1-1-1981

File maintenance by using AVL trees (an implementation of payroll system).

Alice Ming-Mei Chen

FILE MAINTENANCE BY USING AVL TREES

(AN IMPLEMENTATION OF PAYROLL SYSTEM)

by

Alice Ming-Mei Chen

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science

Division of Computing and Information Science

Mathematics Department

Lehigh University

1981

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

Sept 15, 1981
(date)

Professor in Charge

Head of Division

Acknowledgements

The author thanks Professor Samuel L. Gulden for his comments and suggestions, thus making possible the results of this thesis. Thanks are also due to her husband and mother for their love and encouragement.

Table of Contents

List of Figures

List of Tables

Abstract

File Maintenance by Using AVL Trees
(An Implementation of Payroll System)

by Alice Ming-Mei Chen

Binary search tree are in competition with other methods for organization files, such as : hash-coding or scatter storage techniques; linear lists, either sequentially allocated or chained; other kinds of trees, such as : "tries", or multiway trees of various kinds and methods that are based on a combination of such techniques as indexed-sequential file organization.

Binary search trees are one of the most flexible and best understood techniques for organizing large files. Because of this, they have received a great deal of attention in recent years, and their properties are now better understood than those of most other file organization methods. Their practical importance comes mainly from the fact that they perfom with reasonable efficiency all of the common operations on files: random and sequential processing of a file, insertion and deletion of records, and restructuring of the file. And they can be allocated in reasonable ways ln back-up

storage devices with restricted access. In addition to their practical importance, they are of theoretical interest because they generate mathematical problems which arise in many other areas of information processing : sorting, coding and information theory, and others.

This thesis present a method of designing an payroll system by using AVL trees and file structure. A noteworthy aspect of the tree algorithm is the use of recursion and concept of a virtual root.

processes(batch, and on-line) are dicussed, modeled, and programmed.

1. Introduction

Although trees have been long used for the storage and retrieval of information, unfortunately there is a tradeoff between storage (construction) time and retrieval time. To keep retrieval time at a minimum, the tree must be balanced; but posting a new item under this constraint can require a complete reorganization of the tree. Conversely, if the tree is allowed to grow without restriction on its structure, the average number of probes (that is, references to main memory) required for retrieval can approach N/2, depending on the arrival of the items.

If insertion and subsequent queries are the main operations of interest, AVL trees present the overall best qualities.

Binary search represents an important technique for handling structures such as files and directories, dictionaries, and symbol tables. The random growth of a binary search tree can lead in the worst case to a (linked) linear list. Hence several algorithms have been devised to balance or restructure the tree while it is being built,i.e., to keep it close to its optimal form.

The three methods for doing this - height balanced, Weight balanced, and total restructuring - in particular. We show that a height-balance technique, the AVL tree construction, is the most efficient when the main operations on trees are insertion and queries.

AVL trees (named after their inventors, the two Russian mathematicians Adel'son-Vel'skiy and Landis) and their extensions are built according to a height-balance algorithm. Several algorithms such as height-balance (i.e. AVL), weight-balance (i.e. BB and WB)and total restructuring for building balanced binary search are compared. The AVL construction presents less overhead when one is interested in insertion and subsequent queries.

An average of approximately $Log_2(N + 1)$ probes is required to find an item if it is present in an AVL tree. The average number of probes required to post a new item on an AVL tree is given by the sum of the probes required to find the vacancy, to generate the new node, to retrace the tree, and either to restructure the tree and the retrace; this is given by $Log_2(N + 1) + 4.75$ for $N \sim 2000$. This leads, on the average to about 11 probes for retrieval and about 16 probes for posting a new item. For example, consider a computer with a 16-word 0.1-microsecond scratch pad to keep the pushdown list in, a 2-microsecond main core and an appropriate instruction set. Insertion in a 2000-item tree will require about 33-microseconds and retrieving, about 22 microsecond.

2. Definitions

2.1 Node, Root, Level of node, Height, Path Length

Node : A binary tree is a finite set of nodes either empty, in which case we call it a nil node or a nil tree, or the tree $T(T_R,R,T_L)$, where R is a special node called the root and T_R and T_L are respectively, the right and left subtrees of R. Each node S, except a nil node, is the father of its right and left sons. In a binary tree each node has at most two subtrees. If both subtrees are present, they are called the left and right subtree in a binary trees. The subtrees are not interchangeable.

Root : The top node is commonly called root. The root of a tree is defined to be at level 1.

Level of node : A node y which is directly below node x is called a descendant of x; if x is at level i, then y is said to be at a level i + 1. inversely node x is said to be the ancestor of y. The root of a tree is defined to be at level 1.

Height : The maximun level of any element of a tree is said to be its height. if an element has no descendants, it is called a leaf; and an element which is not terminal is an interior node. The number of descendants of an interior node is called its degree. The maximum degree over all nodes is the degree of the tree.

Path Length : The number of branches or edges which have to be traversed in order to proceed from the root to a node x is called the path length of x. The root has path length 1, its direct descendants have path length 2, etc. In general, a node at level i has path length i. The path length of a tree is defined as the sum of the path lengths of all its components. It is also called its internal path length.

2.2 AVL trees

We begin the discussion of AVL trees with the following definitions.

1. A tree is balanced if and only if for every node the heights of its two subtrees differ by at most 1.

2. The length of a tree or a subtree is determined by the number of nodes in the longest path within that tree or subtree.

3. A complete, balanced tree of d levels, B^d , is structured so that all the dependent subtrees from any node all have exactly the same length. An incomplete, balanced tree is similar to a complete balanced tree, but may

have vacancies occurring on the lowest level. Landauer calls this a "balanced tree."

4. An AVL tree of N items, N, is structured so that, for every node of the tree, the lengths of the two subtrees dependent from that node differ at most by one from each other. Balanced trees are, therefore, a special case of AVL trees.

5. An unbalanced tree is none of the above trees.

The Figure 2-1 shows some AVL trees and one which is not.

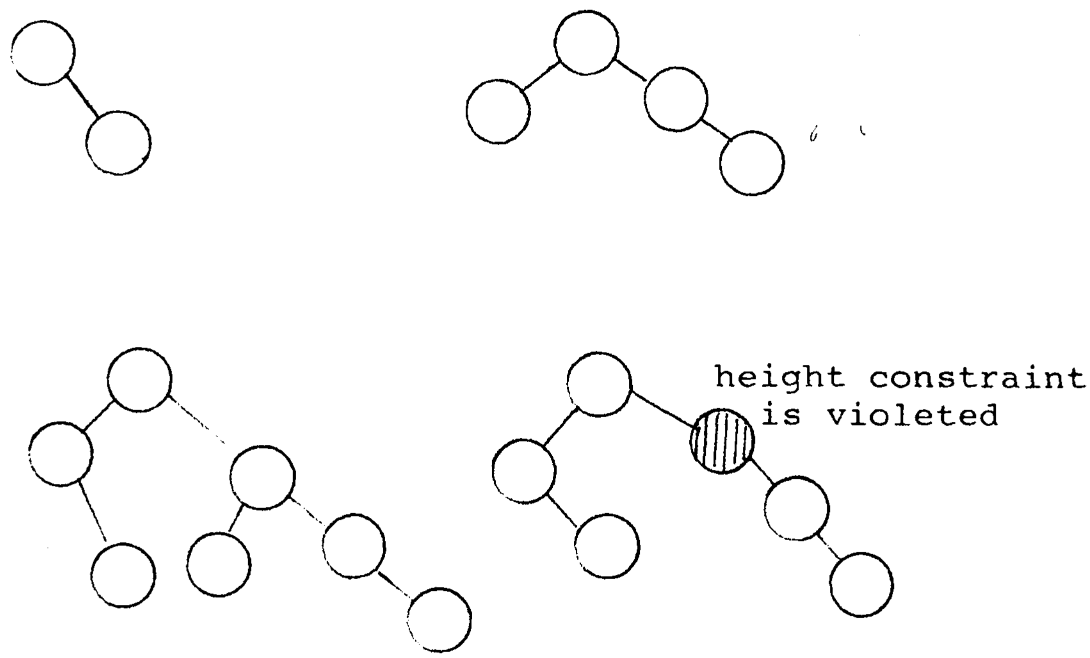

Figure 2-1: AVL trees

2.3 Mintrees of N items

To set an upper limit on the probes required for retrieval, we wish to know the longest path that may exist in an AVL tree of N items. This is done by determining the least number of items required to construct an AVL tree of n levels. Such a tree is called a mintree and is simbolized by M^n. A mintree of n levels may be constructed by taking one item as the root of the tree and placing a mintree of length n - 1 dependent from the root on one side. This step gives the required length of the tree; but to keep it AVL, the other branch from the root must no shorter than n - 2. Let that branch be a mintree of length n - 2.

Table 2-1 show the maximum and minimum number of items that can be stored in an AVL tree of length n, where the maximum is given by 2n - 1; and the minimum $N_n = N_{n+1} + N_{n-2} + 1$.

Tree length, n	Storable number of items maximum	Minimum
1	1	1
2	3	2
3	7	4
4	15	7
5	31	12
6	63	20
7	127	33
8	255	54
9	511	88
10	1,023	143
11	2,047	232
12	4,095	376
13	8,191	609
14	16,383	986
15	32,767	1,596
16	65,535	2,583
17	131,071	4,180
18	262,143	6,764
19	524,287	10,945
20	1,048,575	17,710

Table 2-1: Maximum and minimum number of items storable in AVL tree of lenght n.

3. AVL trees Algorithms

The height constraint prevents these trees from being too far away from a completely balanced tree - indeed, figure 2-1 shows the three "most unbalanced" height balanced trees relative to their respectively number of nodes.

Despite the fact that AVL trees look sparse, the search time they require is only moderately longer than in completely balanced trees as shown in Table 2-1. Thus height-balanced trees satisfy the requirement of short search time.

The more interesting aspect of a scheme for organizing a highly dynamic file is the requirement that insertions and deletions can be performed easily while maintaining the tree within the desired class.

Insertion in AVL trees requires at most one rotation or double rotation. Deletion, on the other hand, may require as many as h/2 transformations (where h is the height of the tree), but on the average (over all AVL trees) the number each transformation requires an amount of work which is independent of tree size(i.e., adjusting a few pointers), the amount of work required to update AVL tree is indeed samll.

In general one rotation is invoked for approximately every two insertions. one is required for every five deletions only. Deletion in AVL trees if therefore about as easy - or as complicated - as insertion.

3.1 AVL trees Insertion

The basic for AVL tree insertion is algorithm as Figure 3-1 and Example 1. We suppose initially that the tree is AVL and that a new item is added to the tree in some previously vacant spot. The path through the tree that leads to this new item is now singled out for examination.

Consider any arbitrary node on this path and assume that the new item is to be posted on its left subtree. (By symmetry the same arguments will apply if the new item were inserted on the right subtree). Three possible conditions could have obtained at this node before the new item was added:

1. Lengths of the left and right subtrees from this node were equal.
2. Right subtree was longer by one than the left.

3. Left subtree was longer by one than the right.

```
Program AVL(master,trans); (*Example 1*)
type
   ref = ^word;
   spell = record
              str : alfa;
              len : integer;
           end;
   person = record
              name : spell;
              sex : char;
              division: spell;
              wage : real;
              dependant : integer;
              overhours: integer;
              taxpay : real;
              netpay : real;
           end;
    word = record
              key : person;
              left, right : ref;
           end;

Procdure search( x : person; var p : ref);
begin
   if p = nil
      then
         begin (*not in tree;insert it*)
            new(P);
            with p^ do
               begin
                  key := x;
                  left := nil;
                  right := nil;
               end
         end
      else
         if x < p^.key
            then
               search(x,p^.left)
            else
               if x > p^.key
                  then
                     search(x,p^.right)
                  else
                     p^.key := x
   end;
```

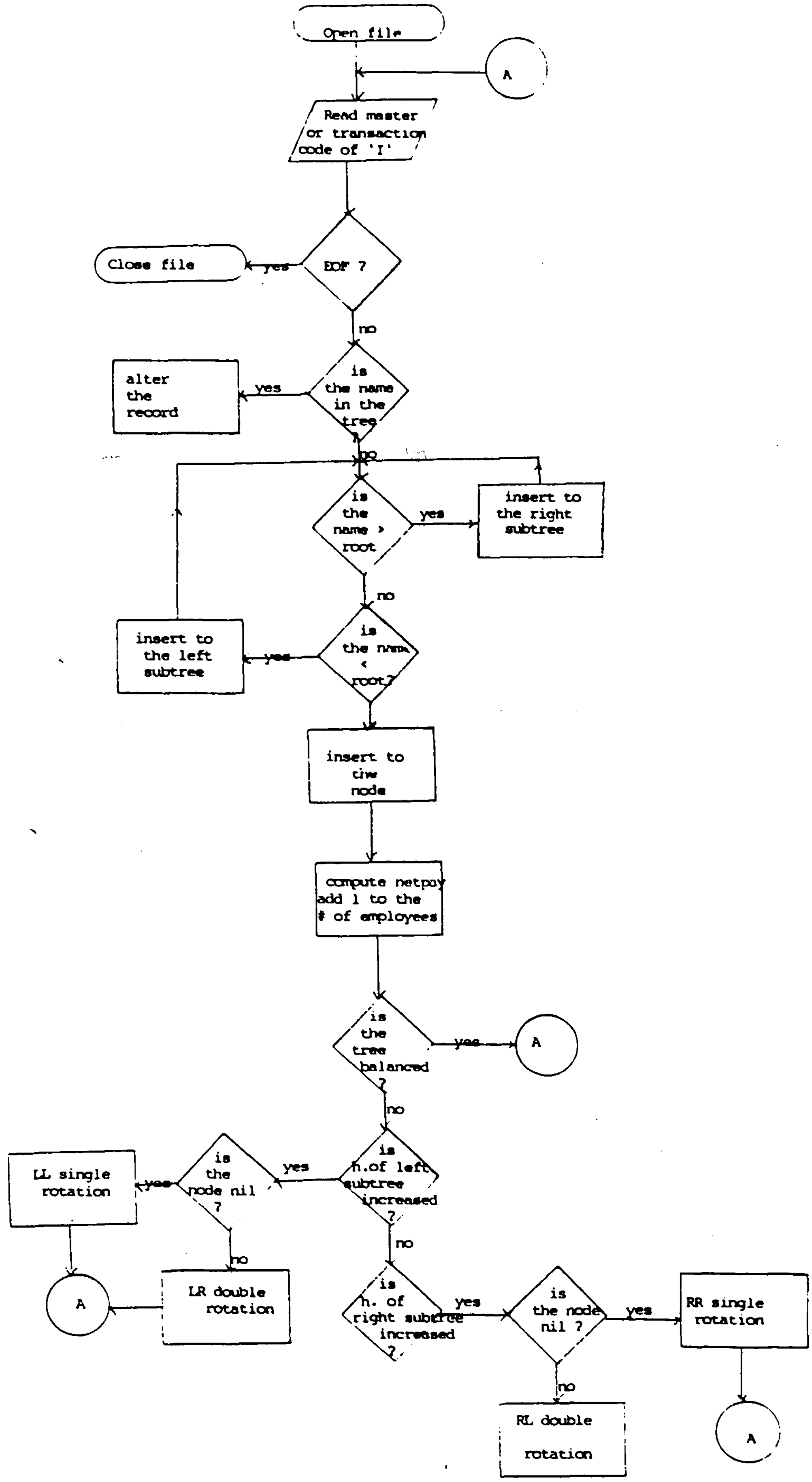

Figure 3-1: Flowchart of AVL trees insertion

After adding the new item, we must examine the tree to see if it is still AVL. The begining step is to examine the node from which the new item is dependent. This node may have had no subtrees at all (Condition 1); or it may have had a right subtree consisting of just one node (Condition 2). If condition 1 obtained, then adding the new item will lengthen the path (generate an "excess"); hence the next prior node on the path must be examined to discover the effects of this excess. If Condition 2 obtained, then adding the new item will not change the length of any subtree of which this node is a member; hence, the potential excess is absorbed, and the process of inserting the new item is completed.

In the event of an unabsorbed excess, we examine the prior node on the singled out path where any one of the three possible conditions might have existed. Again Condition 2 leads to absorption of the excess and termination of the inserting process Condition 1 makes it neessary to back up one more node where the examination is repeated. If this process leads eventually back up to the root, the tree is still AVL; hence, no changes are required, since only nodes on the singled -out path can be effected by the addition of the new item, and all these nodes have been examined and found to be AVL.

If, however, we come upon Condition 3 in the process of retracing the path, the retracing must stop. A change procedure must stop. A change procedure must then be invoked since Condition 3 implies an attempt to lengthen the subtree that is already the longer of the two dependents from this node. The node at which Condition 3 is found is called the "critical" node; it is assumed to have two subtrees of original length, X and X - 1.

The process of node insertion consists essentially of the following three consecutive parts:

1. Follow the search path until it is verified that the key is not already in the tree.

2. Insert the new node and determine the resulting balance factor.

3. Retreat along the search path and check the balance factor at each node.

The procedure in program example 1 describes the search operation needed at each single node, and because of its recursive formulation it can easily accommodate an additional operation "on the way back along the search path." At each step, information must be passed as to whether or not the height of the subtree had increased.

We therefore extended the procedure's parameter list by the Boolean h with the meaning "the subtree height has increased." h must denote a variable parameter since it is used to transmit a result.

The working principle is shown by Figure 3-2. Consider the binary tree which consists of two nodes only. Insertion of key 7 first results in an unbalanced tree (i.e., a linear list). Its balancing involves a RR single rotation, resulting in the perfectly balanced tree(b). further insertion of nodes 2 and 1 result in an imbalance of the subtree with root 4. This subtree is balanced by an LL single rotation(d). The subsequent insertion of key 3 immediately offsets the balanced criterion at the root node 5. Balance is thereafter re-established by the more complicated LR double rotation; the outcome is tree(s). The only candidate for loosing balance after a next insertion is node 5. Indeed, insertion of node 6 must invoke the forth case of rebalancing outlined in programexaple 1. the RL double rotation. The final tree is shown in Figure 3-2.

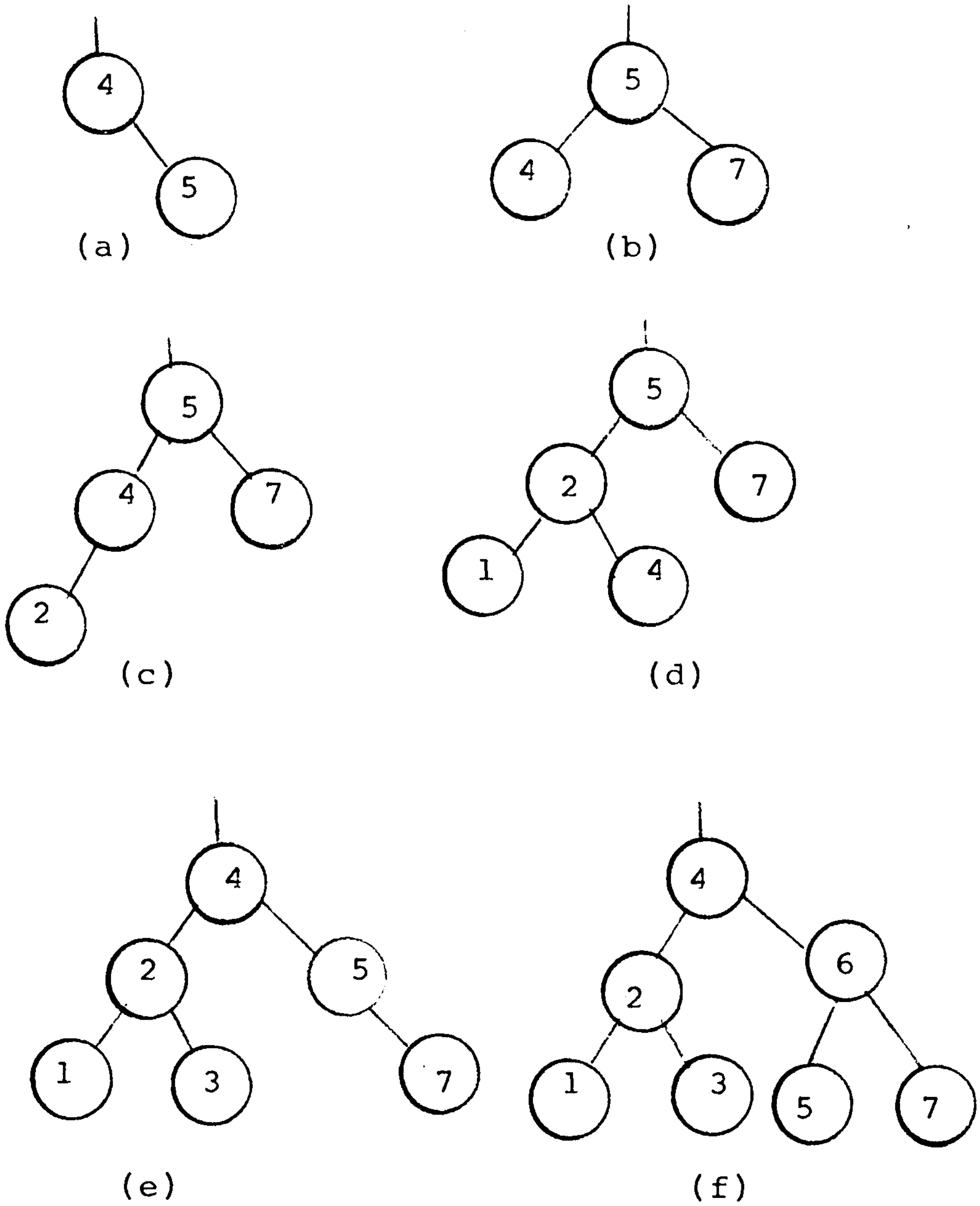

Figure 3-2: Insertions in balanced tree

3.2 AVL trees Deletion

The basic for AVL tree deletion is algorithm as Figure 3-3 and Example 2.

```
Procedure delete( x:spell; var p : ref);
   var q : ref;
   procedure del ( var r: ref);
      begin
         if r^.right <> nil
            then
               del(r^.right)
               begin
                  q^.key := r^.key;
                  q := r;
                  r := r^.left
               end
      end;
   begin (*delete*)
      if p = nil
         then
            writeln(tty,'record not found');
         else
            if x < p^.key
               then delete(x,p^.left)
               else
                  if x > p^.key
                     then
                        delete(x,p^.right)
                     else
                        begin
                           q := p;
                           if q^.right = nil
                              then p := q^.left
                              else
                                 if q^.left = nil
                                    then
                                   p := q^.right
                              else del(q^.left);
                        end
   end;
```

The easy cases are terminal nodes and nodes with only a single descendant. If the node to be deleted has two subtrees, we will again replace it by the rightmost node of its left subtree. A boolean variable parameter h is added with the meaning "the height of the subtree has been reduced." Rebalancing has to be considered only when

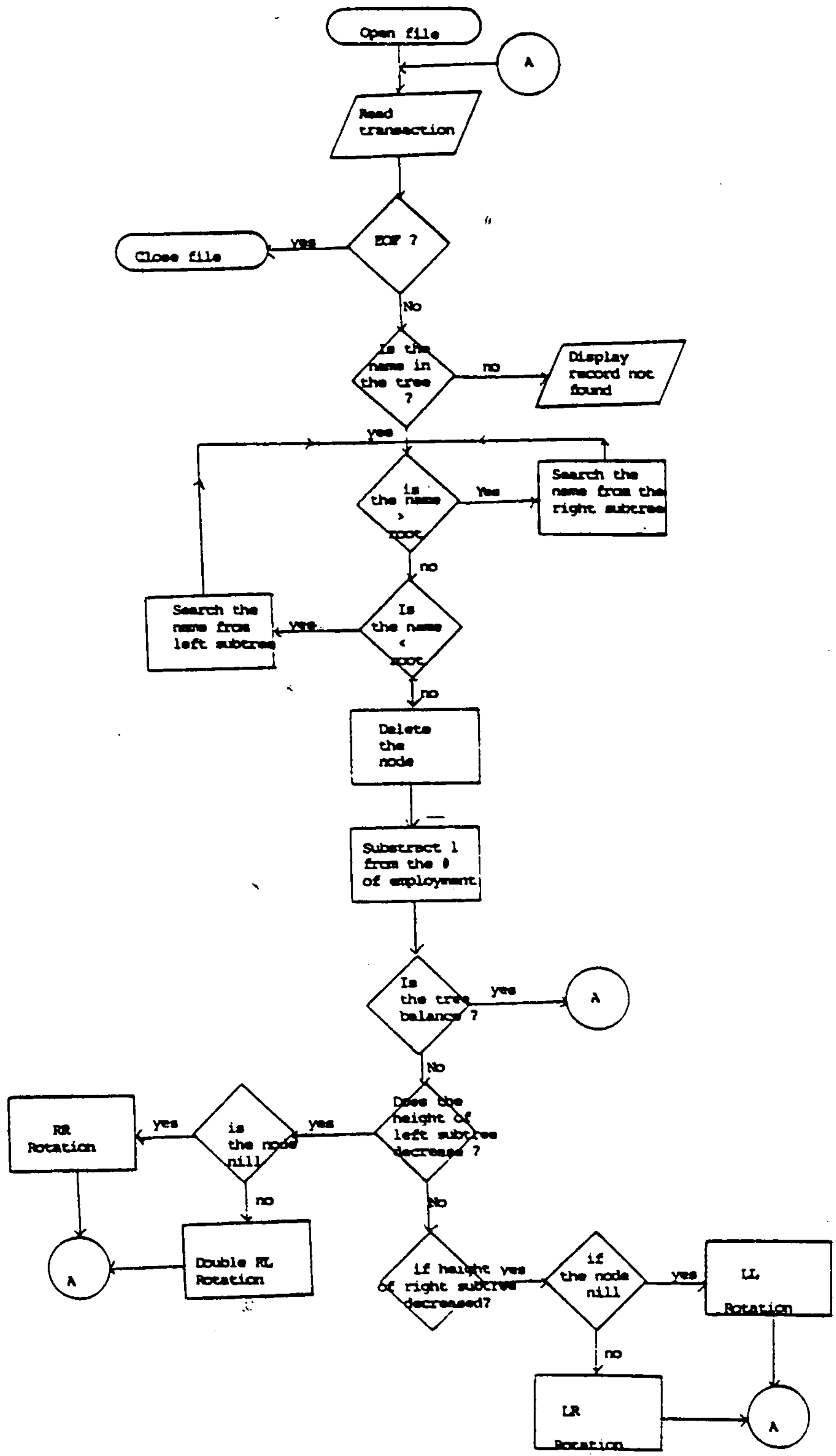

Figure 3-3: The Flowchart of Deletion

h is true. h is assigned the value true upon finding and deleting a node or if rebalancing itself reduces the height of a subtree. Balance1 is applied when the left, balance2 after the right branch had been reduced in height.

The operation of the procedure is illustrated in Figure3-4 Given the balanced tree (a), sucessive deletion of the nodes with keys 4, 8, 6, 5, 2, 1, and 7 results in the tree (b) .. (h).

The deletion of key 4 is simple in itself since it represents a terminal node. However, it results in an unbalanced node 3. Its rebalancing operation involves an LL single rotation. Rebalancing becomes again necessary after the deletion of node 6. This time the right subtree of the root(7) is rebalanced by an RR single rotation. Deletion of node 2, althourgh in itself straight-forward since it has only a single descendant, calls for a complicated RL double rotation. The fourth case, an LR double rotation, is finally invoked after the removal of node 7, which at first was replaced by the rightmost element of its left subtree, i.e., by the node with key 3.

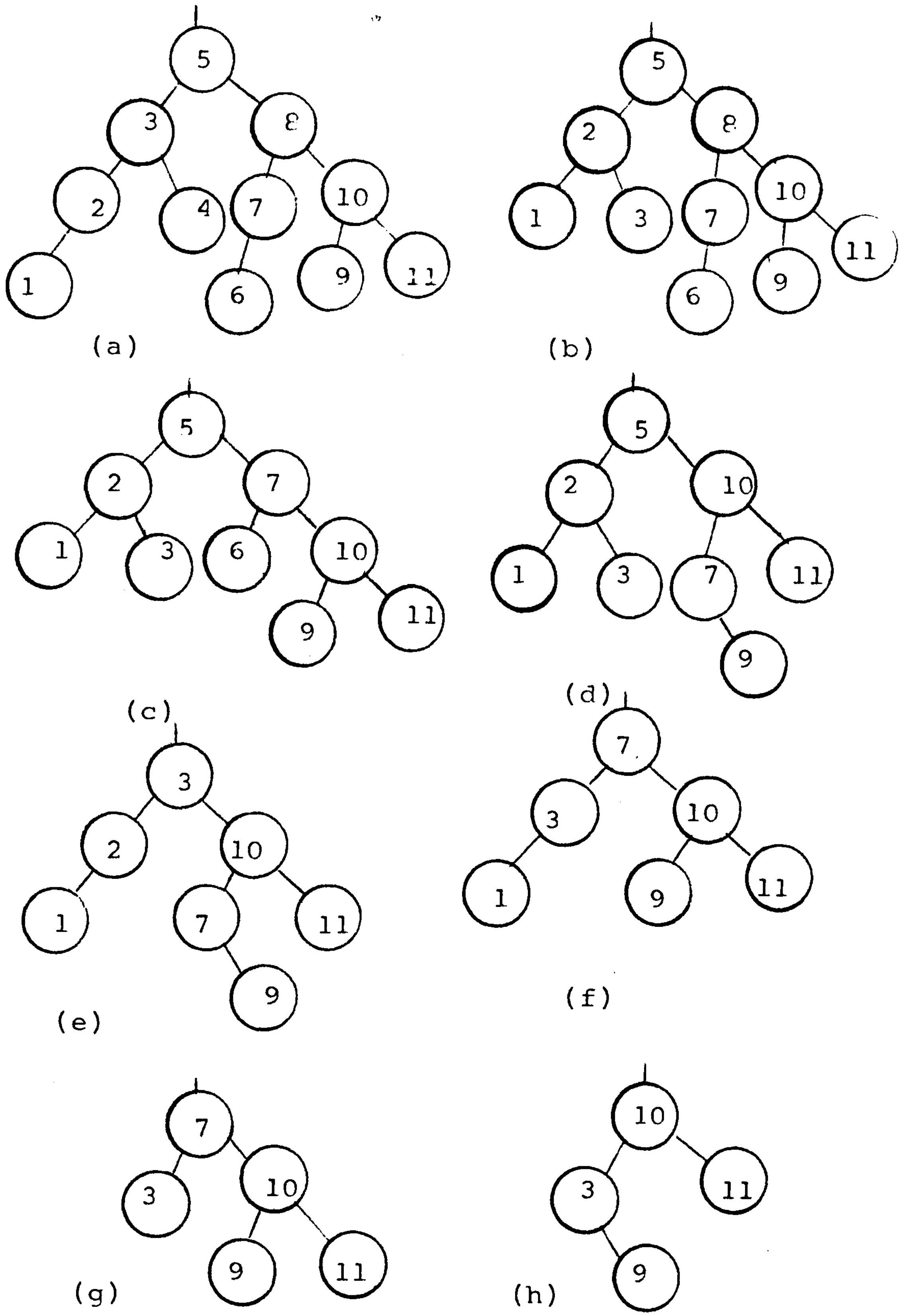

Figure 3-4: Deletions in balanced tree.

4. User's Manual

4.1 Introduction

AVL.pas is a PASCAL program aimed at designing the general purpose of Payroll System for ABC Company, by AVL tree and file structure. This system can contain any number of records, computes the netpay, taxpay of each employee, computes the average salary of each division and print the record you request interactively.

4.2 How to run AVL.pas

1. Execute AVL.pas, type TS if run by batch after the 'Trans', otherwise type TR if run by on-line after the 'Trans'.

2. Type 'B' or 'O' after the requestion "which way are you going to update ?", in which 'B' stand for Batch, 'O' for on-line.

3. Begining to update by typing 'A', 'D', 'I', 'L', 'E', in which stand for alter, delete, insert, list, and end respectively.

4. After a series of request on-line, type 'Y' or 'E' to see the general information, new report, paychecks.

5. After execution there are five files as : nl, n2, ne, re, ch, for the general information, newfile after update, new payroll report, and paychecks respectively.

6. Next month use the newfile as a master file to update the future's transaction file.

4.3 Results

4.3.1 The Output

1. Table 4-1 shows the transaction file.
2. Table 4-2, 4-3, 4-4 shows the new payroll record after update.
3. Table 4-5 shows that hows many employees in each division after update.
4. Table 4-6 shows that the average salary of each divisions after update.
5. Table 4-7 shows the paychecks of ABC Company's.
6. Table 4-8, 4-9, 4-10 shows the new master file after update.

4.3.2 Analysis

The program has process 20 transactions (batch or on-line process): 4 transactions were listed(for on-line only), 10 transactions were deleted, 4 transactions were inserted, 6 transactions were altered. Old master file had 102 records, New masterr file had 96 records.

4.4 Conclusions

After checking the output against copies of transaction file, old master file, new master file, note1, and note2, we can see that this program is precisely able to do the desired batch or on-line updating process.

In response to the requirements of Payroll system as Figure 4-1, the analysis and design has been made and presented with well designed system and program structure for the Payroll system. The analysis has been fully explained and the writer feels the proposed system meets all of the company's immediate needs as well as future expansion.

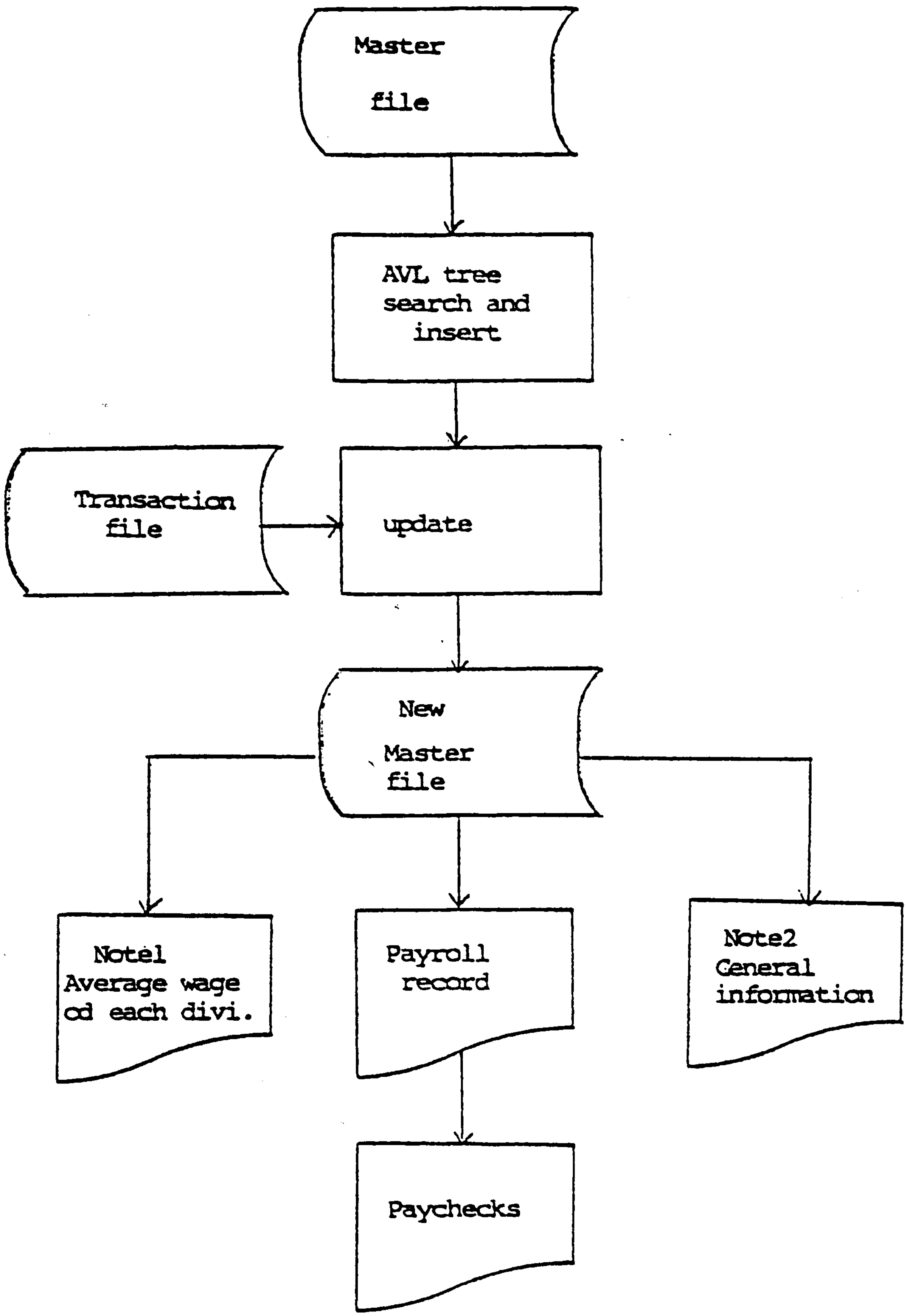

Figure 4-1: The Flowchart of Payroll System

```
D BILL
D YOUNG
D MARK
D SEVILLA
D KOCH
D SEEB
D NIZAR
D QUEEN
D VON
D WEISS
I LIANG F EXPORT 29000 1 10
I HUANG F CONTROL 26000 2 10
I JENNIFER F EXPORT 23000 1 10
I HIGH M EXPORT 32000 0 0
A TRACY M SALES 23000  3 10
A ZIMMERS M CONTROL 36000 3 0
A ALI M ACCOUNTING 46000 2 10
A KING M SALES 46000 0 10
A WAYE M DATA 25000 2 20
A LEIGHT F ACCOUNTING 30600 2 0
```

Table 4-1: The Transaction file of batch, and on-line process

ABC COMPANY
PAYROLL RECORD

PAGE 1

NAME	SEX	DIVISION	WAGE	DEP	HOUR	TAXPAY	NETPAY
ALI	M	ACCOUNTING	46000	2	10	726.67	3106.67
BOSCO	M	SALES	34000	3	10	486.67	2346.67
CANARY	F	DATA	31000	0	0	516.67	2066.67
CARTER	M	ACCOUNTING	35000	2	10	543.33	2373.33
CHANG	M	PERSONNEL	32000	0	20	613.33	2053.33
CHILL	M	PERSONNEL	23000	2	10	343.33	1573.33
COHEN	F	CONTROL	35000	0	0	583.33	2333.33
COX	F	ACCOUNTING	34500	3	0	455.00	2420.00
CRISTOPH	M	ACCOUNTING	33000	2	10	510.00	2240.00
DAVY	F	EXPORT	31000	0	0	516.67	2066.67
DILL	M	EXPORT	38000	0	10	673.33	2493.33
DIVID	F	ACCOUNTING	21000	0	0	350.00	1400.00
EDWARD	M	PERSONNEL	32000	1	20	573.33	2093.33
EDWIN	F	DATA	31000	1	20	556.67	2026.67
EUIN	M	ACCOUNTING	32000	1	20	573.33	2093.33
FILLER	M	EXPORT	23000	0	0	383.33	1533.33
FISH	M	DATA	24500	0	0	508.33	1633.33
FISHER	M	CONTROL	23000	0	0	383.33	1533.33
FLY	F	PERSONNEL	23000	2	10	343.33	1573.33
FORD	M	EXPORT	29000	1	20	523.33	1893.33
FRISBI	F	EXPORT	34000	1	20	606.67	2226.67
FRY	F	EXPORT	24000	0	0	400.00	1600.00
FULLER	M	DATA	34000	3	0	446.67	2386.67
GALLERY	M	SALES	35000	0	25	683.33	2233.33
GEBERT	M	EXPORT	21000	2	0	270.00	1480.00
GHOSH	M	DATA	32000	4	31	497.33	2169.33
GILL	M	CONTROL	23000	0	0	383.33	1533.33
GILMORE	F	DATA	31000	2	10	476.67	2106.67
GOOD	M	DATA	28560	0	0	476.00	1904.00
GORMAN	F	ACCOUNTING	24000	1	0	360.00	1640.00
GOTTLE	M	PERSONNEL	28000	2	10	426.67	1906.67
GULDEN	M	DATA	32000	3	21	497.33	2169.33

Table 4-2: The output of Payroll Record 1

ABC COMPANY
PAYROLL RECORD

PAGE 2

NAME	SEX	DIVISION	WAGE	DEP	HOUR	TAXPAY	NETPAY
HAMMER	M	ACCOUNTING	23000	2	20	383.33	1533.33
HAMPTON	M	SALES	34000	1	12	574.67	2258.67
HERPEN	M	PERSONNEL	21000	2	10	310.00	1440.00
HIGH	M	EXPORT	32000	0	0	533.33	2133.33
HILLMAN	F	ACCOUNTING	31000	2	10	476.67	2106.67
HOOLY	M	DATA	28700	3	10	398.33	1993.33
HUANG	F	CONTROL	26000	2	10	393.33	1773.33
JACKSON	F	CONTROL	20000	2	10	293.33	1373.33
JACOB	M	DATA	41000	2	0	603.33	2813.33
JENNIFER	F	EXPORT	23000	1	10	383.33	1533.33
JONATHAN	M	DATA	23000	2	10	343.33	1573.33
JOSEPH	M	ACCOUNTING	24000	0	0	400.00	1600.00
KENNEDY	M	SALES	23000	2	10	343.33	1573.33
KING	M	SALES	46000	0	10	806.67	3026.67
LASSER	M	SALES	28000	2	0	386.67	1946.67
LEE	M	CONTROL	34000	0	10	606.67	2226.67
LEIGHT	F	ACCOUNTING	30600	2	0	430.00	2120.00
LEWAZ	M	ACCOUNTING	25000	0	0	416.67	1666.67
LIANG	F	EXPORT	29000	1	10	483.33	1933.33
LITZ	M	SALES	25000	3	10	336.67	1746.67
LONG	M	SALES	29800	2	10	456.67	2026.67
MARVIN	M	ACCOUNTING	34000	2	10	526.67	2306.67
MCDONALD	M	SALES	29000	0	0	483.33	1933.33
MILLER	M	CONTROL	30000	0	0	500.00	2000.00
NEMO	M	SALES	23000	2	10	343.33	1573.33
NOMI	M	SALES	29000	2	10	443.33	1973.33
ONASIS	M	CONTROL	23000	2	10	343.33	1573.33
PETER	M	EXPORT	23000	1	40	503.33	1413.33
PHILLIPS	M	PERSONNEL	32000	2	0	453.33	2213.33
REAGAN	F	SALES	31000	4	30	476.67	2106.67
ROHMAN	F	ACCOUNTING	32000	2	10	493.33	2173.33
SEAGLE	M	ACCOUNTING	31000	1	10	516.67	2066.67

Table 4-3: The output of Payroll Record 2

ABC COMPANY
PAYROLL RECORD

PAGE 3

NAME	SEX	DIVISION	WAGE	DEP	HOUR	TAXPAY	NETPAY
SEMPLE	F	PERSONNEL	27000	2	10	410.00	1840.00
SEWIZ	M	PERSONNEL	34000	0	0	566.67	2266.67
SHERIF	M	CONTROL	21000	2	10	310.00	1440.00
SINGVOR	M	ACCOUNTING	21000	2	10	310.00	1440.00
SMITH	M	CONTROL	31000	1	10	516.67	2066.67
SNEER	M	SALES	23000	2	0	303.33	1613.33
SNYDER	M	SALES	25000	3	0	296.67	1786.67
SOOMER	F	ACCOUNTING	29000	1	20	523.33	1893.33
SPIZIKS	M	CONTROL	32000	2	10	493.33	2173.33
STENGLE	M	CONTROL	30000	2	20	500.00	2000.00
STURTWARD	M	PERSONNEL	23000	0	10	423.33	1493.33
SUMMER	M	PERSONNEL	23000	2	0	303.33	1613.33
SUNSHINE	M	ACCOUNTING	25000	0	0	416.67	1666.67
SWAN	M	CONTROL	31000	0	0	516.67	2066.67
SWENSEN	M	CONTROL	21000	0	0	350.00	1400.00
SWING	F	CONTROL	23000	1	0	343.33	1573.33
TAYLOR	M	EXPORT	31000	0	10	556.67	2026.67
TRACY	M	SALES	23000	3	10	303.33	1613.33
TRAN	M	EXPORT	25000	3	10	336.67	1746.67
TSAI	F	CONTROL	23000	2	0	303.33	1613.33
VOGEL	M	EXPORT	23000	2	20	383.33	1533.33
WANG	F	DATA	31000	1	0	476.67	2106.67
WARNER	F	PERSONNEL	30000	2	0	420.00	2080.00
WATTS	F	DATA	34000	0	0	566.67	2266.67
WAYE	M	DATA	25000	2	20	416.67	1666.67
WAYNE	F	DATA	29800	2	20	496.67	1986.67
WERNER	F	EXPORT	26500	1	12	449.67	1758.67
WHISKY	F	EXPORT	26400	0	0	440.00	1760.00
WHITNEY	M	DATA	32000	0	10	573.33	2093.33
WILSON	M	CONTROL	35000	0	0	583.33	2333.33
WOOD	F	EXPORT	32000	3	10	453.33	2213.33
ZIMMERS	M	CONTROL	36000	3	0	480.00	2520.00
TOTAL :	96					$43853.33	$185593.30

Table 4-4: The Output of Payroll Record 3.

```
BEFORE UPDATE THERE ARE
102   EMPLOYEES,
 31   FEMALE EMPLOYEES,
 71   MALE EMPLOYEES ,
 20   EMPLOYEES IN ACCOUNTING DIVISION,
 20   EMPLOYEES IN CONTROL DIVISION,
 18   EMPLOYEES IN DATA DIVISION,
 16   EMPLOYEES IN EXPORT DIVISION,
 13   EMPLOYEES IN PERSONNEL DIVISION,
 15   EMPLOYEES IN SALES DIVISION.

AFTER UPDATE THERE ARE
 96   EMPLOYEES,
 30   FEMALE EMPLOYEES,
 66   MALE EMPLOYEES ,
 18   EMPLOYEES IN ACCOUNTING DIVISION,
 18   EMPLOYEES IN CONTROL DIVISION,
 16   EMPLOYEES IN DATA DIVISION,
 17   EMPLOYEES IN EXPORT DIVISION,
 12   EMPLOYEES IN PERSONNEL DIVISION,
 15   EMPLOYEES IN SALES DIVISION.
```

Table 4-5: The output of General information

```
BEFORE UPDATE

THE AVERAGE WAGE OF ACCOUNTING DIVISION IS $  28055.00
THE AVERAGE WAGE OF CONTROL DIVISION IS $  29100.00
THE AVERAGE WAGE OF DATA DIVISION IS $  31031.11
THE AVERAGE WAGE OF EXPORT DIVISION IS $  27556.25
THE AVERAGE WAGE OF PERSONNEL DIVISION IS $  27384.62
THE AVERAGE WAGE OF SALES DIVISION IS $  29120.00

AFTER UPDATE

THE AVERAGE WAGE OF ACCOUNTING DIVISION IS $  28172.22
THE AVERAGE WAGE OF CONTROL DIVISION IS $  28111.11
THE AVERAGE WAGE OF DATA DIVISION IS $  31160.00
THE AVERAGE WAGE OF EXPORT DIVISION IS $  27700.00
THE AVERAGE WAGE OF PERSONNEL DIVISION IS $  27333.33
THE AVERAGE WAGE OF SALES DIVISION IS $  29120.00
```

Table 4-6: The output of Average Wage of each divisions.

```
***********************************************************
*                                                         *
*  ABC COMPANY                             NO.  1         *
*  BETHLEHEM,PA 18015                      JUNE 1, 1981   *
*                                                         *
*  PAY TO THE                                             *
*  ORDER OF  ALI                           $ 3106.67      *
*                                                         *
*                                                         *
*  FIRST NATIONAL BANK                                    *
*  ALLENTOWN,PA 18101                       SIGNATURE     *
*  01230234286 543910                      -----------    *
*                                                         *
***********************************************************
```

```
***********************************************************
*                                                         *
*  ABC COMPANY                             NO.  2         *
*  BETHLEHEM,PA 18015                      JUNE 1, 1981   *
*                                                         *
*  PAY TO THE                                             *
*  ORDER OF  BOSCO                         $ 2346.67      *
*                                                         *
*                                                         *
*  FIRST NATIONAL BANK                                    *
*  ALLENTOWN,PA 18101                       SIGNATURE     *
*  01230234286 543910                      -----------    *
*                                                         *
***********************************************************
```

Table 4-7: The output of Employees' paychecks.

ALI	M	ACCOUNTING	46000.00	2	10
BOSCO	M	SALES	34000.00	3	10
CANARY	F	DATA	31000.00	0	0
CARTER	M	ACCOUNTING	35000.00	2	10
CHANG	M	PERSONNEL	32000.00	0	20
CHILL	M	PERSONNEL	23000.00	2	10
COHEN	F	CONTROL	35000.00	0	0
COX	F	ACCOUNTING	34500.00	3	0
CRISTOPH	M	ACCOUNTING	33000.00	2	10
DAVY	F	EXPORT	31000.00	0	0
DILL	M	EXPORT	38000.00	0	10
DIVID	F	ACCOUNTING	21000.00	0	0
EDWARD	M	PERSONNEL	32000.00	1	20
EDWIN	F	DATA	31000.00	1	20
EUIN	M	ACCOUNTING	32000.00	1	20
FILLER	M	EXPORT	23000.00	0	0
FISH	M	DATA	24500.00	0	0
FISHER	M	CONTROL	23000.00	0	0
FLY	F	PERSONNEL	23000.00	2	10
FORD	M	EXPORT	29000.00	1	20
FRISBI	F	EXPORT	34000.00	1	20
FRY	F	EXPORT	24000.00	0	0
FULLER	M	DATA	34000.00	3	0
GALLERY	M	SALES	35000.00	0	25
GEBERT	M	EXPORT	21000.00	2	0
GHOSH	M	DATA	32000.00	4	31
GILL	M	CONTROL	23000.00	0	0
GILMORE	F	DATA	31000.00	2	10
GOOD	M	DATA	28560.00	0	0
GORMAN	F	ACCOUNTING	24000.00	1	0
GOTTLE	M	PERSONNEL	28000.00	2	10
GULDEN	M	DATA	32000.00	3	21

Table 4-8: The output of New Master file 1.

HAMMER,LEI	M	ACCOUNTING	23000.00	2	20
HAMPTON	M	SALES	34000.00	1	12
HERPEN	M	PERSONNEL	21000.00	2	10
HIGH	M	EXPORT	32000.00	0	0
HILLMAN	F	ACCOUNTING	31000.00	2	10
HOOLY	M	DATA	28700.00	3	10
HUANG	F	CONTROL	26000.00	2	10
JACKSON	F	CONTROL	20000.00	2	10
JACOB	M	DATA	41000.00	2	0
JENNIFER	F	EXPORT	23000.00	1	10
JONATHAN	M	DATA	23000.00	2	10
JOSEPH	M	ACCOUNTING	24000.00	0	0
KENNEDY	M	SALES	23000.00	2	10
KING	M	SALES	46000.00	0	10
LASSER	M	SALES	28000.00	2	0
LEE	M	CONTROL	34000.00	0	10
LEIGHT	F	ACCOUNTING	30600.00	2	0
LEWAZ	M	ACCOUNTING	25000.00	0	0
LIANG	F	EXPORT	29000.00	1	10
LITZ	M	SALES	25000.00	3	10
LONG	M	SALES	29800.00	2	10
MARVIN	M	ACCOUNTING	34000.00	2	10
MCDONALD	M	SALES	29000.00	0	0
MILLER	M	CONTROL	30000.00	0	0
NEMO	M	SALES	23000.00	2	10
NOMI	M	SALES	29000.00	2	10
ONASIS	M	CONTROL	23000.00	2	10

Table 4-9: The output of New Master file 2.

PETER	M	EXPORT	23000.00	1	40
PHILLIPS	M	PERSONNEL	32000.00	2	0
REAGAN	F	SALES	31000.00	4	30
ROHMAN	F	ACCOUNTING	32000.00	2	10
SEAGLE	M	ACCOUNTING	31000.00	1	10
SEMPLE	F	PERSONNEL	27000.00	2	10
SEWIZ	M	PERSONNEL	34000.00	0	0
SHERIF	M	CONTROL	21000.00	2	10
SINGVOR	M	ACCOUNTING	21000.00	2	10
SMITH	M	CONTROL	31000.00	1	10
SNEER	M	SALES	23000.00	2	0
SNYDER	M	SALES	25000.00	3	0
SOOMER	F	ACCOUNTING	29000.00	1	20
SPIZIKS	M	CONTROL	32000.00	2	10
STENGLE	M	CONTROL	30000.00	2	20
STURTWARD	M	PERSONNEL	23000.00	0	10
SUMMER	M	PERSONNEL	23000.00	2	0
SUNSHINE	M	ACCOUNTING	25000.00	0	0
SWAN	M	CONTROL	31000.00	0	0
SWENSEN	M	CONTROL	21000.00	0	0
SWING	F	CONTROL	23000.00	1	0
TAYLOR	M	EXPORT	31000.00	0	10
TRACY	M	SALES	23000.00	3	10
TRAN	M	EXPORT	25000.00	3	10
TSAI	F	CONTROL	23000.00	2	0
VOGEL	M	EXPORT	23000.00	2	20
WANG	F	DATA	31000.00	1	0
WARNER	F	PERSONNEL	30000.00	2	0
WATTS	F	DATA	34000.00	0	0
WAYE	M	DATA	25000.00	2	20
WAYNE	F	DATA	29800.00	2	20
WERNER	F	EXPORT	26500.00	1	12
WHISKY	F	EXPORT	26400.00	0	0
WHITNEY	M	DATA	32000.00	0	10
WILSON	M	CONTROL	35000.00	0	0
WOOD	F	EXPORT	32000.00	3	10
ZIMMERS	M	CONTROL	36000.00	3	0

Table 4-10: The output of New Master file 3.

References

1. Crogono, Peter. Programming in PASCAL,(2nd ed.), Addison Wesley Publishing Company, Inc., 1978.

2. Jensen, Kathleen and Wirth, Niklaus. User Manual and Report (2nd ed.) Soringer-Veriag, N. Y. 1974.

3. Wirth, Niklaus. Algorithms + Data Structures = Programs. Prentice- Hall, Inc., Englewood Cliffs, New Jersey, 1976.

4. Elson, Mark. Data Structures, Science Research Association, Inc., Chicago, Palo Alto, Toronto.

5. Alagic, Saud and Arbib, A Michael. The Design of Well-Structured and Correct Programs. Springer-Verlag, N. Y. Heidelberg, Berlin.

6. Luccio, Fabrizio and Pagli, Linda. "Correspondence on the Height- Balanced Trees." IEEE Transactions on Computers, Jan. 1976. PP.87-91.

7. Luccio, Fabrizio and Pagli, Linda. "Rebalancing Height-Balanced Trees." IEEE Transactions on Computers, Vol.c-27, no.5, May 1978, PP.386-396.

8. Nievergelt, J. and Reingold, E.M. "Binary Search Trees of Bounded Balanced." Siam J. Comput. Vol.2, no.1, March 1973. PP.33-43.

9. Foste, C.C. "A Generalization of AVL trees." Communications of the ACM. Aug. 1973, Vol. 16, no.8, PP.513-517.

10. Luccio, Fabrizio and Pagli, Linda. "Comment on Genralized AVL Trees." Communications of the ACM. July 1980, Vol. 23, no.7, PP.394-395.

11. Horowitz, Ellis and Zorat, Alessandro. "The Binary Tree as an Interconnection Network : An

Application to Multiprocessors Systems and VLSI." IEEE Transactions on Computers. Vol. c-30, no.4, Apr. 1981 PP.247-253.

12. Martin, W.A. and Ness, D.N. "Optimizing Binary Trees Grown with a Sorting Algoithm." Commuications of the ACM, Feb. 1972, Vol.15 no.2, PP. 88-93.

13. Samet, Hanan. "Deletion in Two-Dimensional Quad Trees." Communications of the ACM. Dec. 19■0, Vol. 23, no.12, PP.703-710.

14. Tan, K. C. "On Foster's Information Storage and Retrieval Using AVL Trees." Communications of the ACM, Sept. 1972, Vol. 15, no. 9 PP.843.

15. Foster, C. C. "Information Storage and Retrieval Using AVL Trees." ACM 20th National Conference/1965. PP.192-205.

16. Karlton, P. L., Fuller, S. H., Scroggs, R. E., and Kaehler, E. B. "Performance of Height-Balanced Trees." Communications of The ACM, Jan. 1976, Vol. 19, no. 1, PP. 23-28.

17. Nievergelt, J. "Bainary Search Trees and File Organization." Computing Survey, Vol. 6, No. 3, Sept. 1974, PP.195-207.

18. Baer, J. L. and Schwab, B. "A Comparison of Tree-Balancing Algorithm." Communications of The ACM, May 1977, Vol. 20, no. 5, PP.322-330.

Vita

Alice Ming-Mei Chen was born on August 9,1945 in Taoyuan, Taiwan, Republic of China. Her mother, Jung-May Peng is living in Taipei and her husband, John Jung-Chung Chen is studying in Industrial Engineering Department, Lehigh University.

From 1963 to 1967, she attended the National Cheng-Chi University in Taipei, Taiwan. After graduation in June 1967, she had taught Algebra in Hsin-Chu Junior Middle School for 5 months. In November 1967, she had started working for Bank of Taiwan, Export Division, Data Processing division, Service Division, Foreign Exchage Division for more than 10 years. During September 1972 and January 1973, she attended the Montana State University, Economics Department. In January 1979, she enrolled in Lehigh University, Department of Mathematics, Computer and Information science division.