CIS 338 (Spring 2012)

Final, 5/22/12

Name (sign)

Name (print)

email

Question	Score
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
EX 1	
EX 2	
EX 3	

Question 1

For the operations and data structures below, indicate the worst-case asymptotic performance. If expected-case performance makes sense, and is different from worst-case, indicate it as well. Use common abbreviations for problem sizes. (For symbol tables, let ***n*** be the number of valid key-value pairs, and ***N*** be the total number of pairs ever inserted in the table. **ED** stands for eager deletion.)

a) insertion sort

b) merge sort

c) quick sort

d) heap sort

e) put(key), sequential search symbol table

f) get(key), open addressing hash table

g) keys(), binary search tree (**ED**)

h) rank(key), binary search tree

i) min(), two-three tree (**ED**)

j) floor(key), red-black tree

k) strongly connected components, (unweighted) directed graph

l) minimum spanning tree, weighted (undirected) graph.

m) shortest path, weighted directed graph with cycles but non-negative weights

Question 2

Draw the open-addressing hash table constructed by inserting a sequence of key-value pairs whose keys are: **128, 303, 486, 181, 106, 444, 358, 490**. Initially, the size of the table, `M`, is 5. Before entering the sixth pair (key **444**), resize the table so that `M` is 10. Use the following functions: `hash(x) = x % M; rehash(x) = x / 100;`

Question 3

Draw the 2-3 Tree constructed by inserting a sequence of key-value pairs whose keys are

h b d r c p a g i x o e m s n

Question 4

Draw the TST (ternary search trie) by inserting a sequence of key-value pairs whose keys are listed below. Circle the nodes that represent keys. *(First use scratch paper to build the data-structure, then draw it **sideways** on this page.)*

hate

cute

love

log

cat

hot

hops

cape

hope

lot

lost

hat

cot

loss

hop

Question 5

I) Regard G on the last page of this exam as a (unweighted, undirected) graph.

a) How many connected components does G have?

b) List G's bridges.

c) List G's articulation points.

d) What is the preorder of the vertices visited in a DFS starting at vertex j?

e) Indicate the shortest path from p to q

II) Regard G as an (unweighted) directed graph.

f) 1 of the components (part I a) is a DAG, list its vertices in topological order.

g) What is the postorder of the vertices visited in a DFS starting at vertex a?

h) Indicate the shortest path from p to q

III) Regard G as a weighted (undirected) graph.

i) Indicate the minimum spanning tree of the component containing vertex a.

j) Indicate the minimum spanning tree of the component containing vertex j.

k) Indicate the shortest path from p to q

IV) Regard G as a weighted directed graph.

l) Indicate the shortest path from p to q.

Question 6

What does it mean for a problem to be in P?

What does it mean for a problem to be in NP?

What does it mean for a problem to be NP-complete?

What would be the implications of finding an algorithm for discovering a minimal coloring for any undirected graph that ran in ~ c V^3 time (where V is the number of vertices in the graph)?

Question 7

Complete the method below to return the rank of **`d`** in an array **`a`**. *Assume* that **`a`** is sorted and that **`d`** is somewhere in **`a`**. Your method should be O(**`lg a.length`**).

```
int rank(double d, double[] a) {
    return rank(d, a, 0, a.length-1);
}
int rank(double d, double[] a, int lo, int hi) {
```

Question 8

Complete the three methods below to implement a Queue.

```
public class Queue {
  private Item[] data = new Item[1];
  private int count = 0; // number of items in the queue
  private int first = 0; // index of the oldest item
  // hint: index of next item is (first+count)%data.length
  public void enqueue(Item item) {









  }
  public Item dequeue() {









  }
  private void resize(int max) {
    assert count < max;









  }
}
```

Question 9

Fill in the blanks to implement a Quick sort class in Java.

```
public final class Quick {
    public static void sort(Comparable[] a) {
?
    }
    private static void sort(Comparable[] a, int lo, int hi) {
        if (hi <= lo) return;
?
?
?
    }
    private static int partition(Comparable[] a, int lo, int hi) {
        // choose a random pivot
        exch(a, lo, StdRandom.uniform(lo, hi+1));
        Comparable pivot = a[lo];
        int i = lo;
        int j = hi+1;
        while (true) {
            while(less(a[++i], pivot) && i<hi);
            while(less(pivot, a[--j]) && lo<j);
?
?
?
?
            exch(a, i, j);
        }
    }
    private static boolean isSorted(Comparable[] a) {
        for (int i=1; i<a.length; i++) {
?
?
        }
        return true;
    }
}
```

Question 10

```
private Node put(Node node, Key key, Value val) {
  if (null == node)
    return ?
  int cmp = key.compareTo(node.key);
  if      (cmp < 0) node.left  = ?
  else if (cmp > 0) node.right = ?
  else node.val = ?
  if (isRed(node.right) && !isRed(node.left))      ?
  if (isRed(node.left)  && isRed(node.left.left)) ?
  if (isRed(node.right) && isRed(node.left))       ?
  node.N = ?
  return ?
}
protected RedBlackNode rotateRight(Node node) {
  RedBlackNode n = (RedBlackNode) node;
  RedBlackNode m = (RedBlackNode) n.left;
  n.left  = ?
  m.right = ?
  m.N     = ?
  n.N     = ?
  m.color = ?
  n.color = ?
  return   ?
}
protected void flipColors(Node node) {
  ((RedBlackNode) node      ).color = ?
  ((RedBlackNode) node.left ).color = ?
  ((RedBlackNode) node.right).color = ?
}
```

Question 11

Complete the max() method of the TernarySearchTrie class. (Your code should be recursive but should not call any other methods except possibly size().)

```
public class TernarySearchTrie<Value> … {

    private class Node {
        char c;
        Value val;
        Node left;
        Node mid;
        Node right;
        int N = 0;
        Node(char ch, Value v) {}
    }

    public String max(String key) {
        if (null != emptyStringVal) return "";
        return max(root, "", 0);
    }

    private String max(Node node, String prefix, int i) {
        assert prefix.length() == i;
        if (null == node || 0 == size(node)) return null;
```

Question 12

Complete the following method to determine if the directed graph, **`g`**, contains a path from **`v`** to **`w`** (**`g`** maps each vertex **`x`** to the set of all vertices **`y`** such that **`<x, y>`** is an edge of the graph.)

```
boolean isPath(Vertex v, Vertex w, Map<Vertex, Set<Vertex>> g){
```

Extra Credit

EX 1) Briefly describe a SymbolTable implementation with keys that implement Java's **`Comparable`** interface having O(lg N) worst-case, and ~c expected-case, asymptotic performance for its **`put()`** and get() operations.

EX 2) Using the construction for reducing an instance of a 3-SAT problem to an instance of a Graph-Coloring problem, construct a graph that is 3-colorable iff the following Boolean formula is satisfiable: **(x V ~y V ~z) & (w V ~x V y)**.

EX 3) Complete the **`rank()`** method of the TernarySearchTrie class.

```
public int rank(String key) {
    if (0 == key.length() return 0;
    int n = rank(root, key, "", 0);
    return (null != emptyStringVal) ? n+1 : n;
}

private int rank(Node n, String k, String prefix, int i) {
```