Midterm

Note

This is a closed-book exam. Each problem accounts for 10 points, unless otherwise marked.

Problems

1. Find an expression for the sum of the i-th row of the following triangle, which is called the **Pascal triangle**, and prove *by induction* the correctness of your claim. The sides of the triangle are 1s, and each other entry is the sum of the two entries immediately above it.

$$\begin{array}{ccccccccc} & & & & 1 & & & & \\ & & & 1 & & 1 & & & \\ & & 1 & & 2 & & 1 & & \\ & 1 & & 3 & & 3 & & 1 & \\ 1 & & 4 & & 6 & & 4 & & 1 \end{array}$$

2. In the so-called implicit representation of a binary tree, the tree nodes are stored in an array, say A, such that

   (a) the root is stored in $A[1]$ and

   (b) the left child of (the node stored in) $A[i]$ is stored in $A[2i]$ and the right child in $A[2i+1]$. (Note: a nonexistent child may be indicated by a special mark/value in the corresponding cell for storing the child.)

   Prove *by induction* that, for complete binary trees, the implicit representation is compact in the sense that, if we label the tree nodes from top to bottom and left to right with the numbers 1 through n (where n is the number of nodes in the tree), then the node labeled i is stored in $A[i]$ for $1 \leq i \leq n$.

3. (15 points) Let $G(h)$ denote the least possible number of nodes contained in an AVL tree of height h. Let us assume that the empty tree has height -1 and a single-node tree has height 0.

   (a) (5 points) Please give a recurrence relation that characterizes (fully defines) G.

   (b) (10 points) Based on the recurrence relation, prove that the height of an AVL tree with n nodes is $O(\log n)$.

4. (15 points) Consider the problem of merging two skylines, which is a useful building block for computing the skyline of a number of buildings. A skyline is an alternating sequence of x coordinates and y coordinates (heights), ending with an x coordinate (as discussed in class).

(a) What is the resulting skyline from merging these two skylines: (1,**3**,3,**6**,6,**2**,8,**7**,10,**3**,14) and (2,**4**,6,**2**,8,**5**,11,**3**,12)?

(b) To obtain a systematic procedure for merging two skylines, one may focus on determining the first two coordinate values to be output and reducing (by two) the length of one of the input skylines. Suppose the two input skylines are $(a_1, a_2, \cdots, a_{2m+1})$ and $(b_1, b_2, \cdots, b_{2n+1})$. How can the first two coordinate values in the output be determined?

(c) Following the thought in the previous subproblem, assume that the first two coordinate values that should be output are a_1 and a_2. We are now looking at two skylines $(a_3, a_4, \cdots, a_{2m+1})$ and $(b_1, b_2, \cdots, b_{2n+1})$, the first being shorter (by two) than in the original input. How can the next two coordinate values in the output be determined?

5. Consider the Knapsack Problem: Given a set S of n items, where the i-th item has an integer size $S[i]$, and an integer K, find a subset of the items whose sizes sum to exactly K or determine that no such subset exists.

   We have discussed in class two approaches to implementing a solution that we designed by induction: one uses dynamic programming (see the Appendix), while the other uses recursive function calls.

   Please present the recursive approach in suitable pseudocode.

6. Show all intermediate and the final AVL trees formed by inserting the numbers 7, 6, 3, 1, 4, 5, and 2 (in this order) into an empty tree. Please use the following ordering convention: the key of an internal node is larger than that of its left child and smaller than that of its right child. If re-balancing operations are performed, please also show the tree before re-balancing and indicate what type of rotation is used in the re-balancing.

7. The input is a set S of n real numbers. Design an $O(n)$ time algorithm to find a number that is *not* in the set. Prove that $\Omega(n)$ is a lower bound on the number of steps required to solve this problem.

8. Consider rearranging the following array into a max heap using the *bottom-up* approach.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	3	5	2	1	8	14	6	4	11	10	12	13	15	9

   Please show the result (i.e., the contents of the array) after a new element is added to the current collection of heaps (at the bottom) until the entire array has become a heap.

9. Consider the following algorithm that, given a sorted sequence of *distinct* integers $a_1, a_2, \cdots, a_n$ (stored in an array A), determines whether there exists an index i such that $a_i = i$.

   Algorithm Special_Binary_Search (A, n);

```
begin
    $Position := Special\_Find(1, n)$;
end

function Special_Find ($Left, Right$) : $integer$;
begin
    if $Left = Right$ then
        if $A[Left] = Left$ then $Special\_Find := Left$
        else $Special\_Find := 0$
    else
        $Middle := \lceil \frac{Left+Right}{2} \rceil$;
        if $A[Middle] < Middle$ then
            $Special\_Find := Special\_Find(Middle + 1, Right)$
        else
            $Special\_Find := Special\_Find(Left, Middle)$
end
```

Draw a decision tree of the algorithm for the case of input size six, i.e., $n = 6$.

Appendix

- The solution of the recurrence relation $T(n) = aT(n/b) + cn^k$, where a and b are integer constants, $a \geq 1$, $b \geq 2$, and c and k are positive constants, is as follows.

$$T(n) = \begin{cases} O(n^{\log_b a}) & \text{if } a > b^k \\ O(n^k \log n) & \text{if } a = b^k \\ O(n^k) & \text{if } a < b^k \end{cases}$$

- Below are several basic generating functions.

generating func.	power series	generated sequence
$\frac{1}{1-z}$	$1 + z + z^2 + \cdots + z^n + \cdots$	$1, 1, 1, \cdots, 1, \cdots$
$\frac{c}{1-az}$	$c + caz + ca^2z^2 + \cdots + ca^nz^n + \cdots$	$c, ca, ca^2, \cdots, ca^n, \cdots$
$\frac{1}{(1-z)^2}$	$1 + 2z + 3z^2 + \cdots + nz^{n-1} + \cdots$	$1, 2, 3, \cdots, n, \cdots$
$\frac{z}{(1-z)^2}$	$z + 2z^2 + 3z^3 + \cdots + nz^n + \cdots$	$0, 1, 2, 3, \cdots, n, \cdots$

- Below is a non-recursive algorithm for determining whether a solution to the Knapsack Problem exists.

```
Algorithm Knapsack ($S, K$);
begin
    $P[0, 0].exist := true$;
    for $k := 1$ to $K$ do
        $P[0, k].exist := false$;
    for $i := 1$ to $n$ do
```

```
        for $k := 0$ to $K$ do
            $P[i,k].exist := false$;
            if $P[i-1,k].exist$ then
                $P[i,k].exist := true$;
                $P[i,k].belong := false$
            else if $k - S[i] \geq 0$ then
                if $P[i-1, k-S[i]].exist$ then
                    $P[i,k].exist := true$;
                    $P[i,k].belong := true$
end
```