COMP2521 20T2 Data Structures and Algorithms

Week 03 Tutorial

Compiling, ADTs, Binary Search Trees

For the BSTree questions below, consider the following implementation of the `BSTree` data structure.

```
typedef struct BSTNode *BSTree;
typedef struct BSTNode {
    int value;
    BSTree left;
    BSTree right;
} BSTNode;
```

1. What does each of these `gcc` options do:

    a. `-E`
    b. `-c`
    c. `-o`
    d. `-g`
    e. `-O3`

2. Consider the following `Makefile`:

    ```
    tw : tw.o Dict.o stemmer.o
    tw.o : tw.c Dict.h WFreq.h stemmer.h
    Dict.o : Dict.c Dict.h WFreq.h
    stemmer.o : stemmer.c stemmer.h
    ```

    If none of the `*.o` files exist, explain how `make` decides which (implicit) rules to apply.

3. At the beginning and end of the `X.h` file, we typically place the following:

    ```
    #ifndef X_H
    #define X_H
    // ... rest of X.h file ...
    #endif
    ```

    Explain the purpose of this and how it achieves this purpose.

4. For each of the sequences below

    - start from an initially empty binary search tree
    - show tree resulting from inserting values in order given

    (a) 4 2 6 5 1 7 3

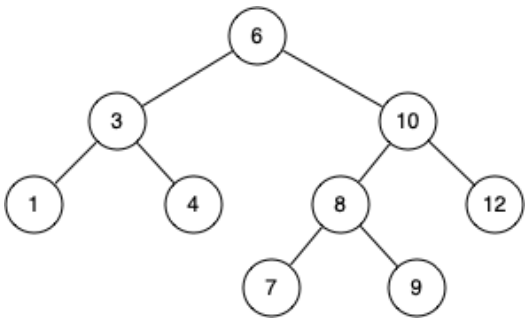
    (b) 5 6 2 3 4 7 1

    (c) 1 2 3 4 5 6 7

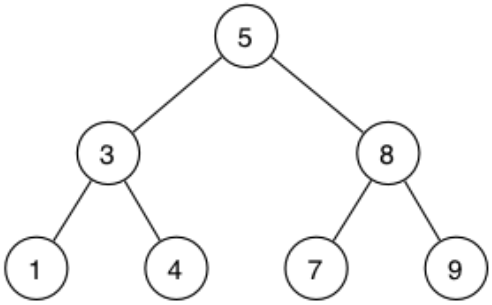
    Assume new values are always inserted as new leaf nodes

5. Give at least two other insertion orders that would produce the same tree as part (a) in the previous question. What do such orders have in common?

6. Show how the following tree would change if we do a right rotation on the node containing 10 followed by a left rotation on the node containing 6.



7. Consider the following tree and its nodes displayed in different output orderings:



Infix Order	1 3 4 5 7 8 9
Prefix Order	5 3 1 4 8 7 9
Postfix Order	1 4 3 7 9 8 5
Level Order	5 3 8 1 4 7 9

What kind of trees have the property that their infix output is the same as their prefix output?

Are there any kinds of trees for which all output orders will be the same?

8. Write a recursive function to count the total number of nodes in a tree. Use the following function header:

```
// count #nodes in Tree
int BSTreeNumNodes(BSTree t) { ... }
```

9. Write a recursive function to compute the height of a tree. The *height* is defined as the length of the longest path from root to a leaf. The *path length* is a count of the number of *links* (edges) on the path. Use the following function header:

```
// compute height of Tree
int BSTreeHeight(BSTree t)  { ... }
```

10. Implement the following function to count number of internal nodes in a given `BSTree`. Reminder: an internal node is one with at least one non-empty subtree. Use the following function interface:

```
int countInternal(BSTree t) { ... }
```

11. Implement the following function that returns the depth of the node containing a given key if such a node exists, otherwise the function returns -1 (when a given key is not in the binary search tree). The depth of a root node is zero. Use the following function interface:

```
int nodeDepth(BSTree t, int key) { ... }
```

12. Implement the following function to compute the *maximum branch length* of a `BSTree`, where the branch length of a `BSTree` is defined as the number of links (edges) on the longest path from the root to a leaf. Use the following function interface:

```
int BSTreeMaxBranchLen(BSTree t) { ... }
```

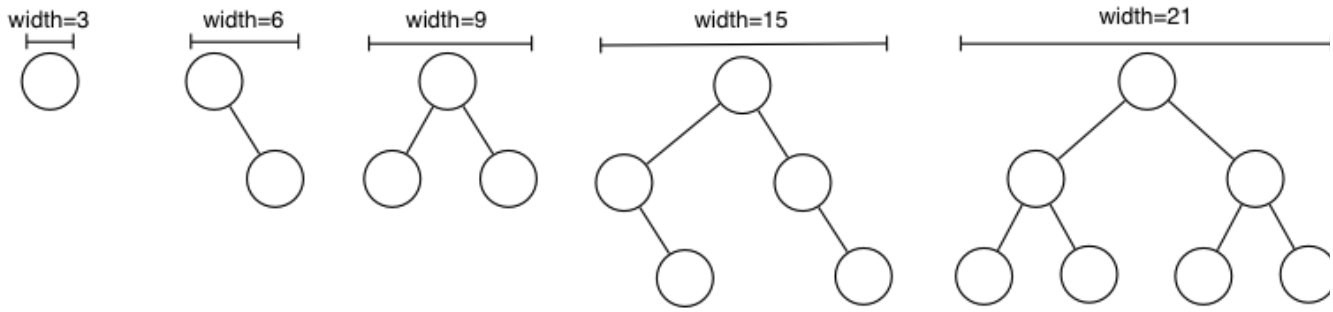
13. Computing the height/depth of trees is useful for estimating their search efficiency. For *drawing* trees, we're more interested in their *width*. Add a new function to the `BSTree` ADT which computes the *width* of a tree, where tree *width* is defined as follows:

    - an empty tree has width zero
    - a tree with just one node has width three (to keep reasonable spacing)
    - all other trees have width which is three more than the combined width of the subtrees

    Use the following function interface:

    ```
    int BSTWidth(BSTree t) { ... }
    ```

    The following diagrams show the widths for some simple trees:



COMP2521 20T2: Data Structures and Algorithms is brought to you by
the School of Computer Science and Engineering at the University of New South Wales, Sydney.
For all enquiries, please email the class account at cs2521@cse.unsw.edu.au
CRICOS Provider 00098G