CSCI 102L - Data Structures

Final Exam

Spring 2011

(2:00pm - 4:00pm, Tuesday, May 10)

Instructor: Bill Cheng

(This exam is closed book, closed notes, closed everything.
No “cheat sheet” allowed.
No calculators, cell phones, or any electronic gadgets.)
CSCI 102L - Data Structures
Final Exam – Spring 2011
Instructor: Bill Cheng

Problems

Problem #1 (24 points) ____________
Problem #2 (12 points) ____________
Problem #3 (17 points) ____________
Problem #4 (16 points) ____________
Problem #5 (18 points) ____________
Problem #6 (16 points) ____________
Total (103 points) ________________

(This exam is closed book, closed notes, closed everything.
No “cheat sheet” allowed.
No calculators, cell phones, or any electronic gadgets.)
Problem 1. Short Answers (24 points)

(a) (3 points) A binary heap can be viewed as a nearly-complete binary tree structure. What is the Max Heap Property for a binary heap? (You may assume that values in the heap are unique.)

(b) (3 points) What is the organizational property obeyed by all the nodes in a binary search tree, i.e., the Binary Search Tree Property? (You may assume that values in the binary search tree are unique.)

(c) (3 points) Let the key of a node in a binary search tree be X (let’s also call this node, “node X”). (1) (1 pt) Please give a definition of Predecessor(X). (2) (2 pts) Given that you are at node X and that node X has two children, how do you find Predecessor(X) by traversing the children (you don’t need to write code to do this, just describe what you need to do)?

(d) (3 points) Continuing from part (c), (1) (1 pt) Please give a definition of Successor(X). (2) (2 pts) Given that you are at node X and that node X has two children, how do you find Successor(X) by traversing the children (you don’t need to write code to do this, just describe what you need to do)?
(e) (3 points) Why is it sometimes better to use a Red/Black Tree or AVL tree instead of a normal Binary Search Tree?

(f) (3 points) Ideal hash function and perfect hashing are not practical and general approaches to handling collisions for hash tables. Please name 3 practical and general approaches of handling collisions for hash tables. Please note that if two answers are very similar in their basic approaches, they will be counted as only one answer.

(g) (3 points) If you know that your hash table will be at most 50% full, what would be a good choice for handling collisions? Please note that more efficient answer (in both space and time) may get more points.

(h) (3 points) If you have a graph that’s represented as an Adjacency List, what type of query would make you wish that you used an Adjacency Matrix implementation instead? You can either give a general description of such a query or give an example.
(a) (12 points) A binary heap is an array-based data structure that can be viewed as a nearly-complete binary tree structure. You are given an array $v$ with the following elements (in sequence):

$$73 \quad 24 \quad 64 \quad 31 \quad 40 \quad 89 \quad 17 \quad 55$$

The elements in $v$ currently do not satisfy the Max Heap Property. Please make $v$ into a Max Heap by performing a sequence of Max-Heapify operations, starting from the bottom of the tree. Please draw all intermediate trees whenever elements are swapped in order to receive full credit. Feel free to add additional text and/or drawings if you want to help explain what you are doing. Please also show what the resulting array $v$ would look like.
In a binary tree, the **level** of a node is the number of branches from the root to the node. Below and on the left, you are given the class definition for a templated **binary tree node**:

```cpp
template <typename T>
class BTNode
{
    public:
        BTNode(T& d) : data(d)
        {
            left = right = NULL;  // invalid level to start
            level = (-1);
        }

    T data;
    BTNode<T>* left;
    BTNode<T>* right;
    int level;
};
```

You must **not** alter the class definition of BTNode (nor can you add anything to it). Above and on the right, you are also given a **partial** class definition for a templated **BinaryTree** class. Your job is to implement the **calc_levels()** function for the templated **BinaryTree** class.

(a) (8 points) Each BTNode has a **level** field which is initialized to −1. Please implement **calc_levels()**, using **recursion**, to correctly set the **level** field for **every node** in the binary tree. *(Hint: you may need a helper function.)*
The questions below are about different ways of traversing a binary tree to print out the node values in a certain order. Given that the ordering is the most important thing, if you give the “right answer” for the “wrong problem”, you will not get any credit for it. So, please make sure you give your answers in the correct order.

(b)  (3 points) What would be the output if you perform a pre-order traversal of the above binary tree?

c)  (3 points) What would be the output if you perform an in-order traversal of the above binary tree?

(d)  (3 points) What would be the output if you perform a post-order traversal of the above binary tree?
(e)  (This problem intentionally left blank so you can use this page for whatever you want.)
Consider the following **Binary Search Tree** (you can also use its equivalent short-hand notation on the right):

(a) (8 points) **Insert** the following values into the above **Binary Search Tree** in the specified order:

15, 13, 53, 52

Draw the resulting binary search tree (please use the short-hand notation to save space) after these insertions are done. Be sure to show some intermediate steps if you want partial credit (in case your final tree is incorrect).
Consider the following **Binary Search Tree** (you can also use its equivalent short-hand notation on the right):

![Binary Search Tree Diagram]

(b) (8 points) For this problem, please note that when you remove a node that has two children, you have two ways to go about it. If you choose one way, you must be consistent and choose the same way throughout this problem.

**Remove** the following nodes from the above **Binary Search Tree** in the specified order:

76, 46, 91

Draw the resulting binary search tree (please use the short-hand notation to save space) after these removals are done. Be sure to show some intermediate steps if you want partial credit (in case your final tree is incorrect).
(c) (This problem intentionally left blank so you can use this page for whatever you want.)
(a) (6 points) Consider the following graph:

If you use an adjacency matrix to represent the above graph, what would the adjacency matrix look like?

(b) (2 points) What is the Big O for the space complexity of adjacency matrix? (If you use a non-standard notation, please define the symbols you use clearly.)

(c) (2 points) What is the Big O for the space complexity of adjacency list? (If you use a non-standard notation, please define the symbols you use clearly.)
Consider the following graph and its adjacency list representation:

![Graph Diagram]

(d) (4 points) Write out the order in which the vertexes in the graph would be visited if a Breadth-First Search were started from vertex 3 given the above adjacency list representation of the graph. (NOTE: There is only one correct answer since you must use the given adjacency list and follow the rules of accessing such a data structure.)

(e) (4 points) Write out the order in which the vertexes in the graph would be visited if a Depth-First Search were started from vertex 3 given the above adjacency list representation of the graph. (NOTE: There is only one correct answer since you must use the given adjacency list and follow the rules of accessing such a data structure.)
(a) (8 points) You are given an input file full of English words with one word per line. The input file can contain duplicate words and it is your job to generate an output that counts the number of occurrences of each distinct word in the input file. For example, if the input file contained the word “goat” 26 times, “goat 26” should appear in your output exactly once. The input file is unordered and you may assume that the input file is well-formed so you do not have to validate the input. Your output does not need to be sorted. Please write a partial C++ program to perform this task. You may assume that the correct header files has been included. In order to receive any credit for this problem, you must use an STL map to store the count for each word.
(b) (8 points) You are given a vector $v$ of integers and a single integer value $X$. The size of the vector may be very large. You must determine if any pair of numbers in the list sum up to the value of $X$. Please write a boolean function `sum_check()` (please see function declaration below) that returns a boolean value indicating whether or not any two values in the list can sum to the value of $X$ (you don’t have to worry about providing the actual two values out of the list).

For example, given the list $[8, 4, 0, -2, 5, 11]$ and $X = 15$, `sum_check()` should return `true` because $4 + 11 = 15$. Given the list $[8, 4, 0, -2, 5, 11]$ and $X = 17$ `sum_check()` should return `false` because no pair of values in the list sum to 17. In order to receive any credit for this problem, you must use either an STL `set` or an STL `map` in your code to make your algorithm run in $O(n \log n)$ time where $n$ is the size of the vector $v$. If your algorithm is correct but runs slower than $O(n \log n)$, e.g., $O(n^2)$, you will receive at most 3 points.

```cpp
bool sum_check(vector<int>& v, int X)
```