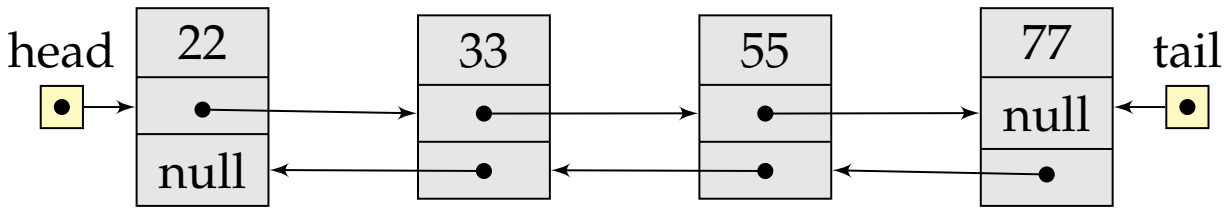
CSCI 2270: Data Structures

Lecture 16: Midterm Review using Doubly Linked List

Ashutosh Trivedi



Department of Computer Science
UNIVERSITY OF COLORADO BOULDER

Announcements

- Midterm Exam:
  - Date: 22nd Feb Friday from 5 PM to 7 PM.
  - Venue: Eaton Humanities 1B50
- Midterm (Special Accommodation):
  - Date: 22nd Feb Friday from 4 PM to 8 PM.
  - Venue: ECES 112
  - Email your TA by the end of the day Wednesday (2/20).
- Midterm (Makeup):
  - Date: 20th Feb Wednesday from 4 PM to 8 PM.
  - 4 hours for accommodation and 2 hours for others.
  - Venue: KCEN N100
- Exam format:
  - Moodle-based
  - 6 MCQs 5 points each — total 30 points
  - 3 Coding questions — 1 of the 3 will be Compulsory (35 points) and you will pick any 1 of the remaining 2 (35 points)
  - Total 100 points
  - Up-to 10 extra credits if you answer both coding questions correctly.

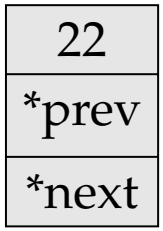
Announcements

- PLEASE CHARGE YOUR LAPTOP BATTERY BEFORE MIDTERM.
- You can use offline resources in the exam (Saved assignments, recitation materials, class notes, reference materials).
- Good luck!

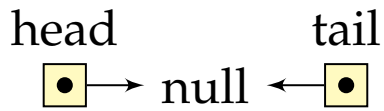
Doubly Linked List: Implementation

Binary Search Trees: Implementation

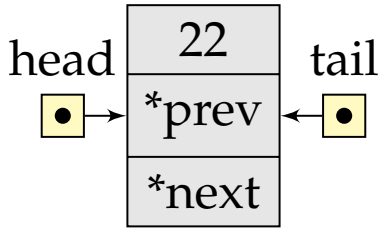
Doubly Linked List: Node



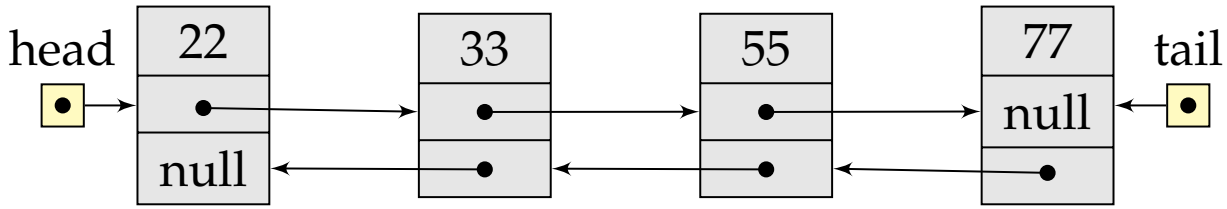
Doubly Linked List: Empty



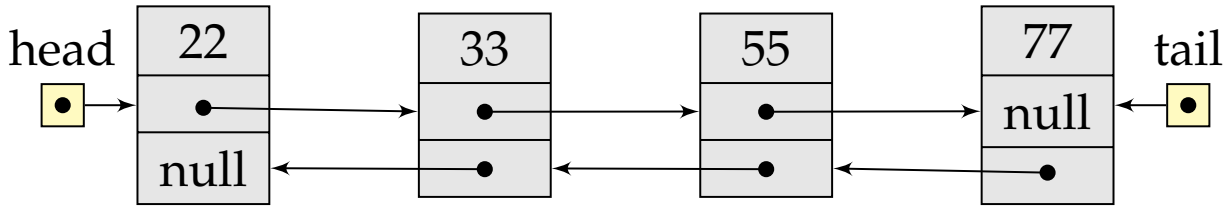
Doubly Linked List: Single Node



Doubly Linked List: General Case

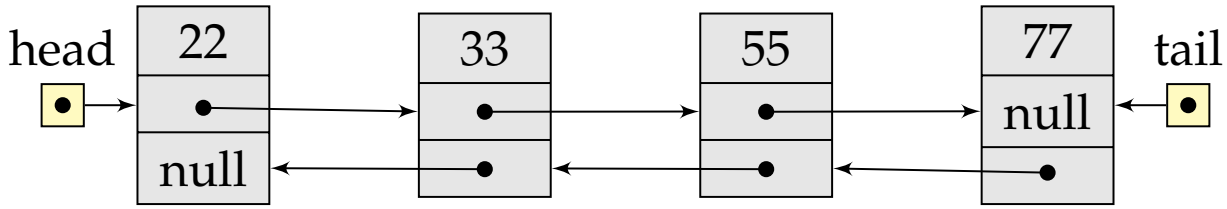


Doubly Linked List: Complexity



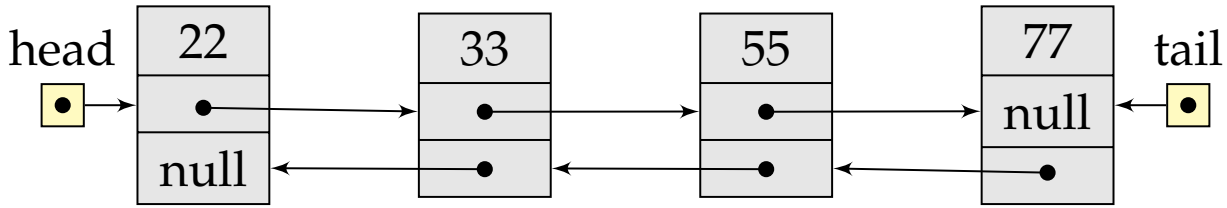
- Insertion:
  - at the head:

Doubly Linked List: Complexity



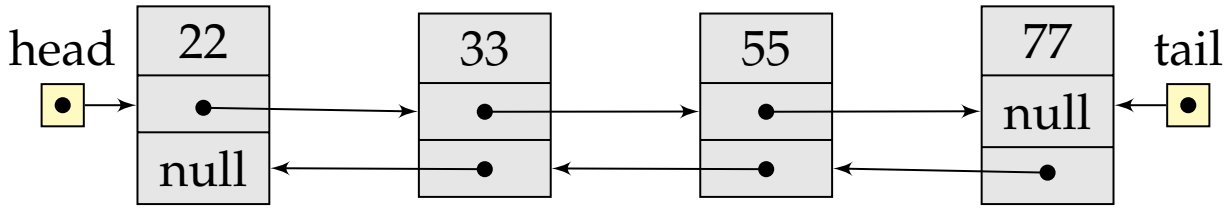
- Insertion:
  - at the head: $O(1)$

Doubly Linked List: Complexity



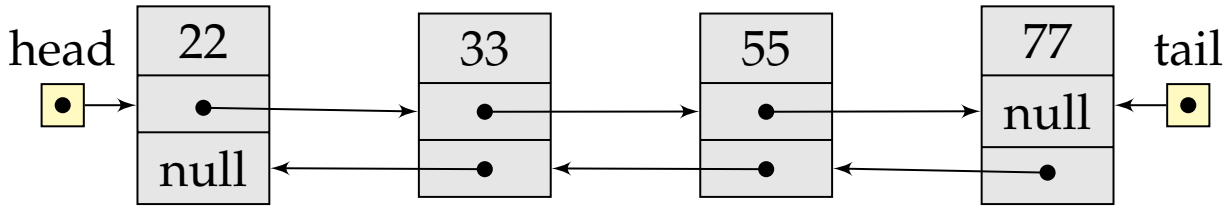
- Insertion:
  - at the head: $O(1)$
  - at the tail:

Doubly Linked List: Complexity



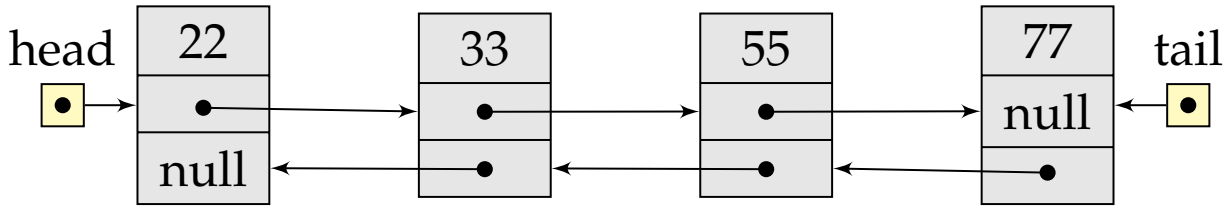
- Insertion:
  - at the head: $O(1)$
  - at the tail: $O(1)$

Doubly Linked List: Complexity



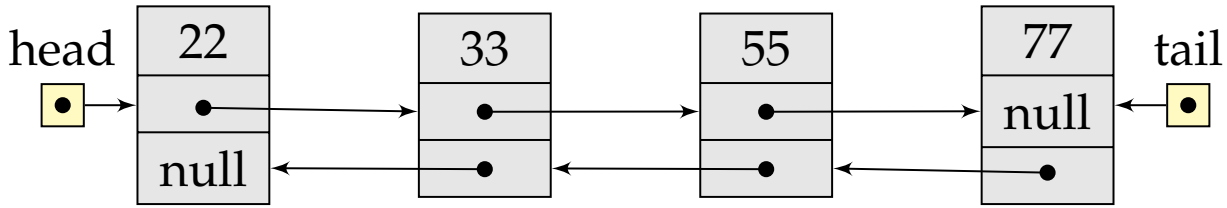
- Insertion:
  - at the head: $O(1)$
  - at the tail: $O(1)$
  - in the middle:

Doubly Linked List: Complexity



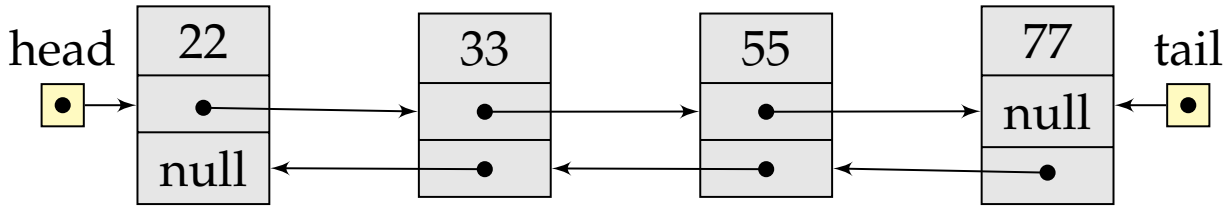
- Insertion:
  - at the head: $O(1)$
  - at the tail: $O(1)$
  - in the middle: $O(1)$

Doubly Linked List: Complexity



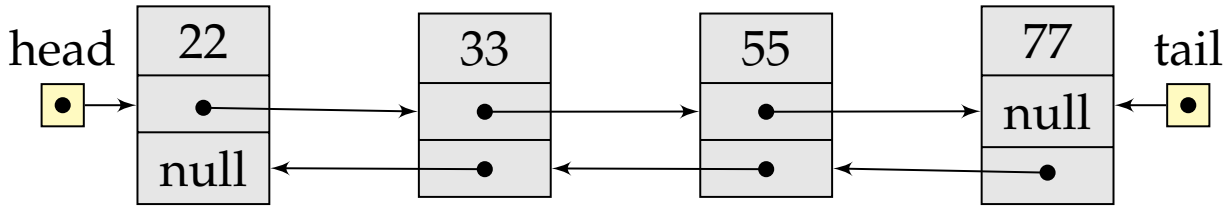
- Insertion:
  - at the head: $O(1)$
  - at the tail: $O(1)$
  - in the middle: $O(1)$
- Deletion:
  - at the head:

Doubly Linked List: Complexity



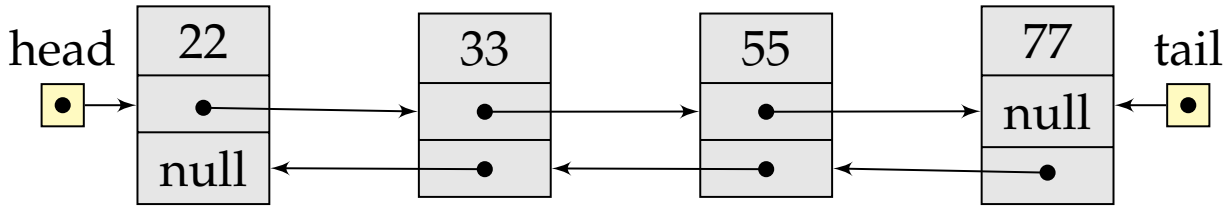
- Insertion:
  - at the head: $O(1)$
  - at the tail: $O(1)$
  - in the middle: $O(1)$
- Deletion:
  - at the head: $O(1)$

Doubly Linked List: Complexity



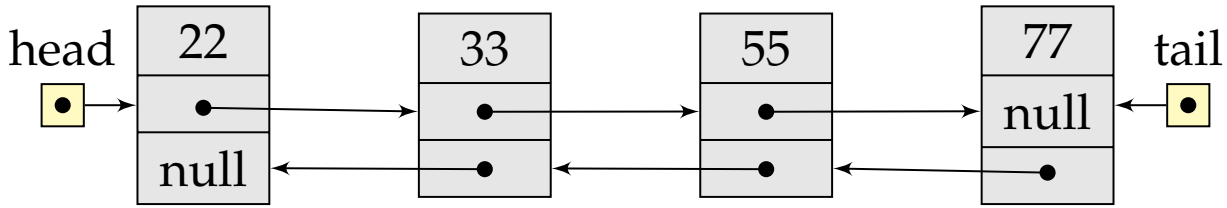
- Insertion:
  - at the head: $O(1)$
  - at the tail: $O(1)$
  - in the middle: $O(1)$
- Deletion:
  - at the head: $O(1)$
  - at the tail:

Doubly Linked List: Complexity



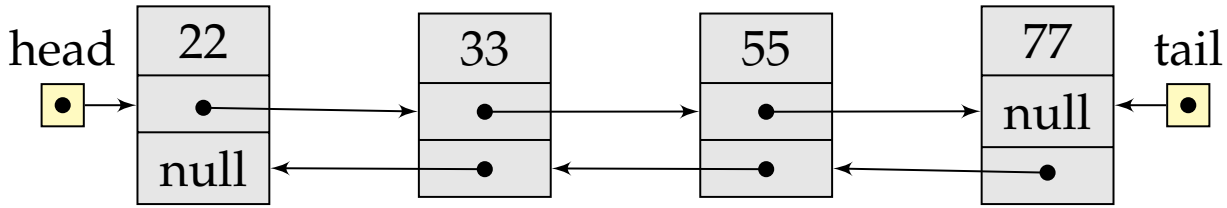
- Insertion:
  - at the head: $O(1)$
  - at the tail: $O(1)$
  - in the middle: $O(1)$
- Deletion:
  - at the head: $O(1)$
  - at the tail: $O(1)$

Doubly Linked List: Complexity



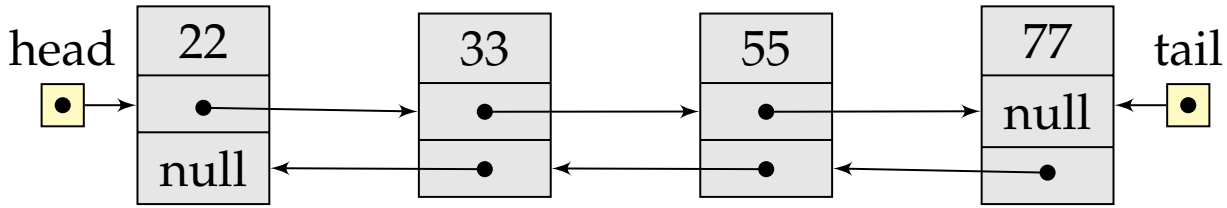
- Insertion:
  - at the head: $O(1)$
  - at the tail: $O(1)$
  - in the middle: $O(1)$
- Deletion:
  - at the head: $O(1)$
  - at the tail: $O(1)$
  - in the middle:

Doubly Linked List: Complexity



- Insertion:
  - at the head: $O(1)$
  - at the tail: $O(1)$
  - in the middle: $O(1)$
- Deletion:
  - at the head: $O(1)$
  - at the tail: $O(1)$
  - in the middle: $O(1)$

Doubly Linked List: Complexity



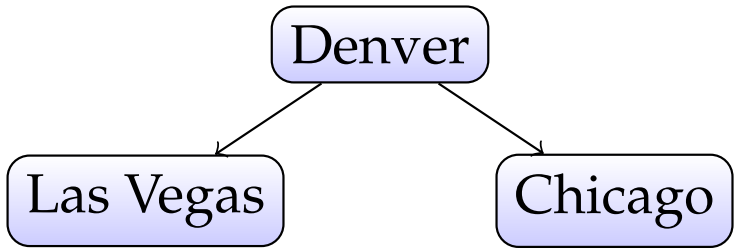
- Insertion:
  - at the head: $O(1)$
  - at the tail: $O(1)$
  - in the middle: $O(1)$
- Deletion:
  - at the head: $O(1)$
  - at the tail: $O(1)$
  - in the middle: $O(1)$
- Traverse: $O(n)$
- Search: $O(n)$

Doubly Linked List: Implementation

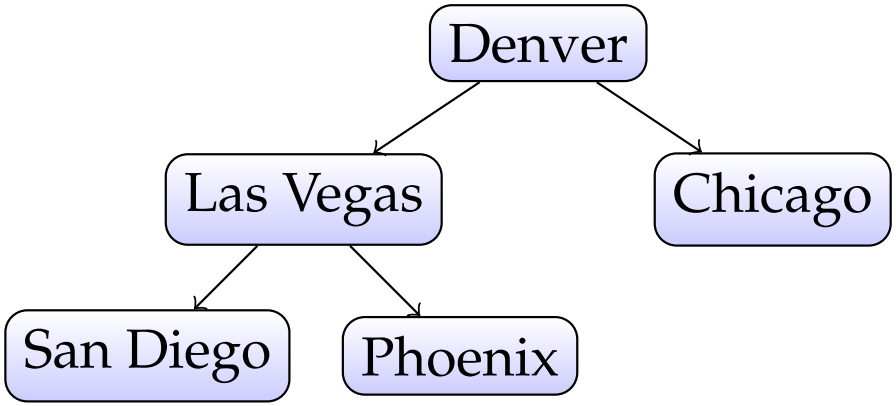
Binary Search Trees: Implementation

Trees: Terminology

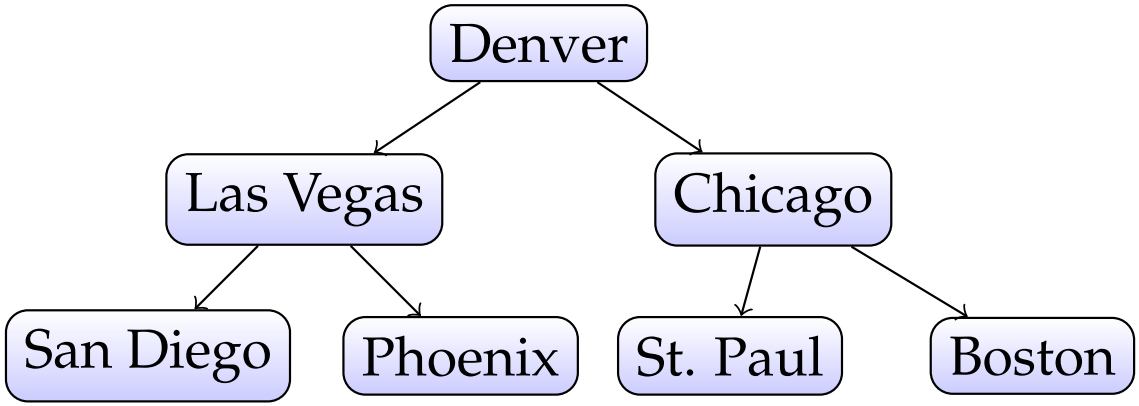
Trees: Terminology



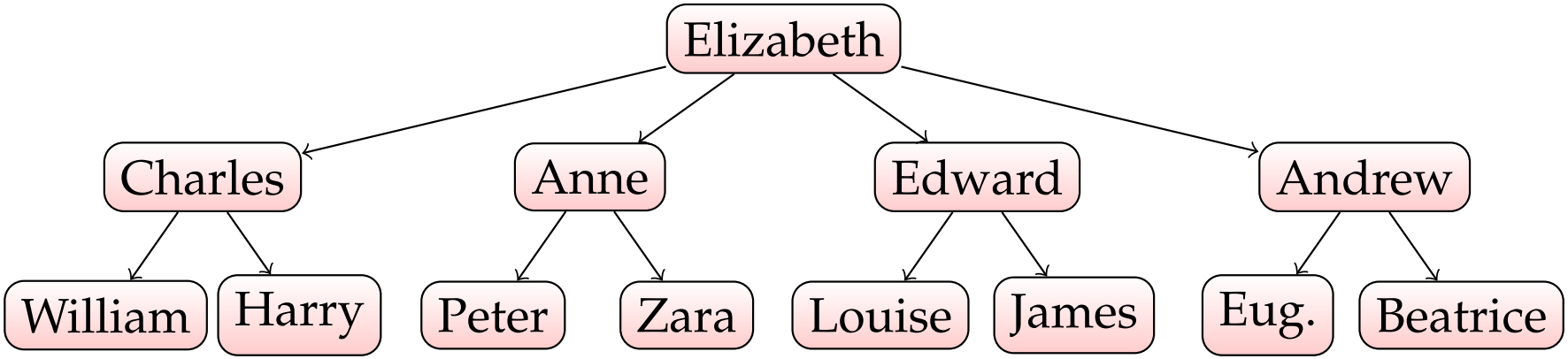
Trees: Terminology



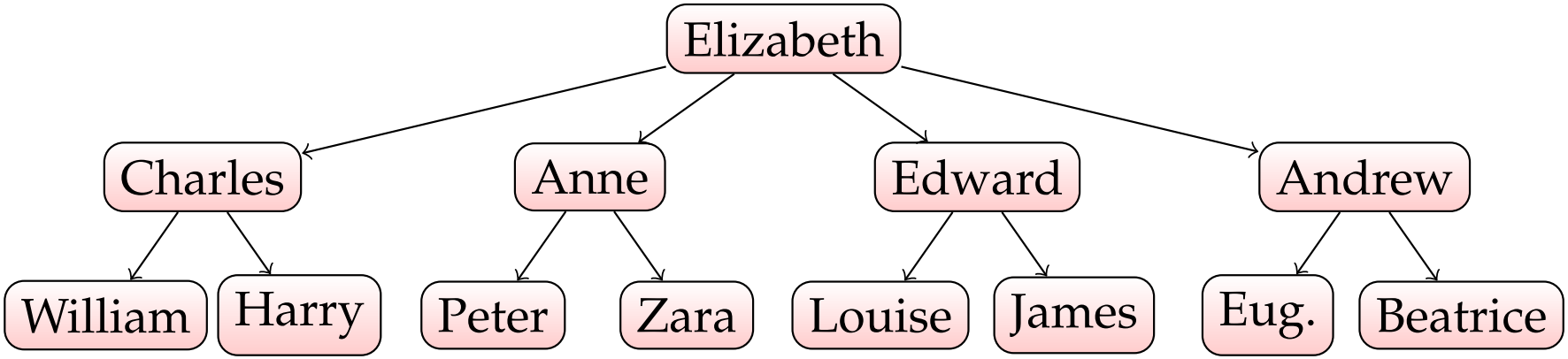
Trees: Terminology



Trees: Terminology

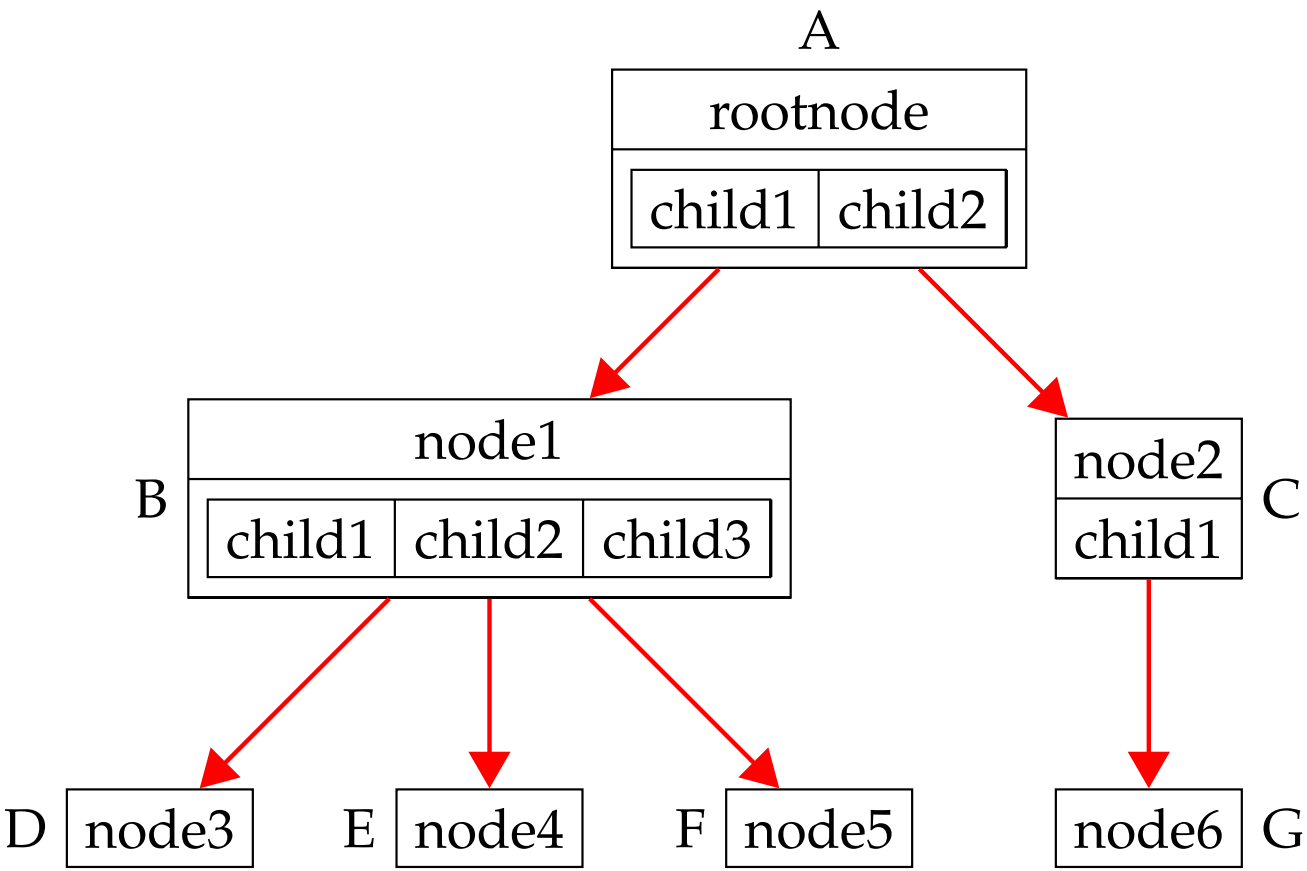


Trees: Terminology

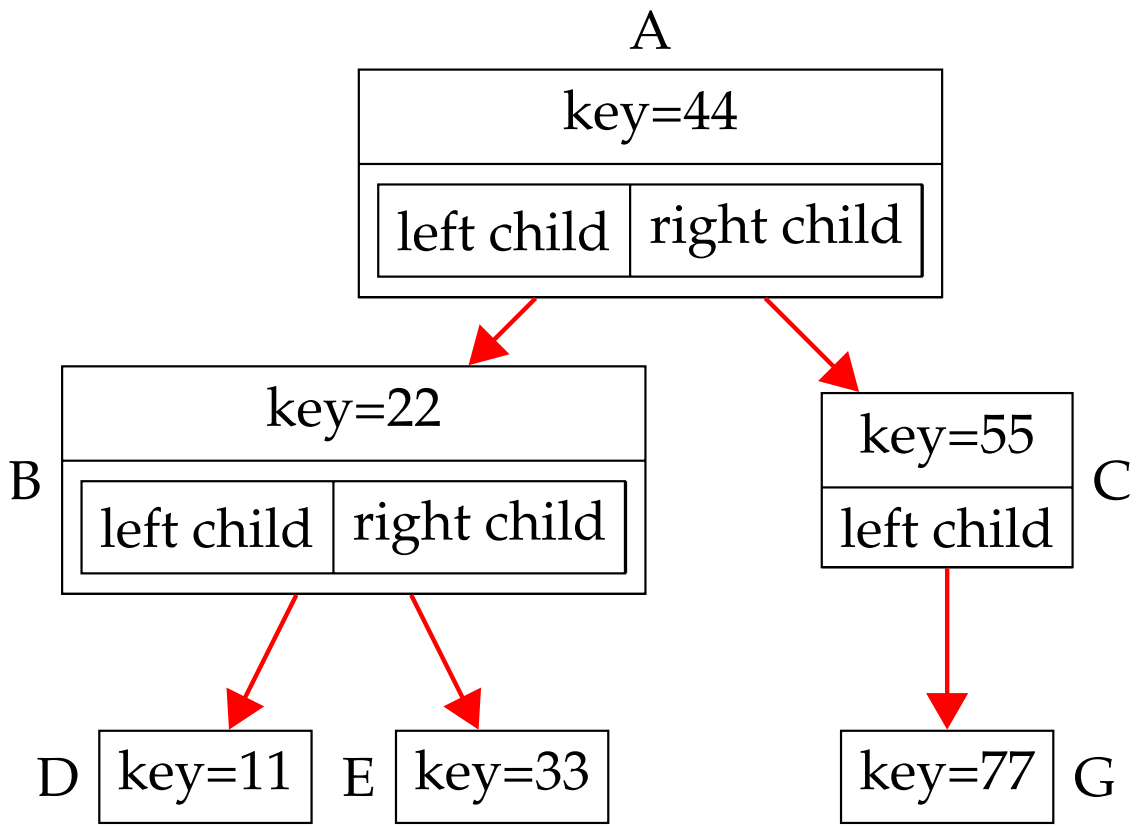


- *arity* of a tree: binary trees, ternary trees, k-ary trees, etc.
- *root* of a tree
- *leaf* of a tree
- *child* of a node
- *parent* of a node
- *ancestor* of a node
- *descendant* of a node
- *sibling* of a node

Trees as a linked structure

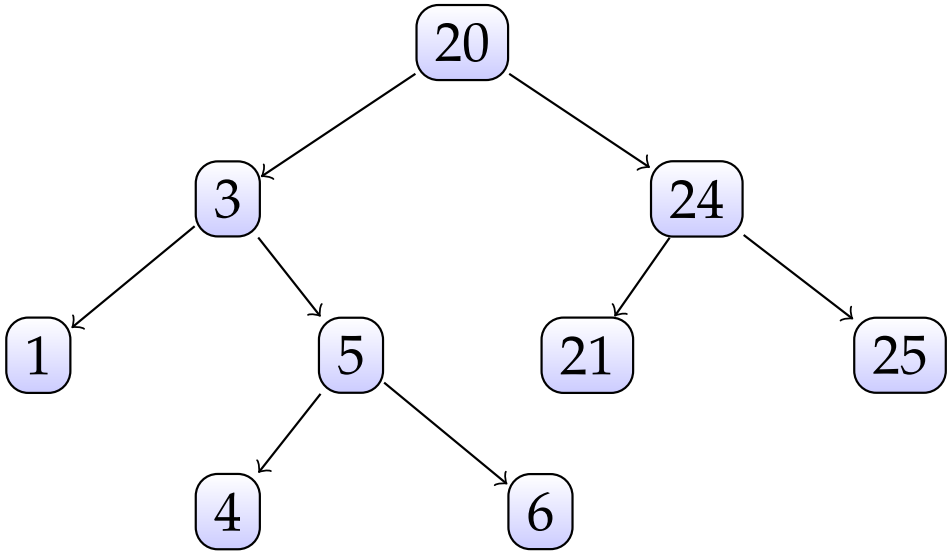


Binary Trees as a linked structure

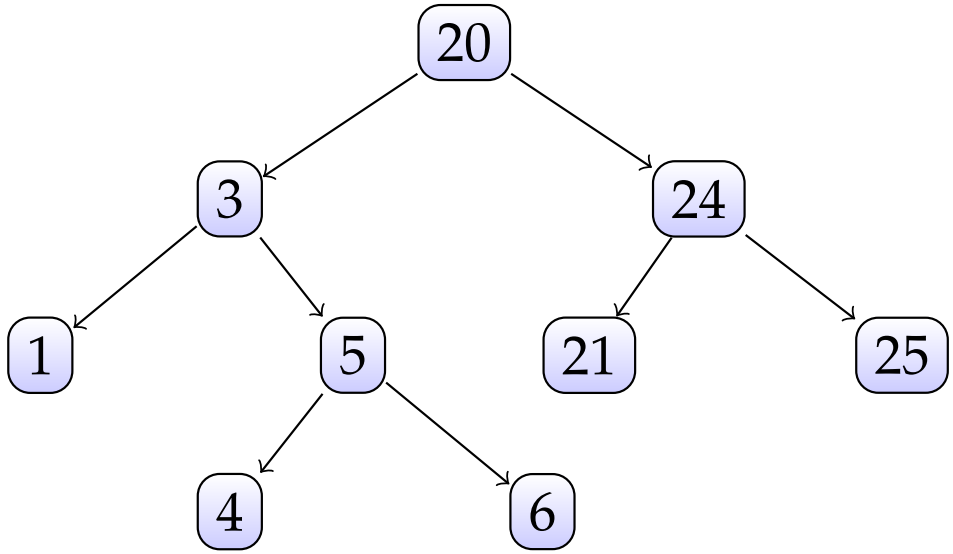


- *left sub-tree* of a node
- *right sub-tree* of a node

Binary Trees



Binary Trees



Properties: If x and y are nodes, and

1. y is in the left sub-tree of x, then

$$y.key < x.key$$

2. y is in the right sub-tree of x, then

$$y.key \geq x.key$$