Midterm 2: Compsci 201

Tabitha Peck

November 13, 2013

Print your name and NetID legibly in ALL CAPITAL letters. Make sure that we can clearly determine L vs. 1 and S vs. 5. It will effect your grade if you do not follow these instructions or we cannot read your name or netID.

Name: __________________________________________

NetID/Login: __________

Honor code acknowledgment (signature) ____________________________

This test has 13 pages (with a help page at the end), be sure your test has them all. Do NOT spend too much time on one question — remember that this class lasts 75 minutes.

In writing code you do not need to worry about specifying the proper import statements. Don’t worry about getting function or method names exactly right. Assume that all libraries and packages are imported in any code you write.
<table>
<thead>
<tr>
<th>Problem</th>
<th>Value</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>34 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>18 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>16 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>12 pts.</td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>80 pts.</td>
<td></td>
</tr>
</tbody>
</table>
1 Lumber yard: 34 points

1.1 4 points

An in-order traversal on the binary-search tree above is: 4, 5, 8, 12, 15. What is the pre-order traversal?

1.2 4 points

Show where the values 1, 3, 11, and 7 (added in order) would be inserted into the binary-search tree above by adding nodes to the diagram.

1.3 4 points

You have a binary-search tree with \( N \) nodes, with values added in random order. Which of the following algorithms can be performed in \( O(\log N) \) time on the tree? You may select more than one.

(a) Inserting a node into the tree
(b) Computing the height of the tree
(c) Searching for a value in the tree
(d) Finding the maximum value in the tree
(e) None of the above

You have a binary-search tree with \( N \) nodes, with values added in numerical order. Which of the following algorithms can be performed in \( O(\log N) \) time on the tree? You may select more than one.

(a) Inserting a node into the tree
(b) Computing the height of the tree
(c) Searching for a value in the tree
(d) Finding the maximum value in the tree
(e) None of the above
The following code uses the TreeNode class:

```java
public class TreeNode {
    public int myValue;
    public TreeNode myLeft;
    public TreeNode myRight;

    public TreeNode(int val) { myValue = val; }
}
```

1.4 6 points

The method below, `oneChildCount` returns the number of nodes in the tree with only one child. The original tree at the beginning of this problem has 2 nodes with only one child, 4 and 12.

```java
public int oneChildCount(){
    return oneChildCount(myRoot);
}

public int oneChildCount(TreeNode cur){
    if(cur == null){
        return 0;
    }

    int total = oneChildCount(cur.myLeft) + oneChildCount(cur.myRight);

    if(cur.myLeft == null || cur.myRight == null){
        if(!( cur.myLeft == null && cur.myRight == null)){
            return total + 1;
        }
    }

    return total;
}
```

What is the recurrence relation and running time of this code for an unbalanced tree? The recurrence relation chart is below.

\[ T(1) = \] 
\[ T(N) = \] 

Running time: 

\[ T(n) = T(n/2) + O(1) \quad O(\log n) \]
\[ T(n) = (T(n - 1) + O(1) \quad O(n) \]
\[ T(n) = 2T(n/2) + O(1) \quad O(n) \]
\[ T(n) = 2T(n/2) + O(n) \quad O(n\log n) \]
\[ T(n) = T(n - 1) + O(n) \quad O(n^2) \]
1.5 8 points

Complete the method below that returns the total number of leaves in a binary tree. Calling `countLeaves` of the original tree at the beginning of **Problem 1** would return 2 as there are two leaves, 5 and 15.

```java
public int countLeaves()
{
    return countLeaves(myRoot);
}

public int countLeaves(TreeNode cur){
    // Method body
}
```

1.6 3 points

What is the minimum number of non-leaf nodes in an 8-node binary tree? For partial credit, cross off all answers you know to be incorrect.

(a) 1
(b) 2
(c) 3
(d) 4
(e) 5

1.7 3 points

What is the maximum number of nodes you can have in a tree of height $h$? For partial credit, cross off all answers you know to be incorrect.

(a) $2^h + h$
(b) $2^{h-1}$
(c) $2^h + 1$
(d) $2^{h+1}$
(e) $2^h - 1$
2 A heap of fun: 18 points

When a priority queue is implemented using a min-heap it is stored in an array such that the root of the
heap is at index 1.

2.1 3 points

Draw the heap that is stored in the following array:

```
3 7 4 9 8 6 12
```

2.2 3 points

Draw the resulting heap after adding the value 5.
2.3 3 points

If a node is found at index value $k$ what would be the index of the parent node?

2.4 5 points

Two common operations on a priority queue are $add(value)$ - add $value$ to the priority queue, and $remove()$ - remove and return the smallest value in the queue (the smallest value in a min-heap). What is the worst case running time for $add(value)$ and $remove()$ on a heap? Explain your answer.

$add(value)$: __________

$remove()$: __________

Explain your answer:

2.5 4 points

A priority queue can also be implemented using a balanced binary-search tree. Explain in brief why using a heap would be better than using a balanced binary-search tree to implement a priority queue.
3  aMAZEing: 16 points

In this problem you will be adding code to two methods, `getPath` and `isClear` in the class `ExamMaze`.

The recursive method `getPath` takes as parameters a `char[][] maze` and the row and column index locations of the starting and ending points in a maze, `curX, curY, endX, endY` such that the first time `getPath` is called, the starting point in the maze is `(curX, curY)` and the ending point is `(endX, endY)`.

The `maze` is saved as a `char[][]` such that a square foot of space is represented by a period, and a block is represented by a `#`. The state of the maze in row `i` and column `j` is stored in `maze[i][j]`. For the example maze below, `maze[0][0] = '.'` and `maze[1][2] = '#'`. (Spaces are added for easy reading.)

```
. . # . .
. . # # .
. . . . .
# # # . .
```

The method `getPath` will update `char[][] maze` with ANY possible path from `(curX, curY)` to `(endX, endY)` such that the path may only go through spaces represented by `.` not through `#`. Travel through the maze may only go North, South, East, or West, not diagonally.

The final path should be marked by ‘X’ along a possible route. For example `getPath(maze, 0, 0, 4, 4)` on the above maze could update `maze` as follows:

```
X X # . .
. X # # .
. X X X X
# # # . X
```

If there is no path from `start to end` then `getPath` will have no ‘X’s added to the maze.

The helper method `isClear` takes as parameters the current state of the maze, `char[][] maze`, and integers `x` and `y`. The method `isClear` returns `true` if location `(x, y)` is an open space in the maze (denoted by '.') and `false` otherwise.

Complete the code on the following page as denoted by the comments.
public class ExamMaze {

    public boolean getPath(char[][] maze, int curX, int curY, int endX, int endY) {
        if (curX == endX && curY == endY) {
            maze[curX][curY] = 'X';
            return true;
        }

        // check if (curX, curY) is clear

        // mark maze to save current state

        int[] travelX = {-1, 0, 0, 1};
        int[] travelY = {0, -1, 1, 0};

        // add your recursive calls here

        // backtracking - unmark your maze as needed

        return false;
    }

    public boolean isClear(char[][] maze, int x, int y) {
        if (x < 0 || x > (maze.length - 1))
            return false;

        // complete isClear

        return true;
    }
}


4 Not Bogo: 12 points

Below is code from three sorting algorithms that we discussed in class. Questions are on the following page.

Algorithm 1

```java
public void sort2(int[] a){
    if(a.length > 1){
        int half = a.length/2;
        int[] a1 = Arrays.copyOfRange(a, 0, half);
        int[] a2 = Arrays.copyOfRange(a, half, a.length);
        sort2(a1);
        sort2(a2);
        sort2Helper(a, a1, a2);
    }
}

private void sort2Helper(int[] array, int[] a1, int[] a2){
    int len1 = a1.length; int len2 = a2.length;
    int it1 = 0; int it2 = 0;
    for(int i=0; i < array.length; i++){
        if(it2==len2 || (it1 < len1 && a1[it1] < a2[it2])){
            array[i] = a1[it1];
            it1++;
        } else{
            array[i] = a2[it2];
            it2++;
        }
    }
}
```

Algorithm 2

```java
public void sort1(int[] a){
    for(int i=1; i < a.length; i++){
        int temp = a[i];
        int j;
        for(j= i-1; (j>= 0 && temp < a[j]); j--)
            a[j+1] = a[j];
        a[j+1] = temp;
    }
}
```

Algorithm 3

```java
public void sort3(int[] a){
    PriorityQueue<Integer> q = new PriorityQueue<Integer>();
    for(int i: a)
        q.add(i);
    int i = 0;
    while(!q.isEmpty()){
        a[i] = q.remove();
        i++;
    }
}
```
4.1 6 points
Given the array, \([1, 2, 3, 4, 5, 6, \ldots, n]\), where \(n\) is a large number, what is the running time for each algorithm?

Algorithm 1: ________________

_Briefly explain your answer_

Algorithm 2: ________________

_Briefly explain your answer_

Algorithm 3: ________________

_Briefly explain your answer_

4.2 6 points
Given a randomly ordered array of \(n\) elements, what is the running time for each algorithm?

Algorithm 1: ________________

_Briefly explain your answer_

Algorithm 2: ________________

_Briefly explain your answer_

Algorithm 3: ________________

_Briefly explain your answer_
String

- `.length()` Get the length of the `String`. O(1).
- `.charAt(i)` Get the `char` at index `i`. O(1).
- `.split(" ")` Split a string by spaces and store it in a string[].
- `.substring(i, j)` Get the substring between indices `i` and `j`. Index `i` is `inclusive`, and index `j` is `exclusive`. O(1). For example:

  ```java
  String x = "abcdefg";
  String y = x.substring(2, 4);
  // y now has the value "cd"
  ```

`ArrayList<T>` // Where `T` is a type, like `String` or `Integer`

- `.add(i, X)` Add element `X` to the list at index `i`. If no `i` is provided, add an element to the end of the list. Adding to the end runs in O(1).
- `.get(i)` Get the element at position `i`. Runs in O(1).
- `.set(i, X)` Set the element at position `i` to the value `X`. O(1).
- `.size()` Get the number of elements. O(1).

`HashSet<T>` // Where `T` is a type, like `String` or `Integer`

- `.size()` Compute the size. O(1).
- `.add(X)` Add the value `X` to the set. If it’s already in the set, do nothing. O(1).
- `.contains(X)` Return a `boolean` indicating if `X` is in the set. O(1).
- `.remove(X)` Remove `X` from the set. If `X` was not in the set, do nothing. O(1).

`HashMap<K, V>` // Where `K` and `V` are the key and value types, respectively.

- `.size()` Compute the size. O(1).
- `.containsKey(X)` Determines if the map contains a value for the key `X`. To get that value, use `.get()`. O(1).
- `.get(X)` Gets the value for the key `X`. If `X` is not in the map, return `null`. O(1).
- `.put(k, v)` Map the key `k` to the value `v`. If there was already a value for `k`, replace it. O(1).
- `.keySet()` Return a `Set` containing the keys in the map. Useful for iterating over. O(1).

To iterate over a `HashSet<T>`, use

```java
for (T v : nameOfSet) {
    // v is the current element of the set.
}
```

This can be combined with `HashMap`'s `.keySet()` to iterate over a `HashMap`. 