CSC 120 Algorithms and Data Structures **Spring Semester, 2002**

Review Final Exam

Exam Thursday, 5/16, 1:30-3:30pm.

The exam is NOT open book exam, but you can bring one page (8 by 11in) with relevant math formulas, including the master method. DO NOT WRITE solutions to specific recurrences, or running times of specific algorithms, or complexities of specific functions. do NOT write solutions of problems, just formulas you think you might need and can't memorize. No outlines of algorithms! No calculators!

Review Tuesday, 5/14, 4-6pm, Adams 201C.

What the exam covers

Material we have covered in lectures, assigned textbook readings, homeworks, labs, quizzes, midterm exam.

Complexity:

Evaluating complexity of non-recursive and recursive algorithms.

Evaluating asymptotic complexity of functions in terms of Big-Oh, Big-Theta and Big-Omega

Ordering functions according to their asymptotic growth rate.

Using the master method.

Algorithms and data structures:

Algorithm design. Divide and conquer algorithms. Analyzing complexity of divide and conquer algorithms. Design an algorithm that solves a problem using a data structure that we studied.

Sorting algorithms and their complexities.

ADTs: list, stacks, queues, priority queue, dictionary, dynamic set. Operations, their complexities in various implementations (using arrays, linked lists, hash tables, heaps, BST). Best-case, worst-case, average-case complexities for the algorithms on various data structures.

Applications of ADTs.

Data Structures: linked lists, hash tables (and hash functions), heaps, trees, BST trees. Tracing various operations (like search, delete, insert, etc., on that data structures), including the tree traversal algorithms. Writing algorithms using that data structures, evaluating the complexity of your algorithms.

Graphs: definitions, representations. BFS searches on a graph (no tracing). You should know the main steps on which the algorithm is based, reason about the complexity.

This review set is not exhaustive.

Review homework problems, exam1 and quizzes for additional problems.

I will NOT ask you to trace merge sort or quick sort. I will NOT ask you to solve a recurrence using the recurrence tree method. I will NOT ask you to do a proof by induction.

All procedures describing algorithms should be given in pseudo-code, but they should be algorithms (i.e. be clear about the steps, and also make sure they terminate.)

1. True or False

$20n^3 + 10n \log\, n + 5$	is $O(n \log\, n)$
2^{100}	is $\Theta(1)$
$\log\,(n^x)$	is $O(\log\, n)$ *where* x *is constant*
$500 \log^5 n + n + 10$	is $O(n)$
$0.5 \log\, n$	is $\Theta(n)$

2. Order the following list of functions by the big-Oh asymptotic growth rate

$$6 \log\, n,\ \log \log\, n,\ 2^{\log\, n},\ n\sqrt{n},\ n^2$$

3. Express as a function of the input size n the worst-case running time $T(n)$ of the following algorithm

```
int Me(int n, int A[], int k)
{
    int tot = 0;

    for (int i=n; i>=0; i--) {
       if (i==k) {
          cout << i << endl;
          tot++;
       }
    }
    return tot;
 }
```

What is the asymptotic time complexity?

4. Give the worst-case and the best-case recurrences expressing the running time of the following algorithm manipulating a BST rooted at p.

```
int You(int &p, int a, int b)
{
    int tot = 0;
    if (p==0) return 0;
    if ( p > a && p < b) {
        cout << p << endl;
        tot++;
    }
    tot += You( Left(p), a, b);
    tot += You( Right(p), a, b);
    return tot;
}
```

What are the best- and the worst- case time complexities of this algorithm?

5. Using the master method estimate the complexity of the recursive algorithms which run times are expressed by the recurrences:

   (a) $T(n) = 21T(\frac{2n}{3}) + \Theta(n^2 \log^3 n)$

   (b) $T(n) = 9T(\frac{n}{3}) + \Theta(n^2)$

   (c) $T(n) = 10T(\frac{n}{2}) + \Theta(n^4 \log\, n)$

6. Fill in the following table with the asymptotic time complexities (use expected time complexities when appropriate). Be sure to indicate which time complexities are worst-case and which are expected-case. Let n be the number of elements in the ADT. Let m be the number of slots in the hash table.

	BST	heap	hashing with chaining	doubly linked sorted list
SEARCH (for key k)				
DELETE(k) (when given the location of k)				
MAXIMUM				

7. In a directed graph, the *in-degree* of a vertex is the number of edges entering it. Let n be the number of vertices, and m the number of edges in the graph, and let $d_I(v)$ be the in-degree of vertex v.

   Given an **adjacency matrix** representation of a directed graph, and a specified vertex v, how would you best compute the in-degree of v? Write your algorithm in pseudo code, analyze the time complexity of your algorithm.

8. An undirected graph is *bipartite* if its vertex set V can be partitioned into two sunsets S and T (i.e. $V = S \cup T, S \cap T = \emptyset$) so that every edge in E connects a vertex in S to a vertex in T.

   Describe a brute force algorithm for checking if an undirected graph is bipartite. What's the time complexity of your algorithm.

9. Show the hash table obtained when inserting the keys 26, 18, 19, 32, 4, and 65 into a hash table (in the given order) with collisions resolved by chaining. Let the table have 7 slots and let the hash function h be such that $h(26) = 3$, $h(18) = 6$, $h(19) = 0$, $h(32) = 0$, $h(4) = 3$, $h(65) = 3$.)

   You need just show the final result. Be careful to correctly show the order that would result within the chains/lists.

10. Give pseudo-code for a procedure DECREASE-KEY(i, k) that modifies the given binary heap A by decreasing the value of $A[i]$ to k. (If $A[i] \leq k$ then no change should be made.) You can use (without describing) any of the standard binary heap procedures that we've studied. You should give the most efficient implementation you can.

    Now analyze the asymptotic time complexity of your algorithm. Be sure to explain!

11. You are to implement a caller-id system which supports the operations given in b)-d) below. Pick a data structure that is best suited for the problem (i.e. the required operation will run as efficiently as possible). You should very clearly describe how the data structure is to be applied (e.g. what is used as the key, what the associated data is,...). Also be sure to give the complexity for <u>each</u> of the operations.

    (a) Describe your data structure choice first.

    (b) Give the time complexity analysis for inserting into the system of new item that consists of a phone number, address, and name.

    (c) Give the time complexity analysis for searching: given a phone number, return the address and name.

    (d) Give the time complexity analysis for deletion: given a phone number, remove the corresponding item from the system.

12. Write an algorithm to print all keys in BST between two given keys, k_1 and k_2.