Name:

Score:

1	2	3	4	5	6	7	8	SUM

CRN 21546 CIS 621 Winter 2009

Midterm Exam

(due in class on Monday, February 23)

This is the usual open-everything, but no outside help take-home test. Chris Wilson (`cwilson@cs.uoregon.edu`, office hours MWF at 2) has graciously agreed to answer your questions about typos, misunderstandings, and such. You can also check the "Class News", where I will post "frequently asked questions" about the test (and my answers). You should not spend more than five hours solving the problems (even though four should be sufficient). Make sure that your answers are neat and legible – no first drafts, please.

Brute Force

1. Given the dimensions of a sequence of compatible matrices, the Matrix-Chain Product problem is that of finding an optimal parenthesization of a multi-operand expression, with the cost measure being the total number of scalar multiplications performed.

1. What is the complexity of the brute force evaluation of the minimum cost of a matrix-chain product of n matrices?
2. Compute the minimum cost of matrix-chain product of four matrices of dimensions $2 \times 20, 20 \times 5, 5 \times 6$, and 6×1, in this order. and the corresponding optimal parenthesization of the product

$$A \times B \times C \times D$$

Loop Invariant

2. Prove the correctness of the "Russian Peasant" multiplication algorithm below by providing a useful loop invariant with appropriately argued initialization, maintenence and final properties. State the algorithm's complexity, assuming that A and B are positive integers.

```
a:=A; b:=B; c:=0;
while b>0 do
  if even(b) then {a:=a+a; b:=b div 2} else {c:=c+a; b:=b-1};
return(c)
```

3. A candidate has *majority* of a vote if more than half voters support her. Describe the result of the following algorithm on an array whose entries contain names (arbitrary strings) of candidates. Prove your answer by a loop invariant argument. (**Hint:** consider the possible majority candidate among the votes $\texttt{cand}^{\texttt{count}} \cup \texttt{vote[index..n]}$)

```
procedure find(vote:  array[1..n] of name);
name cand; int index, count;
begin count:=0;
     for index:=1 to n do
     if count=0 then begin cand:= vote[index]; count:=1 end
                else if cand=vote[index]
                        then count:=count+1
                        else count:=count-1
end
```

Amortized Complexity

4. Define an abstract data type *MaxBag (MB)* to hold (possibly repeated) integer values $1, \ldots, n$, with the following operations:

Initialize: $\rightarrow$ *MB* (create an empty MB)
Increment: $MB \rightarrow MB$ (add a copy of '1' to MB)
Add: $Int \times Int \times MB \rightarrow MB$ (delete a copy of each of the two integers – assuming they are present in MB – and add a copy of their sum to MB)
DeleteMax: $MB \rightarrow MB$ (delete a copy of the maximum integer in MB).

Thus, for instance, *Add(3,2,{1,2,3,5})={1,5,5}*

and *DeleteMax({1,2,3,5,5}={1,2,3,5}*

Implement *MaxBag* with constant time amortized complexity of all operations.

5. We have seen that disjoint sets with operations *Union* and *Find* can be implemented very efficiently when *Union* is performed guided by the *rank*, and *Find* involves *path contraction*.

1. What is the time complexity when *path contraction* is **not** performed?
2. Show the amortized time complexity when *Union* is guided by *size*.
3. Show that the amortized time complexity for *Union* without any heuristics and *Find* with *path contraction* is $\Omega(\log n)$.
   (**Hint:** Perform the following sequence of operations:
   Build via $n-1$ Unions a set represented by
   a binomial tree T of order h;
   do $n = 2^h$ times
   {*Union* T with a single node set by
   making that node the overall root of the new T;
   Find the deepest node of T compressing the find path; }

Greedy Algorithm

6. This problem pertains to the construction of an optimal prefix-free binary code for n messages by the Huffman algorithm. Assume that the probabilities of message transmission are given in the non-decreasing order: $p_1 \leq p_2 \leq \cdots \leq p_n$. (An essential assumption!)

1. **Warm-up:** Draw a Huffman tree for six messages with the following probability of transmission: .10, .10, .15, .20, .21, and .24. Use boxes for leaves and circles for internal nodes.
2. Propose a linear implementation of an algorithm that constructs a Huffman tree, given transmission probabilities in the non-decreasing order.

Divide and Conquer

7. Show how to find the majority element (if one exists, see **3.** above) in linear time using the order statistics ("k-th largest") algorithm.

8. A binomial tree of order k, B_k is obtained from two copies of B_{k-1}, where one is made a principal subtree of the other (B_0 being the trivial tree of one node.) Nodes of such a tree can be represented in an array $A[1..2^k]$ so that the two definitional binomial trees of order $k-1$ are represented in $A[1..2^{k-1}]$ and $A[2^{k-1}+1..2^k]$ (with the root in $A[1]$).

Design and prove correct a linear-time algorithm that heapifies a binomial tree of order k, B_k with node values stored in an array $A[1..n]$ (for $n = 2^k$), as above.