Algorithms and Data Structures 2

Recap Lectures 13-14

Dr Michele Sevegnani

School of Computing Science
University of Glasgow

michele.sevegnani@glasgow.ac.uk

Topics we covered so far

- **Binary trees**
- **Rooted trees with unbounded branching**
- **Binary search trees (BSTs)**
- **Querying a tree**
- **Computation of tree parameters**
- **Operations**
  - Insertion
  - Deletion
- **Randomly build BSTs**
- **BSTs with equal keys**

Question 1

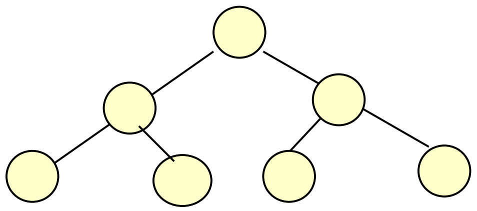
- **Show that perfectly balanced trees of height 2 and 3 contain 7 and 15 nodes respectively**

- **State, with a brief reason, how many nodes are in a perfectly balanced binary tree with height h**
  - $1 + 2 + 2^2 + \ldots + 2^k = 2^{k+1} - 1$

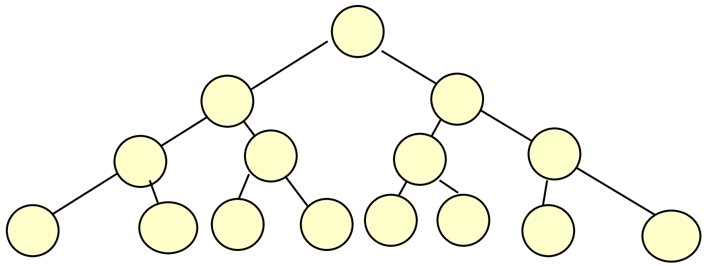
- **What extra property turns a binary tree into a binary search tree?**

Question 1: solution

- **Perfectly balanced trees of depths 2 and 3 have the following structures**



height 2
7 nodes



height 3
15 nodes

Question 1: solution (cont.)

- **Consider a perfectly balanced binary tree of height h**
  - At height 0 there is 1 node
  - At height 1 there are 2 nodes
  - At height 2 there are 4 nodes
  - …
  - …
  - At height h there are 2^h nodes

- **In total there are $2^0 + 2^1 + 2^2 + \ldots + 2^h = 2^{h+1} - 1$ nodes**

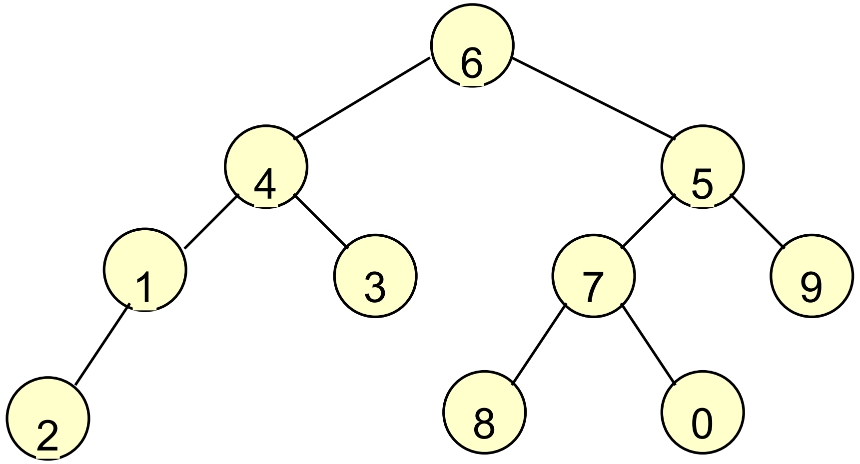
Question 1: solution (cont.)

- **The extra property is that the inorder traversal is in sorted order**

Question 2

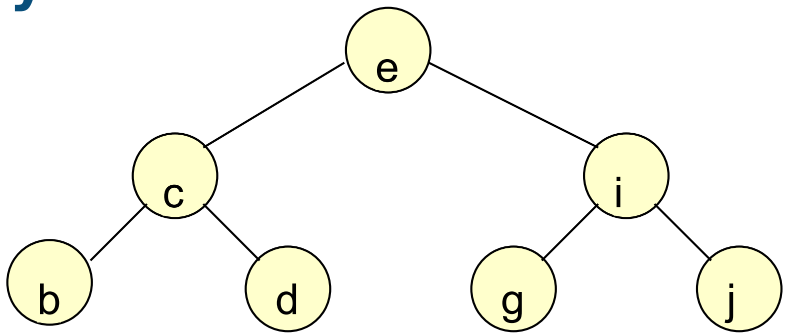
- **Draw one example each of a balanced binary tree and an extremely unbalanced binary tree**

- **Find the inorder, preorder and postorder traversal of the binary tree below**

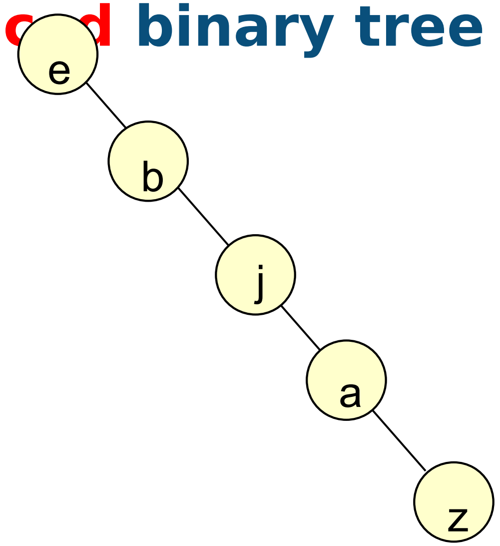


Question 2: solution

- **Example balanced binary tree**

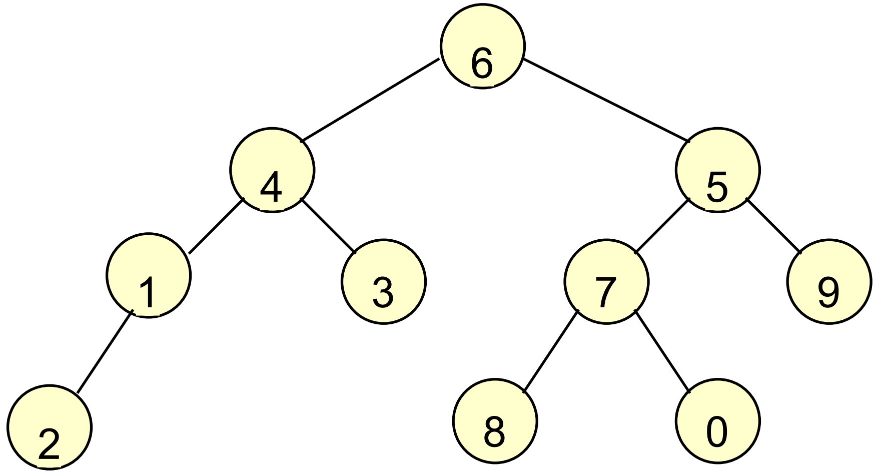


- **Example extremely unbalanced binary tree**



Question 2: solution (cont.)

- **Inorder traversal ☾ (1) left subtree, (2) root, and (3) right subtree**
  - 2,1,4,3,6,8,7,0,5,9
- **Preorder traversal ☾ (1) root subtree, (2) left, and (3) right subtree**
  - 6,4,1,2,3,5,7,8,0,9
- **Postorder traversal ☾ (1) left subtree, (2) right, and (3) root subtree**
  - 2,1,3,4,8,0,7,9,5,6



Question 3

- **Explain why an algorithm for finding a node in a binary search tree that contains the maximum number, n, of nodes for its height, has logarithmic complexity**

- **What is the complexity for a search of a binary search tree that contains no right subtrees?**

Question 3: solution

- **A BST containing the maximum number n of nodes for its depth is perfectly balanced**

- **As in Question 1, $n = 2^{h+1} - 1$ where h is the height of tree**

- **Search in BST has complexity $O(h)$**
  - $\log n \approx h + 1$
  - $O(h)$ is $O(\log n)$

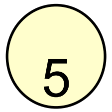
- **If a tree has no right subtrees, it is a linked list and $h = n$. Hence complexity is $O(n)$**

Question 4

- **Build a binary search tree by adding the following nodes in the given order: 5,1,18,9,7,6,15**

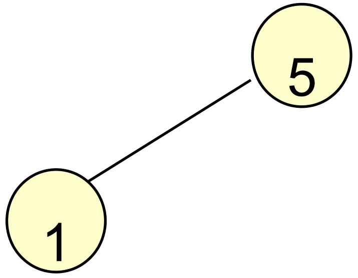
Question 4: solution

- **Build a binary search tree by adding the following nodes in the given order: 5,1,18,9,7,6,15**



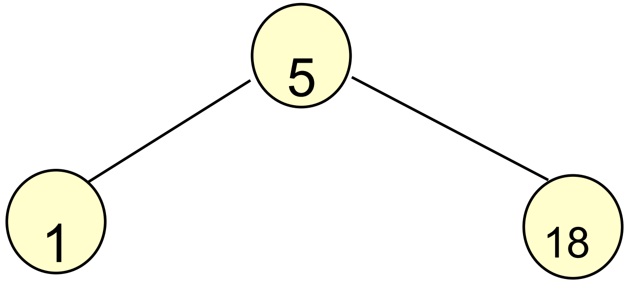
Question 4: solution

- **Build a binary search tree by adding the following nodes in the given order: 5,1,18,9,7,6,15**



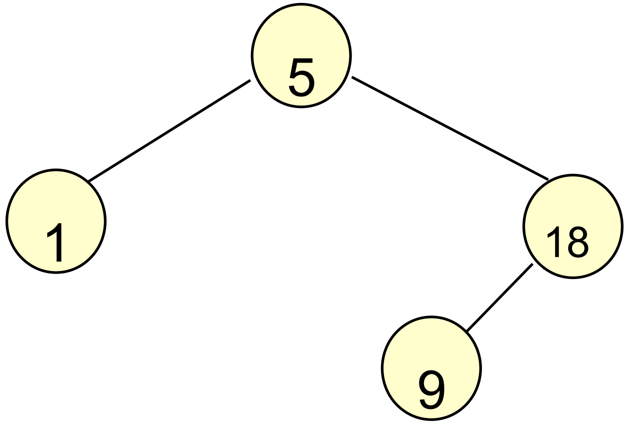
Question 4: solution

- **Build a binary search tree by adding the following nodes in the given order: 5,1,18,9,7,6,15**



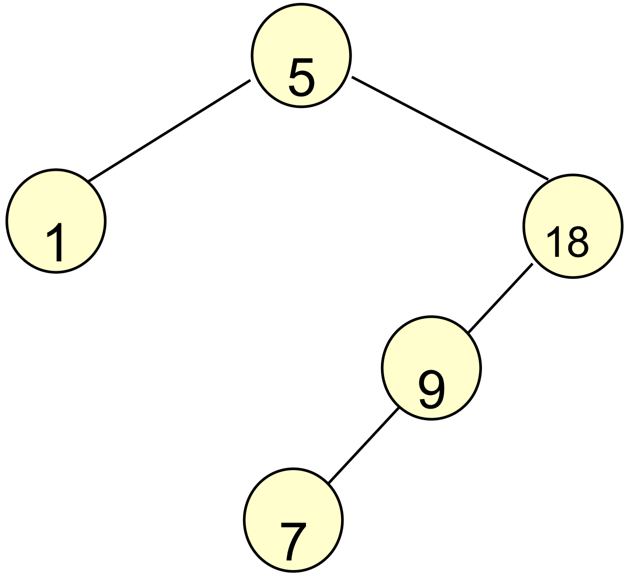
Question 4: solution

- **Build a binary search tree by adding the following nodes in the given order: 5,1,18,9,7,6,15**



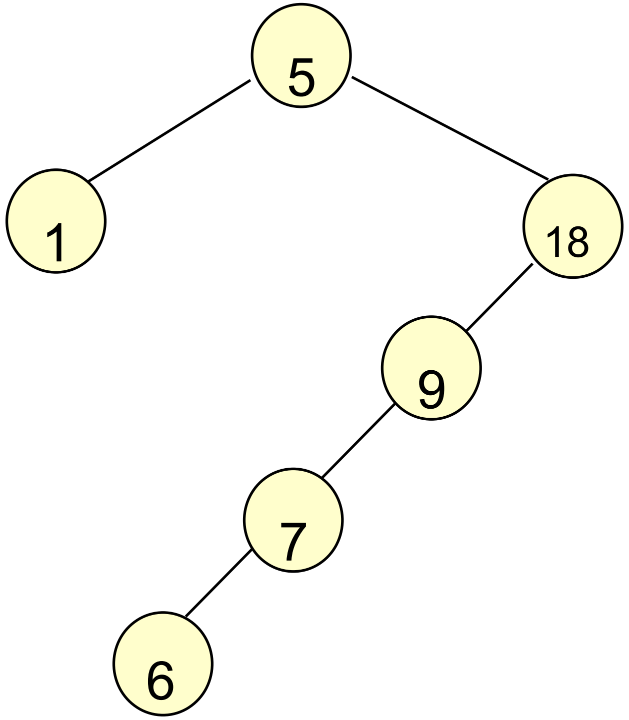
Question 4: solution

- **Build a binary search tree by adding the following nodes in the given order: 5,1,18,9,7,6,15**



Question 4: solution

- **Build a binary search tree by adding the following nodes in the given order: 5,1,18,9,7,6,15**



Question 4: solution

- **Build a binary search tree by adding the following nodes in the given order: 5,1,18,9,7,6,15**