Generic Data Structure Viewer

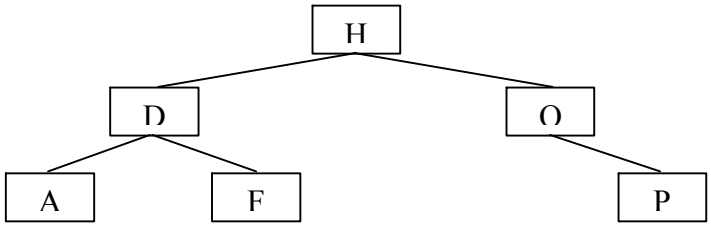
Definition of Array Representation of Data Structures

Binary Tree

The GDSV array representation of the binary tree follows the normal convention. The following is a description of the algorithm:

- The 0^{th} position is ignored (left *null*)
- Given the position of a node, i
  - Left child: $i * 2$
  - Right child: $i * 2 + 1$

Here is an example of the algorithm given the binary tree below:



All we do is go through the algorithm recursively. First we start with an empty array length 2^l (where l is the number of levels in the tree).

First we start with an array of size $2^3 = 8$:

index	0	1	2	3	4	5	6	7

In the case of the GDSV, this will be an array of Objects. Thus, for this example we will be filling the array with Character's. The 0^{th} position in the array is ignored.

Place "H" into the array at position 1 (root starts at position 1)

index	0	1	2	3	4	5	6	7
		H						

It's left child ("D") will be $i * 2$: $1 * 2 = 2$
It's right child ("O") will be $i * 2 + 1$: $1 * 2 + 1 = 3$

index	0	1	2	3	4	5	6	7
		H	D	O				

Continue this process recursively until all nodes have been placed into the array.

index	0	1	2	3	4	5	6	7
		H	D	O	A	F		P

You may notice that the array is a level-order representation of the binary tree.

CODE: The following is sample code that will create the array representation of a binary tree:

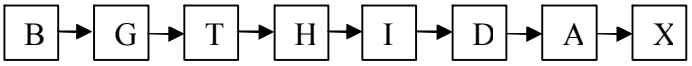
```
private Object[] arrayRepresentationOfDataStructure() {
   Object[] arrayRep = new Object[(int)Math.pow(2,height())];
   arrayRepHelper(arrayRep, myRoot, 1);
   return arrayRep;
}

private void arrayRepHelper(Object[] array, TreeNode root, int position) {
   if(root==null) return;
   array[position] = root.getValue();
   arrayRepHelper(array, root.getLeft(), position * 2);
   arrayRepHelper(array, root.getRight(), position * 2 + 1);
}
```

NOTE: The above code assumes that…

1. The binary tree is constructed using the AP TreeNode class
2. *myRoot* is a reference to the root TreeNode
3. *height()* has been coded to return the height of the binary tree (returns the number of levels)

LinkedList Tree

The linked-list is represented as a normal array; the linked-list is merely copied into an array. Example:



The above linked-list would become the array shown below:

index	0	1	2	3	4	5	6	7
	B	G	T	H	I	D	A	X

CODE: The following is sample code that will create the array representation of a linked-list:

```
import java.util.ArrayList;

private Object[] arrayRepresentationOfDataStructure() {
   ArrayList tempList = new ArrayList();
   for(ListNode currentNode = front;
      currentNode!=null;
      currentNode = currentNode.getNext()) {
      tempList.add(currentNode.getValue());
   }
   return tempList.toArray();
}
```

NOTE: The above code assumes that…

1. The linked-list is constructed using the AP ListNode class
2. *front* is a reference to the first node in the linked-list

Stack

Queue