CSci 115 Midterm

Summer 2000

39 points

There are 16 problems. You are not required to do all of them.

Total of 11 pages.

Show your work.

1. (2 points) Using the **definition** of **Big-Oh**, show that $(n+1)(n-1)/2$ is $O(n^2)$.

2. (2 points) Let $f(n) = n^2$ and $g(n) = n \log_2 n$. Show that $g(n)$ is $O(f(n))$.

3. (3 points) Linked List: Consider the list whose elements are of the type defined as follows:

```
/* C/C++ version 1 */
struct list {
   int         key;
   struct list *link;
};

struct list x;
```

   Alternatively, you may declare your own list structure, but please stay with the specifications.

   Write a recursive function member in C or C++ to test whether a is a member of the list x (i.e., whether an element with the key a occurs in the list x). If so, then deliver the value true, otherwise false.

4. (3 points) Knapsack Algorithm 0/1: Let us consider a recursive solution to a simplified version of the classical knapsack problem in which we are given target t and a collection of positive integer weights w_1, w_2, $\ldots$, w_n. We are asked to determine whether there is some selection from among the weights that totals exactly t. For example, if $t = 1 - 0$, and the weights are 7, 5, 4, 4, and 1, we could select the second, third, and fifth weights, since $5 + 4 + 1 = 10$.

   The image that justified the name "knapsack problem" is that we wish to carry on our back no more than t pounds, and we have a choice of items with given weights to carry. We presumably find the items' utility to be proportional to their weight, so we wish to pack our knapsack as closely to the target weight as we can.

   You are asked to write a function knapsack that operates on an array

   weights: array [1..n] of integer.

   A call to knapsack(x, i) determines whether there is a collection of the elements in weight[i] through weight[n] that sums to exactly t, and prints these weights if so.

   State the recursive version of the knapsack 0/1 algorithm in pseudocode.

5. (2 points) Find the maximum profit that can be obtained by filling the knapsack with a combination of the five objects. Each object can be used only once. Assuming that a fraction of an object may be placed in the knapsack in order to completely fill the knapsack.

   Show and explain your work.

   $N = 5$
   $M = 30$ (* capacity of the knapsack *)
   $(p_1, p_2, \ldots, p_5) = (8, 5, 10, 15, 7)$
   (* Profits associated with the objects *)

   $(w_1, w_2, \ldots, w_5) = (6, 10, 9, 5, 12)$
   (* Weights associated with the objects *)

6. (4 points) Give, using the "big-oh" notation, the worst case running time of the selection sort. **Prove your claim.**

```
Selection(int a[], int N) {
   int i, j, min, t;
   for (i=1; i<N; i++) {
      min = i;
      for (j=i+1; j<=N; j++)
         if (a[j] < a[min]) min = j;
      t = a[min]; a[min] = a[i]; a[i] = t
   }
}
```

7. (4 points) Give, using "big oh" notation, the worst case running time of the following function as a function of n. Prove your claim.

```
function rec (n: integer): integer;
begin
   if n<=1 then
      return (1)
   else
      return (rec(n-1) + rec(n-1) + rec(n-1))
end;
```

8. (3 points) Backtracking Algorithm: Given an $n \times n$ board, state the backtracking algorithm for the knight's tour in pseudocode.

9. (4 points) Divide and Conquer: Tower of Hanoi.

   The initial set up is shown in the figure below. The objective is to move the five disks to peg C. Only the top disk on any peg may be moved to any other peg, and a larger disk may never rest on a smaller one. Write a recursive solution in C or C++.

```
\#include<stdio.h>
void towers(int n, char frompeg, char topeg, char auxpeg);

main() {
   int n;
   scanf("%d", &n);
   towers(n, 'A', 'C', 'B');
   return 0;
} /* end */
```

10. (1 point – no partial credit) Give the inorder traversal of the following tree:

11. (1 point) Calculate the external path length of the tree given in problem 1. Root is at level 1. Show your work.

12. (3 points) Huffman's Algorithm: You are given the following data structures declarations in Pascal. For C or C++ users, use similar declarations.

```
Const maxsize = 100; maxsym = 26;
Type
   treeelt = record
      leftchild, rightchild, parent : integer
      end;
   alphabetelt = record
      symbol : char;
      probability : real;
      Leaf : integer
   forestelt = record
      weight : real;
      root : integer
   end;
Var
   TREE : array [1..maxsize] of treeelt;
   ALPHABET : array [1..maxsym] of alphabetelt;
   FOREST : array [1..maxsym] of forestelt;

etc.
```

You are given some description of two subprograms in Pascal. For C and C++ users, use the same names for the subprograms, respectively.

```
procedure lightones ( var least, second : integer );
(* sets least and second to the indices in FOREST of the
   trees of smallest weight.  We assume lasttree >= 2. *)

function create ( lefttree, righttree : integer ) : integer;
(* returns new node whose left and right children are
   FOREST[lefttree], root, and FOREST[righttree].root *)
```

Write a procedure (in pseudocode) for the Huffman's algorithm.

13. (3 points) Construction of Huffman tree.

    Suppose characters *a*, *b*, *c*, *d*, *e* have probabilities 0.32, 0.15, 0.33, 0.08, 0.12, respectively.

    Find an optimal Huffman code (1 point).

    Draw the Huffman tree and detail its construction (1.5 point).

    Show your work.

    What is the average code length? (0.5 point)

14. (2 points) Optimal Search Tree: Given the following set of frequencies:

internal node		x_1		x_2		x_3	
external node	y_0		y_1		y_2		y_3
frequencey	1	5	2	3	1	4	5

Construct an optimal search tree for this data.
Show all your work.

15. (2 points) Given n nodes, write a recursive algorithm (in pseudocode) to construct a perfectly balanced binary tree with n nodes.

16. (2 points) Fibonacci Tree Construction (using Wirth's convention with root is at level 1):

    A Fibonacci tree of height 0 is the empty tree.
    Construct a Fibonacci tree with height 5. (1.75 points)
    How many nodes are required? (0.25 point)