

# Logarithms

(1)

$$\cdot \log_a(a^x) = x$$

$$\cdot a^{\log_a x} = x$$

$$x = a^y \Rightarrow a^{\log_a x} = a^{\log_a a^y} = a^y = x$$

$$\cdot a^{x+y} = a^x \cdot a^y \qquad \log_a(x \cdot y) = \log_a x + \log_a y$$

$$a^{x \cdot y} = (a^x)^y$$

$$\log_a(x^y) = y \cdot \log_a x$$

$$\cdot a^{\log_b x} = x^{\log_b a} \quad (\text{Swap Rule})$$

$$b^{\log_b a^{\log_b x}}$$

$$= b^{\log_b x \cdot \log_b a}$$

$$= (b^{\log_b x})^{\log_b a} = x^{\log_b a}$$

$$\cdot \log_a x = \log_b (b^{\log_a x}) = \log_b (x^{\log_a b})$$

$$= \log_a b \cdot \log_b x \quad (\text{Transformation Rule})$$

Example

$$\log_2 x = \log_2 10 \cdot \log_{10} x$$

$$\approx \frac{10}{3} \cdot \log_{10} x$$

$$\Rightarrow \log_a x = \Theta(\log_b x) = \Theta(\lg x)$$

(2)

MergeSort( $l, r$ )

% data in array A

if  $l < r$  then

$m := (l+r)/2$

% integer division

MergeSort( $l, m$ )

MergeSort( $m+1, r$ )

Merge( $l, m, r$ )

Merge( $l, m, r$ )

Copy  $A[l, m]$  into new array B

Copy  $A[m+1, r]$  into new array C

while B and C not empty

find smallest among first elements of B and C

put it into  $A[l, r]$

Merge ( $A, l, m, r$ )

array  $B, C$ ;

for  $i := 1$  to  $m - l + 1$  do

$B[i] := A[i]$ ;

for  $i := 1$  to  $r - m$  do

$C[i] := A[m + i]$ ;

$i := 1$ ;  $j := 1$ ;  $k := 1$ ;

while  $j \leq B.length$  and  $k \leq C.length$

if  $B[j] < C[k]$

then  $A[i] := B[j]$ ;

$j++$ ;

else  $A[i] := C[k]$ ;

$k++$ ;

$i++$ ;

if  $j > B.length$

then while  $k \leq C.length$  do

$A[i] := C[k]$ ;  $i++$ ;  $k++$ ;

else while  $j \leq B.length$  do

$A[i] := B[j]$ ;  $i++$ ;  $j++$ ;

3

# Running Time of Merge Sort

$T(n)$  worst case running time for array of length  $n$

recurrence relation

$$T(n) = 2 T(n/2) + \Theta(n), \quad n > 1$$

(under  $2 T(n/2)$ )  
running the alg on 2 halves

$$T(1) = D = \Theta(1)$$

What is  $T(n)$ ? In  $\Theta$ -notation?

Suppose  $n = 2^k \Rightarrow k = \lg n$

$$\begin{aligned}
 T(n) &= 2 T(n/2) + \Theta(n) \\
 &= 2 (2 T(n/2 \cdot 2) + \Theta(n/2)) + \Theta(n) \\
 &= 2 \cdot 2 T(n/2 \cdot 2) + 2 \Theta(n/2) + \Theta(n) \\
 &= 2^2 T(n/2^2) + 2 \Theta(n) \\
 &= 2^2 (2 T(n/2^3) + \Theta(n/2^2)) + 2 \Theta(n) \\
 &= 2^3 T(n/2^3) + \Theta(n) + 2 \Theta(n) \\
 &\dots \\
 &= 2^k T(n/2^k) + k \Theta(n) \\
 &= n \Theta(1) + \lg n \cdot \Theta(n) = \Theta(n \cdot \lg n)
 \end{aligned}$$

Substitution Method

(5)

## Binary Search

Input: sorted array  $A$ , element  $x$  to be found in  $A$

1. Divide: compare  $x$  with middle element in  $A$
2. Conquer: recurse on one subarray
3. Combine: return from call (no work)

Recurrence for Running Time

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 1 \cdot T(n/2) + \Theta(1) & \text{if } n > 1 \end{cases}$$

## Substitution Method

$$n = 2^k$$

$$T(n) = T(n/2) + \Theta(1)$$

$$= T(n/2 \cdot 2) + \Theta(1) + \Theta(1)$$

$$= T(n/2^k) + k \Theta(1)$$

$$= \Theta(1) + k \cdot \Theta(1)$$

$$= \Theta(1) + \lg n \cdot \Theta(1) = \Theta(\lg n)$$

6

# Powering a number

input: number  $x$ , integer  $n > 0$   
Output:  $x^n$

In cryptography (RSA), powering modulo some  $k$   
is basic operation,

i.e.,  $x^n \text{ mod } k$       ( $x \cdot y \text{ mod } k$   
=  $(x \text{ mod } k) \cdot (y \text{ mod } k) \text{ mod } k$ )

⇒ multiplication mod  $k$  is a special operation

$5 \cdot 8 \text{ mod } 3 = 2 \cdot 2 \text{ mod } 3$
"                          "                          "
$40 \text{ mod } 3$ $4 \text{ mod } 3$
"                          "
1                              1

```
pow(x, n)
  p := 1
  for i := 1 to n do
    p := p * x
  return p
```

Running time  $\theta(n)$

Ideas:  $n = 2m \Rightarrow x^n = x^m \cdot x^m$

$n = 2m+1 \Rightarrow x^n = x^m \cdot x^m \cdot x$

power(x, n)

if  $n = 1$  then return  $x$

else

$p := \text{power}(x, n \text{ div } 2)$

combine { if  $n \bmod 2 = 0$   
then return  $p * p$   
else return  $p * p * x$

integer division

divide

Running Time

$$T(n) = \begin{cases} \theta(1) & \text{if } n = 1 \\ T(n/2) + \theta(1) & \end{cases}$$

$$\Rightarrow T(n) = \theta(\lg n)$$

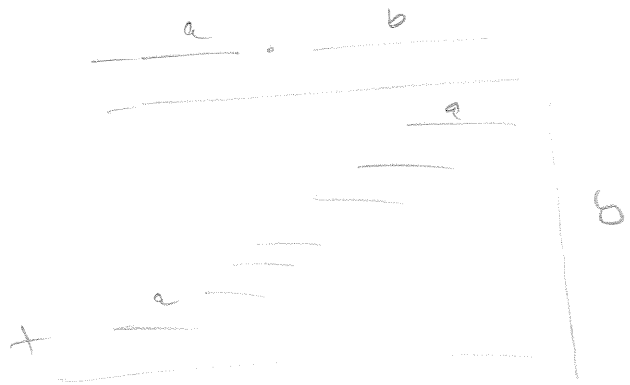
(3)

## Multiplication of binary numbers

$$a = \underbrace{110 \dots 1}_{u \text{ bit}}$$

$$b = \underbrace{01 \dots 1}_{u \text{ bit}}$$

Cost of  $a * b = u^2$  digit multiplications



Example

$$23 * 14 = (2 * 10 + 3) * (1 * 10 + 4)$$

$$= (2 * 1) * 10^2 + \underbrace{(3 * 1 + 2 * 4)} * 10^1 + (3 * 4)$$

Save one multiplication

$$(3 * 1 + 2 * 4) = (2 + 3) * (1 + 4) - 2 * 3 - 3 * 4$$



(9)

In general

$$a = a_1 10^{u/2} + a_0$$

$$b = b_1 10^{u/2} + b_0$$

$$\Rightarrow a \cdot b = (a_1 \cdot b_1) 10^u + \underline{(a_1 \cdot b_0 + a_0 \cdot b_1)} 10^{u/2} + a_0 b_0$$

Trick:  $a_1 \cdot b_0 + a_0 \cdot b_1$

$$= (a_1 + a_0) \cdot (b_1 + b_0) - \underbrace{(a_1 \cdot b_1)}_{\checkmark} - \underbrace{(a_0 \cdot b_0)}_{\checkmark}$$

1 multiplication instead of 2

$T(u) := \underline{\# \text{ digit multiplications}}$  for input  $u$  digits

$$T(u) = \begin{cases} 1 & u = 1 \\ 3 T(u/2) + \Theta(u) & u > 1 \end{cases}$$