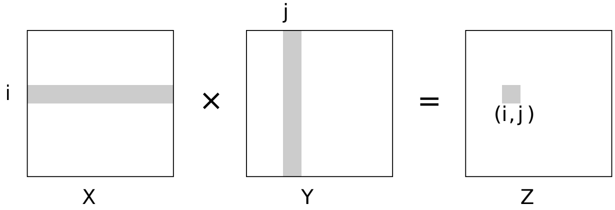
/Sub-
type
/Tex-
t/F
1/T
(Video)/C
tents
(video/11.

Matrix Multiplication

The product of two $n \times n$ matrices x and y is a third $n \times n$ matrix $Z = XY$, with

$$Z_{ij} = \sum_{k=1}^{n} X_{ik} Y_{kj}$$

where Z_{ij} is the entry in row i and column j of matrix Z.



Calculating Z directly using this formula takes $\Theta(n^3)$ time.
/Subtype /Text/F 1/T (Video)/Contents

Matrix Multiplication: Blocks

- decompose the input matrices into four blocks each (cutting the dimension n in half):

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \qquad Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

$$XY = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$
$$= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

/Subtype /Text/F 1/T (Video)/Contents (video/11.9-blocks [redacted]

The naive approach fails again

- 8 recursive calls
- subinstances of dimension $\frac{n}{2}$
- cn^2 time to add results

$$T(n) = 8 \cdot T\left(\frac{n}{2}\right) + cn^2$$

- From Master Theorem $T(n) \in \Theta(n^3)$
- (not technically "cubic algorithm", input size n^2.)

/Subtype /Text/F 1/T (Video)/Contents (video/11.10-8bad.[redacted]

Matrix Multiplication: Strassen Decomposition

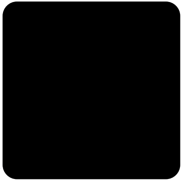
Strassen found a decomposition that re-uses subproblems, getting us from 8 to 7

$$XY = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

where

$$\begin{aligned} P_1 &= A(F - H) & P_5 &= (A + D)(E + H) \\ P_2 &= (A + B)H & P_6 &= (B - D)(G + H) \\ P_3 &= (C + D)E & P_7 &= (A - C)(E + F) \\ P_4 &= D(G - E) \end{aligned}$$

/Subtype /Text/F 1/T (Video)/Contents (video/11.11-strassen.mkv)



Matrix Multiplication: Strassen Decomposition

Strassen found a decomposition that re-uses subproblems, getting us from 8 to 7

$$XY = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

where

$$\begin{aligned} P_1 &= A(F - H) & P_5 &= (A + D)(E + H) \\ P_2 &= (A + B)H & P_6 &= (B - D)(G + H) \\ P_3 &= (C + D)E & P_7 &= (A - C)(E + F) \\ P_4 &= D(G - E) \end{aligned}$$

/Subtype /Text/F 1/T (Video)/Contents (video/11.12-strassen-2.mkv)