CS 308 Data Structures

Fall 2000 - Dr. George Bebis

Final Exam

Duration: 12:00 - 2:00 pm

Name:

1. **True/False** (2 pts each) **To get credit, you must (very briefly) for your answers !!**

(1.1) **T F** The maximum number of nodes in a tree that has L levels is 2^L

(1.2) **T F** A queue should be used when implementing Breadth First Search (BFS).

(1.3) **T F** The largest value of a binary search tree is always stored at the root of the tree.

(1.4) **T F** A complete tree is also a full tree.

(1.5) **T F** In a binary tree, every node has exactly two children.

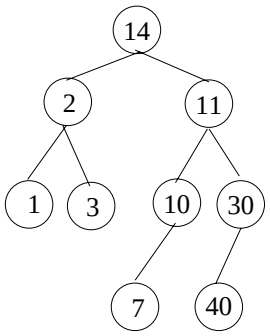
(1.6) **T F** Tree operations typically run in $O(d)$ time where d is the number of nodes in the tree.

(1.7) **T F** To delete a dynamically allocated tree, the best traversal method is *postorder*

(1.8) **T F** In a heap, the left child of a node is always less than the right child of a node.

(1.9) **T F** Implementing a priority queue using heaps is more efficient than using linked lists.

(1.10) **T F** The ancestors of node 10 are nodes 11, 2, and 14.



(1.11) **T F** Every binary tree is either complete or full.

(1.12) **T F** Heaps are useful for searching binary trees efficiently.

(1.13) **T F** A complete directed graph with 8 vertices has 64 edges.

(1.14) **T F** The linked-list implementation of a graph is more efficient in finding whether two vertices are directly connected or not.

(1.15) **T F** The order in which elements are inserted in a binary search tree in unimportant.

2. **Short answers** (3 pts each)

(2.1) What is the number of nodes in a full tree with L levels ? Prove it (show all the steps carefully).

(2.2) What is the maximum number of levels (height) of a tree with N nodes ? What is the minimum number of levels (height) of a tree with N nodes ? Justify your answers.

(2.3) Assume A is an array-based tree with 70 nodes.

What is the index of the first leaf node ?

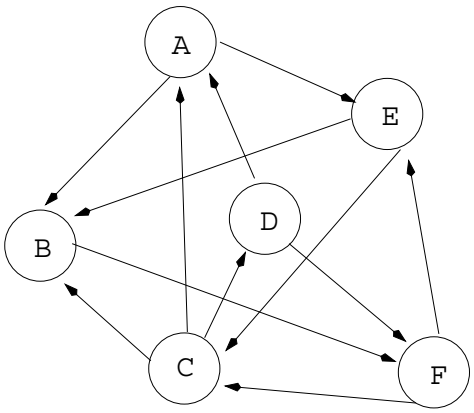
Who is the parent of A[50] ?

Who are the children of A[10] ?

How many leaf nodes does the tree have ?

(2.4) We have discussed two different approaches to implement a priority queue. Which are these two approaches ? How do they compare in terms of efficiency (time wise) ? Justify your answer.

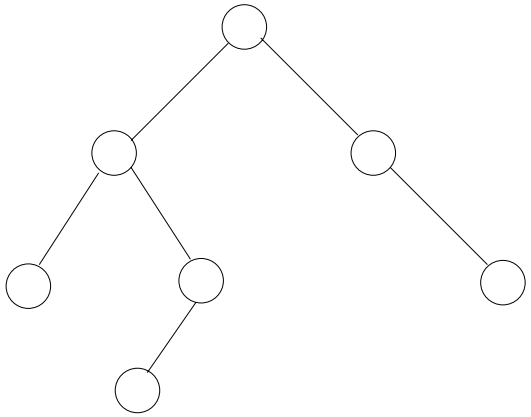
(2.5) Given the graph below, draw its adjacency matrix representation (store the vertices in alphabetical order)



(2.6) Using the graph in (2.5), is there a path from A to D ? Demonstrate how Depth First Search (DFS) solves this problem (to get credit, you need to show all the steps clearly).

(2.7) A graph can be represented using either an adjacency list or an adjacency matrix representation. Compare the two approaches (list their advantages/disadvantages)

(2.8) Label the following binary tree with numbers from the set {6,22,9,14,13,1,8} so that it is a legal binary search tree (choose the numbers in any order).

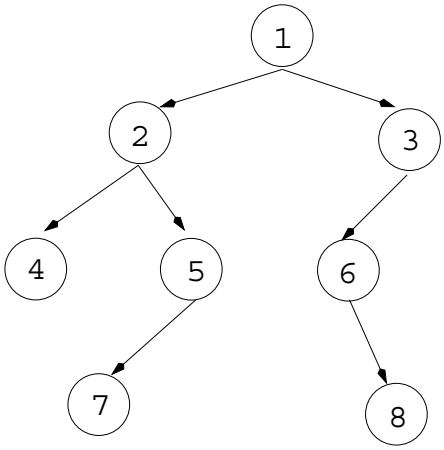


(2.9) Show how the tree in (2.8) would look like after each of the following operations:

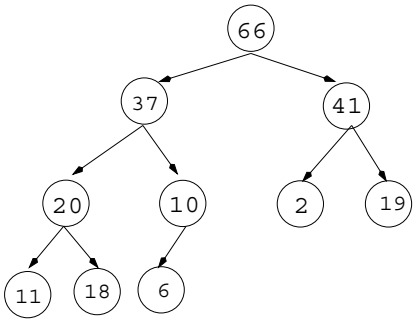
(i) delete 22

(ii) insert 34 (use the tree from step (i))

(2.10) Given the tree shown below, show the order in which nodes in the tree are processed by preorder traversal.



(2.11) A priority queue is implemented as a heap. Show how the heap shown below would look like after each of the following operations:



(i) pq.Enqueue(38);

(ii) pq.Enqueue(102); (use the heap from step (i))

(2.12) Continue problem (2.11)
(iii) pq.Dequeue(x); What is the value of x ? (use the heap from step (ii))

(v) pq.Dequeue(y); What is the value of y ? (use the heap from step (iii))

(2.13) Heaps are usually implemented using arrays. Why ? (be specific). What property of heaps allow us to implement them using arrays ?

(2.14) Suppose *N* elements are inserted in order, from smallest to largest, into a binary search tree. Describe the efficiency of searching for an element in the tree in terms of Big-O notation.

(2.15) Trace the function below and describe what it does.

```
template<class ItemType>
int Mystery(TreeType<ItemType> *tree, int &n)
{
 if(tree != NULL) {
   n++;
   Mystery(tree->left, n);
   Mystery(tree->right, n);
 }
}
```

3. **Code**

(3.1) (10 pts) Write a function that returns the largest value in a binary search tree (full credit will be given only to the most efficient solutions).

(3.2) (10 pts) Write a boolean member function *IsBST* that determines if a binary tree is a binary search tree.

(3.3) (5 pts) Give the pseudo-code of the Depth-First-Search approach. How is it different from the Breadth-First-Search approach ?