Data Structure

Lecture#15: Non-Binary Trees 2 (Chapter 6)

U Kang
Seoul National University

In This Lecture

- Main ideas in implementations of general trees
- Compare advantages and disadvantages of implementations
- Motivation and main ideas of sequential implementation

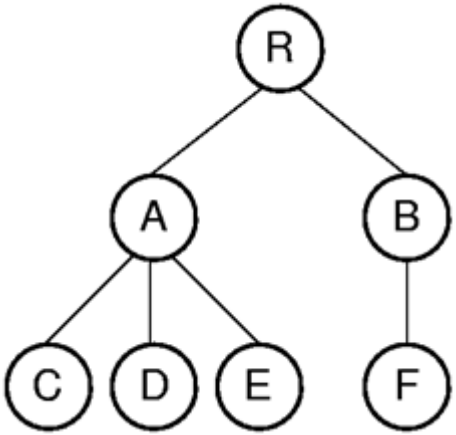
General Tree Implementation

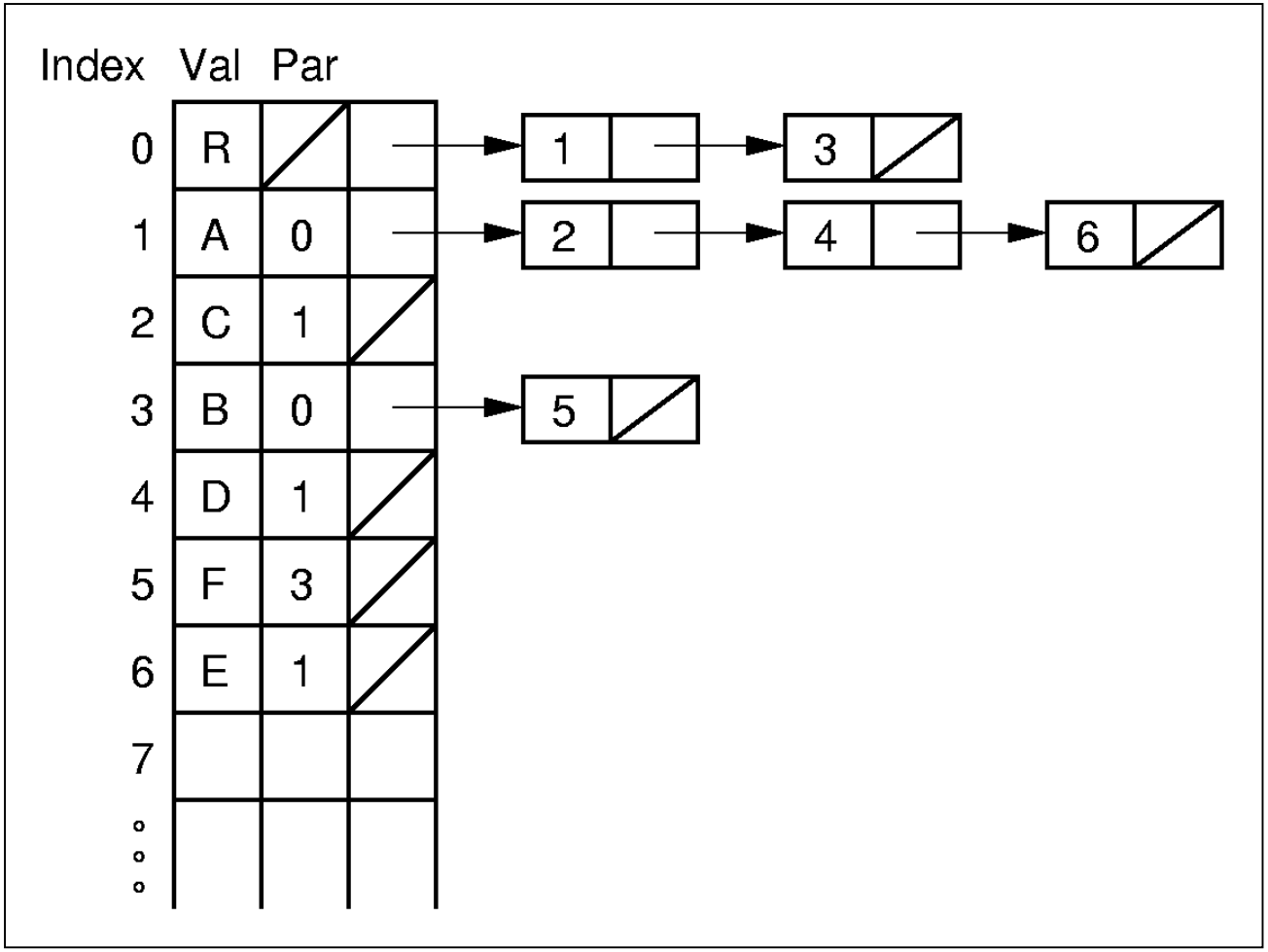
1. List of Children
2. Left-Child/Right-Sibling
3. Dynamic Node
4. Dynamic “Left-Child/Right-Sibling”

Evaluation criteria: how well each implementation supports

- parent();
- leftmostChild();
- rightSibling();

1. Lists of Children

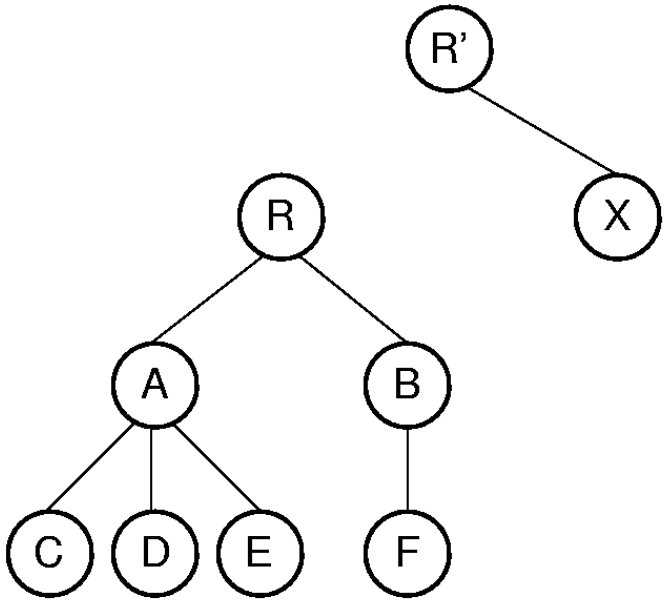




1. List of Children

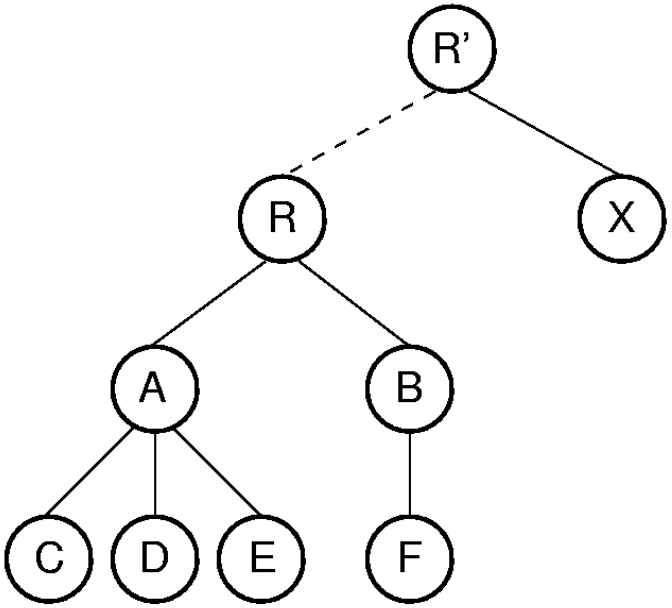
- Advantages
  - parent() is efficient
  - leftmostChild() is efficient
  - Combining two trees is easy if both trees are stored in an array
- Disadvantages
  - rightSibling() is inefficient
  - Problem from array-based implementation: needs to know the number of nodes in advance

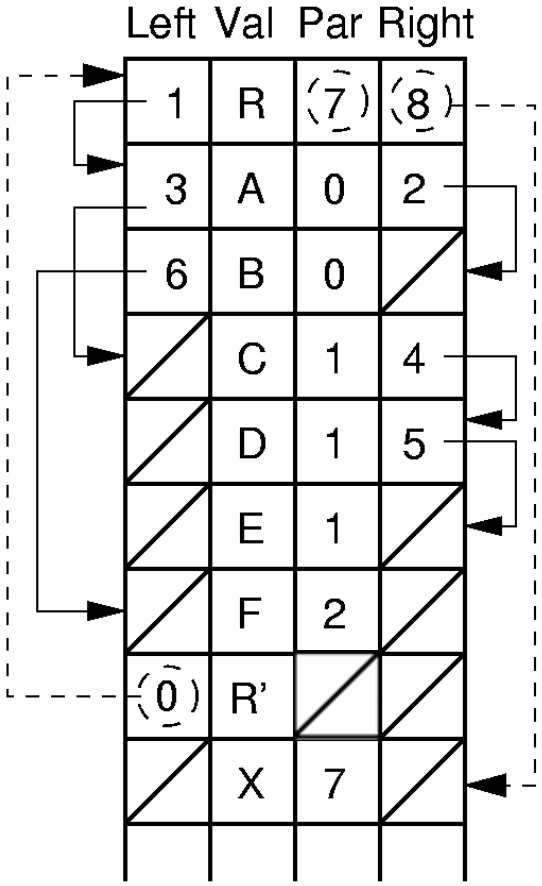
2. Leftmost Child/Right Sibling



Index	Left	Val	Par	Right
0	1	R		
1	3	A	0	2
2	6	B	0	
3		C	1	4
4		D	1	5
5		E	1	
6		F	2	
7	8	R'		
...		X	7	

2. Leftmost Child/Right Sibling

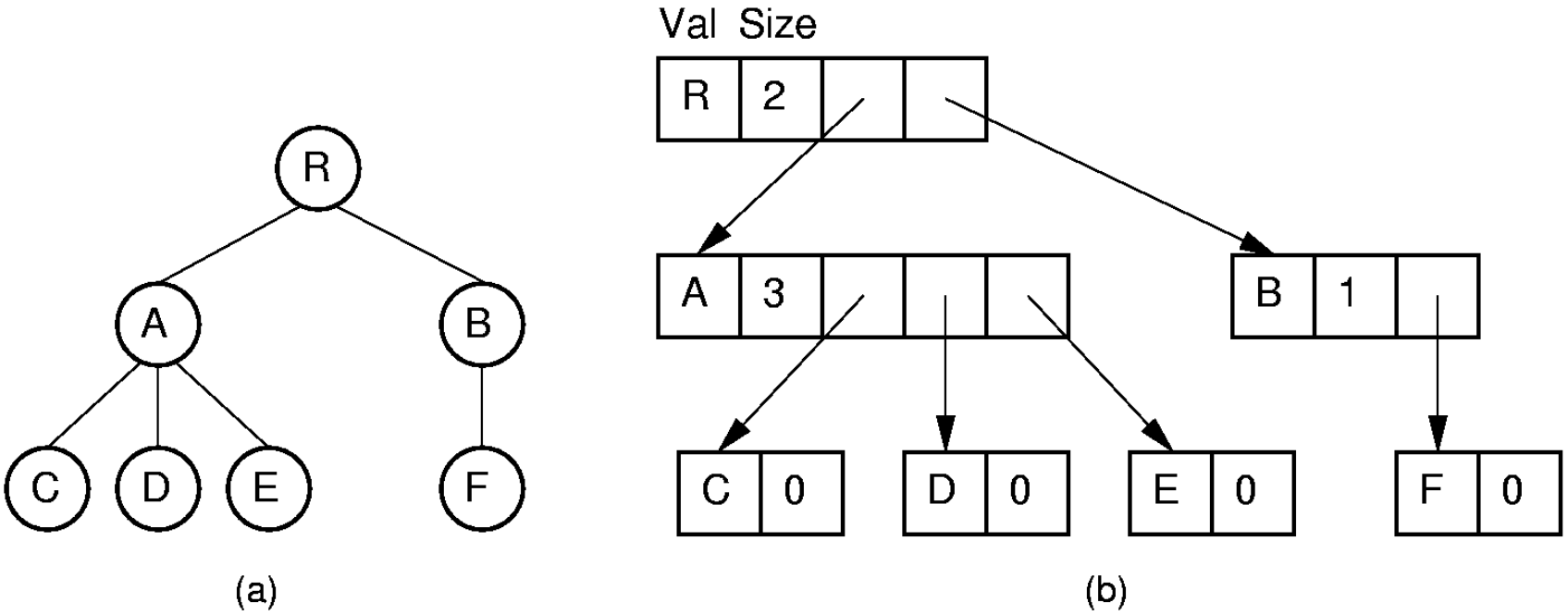




2. Leftmost Child/Right Sibling

- Advantages
  - parent(), leftmostChild(), rightSibling() are efficient
  - Combining two trees is easy if both trees are stored in an array
  - More space-efficient than “1. List of children” approach
- Disadvantages
  - Problem from array-based implementation: needs to know the number of nodes in advance

3. Dynamic Node – ver 1



Link-based implementation of "1. List of children" approach

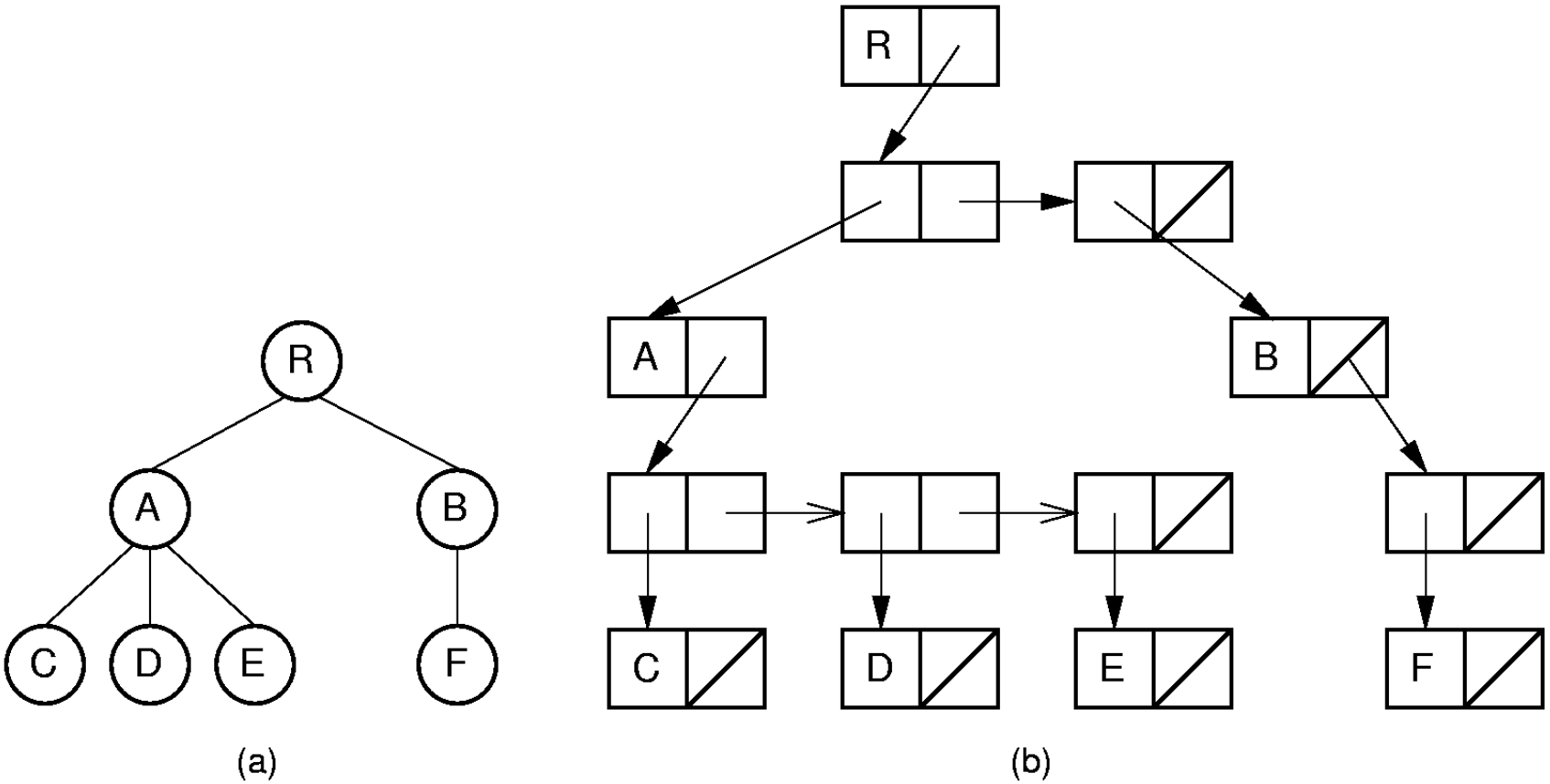
Each node can have a parent pointer as well (omitted for simplicity)

3. Dynamic Node – ver 1

- Advantages
  - parent() is efficient (if parent pointer is stored for each node)
  - leftmostChild() is efficient
  - Combining two trees is easy
  - **No need to know the number of nodes in advance**

- Disadvantages
  - rightSibling() is inefficient
  - **Still, needs to allocate fixed-size array for each node**

3. Dynamic Node – ver 2

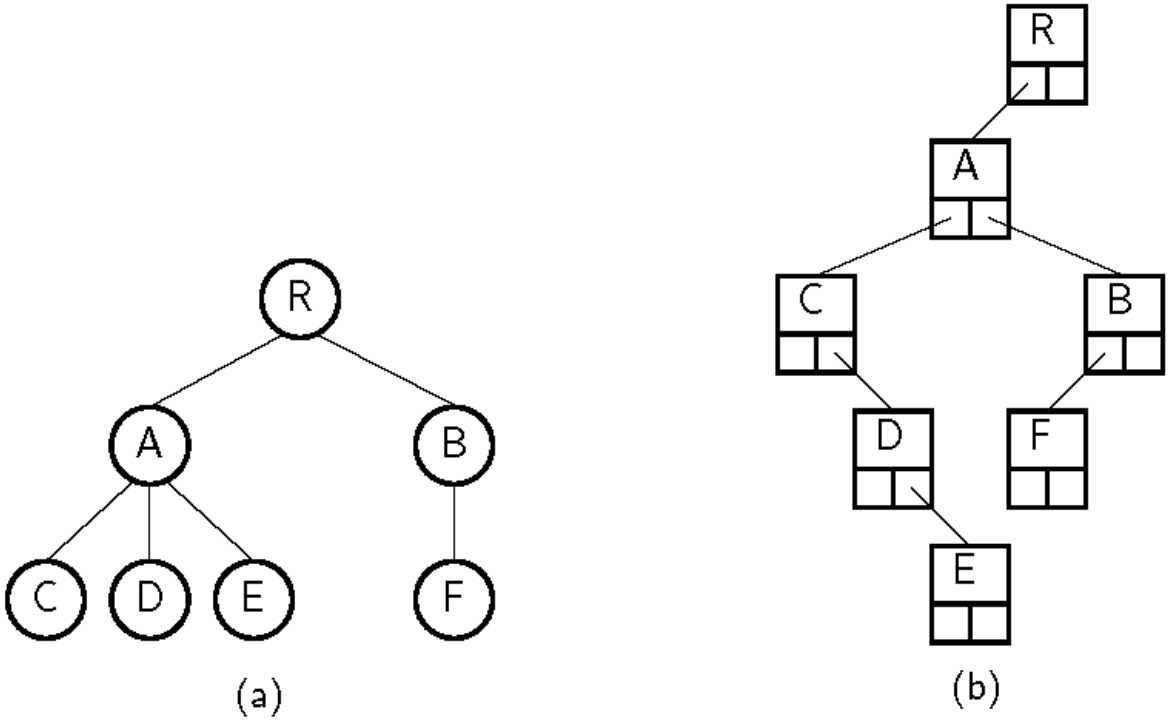


- Each node now requires a fixed amount of space (assuming space for data = space for pointer)

3. Dynamic Node – ver 2

- Compared to ver 1,
  - Ver 2 is more flexible: adding or removing an element is easy
  - On the other hand, ver 2 requires more space than ver 1

4. Dynamic Left-Child/Right-Sibling



Link-based implementation of
"2. Leftmost-Child/Right-Sibling" approach

Each node can have a parent pointer as well (omitted for simplicity)

4. Dynamic Left-Child/Right-Sibling

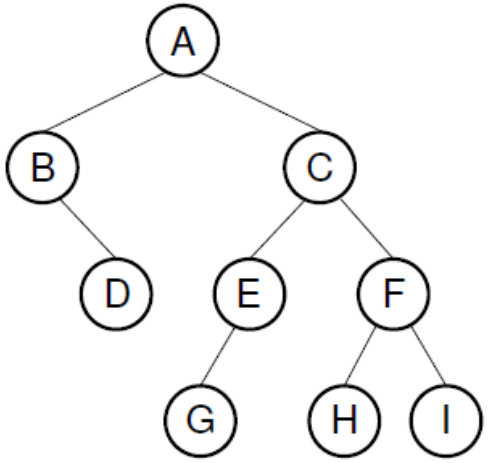
- Dynamic Left-Child/Right-Sibling approach vs. array based “2. Leftmost-Child/Right-Sibling” approach
  - Dynamic Left-Child/Right-Sibling is better: no need to pre-allocate memory
- Dynamic Left-Child/Right-Sibling approach vs. “3. Dynamic Node” approach
  - Dynamic Left-Child/Right-Sibling is better for space: uses less space

Sequential Implementations (1)

- In some cases, we want to focus only on space
  - Goal is to minimize space, without considering the time for parent(), leftmostChild(), rightSibling()
  - Application ?
    - archiving tree to backup disk (bank)
- Sequential tree implementation aims to minimize space to store the tree
  - List node values in the order they would be visited by a **preorder traversal**
  - No pointers are stored
  - Saves space, but allows only sequential access
  - Need to retain tree structure for reconstruction

Sequential Implementations (2)

- For binary trees
- Idea 1) use a symbol to mark **null** links
  - AB/D//CEG///FH//I//
  - / : null link
  - What is the amount of space overhead?

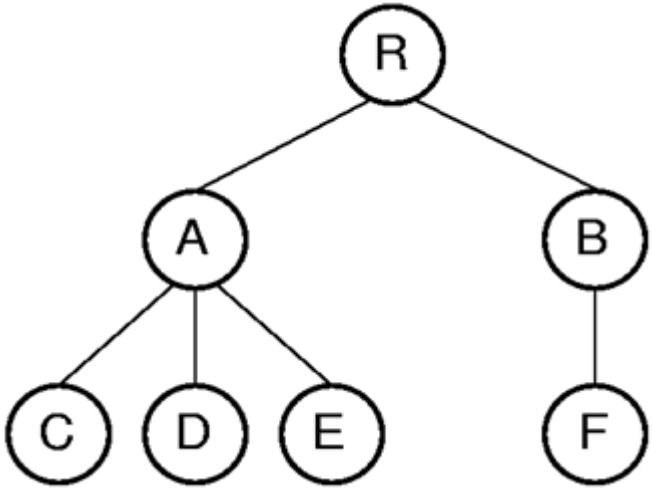


- How can we further improve idea 1, especially for full binary tree?

- Idea 2) use a bit to indicate internal nodes.
  - A'B'/DC'E'G/F'HI
  - ': internal node. / : null link
  - No / for full binary tree
  - For full binary tree, space overhead? (assume each node requires 4 bytes which include the bit)

Sequential Implementations (3)

- For general trees, mark the end of each subtree with)



RAC)D)E))BF)))

Can we use the same technique to store binary trees? Why or why not?

Sequential Implementations (4)

- Exercise: reconstruct a general tree from the sequential representation XAD)E))B)CG)H)))

Summary

- Main ideas in implementations of general trees
  - Evaluation criteria
- Compare advantages and disadvantages of implementations
  - Operations, running time, and space
- Motivation and main ideas of sequential implementation
  - Reconstruct trees from sequential representations

Questions?