Programming Assignment #2

CSCE221 - Data Structure and Algorithms
Instructor: Mohsen Aznaveh
Texas A&M University

Due: Monday, March 30, 2020 11:59 pm

PA2 Instructions

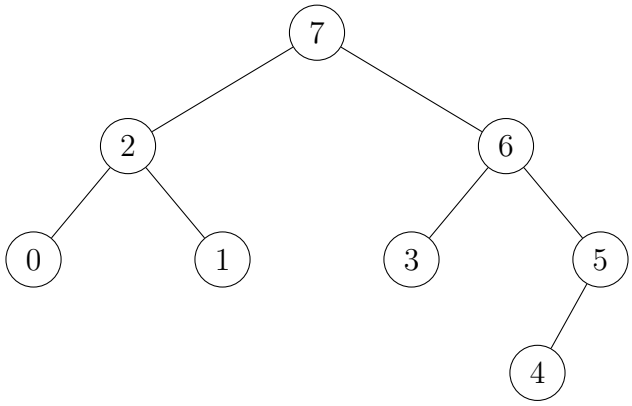
- Basic task before starting assignment
  - Download PA2.zip from piazza and Unzip it.
  - Move PA2 folder to your CSCE-UIN local GitHub repository.
  - Go to terminal, see the new changes in your git repository by running ***git status*** command.
  - Add new changes i.e. a new PA2 folder, using command ***git add filename***
  - Make new commit of this change using ***git commit -m "commit message"***
  - Push the change using ***git push*** command.
- Once above tasks are done, you have basically pushed the PA2 code base without solution on your GitHub.
- Now start working on the assignment, complete it and push the completed assignment on GitHub along with **makefile** and **PDF** file which contains answers to few non-coding based questions and your explanations as part of this assignment.
- The PDF can be hand-written then scanned, or LaTeX, MS Word, what-ever you prefer. But only include the one and only one final PDF, not *docx, *tex, etc.

Important Notes

- Following actions will led to 0 points for this assignment:
  - Manually uploading your assignment using upload option on GitHub. This can be detected.
  - No makefile submission.
  - Using g++ *.cpp in your makefile.
  - Code plagiarism and cheating. It will be checked using MOSS.

Question 1

(20 points)
Consider a data structure that uses an array to save a tree. The tree has N nodes and they are numbered 0 to $N-1$. In this data structure element ith in the array store the parent number. For example this tree:



It is stored as

2	2	7	6	5	6	7	-1

- In the file `q1.cpp` add the code to print the children of a given node p. Here is the signature of the function

```
void child(int p, const std::vector<int> Parent)
```

  `p` is the node that we want to know the children and `Parent` is the array that all the children are saved. For above example if we call function `child(6,Parent)` 3, 5 must be printed.

- What is the running time order of `child` function?

Question 2

(20 points)
File `BinarySearchTree.h` is included with this document. You can find the explanation of this file in the book

- Method `printTree` is already implemented. What kind of a tree traversal is that?

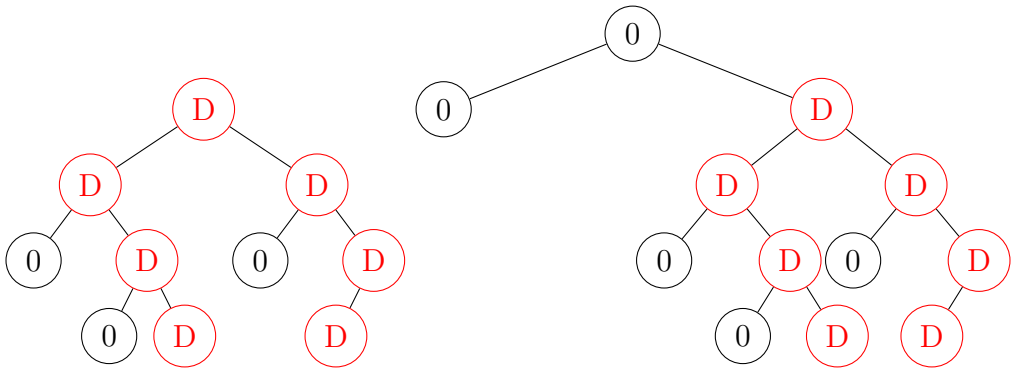
- In the same fashion as `printTree` you can write methods to print the tree using another way of tree traversal. `preorder` is added to this file. Complete the private method with the signature `void preorder (BinaryNode *t, ostream & out) const` so it prints the tree in a preorder fashion.

- The public method `preorder` is already implemented. Look at the `q2.cpp` and explain what happens when we call this method.

- Tree traversals can be used to find depth of the tree. There are two methods `max_depth` and `min_depth` in the `BinarySearchTree.h`. Add the code to compute maximum and minimum depth of the tree.
- If the tree is an AVL tree what is the maximum difference between max and min depth. In the file `q2.cpp` there is a test for adding random nodes to the tree. How much is a tree unblanced after adding 100000 random nodes?
- The method `remove` replace the minimum of right child subtree for a node with two children. As we discussed in the class we can use the maximum of left subtree. Add the code to method `remove_left` to do so. Add test for the new method to see how it works.

Question 3

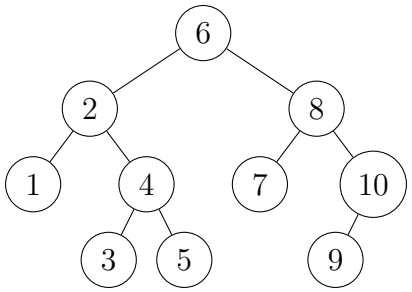
(20 points)
The diameter of a tree (sometimes called the width) is the number of nodes on the longest path between two end nodes. The diagram below shows two trees each with diameter seven, the nodes that form the longest path are shaded in red (note that there is more than one path in tree of length seven, but no path longer than seven nodes).



- In `BinarySearchTree.h` file, implement `int diameter (BinaryNode *t ) const` which will return the diameter of a Binary Tree.
- Also mention the space and time complexity of your algorithm in **PDF** file.

Question 4

(30 points)
Level Order Traversal is the Breadth First Search of a Binary Tree where you traverse nodes each level from top to bottom and in each level, you move from left to right.

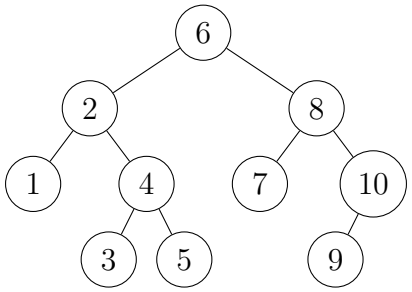


Level Order Traversal of above Tree: 6 2 8 1 4 7 10 3 5 9

- Implement `void levelorder( BinaryNode *t, ostream & out ) const` which will print the level order traversal of a Binary Tree in `BinarySearchTree.h`.
- **Note**: Implement your own Queue Data Structure instead of using C++ STL. Queue implementation is part of your Lab exercise.

Question 5

(30 points)
In a Binary Tree. The lowest common ancestor (LCA) between two nodes n1 and n2 is defined as the lowest node in the tree that has both n1 and n2 as descendants (where we allow a node to be a descendant of itself).



- Follwoing are few LCA for above tree:
  - LCA(2,8) = 6
  - LCA(1,5) = 2
  - LCA(4,9) = 6
  - LCA(3,5) = 4
- Implement `void LCA(BinaryNode *t, int x, int y, ostream & out) const` which will print LCA of 2 nodes in `BinarySearchTree.h`. If there is no LCA for given 2 nodes, print "Do not exist".
- Also mention the space and time complexity of your algorithm in **PDF** file.