The master method is a process of runtime analysis for algorithms using recursive calls, similar to algorithms using divide and conquer, to arrive at solutions.

## ANALYZING RUNTIME OF RECURSIVE CALLS

### EXAMPLE USING KARATSUBA MULTIPLICATION ALGORITHM

Karatsuba algorithm reduces multiplication of two n-digit numbers to at most $n^{\log_2 3} \approx n^{1.58}$ single-digit multiplications compared to the classical "grade-school" algorithm which requires $n^2$ single-digit products.

LET $x = 5678$, $y = 1234$

1) Break operands x and y into half size

   LET $a = 56$, $b = 78$, $c = 12$, $d = 34$ (Multiplying a or c by 100 gives the original magnitude)

2) Compute $a \cdot c = 56(12) = 672$, $z_2$

3) Compute $b \cdot d = 78(34) = 2652$, $z_0$

4) Compute $(a+b)(c+d) = (56+78)(12+34) = 134(46) = 6164$

5) Compute results of step 4 - step 3 - step 2 $= 2840$, $z_1$

6) Combine the results of steps 2, 3, and 5 with appropriate zero padding

| | |
|---|---|
| $z_2\ (100)^2$ | $672(10000)$ |
| $z_1\ (100)^1$ | $2840(100)$ |
| $+\ z_0\ (100)^0$ | $2652$ |
| RESULT | $7006652$ |

LET $T(n)$ be the maximum number of operations this algorithm needs to multiply two n-digit numbers.

RECURRENCE - Expresses $T(n)$ in terms of running time of recursive calls. composed of two parts, the base case and the general case

BASE CASE - Occurs when there is no further recursion. For example, the base case of the Karatsuba algorithm would occur when the original operands get broken down $n/2$ each time until single digit multiplication can be achieved. $T(1) \leq$ a constant

# ANALYZING RUNTIME OF RECURSIVE CALLS (con't)

## EXAMPLE USING KARATSUBA MULTIPLICATION ALGORITHM (con't)

GENERAL CASE - Occurs when not in the base case and making the recursive calls. Analyzed by recording work done by recursive call and work outside of recursive calls.

$$\text{For all } \underset{\text{BASE CASE}}{n > 1} : T(n) \leq \underbrace{3T(n/2)}_{\substack{\text{WORK DONE BY} \\ \text{RECURSIVE} \\ \text{CALLS} \\ \text{STEP 1-5}}} + \underbrace{O(n)}_{\substack{\text{WORK DONE} \\ \text{TO RETURN} \\ \text{FUNCTION} \\ \text{STEP 6}}}$$

ASSUMPTION: All subproblems analyzed with the master method have equal size at that level. That is, breaking a 4-length problem up will result in two 2-lengths subproblems.

# FORMAT OF MASTER METHOD RECURRENCES

Base case: $T(n) \leq$ a constant for all sufficiently small $n$. That is through recursive calls we will eventually reach a subproblem size of $n$ that is sufficiently simple and can be solved in constant time.

GENERAL CASE: For all larger $n$, $T(n) \leq \underbrace{aT(n/b)}_{\text{RECURSION}} + \underbrace{O(n^d)}_{\text{COMBINE}}$

where, $a$ = number of recursive calls (SUBPROBLEMS) made, $a \geq 1$
$b$ = input size dividing factor, $b > 1$ (breaks problem into smaller subproblems)
$d$ = exponent in running time of combine step, $d \geq \emptyset$

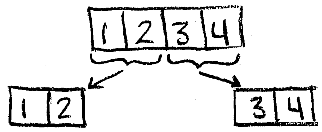
and $a, b$, and $d$ are independent of $n$

$$T(n) = \begin{cases} O(n^d \lg n), & \text{if } a = b^d \text{, Log base does not matter (ONLY DIFFERS BY CONSTANT FACTOR)} \\ O(n^d), & \text{if } a < b^d \text{, work dominated by combine step} \\ O(n^{\log_b a}), & \text{if } a > b^d \text{, Log base does matter (CONSTANT FACTOR AFFECTS RUNTIME LINEAR VS QUADRATIC TIME)} \end{cases}$$

# Analyzing Runtime of Recursive Calls (con't)

## Merge Sort

Find a: For each problem merge sort makes 2 subproblems




$a = 2$

Find b: Merge sort divides each input problem evenly in half.

$b = 2$

Find d: The combine step has to look at each input value once during the merge.

$d = 1$ (linear time)

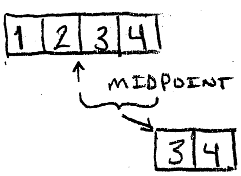
Which case of the Master Method?

$a = 2,\ b^d = 2^1 = 2 \rightarrow a = b^d \rightarrow T(n) = O(n^d \lg n) \rightarrow$

$O(n^1 \lg n)$ ✓

## Binary Search

Find a: For each problem binary search eliminates half the problem set to make 1 smaller subproblem




Find 3

midpoint $= 2 < 3$, solution cannot be located at midpoint or below

$a = 1$

Find b: Binary search eliminates half the problem set until reaching the base case.

$b = 2$

Find d: Only one comparison to see how value relates to midpoint which is always constant no matter input size.

$d = \emptyset$

Which case of the Master Method?

$a = 1,\ b^d = 2^{\emptyset} = 1 \rightarrow a = b^d \rightarrow T(n) = O(n^d \lg n) \rightarrow$

$O(n^{\emptyset} \lg n) \rightarrow O(\lg n)$

# ANALYZING RUNTIME OF RECURSIVE CALLS (con't)

## NAIVE RECURSION ON INTEGER MULTIPLICATION

Recall from Karatsuba algorithm how the operands $x = 5678$ and $y = 1234$ were split into 4 subproblems a, b, c, and d each of half the number of digits as the original operand. This is how the naive recursion begins.

FIND a: From above the original problem is split into 4 subproblems.

$$a = 4$$

FIND b: From above, each subproblem has half the original size as its respective parent. x has 4 digits, a and b both have 2 digits.

$$b = 2$$

FIND c: There is an addition and single-digit multiplication for each base case which at the end of the recursive calls are 8 multiplications and additions and there are 8 digits in the original operands, x and y, leading to a linear runtime.

$$d = 1$$

WHICH CASE OF THE MASTER METHOD?

$$a = 4,\ b^d = 2^1 = 2 \rightarrow a > b^d \rightarrow T(n) = O(n^{\log_2 4}) \rightarrow$$

$$O(n^2)$$

(con't on next sheet)

# ANALYZING RUNTIME OF RECURSIVE CALLS (con't)

## KARATSUBA ALGORITHM

FIND a: Due to using Gauss' trick for integer multiplication instead of 4 subproblems there are 3 subproblems created with each recursion call.

$a = 3$

FIND b: Similar to the naive attempt at recursion.

$b = 2$

FIND d: The same number of single-digit multiplication and addition is needed to combine the results as the number of digits in the original operands, so the combine step runs in linear time

$d = 1$

WHICH CASE OF THE MASTER METHOD?

$$a = 3,\ b^d = 2^1 \rightarrow a > b^d \rightarrow T(n) = O(n^{\log_2 3}) \rightarrow$$

$O(n^{1.58})$ BETTER THAN "GRADE SCHOOL" $O(n^2)$

## STRASSEN'S MATRIX MULTIPLICATION ALGORITHM

From previous work on page 10 of the Divide and Conquer Model notes.

FIND a: There are 7 recursive calls made instead of 8. $a = 7$

FIND b: Each matrix is divided in each dimension by 2. $b = 2$

FIND d: Merging the seven products is linear in each dimension of the matrix so $n \times n$ or $n^2$. $d = 2$

WHICH CASE OF THE MASTER METHOD?

$$a = 7,\ b^d = 2^2 = 4 \rightarrow a > b^d \rightarrow T(n) = O(n^{\log_2 7}) \rightarrow$$

$O(n^{2.81})$

(con't on next sheet)

# ANALYZING RUNTIME OF RECURSIVE CALLS (con't)

## FICTITIOUS RECURRENCE

Assume an algorithm similar to merge sort, but the combine step has quadratic runtime, $O(n^2)$

FIND a: $a = 2$

FIND b: $b = 2$

FIND d: $d = 2$

WHICH CASE OF THE MASTER METHOD?

$a = 2,\ b^d = 2^2 = 4 \rightarrow a < b^d \rightarrow T(n) = O(n^d) \rightarrow O(n^2)$

Work dominated by combine step.