CMPT 225
Tree Sort
Last Lectures

- Balanced binary search tree
Learning Outcomes

At the end of the lectures related to trees, students will be able to:

- Abstract Data Types (ADTs):
  - define the following data structures:
    - Binary search tree
    - Balanced binary search tree (AVL)
    - Binary heap
  and demonstrate and trace their operations
- implement the operations of binary search tree and binary heap
- implement and analyze sorting algorithms: tree sort and heap sort
- write recursive solutions to non-trivial problems, such as binary search tree traversals
Today’s Menu

Our goal is to

- Understand how tree sort works
- Sort an array using tree sort
- Analyze time/space efficiency of tree sort
Tree Sort

• Takes advantage of the ordering property of binary search trees (or AVL trees) to sort the data

• **Algorithm:**

  array = array of elements  
  tree = empty binary search tree (or AVL)

  for index = 0 to (elementCount − 1) do
    insert array[index] into tree

  for index = 0 to (elementCount − 1) do
    get next **inorder** element in tree
    store the element in array[index]
Let’s try!
Time Complexity Analysis of Tree Sort

• Time complexity of Tree Sort algorithm:
  - building the BST tree
    • average case:
    • worst case:
  - building the AVL tree:
  - traversing tree and inserting back into array:

• Overall cost:
Space Complexity Analysis of Tree Sort

• How much space (memory) does tree sort require to execute?
  ◦ Tree sort algorithm requires an additional $n$ space for the tree which can create a space problem when $n$ is large

• Therefore, its space efficiency is ->
Let’s update ...

- Table Comparing Sort Algorithms
Summary

- **Tree Sort**
  - Understood how tree sort works and we tried it
  - Analyzed time/space efficiency of tree sort
Next Lecture

- Heap