CPS 231 Final Exam
Fall 2004
2:00-5:00pm, Friday December 17th
Closed book exam

NAME:______________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Max</th>
<th>Obtained</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>120</td>
<td></td>
</tr>
</tbody>
</table>

You can use any of the algorithms covered in class without describing them. It is ok to describe algorithms with words (and a few accompanying pictures if it helps the description). Give enough details, but keep it concise. It would be best if you can fit your solutions in the space provided. Feel free to use extra paper if necessary.

Good luck!
[15 points] **Problem 1 Part I:**

(a) The summation $\sum_{i=0}^{n} 4^i$ is $\Theta(\phantom{\frac{1}{2}})$. 

(b) The worst case running time of Quicksort is $\Theta(\phantom{\frac{1}{2}})$. 

(c) Any comparison-based sorting algorithm must have a worst case time of at least $\Omega(\phantom{\frac{1}{2}})$. 

(d) Assume you have $n$ positive integers in the range 1 through $k$. Counting Sort sorts the $n$ integers in $O(\phantom{\frac{1}{2}})$ time using $O(\phantom{\frac{1}{2}})$ additional space. 

(e) The median of an array of $n$ elements can be computed in $O(\phantom{\frac{1}{2}})$ time. 

(f) Consider a binary search tree. Searching for an element in the tree takes at most $O(\phantom{\frac{1}{2}})$ time. 

(g) If a data structure supports an operation FOO such that a sequence of $n$ FOO’s takes $O(n)$ time in the worst case, then the amortized time of a FOO operation is $\Theta(\phantom{\frac{1}{2}})$. 

(h) If a data structure supports an operation FOO such that the amortized time of a FOO operation is $\Theta(\log n)$, is it true that the worst-case running time of a sequence of $n$ FOO’s takes $O(n \log n)$ time in the worst case? 

(i) If a data structure supports an operation FOO such that the amortized time of a FOO operation is $\Theta(\log n)$, is it true that the worst-case running time of a single FOO takes $O(\log n)$ time? 

(j) What is the maximum number of edges in an undirected graph with $n$ vertices, where each vertex has degree at most $k$? 

(k) How long does it take to run Prim’s algorithm on a graph $G = (V, E)$? Assume the priority queue is implemented as a heap. 

(l) Let $G = (V, E)$ be a weighted graph. Under what assumption does Dijkstra’s algorithm correctly compute shortest paths? 

(m) A connected undirected acyclic graph with $n$ vertices has edges. 

(n) Is it true that the shortest path between two vertices $u, v$ in a directed graph with all edge weights equal to 1 can be computed in $O(|V| + |E|)$ time? 

(o) Is it true that $P = NP$?
[10 points] Problem 1 Part II:

(a) Describe how can you implement DELETE($T, x$) in a splay tree $T$ using the SPLAY operation.

(b) A “smart” friend claims that he has discovered a data structure which uses only comparisons between elements and supports both INSERT and DELETE-MIN in $O(1)$ time worst-case. Why is this impossible?
Problem 2:

Suppose we have a straightforward algorithm for a problem that runs in \( \Theta(n^2) \) time for inputs of size \( n \). Suppose we devise a Divide-and-Conquer algorithm that divides an input into two inputs half as big, solves the two subproblems recursively and combines the solutions to get a solution for the original input. Dividing the input into two halves is done in \( O(n) \) time, while combining the solutions together takes \( O(n \log n) \) time.

Write a recurrence for this algorithm and solve it. Is the Divide-and-Conquer algorithm more or less efficient than the straightforward scheme?
[15 points] Problem 3:

The binomial coefficient $C(n, k)$ counts the number of ways of choosing $k$ distinct items from a set of $n$ items. It can be defined as follows:

\[
C(n, k) = C(n - 1, k - 1) + C(n - 1, k) \quad \text{for } n > 0 \text{ and } k > 0
\]

\[
C(n, 0) = 1 \quad \text{for } n \geq 0
\]

\[
C(0, k) = 0 \quad \text{for } k > 0
\]

The procedure BINO($n, k$) below computes $C(n, k)$ using the recursive formulation above.

Bino($n, k$)
  if ($k = 0$)
    return 1
  else if ($n = 0$)
    return 0
  else
    return Bino($n-1, k-1$) + Bino($n-1, k$)
end

(a) Write the recurrence for the running time $T(n, k)$ of BINO($n, k$) and show that it is exponential.
(b) Describe a dynamic programming algorithm for calculating $C(n, k)$ and analyze its running time.
Binary search of a sorted array takes logarithmic time, but the time to insert a new element is linear in the size of the array. We can improve the time for insertion by keeping several sorted arrays.

More precisely, suppose that we want to support SEARCH and INSERT on a set of \( n \) elements. Let \( k = \lceil \lg(n + 1) \rceil \), and let the binary representation of \( n \) be \( \langle b_{k-1}, b_{k-2}, \ldots, b_1, b_0 \rangle \). We have \( k \) sorted arrays \( A_0, A_1, \ldots, A_{k-1} \), where each array \( A_i \) is either full or empty depending on whether the corresponding bit \( b_i \) in the binary representation of \( n \) is 1 or 0. The size of array \( A_i \) is \( 2^i \), for all \( i = 0, 1, \ldots, k - 1 \). Therefore the total number of elements in all \( k \) arrays is \( \sum_{i=0}^{k-1} b_i \cdot 2^i = n \).

Although each individual array is sorted, there is no relationship between the elements in different arrays.

Example: A set of \( n = 13 = 1101_2 \) elements stored in \( k = 4 \) sorted arrays.

(a) Describe how to perform the SEARCH operation for this data structure and analyze its worst-case running time.

(b) Describe how to INSERT a new element into this data structure and analyze its worst-case running time.
   
   (Hint: Two sorted arrays can be merged into one sorted array in time linear in the total length.)

(c) Analyze the amortized running time of an INSERT operation. You can use any method you want.
   
   (Hint: Assume you start with an initially empty structure and compute the sum of the running times of the first \( n \) INSERTs.)
[20 points] Problem 5:

Suppose the degree requirements for a computer science major are organized as a dag (directed acyclic graph), where vertices are required courses and an edge \((x, y)\) means course \(x\) must be completed prior to beginning course \(y\). Make the following assumptions:

- All prerequisites must be obeyed.
- There is a course, CPS1, that must be taken before any other course.
- Every course is offered every semester.
- There is no limit to the number of courses you can take in one semester.

Describe an efficient algorithm to compute the minimum number of semesters required to complete the degree and analyze its running time.
20 points  **Problem 6:**

Consider an undirected weighted graph which is formed by taking a binary tree and adding an edge from *exactly one* of the leaves to another node in the tree. We call such a graph a *loop-tree*. An example of a loop-tree could be the following:

Let $n$ be the number of vertices in a loop-tree and assume that the graph is given in the normal edge-list representation without any extra information. In particular, the representation does not contain information about which vertex is the root.

(a) How long time would it take Prim’s or Kruskal’s algorithms to find the minimal spanning tree of a loop-tree? Make your bound as tight as possible.

(b) Describe and analyze a more efficient algorithm for finding the minimal spanning tree of a loop-tree. Remember to prove that the algorithm is correct.