CPS 130 Final Exam

Summer 2002

7:00-10:00pm, Thursday June 27th
Closed book exam

NAME:____

Problem	Max	Obtained
1	15	
2	20	
3	10	
4	15	
5	30	
6	10	
7	10	
8	20	
Total	130	

You can use any of the algorithms/procedures covered in class without describing them. It is okay to describe algorithms with words (and a few accompanying pictures if it helps the description).

Good luck!

HONOR CODE

I have obeyed the honor code.

SIGNATURE:____

[*15 points*] **Problem 1:**

(a) The summation $\sum_{i=0}^{n} 4^i$ is $\Theta($ $\quad\quad\quad$ $)$.

(b) The worst case running time of Quicksort is $\Theta($ $\quad\quad\quad$ $)$.

(c) Any comparison-based sorting algorithm must have a worst case time of at least $\Omega($ $\quad\quad\quad$ $)$.

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

(e) The median of an array of n elements can be computed in $O($ $\quad\quad\quad$ $)$ time.

(f) Consider a binary search tree. Searching for an element in the tree takes at most $O($ $\quad\quad\quad$ $)$ 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 $\Theta($ $\quad\quad\quad$ $)$.

(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 $G = (V, E)$?

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

(m) Given two decision problems A and B, is it true that if $A \leq_P B$ and $A \in P$, then $B \in P$?

(n) Assume that you have found a polynomial time algorithm to solve a decision problem that is known to be NP-complete. What would this imply?

(o) Is it true that $P = NP$?

[*20 points*] **Problem 2:**

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

(b) A "smart" UNC Chapel-Hill professor 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 is impossible?

(c) Order the following functions in increasing order of asymptotic growth. If two functions are of the same order, then indicate this.

$$\log \log n, n^3, 3^{\log_3 n}, n \log n, 7^{\log_2 n}, 2^n, n$$

(d) Let G be an arbitrary weighted graph and let $\delta(u, v)$ denote the shortest distance from vertex u to vertex v. Is it true that $\delta(a, c) = \delta(a, b) + \delta(b, c)$, for any vertices a, b, c in the graph? If it is true, prove it. If not, give a counterexample.

[*10 points*] **Problem 3:**

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 4:**

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:

$$
\begin{array}{ll}
C(n,k) = C(n-1,k-1) + C(n-1,k) & \text{for } n > 0 \text{ and } k > 0 \\
C(n,0) = 1 & \text{for } n \geq 0 \\
C(0,k) = 0 & \text{for } k > 0
\end{array}
$$

The procedure $\textsc{Bino}(n,k)$ 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 $\textsc{Bino}(n,k)$ and show that it is exponential.

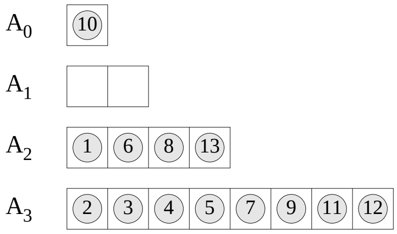
(b) Describe a dynamic programming algorithm for calculating $C(n, k)$ and analyze its running time.

[*30 points*] **Problem 5:**

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 $A_0, A_1, ..., A_{k-1}$, 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 $i = 0, 1, ..., k-1$. 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 of the aggregate, accounting or potential methods.

(*Hint: It may be easiest to use the aggregate method.*)

[*10 points*] **Problem 6:**

Consider an undirected graph G with n vertices and m edges.

(a) Describe an algorithm that counts the connected components of G using a union-find data structure and analyze the running time.

(b) Can you count the connected components of G in time $O(n + m)$? If so, how?

[*10 points*] **Problem 7:**

Describe an algorithm to determine if an undirected graph $G = (V, E)$ is a tree and analyze its running time. Argue briefly that your algorithm is correct.

[*20 points*] **Problem 8:**

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.