

CSIS 3103
Ch 6:
Binary Search Tree Application
Heaps

Binary Tree Application Index for a Term Paper

- The index should show each word in the paper followed by the line number on which it occurred
- The words should be displayed in alphabetical order

Term Paper Index

Analysis

- Store each word and its line number as a string in a tree node
- If a word occurs on multiple lines, the line numbers should be listed in ascending order:
a, 3
a, 13
are, 3
- Display the words in ascending order by performing an inorder traversal

Term Paper Index

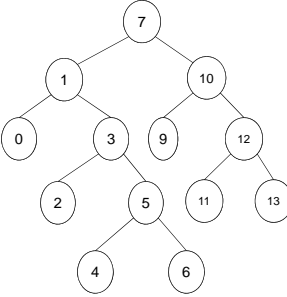
Design

- Use `TreeSet<E>` class (Java API class with similarities to the binary search tree)
- Write a class `IndexGenerator` with a `TreeSet<String>` data fields

Data Field	Attribute
<code>private TreeSet<String> index</code>	The search tree used to store the index.
Method	Behavior
<code>public void buildIndex(Scanner scan)</code>	Reads each word from the file scanned by <code>scan</code> and stores it in <code>tree index</code> .
<code>public void showIndex()</code>	Performs an inorder traversal of <code>tree index</code> .

Full Binary Trees

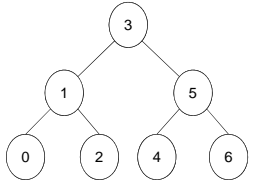
- Trees grow from the top down
- Each new value is inserted as a new leaf node
- In a *full* binary tree all nodes have either 2 children or 0 children (the leaf nodes)



Perfect Binary Trees

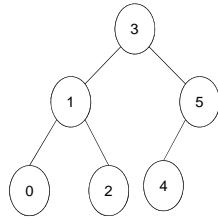
A *perfect binary tree* is a full binary tree of height n with exactly $2^n - 1$ nodes

- In this example, $n = 3$ and $2^n - 1 = 7$



Complete Binary Trees

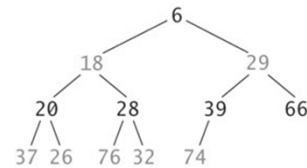
A *complete binary tree* is a perfect binary tree through level $n - 1$ with some extra leaf nodes at level n (the tree height), all toward the left



Heap Properties

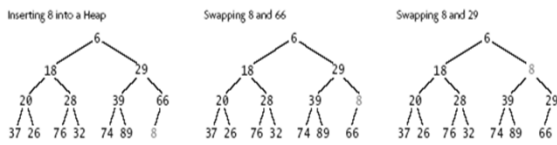
A *heap* is a complete binary tree with the following properties

- The value in the root is the smallest item in the tree
- Every subtree is a heap



Algorithm for Inserting in a Heap

1. Insert the new item in the next position at the bottom of the heap.
2. **while** new item is not at the root and new item is smaller than its parent
3. Swap the new item with its parent, moving the new item up the heap.



Algorithm for Removal from a Heap

1. Remove the item in the root node by replacing it with the last item in the heap (LIH).
2. **while** item LIH has children and item LIH is larger than either of its children
3. Swap item LIH with its smaller child, moving LIH down the heap.

