Introduction To Binary Trees

- Erel Segal-Halevi

Data structures

- Insert 5
- Insert 3
- Insert 7
- Contains 4? - False
- Contains 5? - True
- Insert 4
- Insert 6
- Contains 4? - True
- Contains 5? - True

Solution #1 – Array

• Insert 5	• [5,_,_,_,_,_,_,_,_,_]
• Insert 3	• [5,3,_,_,_,_,_,_,_,_]
• Insert 7	• [5,3,7,_,_,_,_,_,_,_]
• Contains 4? - False	• (Linear search)
• Contains 5? - True	• (Linear search)
• Insert 4	• [5,3,7,4,_,_,_,_,_,_]
• Insert 6	• [5,3,7,4,6,_,_,_,_,_]
• Contains 4? - True	• (Linear search)
• Contains 5? - True	• (Linear search)

Quick insertion, slow search

Solution #2 – Ordered Array

• Insert 5	• [5,_,_,_,_,_,_,_,_,_]
• Insert 3	• [3,5,_,_,_,_,_,_,_,_]
• Insert 7	• [3,5,7,_,_,_,_,_,_,_]
• Contains 4? - False	• (Binary search)
• Contains 5? - True	• (Binary search)
• Insert 4	• [3,4,5,7,_,_,_,_,_,_]
• Insert 6	• [3,4,5,6,7,_,_,_,_,_]
• Contains 4? - True	• (Binary search)
• Contains 5? - True	• (Binary search)

Quick search, slow insertion

Solution #3 – **Binary Tree**

Goal: Quick search, quick insertion.

Idea:

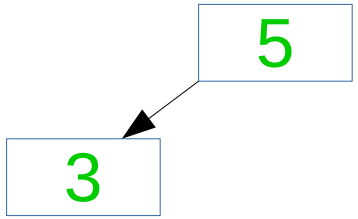
- No array;
- Each number is in its own "node";
- Smaller numbers go to left;
- Larger numbers go to right.

Solution #3 – **Binary Tree**

- Insert 5
- Insert 3

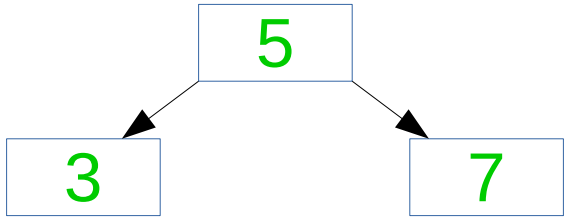
Solution #3 – **Binary Tree**

- Insert 5
- Insert 3
- Insert 7



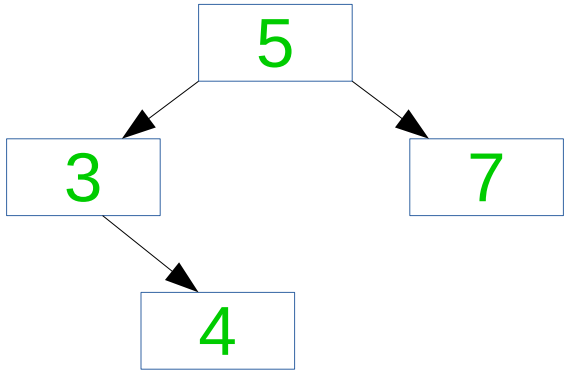
Solution #3 – **Binary Tree**

- Insert 5
- Insert 3
- Insert 7
- Insert 4



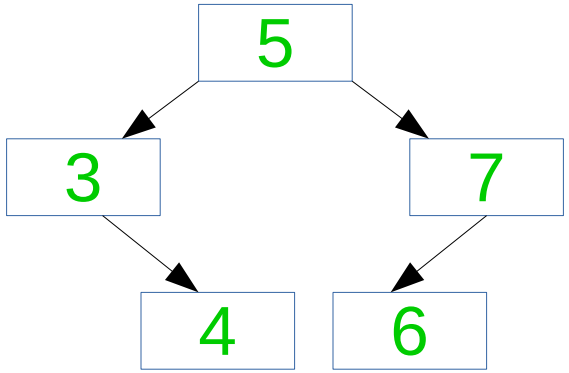
Solution #3 – **Binary Tree**

- Insert 5
- Insert 3
- Insert 7
- Insert 4
- Insert 6



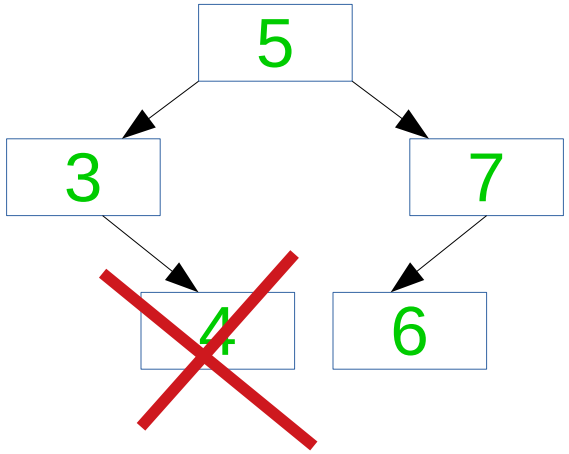
Binary Tree

- Insert 5
- Insert 3
- Insert 7
- Insert 4
- Insert 6



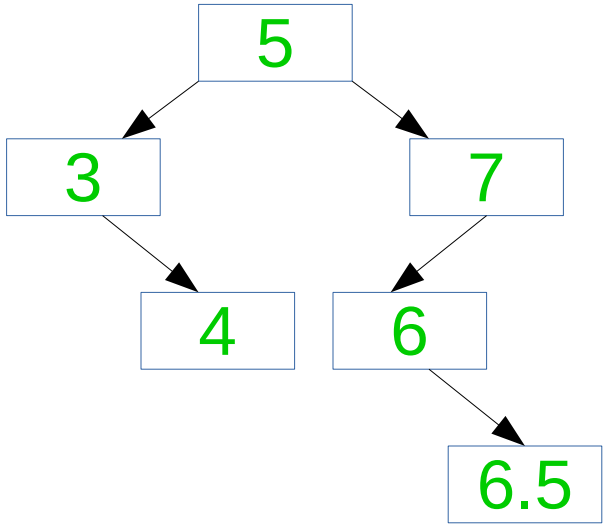
Binary Tree – remove node with **no children**:

- Insert 5
- Insert 3
- Insert 7
- Insert 4
- Insert 6
- Remove 4 – *just delete the node.*



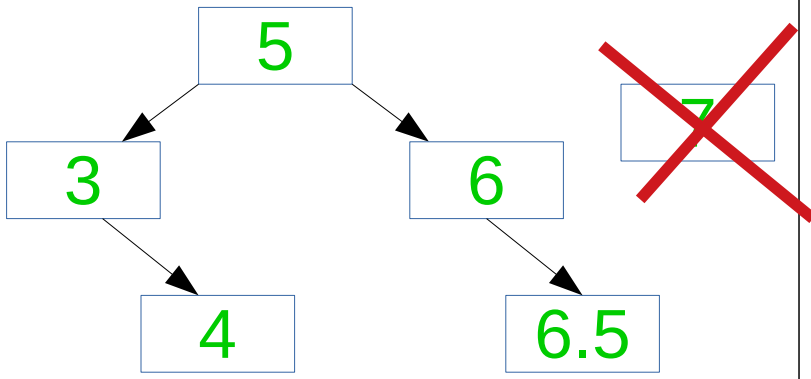
Binary Tree – remove node with **one child**:

- Insert 5
- Insert 3
- Insert 7
- Insert 4
- Insert 6
- Remove 7



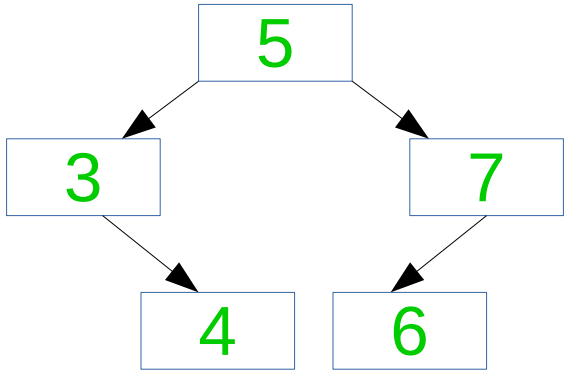
Binary Tree – remove node with **one child**:

- Insert 5
- Insert 3
- Insert 7
- Insert 4
- Insert 6
- Remove 7 – *take left sub-tree and move it instead of the deleted node.*



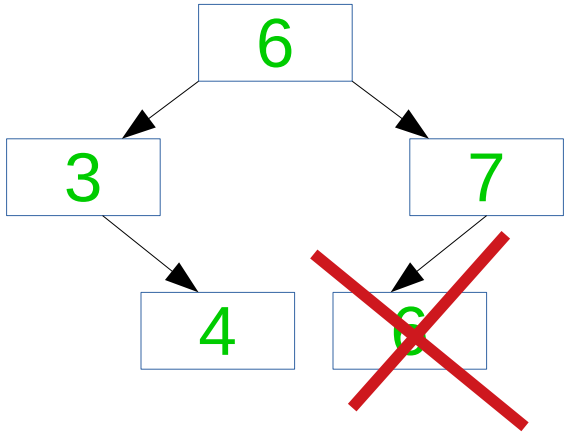
Binary Tree – remove node **with two children**:

- Insert 5
- Insert 3
- Insert 7
- Insert 4
- Insert 6
- Remove 5



Binary Tree – remove node **with two children**:

- Insert 5
- Insert 3
- Insert 7
- Insert 4
- Insert 6
- Remove 5 – copy smallest number in right sub-tree to deleted node; delete smallest element recursively.



Binary Tree – further information

- http://www.algolist.net/Data_structures/Binary_search_tree/Removal
-