# Data Structure: Homework #2

Due Date: Oct. 12th, 2010

Shumin Guo

October 21, 2010

1. (10 points) Describe in pseudo-code, a linear-time algorithm for reversing a queue Q. To access the queue, you are only allowed to use the basic functions of the queue ADT defined as follows (Hint: Using a stack, the basic stack functions defined in the textbook and in the class).

```
class Queue {
public:
    int size();
    bool isEmpty();
    Object& front()
    void enqueue(Object o);
    Object dequeue()
};
```

**Pseudocode With Queue**

**Algorithm** reverseQueue(Q):

**Input:**Queue(Q) with length N($N \geq 0$).
**Output:**Reversed Queue.
$x \leftarrow N$
**while** $x > 0$ **do**
    $X \leftarrow$ Q.dequeue()
    Q.enqueue(X)
    $x \leftarrow x - 1$
**end while**

**Pseudocode With Stack**

**Algorithm** reverseQueue(Q):

**Input:**Queue(Q) with length N($N \geq 0$).
**Output:**Reversed Queue.
$S \leftarrow$ New(Stack) AND Size(S) $\geq$ N
**while** Q Not Empty **do**
    $X \leftarrow$ Q.dequeue()
    S.Push(X)
**end while**
$x \leftarrow N$
**while** S Not Empty **do**
    $X \leftarrow$ S.Pop()
    Q.enqueue(X)
**end while**

2. (20 points) Let T be the binary tree as shown in the following figure. (See Figure 1)

   (a) Give the output of the preorder tree traversal of T. A B D G H E C F

   (b) Give the output of the postorder tree traversal of T. G H D E B F C A

   (c) Give the output of the inorder tree traversal of T. G D H B E A F C

   (d) Give the output of the level order tree traversal of T. A B C D E F G H

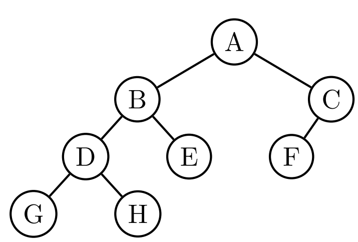


Figure 1: Tree Struture for Question Two.

3. (10 points) Draw a (single) binary tree T, such that

   (a) Each internal node of T stores a single character

   (b) A preorder traversal of T yields A B C D F G E

   (c) An inorder traversal of T yields C B F D G A E $\boxed{\text{See Result in Figure 2}}$

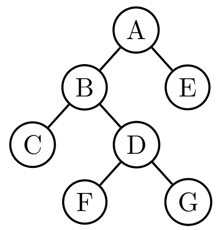


Figure 2: Binary Tree.

4. (10 points) Let T be the tree as shown in the following figure $\boxed{\text{See Figure 3}}$

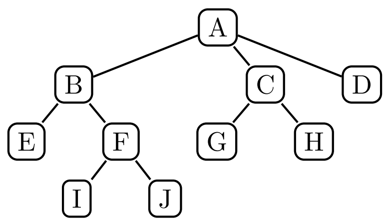


Figure 3: Tree Struture for Question Four.

   (a) Give the output of the preorder tree traversal of T. $\boxed{\text{A B E F I J C G H D}}$

   (b) Give the output of the postorder tree traversal of T. $\boxed{\text{E I J F B G H C D A}}$

5. (20 points) Answer the following questions (This question is for graduate students only, but undergraduate students are welcome to finish this question to get bonus points).

   (a) What is the minimum height h of a binary tree with n nodes. $\boxed{\lceil log_2 \frac{n+1}{2} \rceil}$

   (b) What is the maximum height h of a binary tree with n nodes. $\boxed{\text{n-1}}$

6. (20 points) Answer the following questions (This question is for graduate students only, but undergraduate students are welcome to finish this question to get bonus points).

   (a) What is the minimum number of external nodes for a full binary tree (proper binary tree) with height h? Justify your answer.

   **ANSWER:**
   In order to get a minimum number of external(leaf) nodes, we need to assign as many nodes as possible to be as internal nodes, but for a proper binary tree, a node either have zero or two children. So we can try to build a tree whose deepest nodes are two external nodes and for other levels there is only one external node for each level. The prototype of this kind of tree is shown in the picture below. And we can easily calculate the minimum number of external nodes to be $h+1$. $\boxed{\text{Please see Figure 4 for a sample tree.}}$

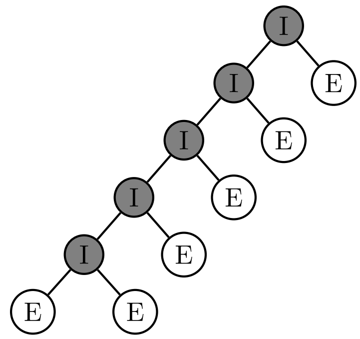

Figure 4: Tree structure for minimum number of external node.

(b) What is the maximum number of external nodes for a full binary tree (proper binary tree) with height h? Justify your answer.
**ANSWER:**
A proper binary tree with height h will have the maximum number of external nodes if this tree becomes a perfect tree. This means for level l of this tree it will have $2^l$ nodes in this level and for level h(the height), we will have $2^h$ nodes, which are all external nodes.

(c) Let T be a full binary tree (proper binary tree) with height h and n nodes, Show that $log_2(n+1) - 1 \leq h \leq \frac{n-1}{2}$.
**ANSWER:**
According to the answers of the above two questions, we know that for a proper binary tree, the height $h$ will be the highest if there are minimum number of external nodes($en = h+1$ where the number of internal nodes will be $in = h$) and $h$ will be the lowest when there are maximum number of external nodes($en = 2^h$ and at this time the number of internal nodes is $in = 2^h - 1$).
And according to the above description, we can get the relationship between the height of tree and the total number of nodes as follows:
Maximum height: $n = in + en = h + h + 1 = 2h + 1 \Rightarrow h = \frac{n-1}{2}$
Minimum height: $n = in + en = 2^h - 1 + 2^h \Rightarrow h = log_2(n+1) - 1$
To Sum up, we have: $\boxed{log_2(n+1) - 1 \leq h \leq \frac{n-1}{2}}$.
Done.

(d) For which values of n and h can the above lower and upper bounds on h be attained with equality?
**ANSWER:**
When $\boxed{n = 2h + 1}$ the upper bound can attain equality.
When $\boxed{n = 2^{h+1} - 1}$ the lower bound can attain equality.

7. (10points) For the set of $\{2, 6, 10, 9, 4, 1, 13\}$ of keys, draw binary search trees of height 2, 3, 4, 5, and 6. $\boxed{\text{See Result on Figure 5.}}$

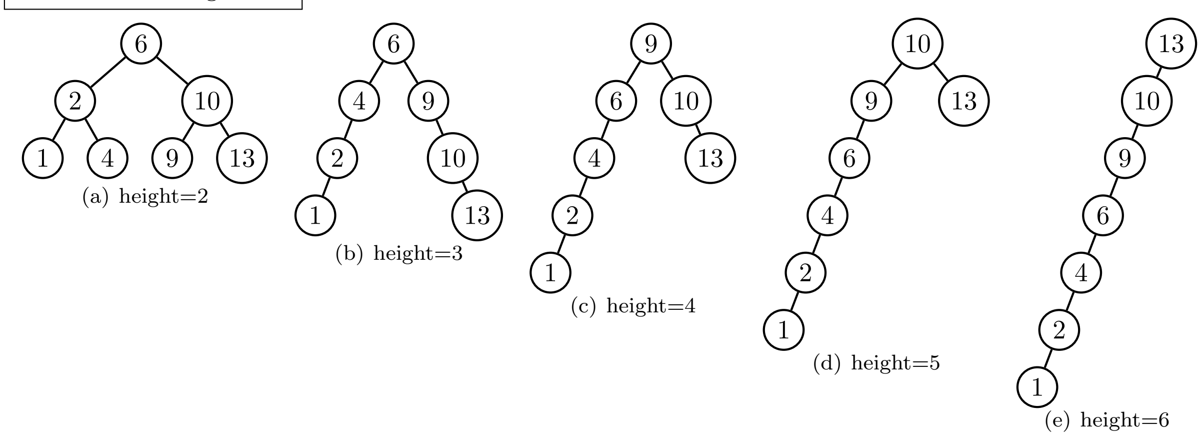


Figure 5: BST with different depth.

8. (15 points) Consider the Binary Search Tree (BST) to the following.

   (a) (5 points) Draw a circle around each node that would be visited in a search for the value 40.
   Please see Figure 6.

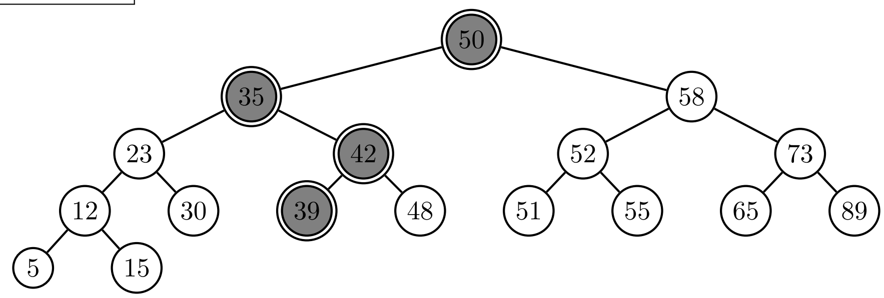


Figure 6: Tree Struture for Question Seven(a).

   (b) (10 points) Show how the tree would appear if the value 35 were deleted. Assuming duplicates are stored in the left subtree.
   **ANSWER:**
   We can pick the direct successor of 35 and place it onto 35's position, in this particular case, 39 is the direct successor. So the result is See Figure 7.

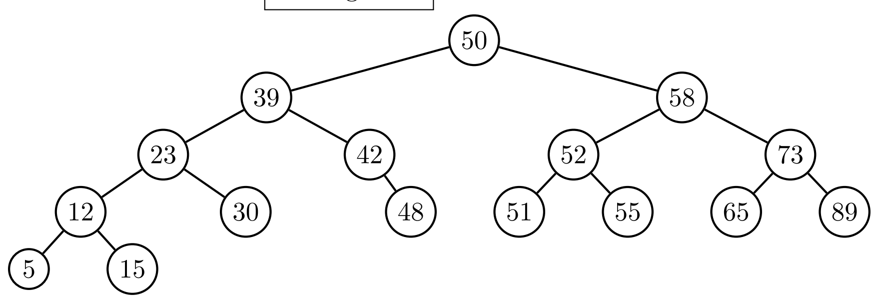


Figure 7: Tree Struture for Question Seven(b).

9. (10points). Suppose that myBST is an empty Binary Search Tree. The following operations are performed in the given order. Draw the resulting tree. Assuming duplicates are inserted in the left subtree.
   myBST.insert(20)
   myBST.insert(28)
   myBST.insert(40)
   myBST.insert(12)
   myBST.insert(25)
   myBST.insert(23)
   myBST.insert(24)
   myBST.insert(16)
   myBST.delete(23)
   myBST.insert(24)
   Please See Figure 8.

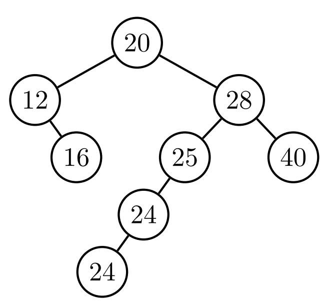


Figure 8: BST structure after insert.

10. (20 Points). Suppose you are given the following array, which represents a complete binary tree:

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Key | 12 | 25 | 72 | 49 | 64 | 58 | 52 | 92 | 8 | 19 | 43 | 27 | 98 | 16 | 37 | 33 | 6 |

Table 1: Array Representation of a Binary Tree.

(a) (5 points) Draw the complete binary tree represented by this array. Please See Figure 9

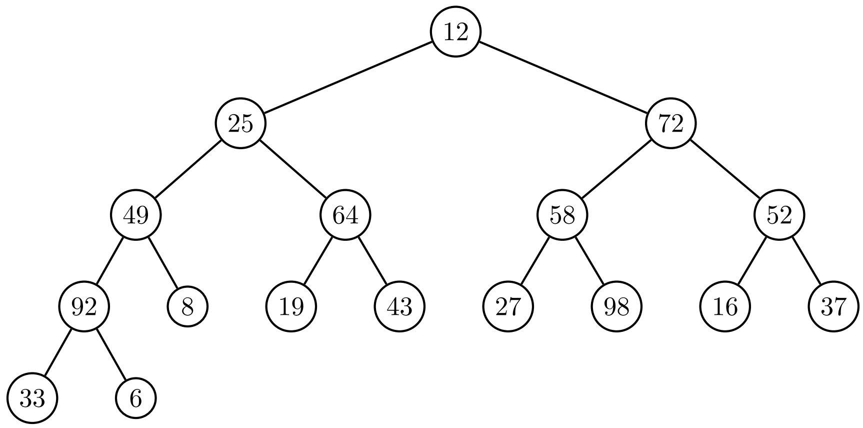


Figure 9: Complete Binary Tree Representation of List.

(b) (10 Points) Is this binary tree a binary heap? If not, build a heap out of this array using the BottomUpHeap() we learned in the class. No, this is not a binary heap. Please see result on Figure 10

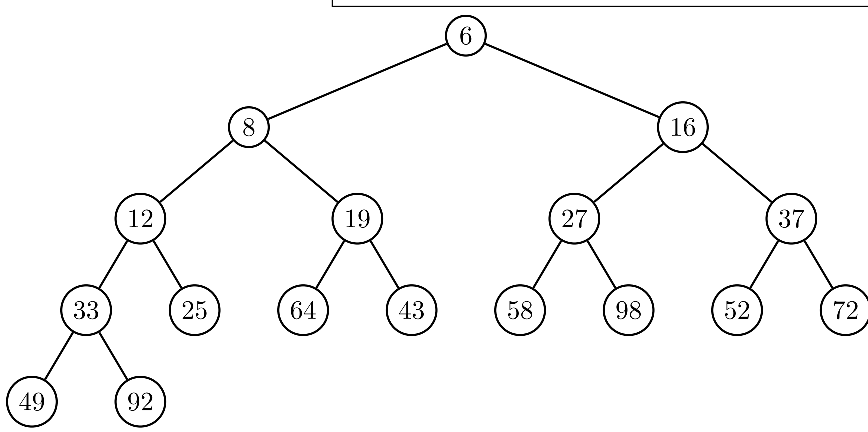


Figure 10: Complete Binary Tree Representation of List.

(c) (5 Points) List the values from the heap as they would be printed out by an inorder traversal of the heap, where the **visit()** function prints the value of the current node.

49, 33, 92, 12, 25, 8, 64, 19, 43, 6, 58, 27, 98, 16, 52, 37, 72