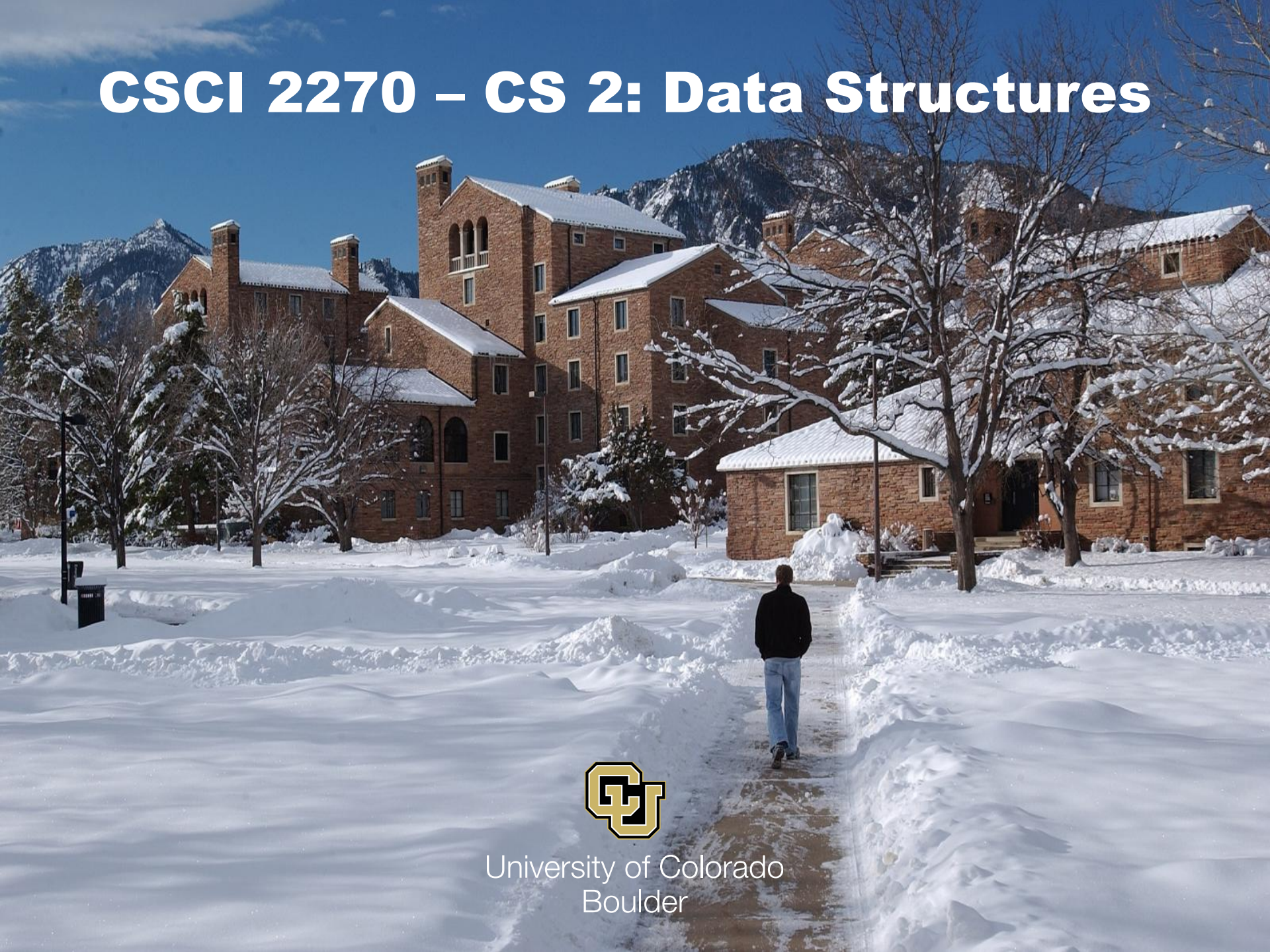
CSCI 2270 – CS 2: Data Structures



University of Colorado
Boulder

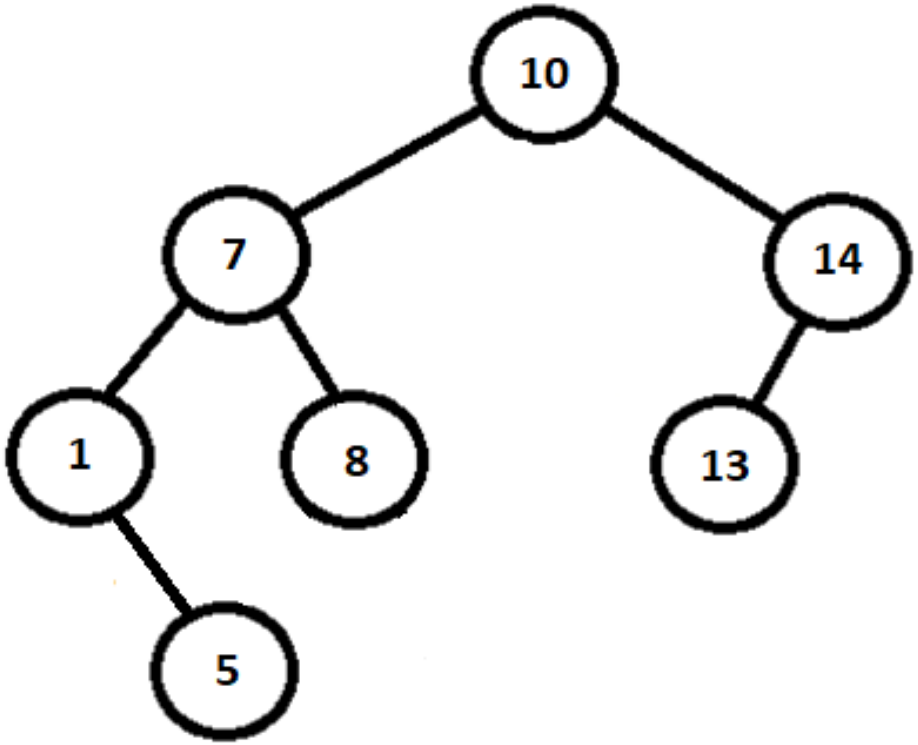
Reminders

Topics

- Binary Search Tree Implementation

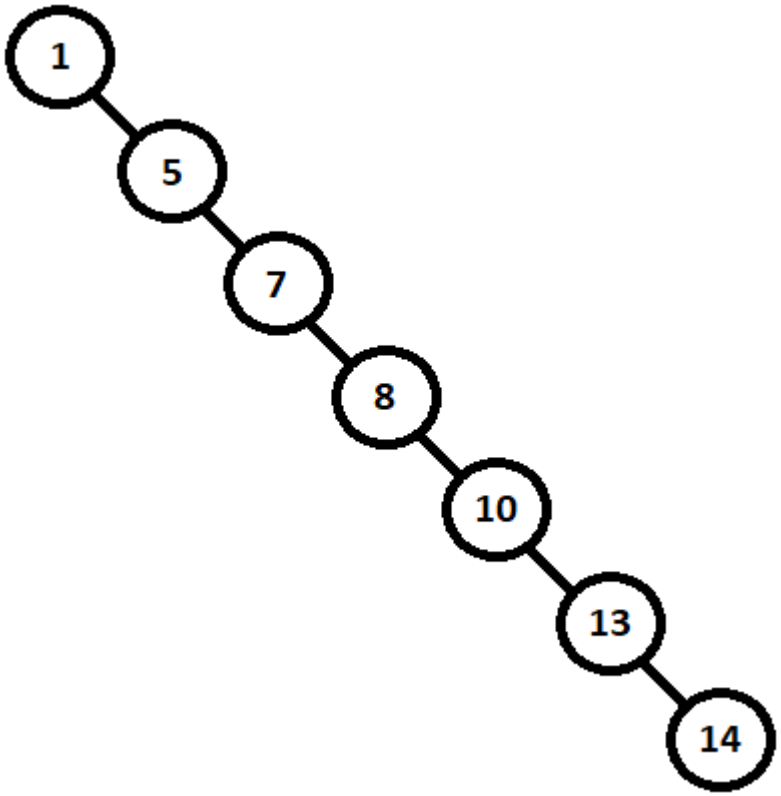
Balanced BST

- **10, 7, 14, 1, 8, 13, 5**

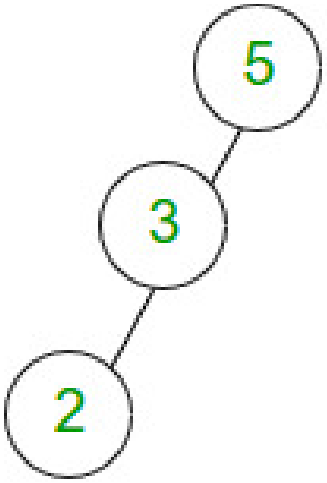


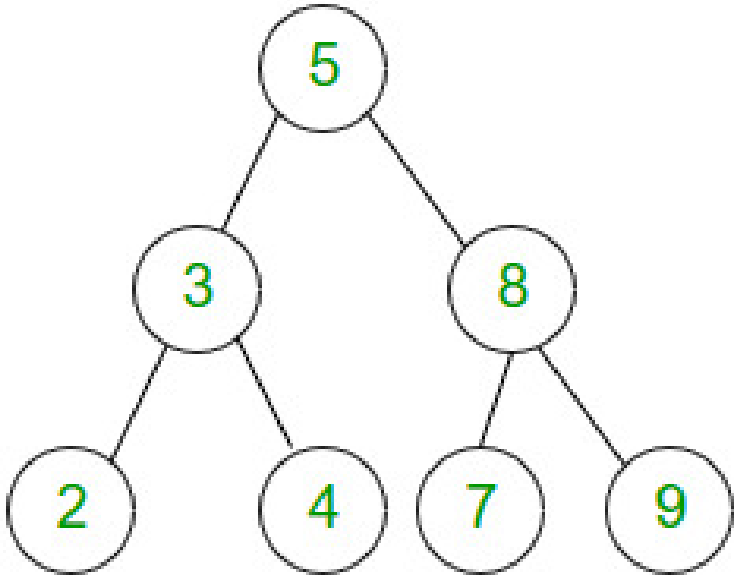
Unbalanced BST

- **1, 5, 7, 8, 10, 13, 14**



Number of Nodes





Height of a BST

searchBST(44)

Do we need to search entire tree?

No, we can take advantage of the ordered nature of the BST.

Solve for the height of the tree in terms of the number of nodes.

$n = 2^{h+1} - 1$

$n + 1 = 2^{h+1}$

$\log (n + 1) = h + 1$ // how did we get this? exp = log

$\log (n + 1) - 1 = h$

Height of a BST

As an example, let's find the height of a 500 element BST.

$h = \log_2 (n + 1) - 1$

$h = \log_2 (500 + 1) – 1$

$h = \log_2 (501) - 1$

$h = 8.97 – 1$

$h = 7.97$

Ignore the fractions! So, the tree has a height of 8.

Searching a BST

searchBST(44); // assume 44 is a leaf node

Do we need to search entire tree?

No, we can take advantage of the ordered nature of the BST.

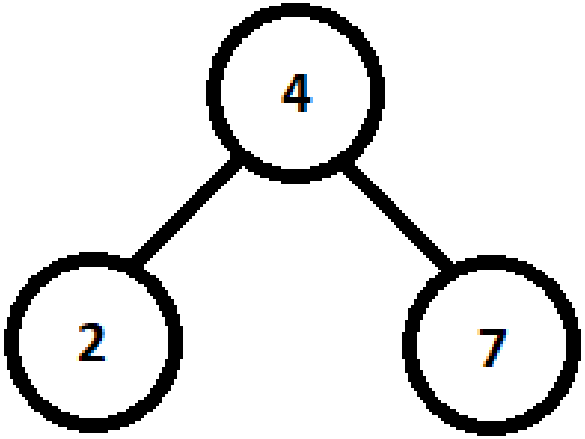
Max of h+1 comparisons to perform a search.

So, 8+1 = **9 comparisons** max to find a node.

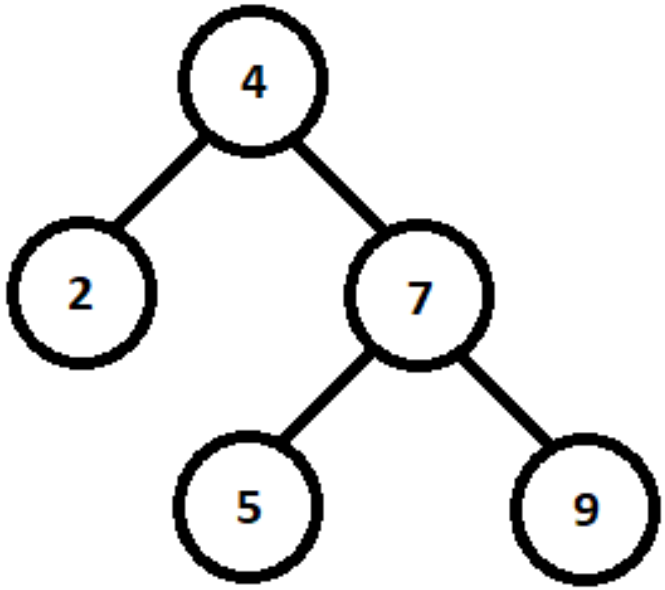
Traversing a BST

- In-order
  - Traverse the left subtree
  - Visit the node
  - Traverse the right subtree
- Pre-order
  - Visit the node
  - Traverse the left subtree
  - Traverse the right subtree
- Post-order
  - Traverse the left subtree
  - Traverse the right subtree
  - Visit the node
- **Each traversal algorithm is recursive!**

Traversing a BST

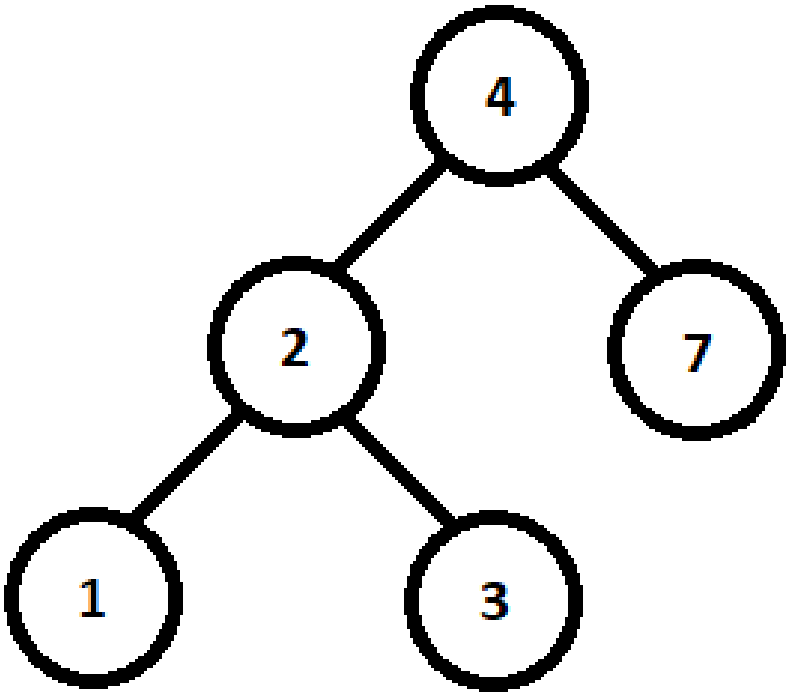


Traversing a BST



Traversing a BST

- **What is the in-order of this tree?**



Traversing a BST

- Example uses:
  - Pre-order: copy a tree to array, back to a tree
  - In-order: ascending order
  - Post-order: delete from leaf

Implementing a BST

- Two types of BST implementations
  - Iterative
  - Recursive
- Let's look at recursive BST (BST.cpp)

Questions?