CSC 421: Algorithm Design and Analysis

Sample Midterm Exam Questions

1. True or False?

_____ *Generate-and-test* is a brute force approach to solving a problem, in which candidate solutions are generated systematically and tested to see if they are legitimate solutions.

_____ If a problem has been proven to be *nP-hard*, then there cannot exist a polynomial-time algorithm for solving that problem.

_____ A *heap* is a binary search tree with additional properties that make it well-suited for implementing a priority queue.

2. Short Answer

A. In class, we discussed several examples that demonstrated a *space vs. time tradeoff*. Describe two (2) such algorithms that were able to improve performance time by sacrificing memory, or vice versa.

B. Consider the following algorithm for finding the median of a list of N unique numbers.

1. Create an empty TreeSet of Integers.
2. Traverse the list of numbers, adding each number to the TreeSet.
3. Iterate through the values in the TreeSet, stopping at the (N/2)nd number reached and returning it.

What is the Big-Oh complexity of this algorithm? Justify your answer.

3. Multipart Question

Consider a new, divide-and-conquer method that has been added to the `BinaryTree` class.

```
public class BinaryTree<E> {
  protected TreeNode<E> root;

  // CONSTRUCTOR & METHODS AS BEFORE

  /**
   * Stores the same value in every existing node in the tree (without
   * altering the structure of the tree).
   *    @param val the value to be stored
   */
  public void storeEverywhere(E val) {
      this.storeEverywhere(this.root, val);
  }

  private void storeEverywhere (TreeNode<E> current, E value) {
      // TO BE COMPLETED IN PART B
  }
}
```

A. Would objects of this class behave any differently if the field `root` were declared to be `private` instead of `protected`? If so, explain how. If not, what reason might there be for declaring it to be `protected`?

B. Complete the definition of the recursive `storeEverywhere` method, which stores the same value in every node of a tree (without altering the structure of the tree).

C. Assuming the tree is balanced, what is the recurrence relation that describes the behavior of your recursive `storeEverywhere` method?

D. What is the big-Oh efficiency of `storeEverywhere` when called on a balanced binary tree of N nodes? Justify your answer.