CSC 236 T15 on Trees

**This is a team assignment designed as an in-class activity.**

Note that this team assignment was inspired by an article by Kuba Karpierz and Steven A. Wolfman <http://dl.acm.org/citation.cfm?id=2538862.2538902&CFID=447895813&CFTOKEN=40660136> and also an assignment created by Matt Lang, of Moravian College. See <http://cspogil.org/tiki-index.php> for more information.

**Directions for use:**

* To use this form effectively, sign into a Google account.
* Then under “File” choose “Make a Copy” in order to be able to edit.
* Share with all team members, but allow Recorder to do the recording.
* Each yellow box should be filled with an appropriate team response..
* Download as *yourteamname-T15.docx* and upload to Moodle

First, rotate and confirm the new roles and complete the form below for assigned roles of each member.  Try to assign a role to each member that they have not yet had.

## Member Roles

* If you have only four people, combine Quality Control Officer & Process Analyst
* If you have only three people, also combine Recorder & Spokesperson

|  |  |
| --- | --- |
| **Team Roles** | **Member Name** |
| **Facilitator:** | **Jon Jeffrey** |
| **Recorder:** | **John Hellrung** |
| **Spokesperson:** | **J.T. Spradlin** |
| **Quality Control Officer:** | **Eric Rhodes** |
| **Process Analyst:** | **Eric Rhodes** |

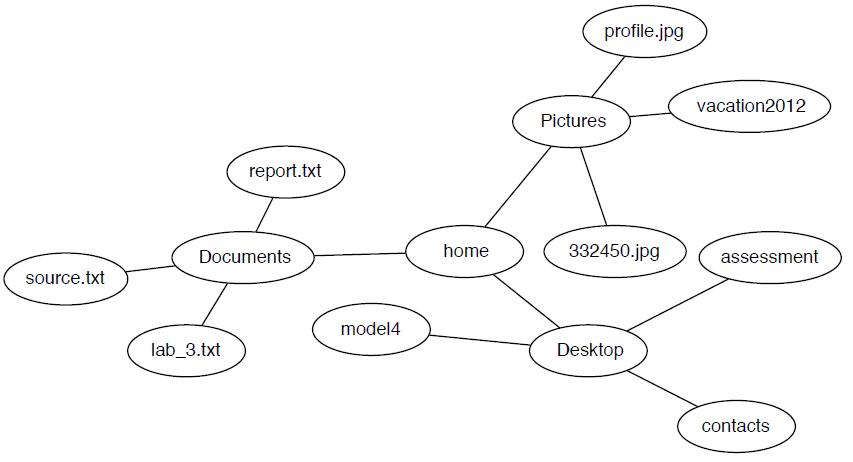
## On Trees

Recall that a **tree** is a widely used data structure that simulates a hierarchical tree structure, with a root value and a set of linked nodes which are each subtrees.  Note that an important property of a tree structure is that if you start at the root node, there is a unique path to any given node in the tree.

A **binary tree** is a tree in which each node has a value and at most two child nodes, usually called left child and right child which are themselves the roots of subtrees.

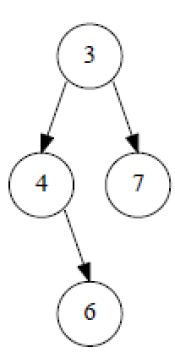
A **binary search tree** is just a binary tree with the extra property that holds for every node in the tree: the values in the left subtree are less than the value at that node and the values in the right subtree are greater than the value at that node .

Consider the following image:



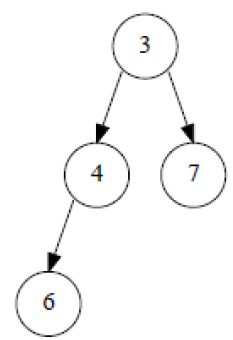
|  |  |
| --- | --- |
| The above is not drawn as we expect a tree, but could it be represented as a tree if directions were added to all of the edges? Why or why not? | It could be a valid tree as long as none of the arrows went against the flow of the tree(none were backwards). It would still look awful, but it’d be a valid tree. |
| If it can be represented as a tree, which of the nodes *could be* used as the root node? Name all of the nodes which *could be* the root. | Home, Desktop, Documents, and Pictures could all be used as the root node. |
| As a binary tree? Why or why not? | No. This is because some of the elements have more than two children. For instance, the element ‘documents’ has 4 connections. If one of them is its parent, then that leaves 3 as its children. |
| As a binary search tree?  Why or why not? | No, because it is not even normal tree, let alone a binary search tree. |

Consider:



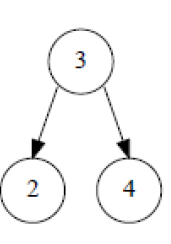
|  |  |
| --- | --- |
| Could the above be represented as a tree? a binary tree? a binary search tree? Explain. | Tree- yes, it has a valid flow from a root. Binary tree- yes, no parent has more than 2 children. Binary search tree- no, the root has a child on both sides that are both higher than itself. The left must be lower to be a binary search tree. |

Consider:



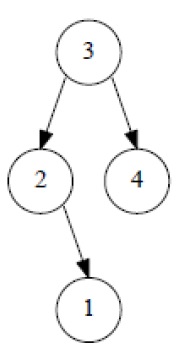
|  |  |
| --- | --- |
| Could the above be represented as a tree? a binary tree? a binary search tree? Explain. | It could be represented as a tree because it has a valid flow from the root. It could be represented as a binary tree because there is a valid flow from the root and each child has 2 or less children. It couldn’t be a binary search tree because the leaf holding 6 is greater than both 3 and 4 and should be placed to the left of 7 in order to be a valid binary search tree. |

Consider:



|  |  |
| --- | --- |
| Could the above be represented as a tree? a binary tree? a binary search tree? Explain. | Tree- yes, valid flow from root. Binary tree- yes, root has two children. Binary search tree- yes. Root has a less than child(2) and a greater than child(4). |

Consider:



|  |  |
| --- | --- |
| Could the above be represented as a tree? a binary tree? a binary search tree? Explain. | Yes, this representation above is a binary search tree but is not a correct search tree because it doesn’t follow the rule of a binary search tree in the second row with the leaf “1”.  Leaf “1” should be connect to the left of the tree branch “2”. |

## Tree Traversals

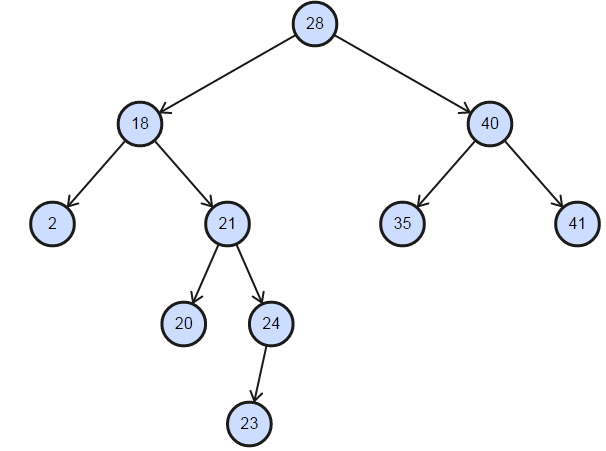
A **tree traversal** is a specific order in which to trace  the nodes of a tree. There are 3 common tree traversals.

1. in-order: left, root, right
2. pre-order: root, left, right
3. post-order: left, right, root

This order is applied recursively.

The **depth** of a node indicates how many edges are between it and the root node. The root node has a depth of zero.The **height** or depth of a tree is the maximum depth of all nodes in the tree.

Consider the following tree:



|  |  |
| --- | --- |
| What is the height of the above tree? | The height of this tree is 4. Because the level with 28 (the root) is zero, the level with 18 is level one, the level with 21 is two, the level with 24 is three and the level with 23 is four. |
| List the order of nodes visited in an in-order traversal of the above. | 2 -> 18 -> 20 -> 23 -> 24 -> 21 -> 28 -> 35 -> 40 -> 41. |
| List the order of nodes visited in an pre-order traversal of the above. | 28 -> 18 -> 2 -> 21-> 20 -> 24 -> 23 -> 40 -> 35 -> 41 |
| List the order of nodes visited in an post-order traversal of the above. | 2-> 20-> 23 -> 24 -> 21 ->18 ->35->41->40 ->28 |

## Efficiency

Let’s compare the efficiency of some common operations on linked lists and trees where N is the number of elements in the linked list or tree.

|  |  |  |
| --- | --- | --- |
| Operation | Efficiency in Linked List and explanation | Efficiency in Binary Tree and explanation |
| find item | O(N) because it may have to search all the way to the end. | O(h) where h = height of the tree because it may have to go all the way to a leaf node. On average this is O(log2(N)) assuming the tree is reasonably balanced. |
| delete item at cursor | O(1) because the program will only need to delete one item and it already has its location. | O(n) this is because the tree may have to transfer all of the nodes after deleting the root node. |
| insert item at cursor | O(1) this is because the program does not need to search for or move anything. All it needs to do is put the item where it’s already pointing. | 0(h - n) h = height of the tree and n is the location of the node is the tree. |

## Real-World Applications

Consider the following situations and determine which data structure seems most appropriate:

|  |  |
| --- | --- |
| **Application** | **True/False Can be easily represented as a tree data structure, and a brief explanation.** |
| An operating system maintains a disk's file system as a tree where file folders act as tree nodes. | **True:** because all folders are subfolders of the “Computer” and all folders are wholly inside of another folder so their contents are subtrees. |
| The biological taxonomy is a tree in which each level can be named: Kingdom, Phylum, Class, Order, Family, Genus, Species. | **False**, because if each level were simply named by that level of classification, this would just be a linked list. Kingdom -> Phylum -> Class, and so on. |
| Kentucky can be represented by a tree in which cities are nodes and roads are edges. | **False**, because two cities may connect to one city causing the tree to be connected and thus not a proper tree by definition. |
| The Internet can be represented as a tree structure in which each page is a node and the links are the edges. | **True**, home pages would be roots. Links to the other pages act as edges that take you to the node that they connect to. Home page would have multiple links to other pages(multiple edges to other nodes). This would hold true for typical websites. |
| The chapters, sections, and subsections of a book. | **True**, because book would be the root which branches off into chapters, the chapters branch off into sections, and the sections branch off into subsections. |

## Suggestions and Submission

Please offer any suggestions for improvement of this activity from the team:

|  |
| --- |
| **Suggestions for improvement** |
| We liked this assignment and corresponding lecture. It really helped to solidify these concepts. This was a great way to teach us about the material in a challenging way without making it too difficult. This assignment should be done in post-order traversal, starting from the bottom page to the top page. |

To submit, the Recorder will download as *yourteamname-T15.docx* and upload to Moodle while all other members will simply upload the name of the assignment (T15) and the names and roles of all team members.