HO CHI MINH UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY
SOFTWARE ENGINEERING DEPARTMENT
ADVANCED PROGRAM IN COMPUTER SCIENCE
COURSE: **DATA STRUCTURE**
LECTURER: Dr. ĐINH BÁ TIẾN

WEEK 02

BINARY SEARCH TREE

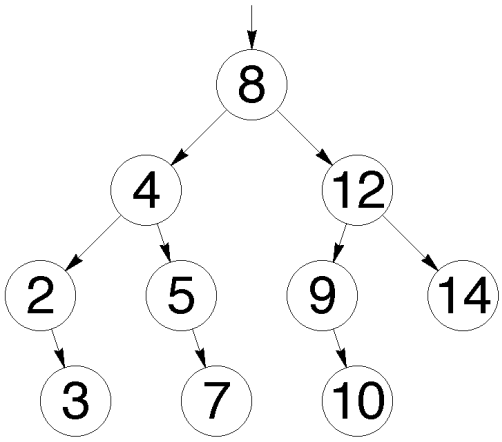
- TRƯƠNG PHƯỚC LỘC
- HỒ TUẤN THANH

HCMC, 2016

1 Binary search tree

1.1 Binary Search Tree

- A binary search tree is a rooted binary tree, whose internal nodes each store a key (and optionally, an associated value) and each have two distinguished sub-trees, commonly denoted left and right.



- Binary search trees keep their keys in sorted order, so that lookup and other operations can use the principle of binary search.

1.2 Searching

- We begin by examining the root node.
  - If the tree is null, the key we are searching for does not exist in the tree.
  - Otherwise, if the key equals that of the root, the search is successful and we return the node.
  - Otherwise, if the key is less than that of the root, we search the left subtree.
  - Otherwise, if the key is greater than that of the root, we search the right subtree.
- This process is repeated until the key is found or the remaining subtree is null. If the searched key is not found before a null subtree is reached, then the item must not be present in the tree.

1.3 Insertion

- Insertion begins as a search would begin; if the key is not equal to that of the root, we search the left or right subtrees as before.
- Eventually, we will reach an external node and add the new key-value pair (here encoded as a record 'newNode') as its right or left child, depending on the node's key

1.4 Traversal

- Once the binary search tree has been created, its elements can be retrieved in-order by recursively traversing the left subtree of the root node, accessing the node itself, then recursively traversing the right subtree of the node, continuing this pattern with each node in the tree as it's recursively accessed.
- As with all binary trees, one may conduct a pre-order traversal or a post-order traversal, but neither are likely to be useful for binary search trees.
- An in-order traversal of a binary search tree will always result in a sorted list of node items (numbers, strings or other comparable items)

1.5 Reference

https://www.cs.usfca.edu/~galles/visualization/BST.html

2 Assignment

2.1 Assignment 01

- Given a hash table with m=11 entries and the following hash function h1 and step function h2:
  - h1(key) = key mod m
  - h2(key) = {key mod (m-1)} + 1
- Insert the keys {22, 1, 13, 11, 24, 33, 18, 42, 31} in the given order (from left to right) to the hash table using each of the following hash methods:
  - Chaining with h1
  - Linear-Probing with h1
  - Double-Hashing with h1 as the hash function and h2 as the step function h(k,i) = (h1(k) + i*h2(k)) mod m
- For the previous h1 and h2, is it OK to use h2 as a hash function and h1 as a step function?
- Why is it important that the result of the step function and the table size will not have a common divisor?
- Suggest how to implement Delete(k) function for a hash table, when using open addressing

2.2 Assignment 02

- Write a small program to do the following tasks:
  1. Load a list of integer numbers from “input.txt” and create a binary seach tree to store them.

2. Output the tree (pre- order) to “out1.txt” file
3. Output the tree (post-order) to “out2.txt” file

2.3 Assignment 03

Write a small program, it load a list of integer numbers from “input.txt”. After it output to “out.txt” file the list whose item is a number & occurrences.

2.4 Assignment 04

- Write a small program to do the following tasks
  1. Load a list from “emotional-dictionary.txt” and create a binary seach tree to store them.
  2. Search by key
  3. Search by content

2.5 Assignment 05

- Write a small program to do the following tasks
  1. Load a list from data of dictionary and create a binary seach tree to store them.
  2. Search by key
  3. Search by content
- Data (unicode): goldendict, colordict, *.dict, *.idx