CS 350 Algorithms and Complexity

Winter 2019

Lecture 10: Divide & Conquer — Trees and Multiplication

Andrew P. Black

Department of Computer Science

Portland State University

What is Divide-and-Conquer?

Solves a problem instance of size n by:

1. dividing it into b smaller instances, of size ~ n/b
2. solving some or all of them (in general, solving a of them), using the same algorithm recursively.
3. combining the solutions to the a smaller problems to get the solution to the original problem.

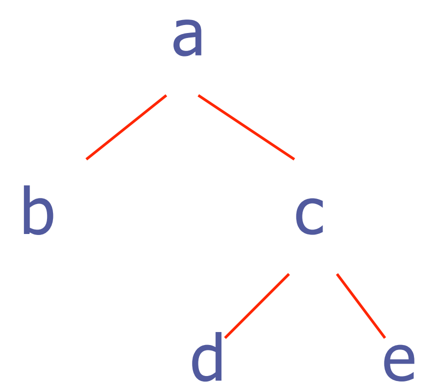
Binary Trees

- The perfect data structure for divide-into-two and conquer.

Binary Trees

- Ex. 1: Classic traversals
    - (preorder, inorder, postorder)
- Algorithm Inorder(T)

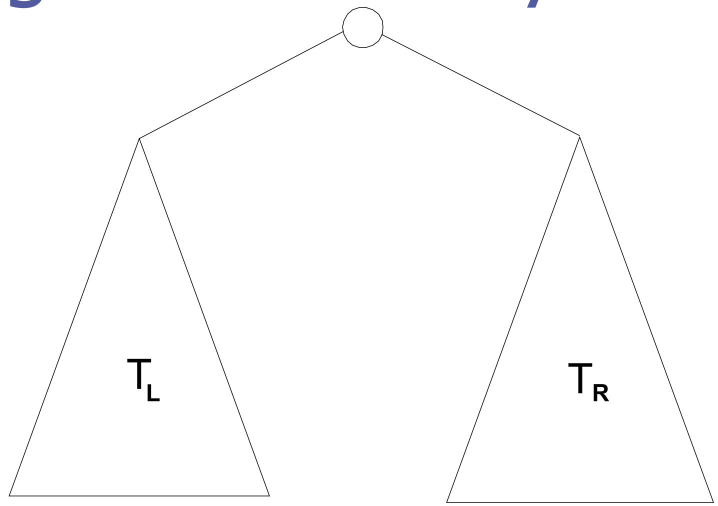
    if $T \neq \varnothing$ then
        Inorder(T_L)
        print(root of T)
        Inorder(T_R)



- Efficiency: $\Theta(n)$
- Could it be better? Worse?

Binary Trees

- Ex. 2: Height of a Binary Tree



$h(T) = \max\{h(T_L), h(T_R)\} + 1$

$\quad$ if $T \neq \varnothing$ and $h(\varnothing) = -1$

Efficiency: $\Theta(n)$

Summing Numbers

- Compute the sum of this array

0	1	2	3	4	5	6	7	8	9	10	11	12
15	3	19	0	21	78	46	94	97	49	57	89	91

- How can treating this as a binary tree help us?

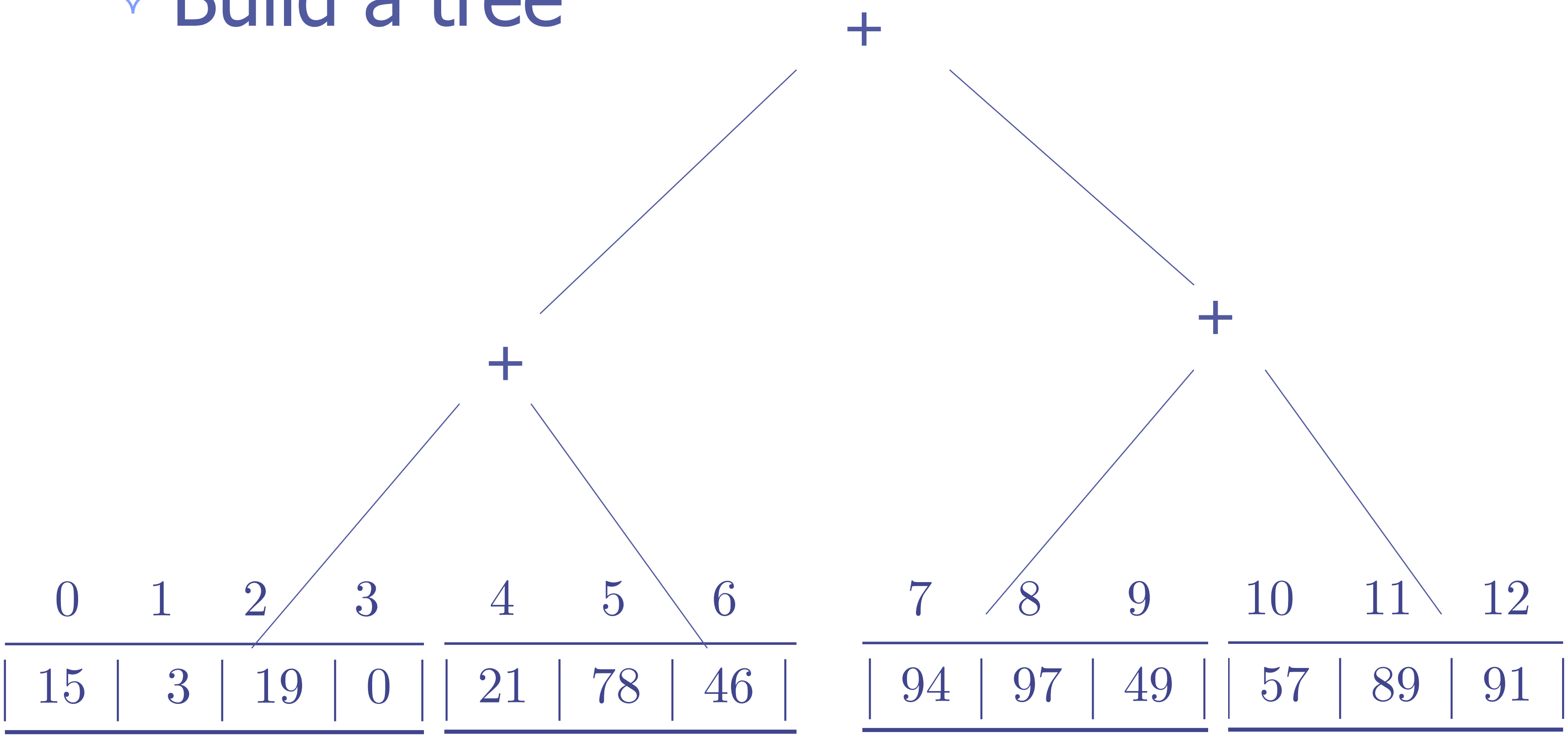
- Build a tree

+

0	1	2	3	4	5	6
15	3	19	0	21	78	46

7	8	9	10	11	12
94	97	49	57	89	91

- Build a tree



- When does this win?

Product of Numbers

- Compute the product of this array

0	1	2	3	4	5	6	7	8	9	10	11	12
15	3	19	0	21	78	46	94	97	49	57	89	91

- How can treating this as a binary tree help us?
- What's different with product, compared to sum?

Problem — Levitin §5.2 Q2

2. The following algorithm seeks to compute the number of leaves in a binary tree.

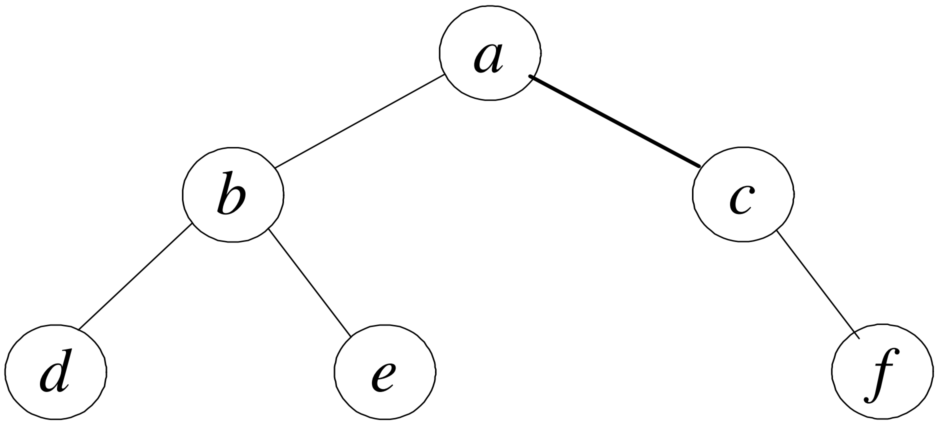
   Algorithm *LeafCounter*(T)
   //Computes recursively the number of leaves in a binary tree
   //Input: A binary tree T
   //Output: The number of leaves in T
   if $T = \varnothing$ **return** 0
   else return $LeafCounter(T_L) + LeafCounter(T_R)$

   Is this algorithm correct? If it is, prove it; if it is not, make an appropriate correction.

Hint: try it on a tree with one node

Problem

- Traverse this tree in pre-order:



A. $a\ b\ d\ e\ c\ f$

B. $d\ e\ b\ f\ c\ a$

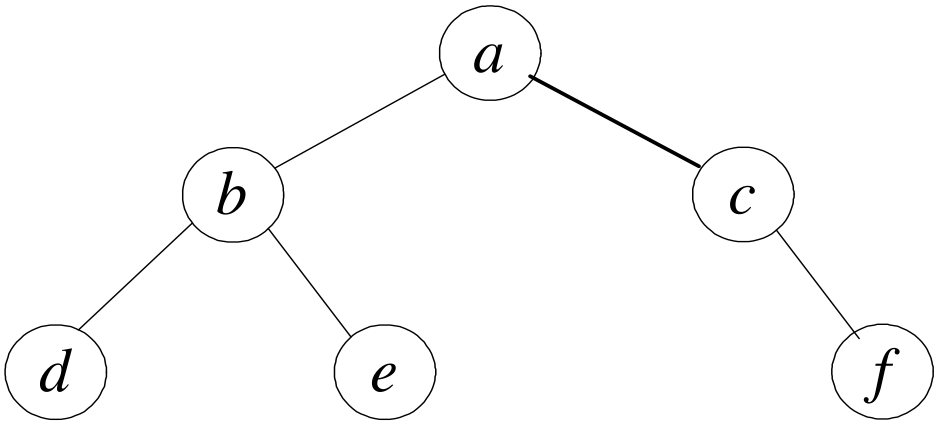
C. $a\ b\ c\ d\ e\ f$

D. $d\ b\ e\ a\ c\ f$

E. $d\ e\ f\ b\ c\ a$

Problem

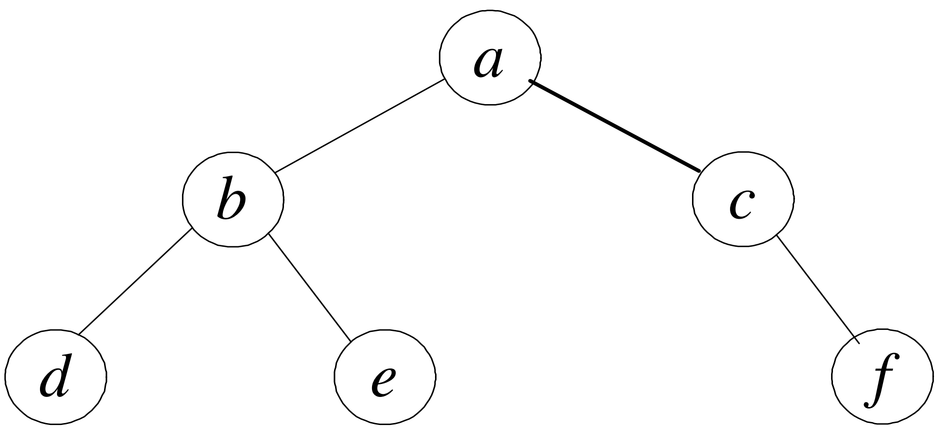
- Traverse this tree inorder:



A. $a\ b\ d\ e\ c\ f$
B. $d\ e\ b\ f\ c\ a$
C. $a\ b\ c\ d\ e\ f$
D. $d\ b\ e\ a\ c\ f$
E. $d\ e\ f\ b\ c\ a$

Problem

- Traverse this tree in post-order:



A. $a\ b\ d\ e\ c\ f$
B. $d\ e\ b\ f\ c\ a$
C. $a\ b\ c\ d\ e\ f$
D. $d\ b\ e\ a\ c\ f$
E. $d\ e\ f\ b\ c\ a$

Karatsuba Multiplication

- The "grade school algorithm" for multiplying n-digit numbers uses $O(n^2)$ multiplications
- Karatsuba's algorithm uses $O(n^{\lg 3}) \cong O(n^{1.585})$ multiplications (and exactly $n^{\lg 3}$ when n is a power of 2)
- What's the approximate ratio of multiplication counts when $n = 2^9 = 512$?

A. 10 B. 13 C. 17 D. 20 E. 50

Basic idea

Let x and y be represented as n-digit strings in some base B. For any positive integer m less than n, one can write the two given numbers as:

$$x = x_1 B^m + x_0 \qquad\qquad y = y_1 B^m + y_0,$$

where x_0 and y_0 are less than B^m. The product is then

$$xy = (x_1 B^m + x_0)(y_1 B^m + y_0) = z_2 B^{2m} + z_1 B^m + z_0$$

where

$$z_2 = x_1 y_1 \qquad z_1 = x_1 y_0 + x_0 y_1 \qquad z_0 = x_0 y_0.$$

These formulae require four multiplications. Karatsuba observed that xy can be computed in only three multiplications, at the cost of a few extra additions.

$$z_1 = (x_1 + x_0)(y_1 + y_0) - z_2 - z_0$$

which holds because

$$z_1 = x_1 y_0 + x_0 y_1$$

$$z_1 = (x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0.$$

Practice

- Multiply 41×20:

  $(40 + 1) \times (20 + 0) = (4\times2)_{\times 100} +$
  $(1\times2 + 4\times0)_{\times 10} +$
  (1×0)

  but $(1\times2 + 4\times0) =$
  $(4+1)\times(2+0) - (4\times2) - (1\times0)$

- Your Turn! Multiply 53×67:

  $53 \times 67 = \quad (\ \ \times\ \ )_{\times 100} +$
  $(\ \ \times\ \ +\ \ \times\ \ )_{\times 10} +$
  $(\ \ \times\ \ )$

- Your Turn! Multiply 53×67:

$$53 \times 67 = (5\times6)_{\times100} + (5\times7 + 3\times6)_{\times10} + (3\times7)$$

$$= 30_{\times100} + (5\times7 + 3\times6)_{\times10} + 21$$

1. $5\times6=30$
2. $3\times7=21$
3. $8\times13=104$

What is $(5\times7 + 3\times6)$? **Don't** spend two multiplications to find out!

A. 35 B. 63 C. 53 D. 74

You put the pieces together:

4153×2067

$$= (41\times20)_{\times10000}$$
$$+ [(41+53)(20+67) - 41\times20 - 53\times67]_{\times100}$$
$$+ (53\times67)$$

Answer:

A. 858471 B. 8485251 C. 8584251

D. None of the above

Strassen's Algorithm

- Three big ideas:
  1. You can split a matrix into blocks and operate on the resulting matrix of blocks like you would on a matrix of numbers.
  2. The blocks necessary to express the result of 2x2 block matrix multiplication have enough common factors to allow computing them in fewer multiplications than the original formula implies.
  3. Recursion.

Apply Strassen's algorithm to compute

$$C = \begin{array}{c} A \\ \left[\begin{array}{cc|cc} 1 & 0 & 2 & 1 \\ 4 & 1 & 1 & 0 \\ \hline 0 & 1 & 3 & 0 \\ 5 & 0 & 2 & 1 \end{array}\right] \end{array} * \begin{array}{c} B \\ \left[\begin{array}{cc|cc} 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 4 \\ \hline 2 & 0 & 1 & 1 \\ 1 & 3 & 5 & 0 \end{array}\right] \end{array}$$

exiting the recursion when $n = 2$, i.e., computing the products of 2-by-2 matrices by the brute-force algorithm.

$$C = \left[\begin{array}{c|c} C_{00} & C_{01} \\ \hline C_{10} & C_{11} \end{array}\right] = \left[\begin{array}{c|c} A_{00} & A_{01} \\ \hline A_{10} & A_{11} \end{array}\right] \left[\begin{array}{c|c} B_{00} & B_{01} \\ \hline B_{10} & B_{11} \end{array}\right]$$

where

$$A_{00} = \begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix}, \quad A_{01} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}, \quad A_{10} = \begin{bmatrix} 0 & 1 \\ 5 & 0 \end{bmatrix}, \quad A_{11} = \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix},$$

$$B_{00} = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix}, \quad B_{01} = \begin{bmatrix} 0 & 1 \\ 0 & 4 \end{bmatrix}, \quad B_{10} = \begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix}, \quad B_{11} = \begin{bmatrix} 1 & 1 \\ 5 & 0 \end{bmatrix}.$$

$$
\begin{aligned}
M_1 &= (A_{00}+A_{11})(B_{00}+B_{11}) = \begin{bmatrix} 4 & 0 \\ 6 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 7 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 8 \\ 20 & 14 \end{bmatrix}, \\
M_2 &= (A_{10}+A_{11})B_{00} = \begin{bmatrix} 3 & 1 \\ 7 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix}, \\
M_3 &= A_{00}(B_{01}-B_{11}) = \begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ -5 & 4 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ -9 & 4 \end{bmatrix}, \\
M_4 &= A_{11}(B_{10}-B_{00}) = \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 6 & -3 \\ 3 & 0 \end{bmatrix}, \\
M_5 &= (A_{00}+A_{01})B_{11} = \begin{bmatrix} 3 & 1 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 5 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix}, \\
M_6 &= (A_{10}-A_{00})(B_{00}+B_{01}) = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 2 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ -2 & -3 \end{bmatrix}, \\
M_7 &= (A_{01}-A_{11})(B_{10}+B_{11}) = \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 6 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ -9 & -4 \end{bmatrix}.
\end{aligned}
$$

$$
\begin{aligned}
C_{00} &= M_1 + M_4 - M_5 + M_7 \\
&= \begin{bmatrix} 4 & 8 \\ 20 & 14 \end{bmatrix} + \begin{bmatrix} 6 & -3 \\ 3 & 0 \end{bmatrix} - \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix} + \begin{bmatrix} 3 & 2 \\ -9 & -4 \end{bmatrix} = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}, \\
C_{01} &= M_3 + M_5 \\
&= \begin{bmatrix} -1 & 0 \\ -9 & 4 \end{bmatrix} + \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix} = \begin{bmatrix} 7 & 3 \\ 1 & 9 \end{bmatrix}, \\
C_{10} &= M_2 + M_4 \\
&= \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix} + \begin{bmatrix} 6 & -3 \\ 3 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 1 \\ 5 & 8 \end{bmatrix}, \\
C_{11} &= M_1 + M_3 - M_2 + M_6 \\
&= \begin{bmatrix} 4 & 8 \\ 20 & 14 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ -9 & 4 \end{bmatrix} - \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ -2 & -3 \end{bmatrix} = \begin{bmatrix} 3 & 7 \\ 7 & 7 \end{bmatrix}.
\end{aligned}
$$

That is,

$$
C = \begin{bmatrix} 5 & 4 & 7 & 3 \\ 4 & 5 & 1 & 9 \\ 8 & 1 & 3 & 7 \\ 5 & 8 & 7 & 7 \end{bmatrix}.
$$

Problem

- Which tree traversal yields a sorted list if applied to a binary search tree?

A. preorder
B. inorder
C. postorder

- Prove it!

Problem — Levitin §5.3 Q8

a. Draw a binary tree with ten nodes labeled.0, 1, 2, ..., 9 in such a way that the inorder and postorder traversals of the tree yield the following lists: 9, 3, 1, 0, 4, 2, 7, 6, 8, 5 (inorder) and 9, 1, 4, 0, 3, 6, 7, 5, 8, 2 (postorder).

b. Give an example of two permutations of the same n labels $0, 1, 2, .., n-1$ that cannot be inorder and postorder traversal lists of the same binary tree.

c. Design an algorithm that constructs a binary tree for which two given lists of n labels $0, 1, 2, .., n-1$ are generated by the inorder and postorder traversals of the tree. Your algorithm should also identify inputs for which the problem has no solution.

Problem — Levitin §5.3 Q8

a. Draw a binary tree with ten nodes labeled.0, 1, 2, ..., 9 in such a way that the inorder and postorder traversals of the tree yield the following lists: 9, 3, 1, 0, 4, 2, 7, 6, 8, 5 (inorder) and 9, 1, 4, 0, 3, 6, 7, 5, 8, 2 (postorder).

Hint:

First, find the label of the root.

Then, identify the left and right subtrees.

Problem — Levitin §5.3, Q11

Chocolate bar puzzle Given an n-by-m chocolate bar, you need to break it into nm 1-by-1 pieces. You can break a bar only in a straight line, and only one bar can be broken at a time. Design an algorithm that solves the problem with the minimum number of bar breaks. What is this minimum number? Justify your answer by using properties of a binary tree.

Hint:

Breaking the chocolate bar can be represented as a binary tree