Xiaofan Wu
Assign 8, part 1
11/20/15

A. Preorder
   DEIAGCJBFKH
B. Inorder
   IEGACJDFBHK
C. Postorder
   IGJCAEFHKBD
D. Level Order
   DEBIAFKGCHJ

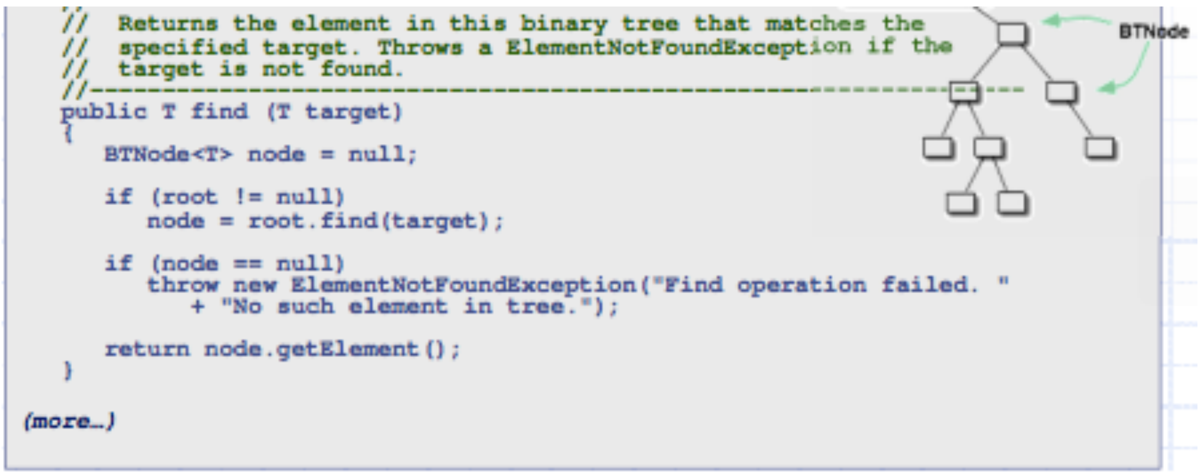
E. Computed Links

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
D	E	B	I	A	F	K			G	C			H									J

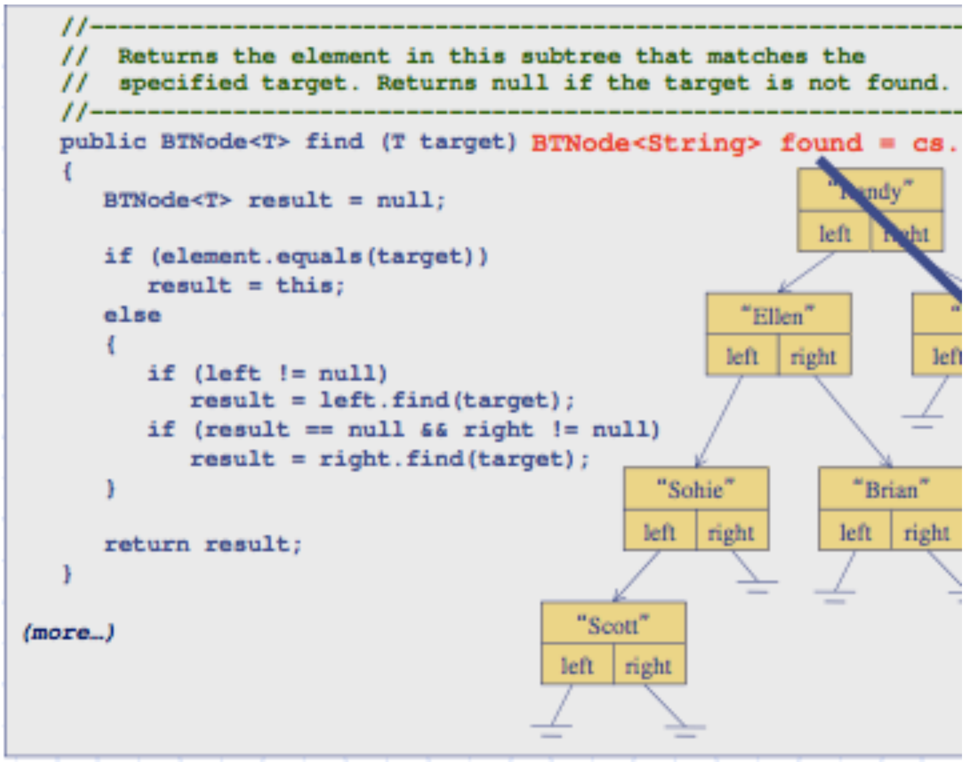
F.

0	1	2	3	4	5	6	7	8	9	10
A	B	C	D	E	F	G	H	I	J	K
6 2	5 10	-1 9	4 1	8 0	-1 -1	-1 -1	-1 -1	-1 -1	-1 -1	7 -1

Xiaofan Wu
Assign 8, part 1
11/20/15



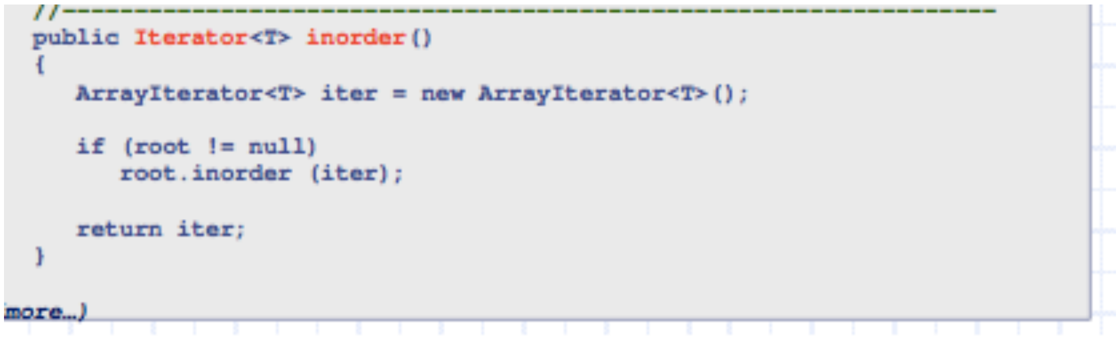
```
   //  Returns the element in this binary tree that matches the
   //  specified target. Throws a ElementNotFoundException if the
   //  target is not found.
   //-----------------------------------------------------------------
   public T find (T target)
   {
      BTNode<T> node = null;

      if (root != null)
         node = root.find(target);

      if (node == null)
         throw new ElementNotFoundException("Find operation failed. "
            + "No such element in tree.");

      return node.getElement();
   }

(more…)
```



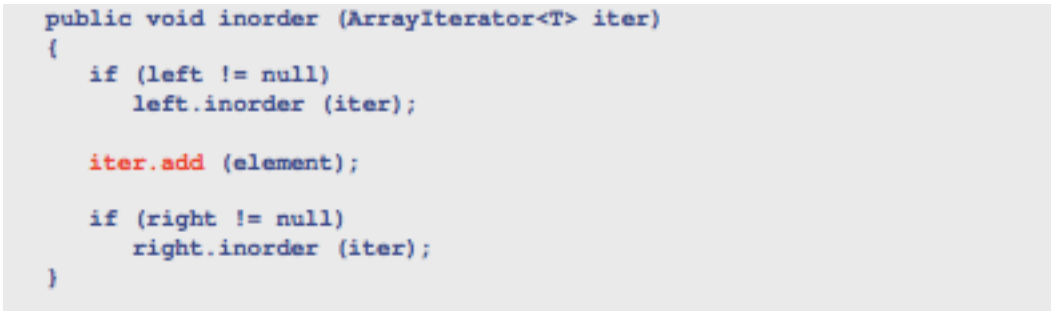
```
   //------------------------------------------------------------
   //  Returns the element in this subtree that matches the
   //  specified target. Returns null if the target is not found.
   //------------------------------------------------------------
   public BTNode<T> find (T target) BTNode<String> found = cs.
   {
      BTNode<T> result = null;

      if (element.equals(target))
         result = this;
      else
      {
         if (left != null)
            result = left.find(target);
         if (result == null && right != null)
            result = right.find(target);
      }

      return result;
   }

(more…)
```

G. O(n). The inorder method in the LBT uses the inorder method in BTNode. In the BTNode inorder method, the if statements and the add are O(1), but the function recursively calls itself and each time it execute statement that is O(1) (.add), so the total will be O(n). Although there are two recursive calls in BTNode method, it is still O(n) because the highest dominate and both are the same, so the highest would be O(n).

Xiaofan Wu
Assign 8, part 1
11/20/15



```
  //-----------------------------------------------------------------
  public Iterator<T> inorder()
  {
     ArrayIterator<T> iter = new ArrayIterator<T>();

     if (root != null)
        root.inorder (iter);

     return iter;
  }

more…)
```



```
   public void inorder (ArrayIterator<T> iter)
   {
      if (left != null)
         left.inorder (iter);

      iter.add (element);

      if (right != null)
         right.inorder (iter);
   }
```

H.O(n). The find method in the LBT uses the find method in BTNode. In the BTNode find method, the if statements and the add are O(1), but the function recursively calls itself and each time it execute statement that is O(1) (.add),, so the total will be O(n). Although there are two recursive calls in BTNode method, it is still O(n) because the highest dominate and both are the same, so the highest would be O(n).