UNIVERSITY OF CAPE TOWN

DEPARTMENT OF MATHEMATICAL STATISTICS

++++++++++++++++++++++++++

THE ADDRESS SORT AND OTHER COMPUTER SORTING TECHNIQUES

by

L.G. Underhill

++++++++++++++++++++++++++

A thesis prepared under the supervision of Professor C.G. Troskie in fulfilment of the requirements for the degree of Master of Science in Operations Research.

++++++++++++++++++++++++++

CONTENTS

I N T R O D U C T I O N

Originally this project was to have been a feasibility study of the use of computers in the library. It soon became clear that the logical place in the library at which to start making use of the computer was the catalogue. Once the catalogue was in machine readable form it would be possible to work backwards to the bookordering and acquisitions system and forwards to the circulation and book issue system.

One of the big advantages in using the computer to produce the catalogue would be the elimination of the "skilled drudgery" of filing. Thus vast quantities of data would need to be sorted. And thus the scope of this project was narrowed down from a general feasibility study, firstly to a study of a particular section of the library and secondly to one particularly important aspect of that section - that of sorting with the aid of the computer.

I have examined many, but by no means all, computer sorting techniques, programmed them in FORTRAN as efficiently as I was able, and compared their performances on the IBM 1130 computer of the University of Cape Town. I have confined myself to internal sorts, i.e. sorts that take place in core.

This thesis stops short of applying the best of these techniques to the library. I intend however to do so, and to work back to the original scope of my thesis.

My thanks go to Mr. Kingwill and Mrs. Snyman of the C.S.I.R. Library for introducing me to this field; to Miss Taylor and Mrs. Freislich at the University of Cape Town Library for explaining to me so clearly the problems of the library; to Professor Troskie, my supervisor and Head of the Department of Mathematical Statistics, for his interest and enthusiasm in this project. He has already applied the best of these techniques, the address sort, to his own research.

Finally I would like to thank Mrs. Cousins for typing the manuscript - especially for her patience with my handwriting and her careful setting out of the figures.

I acknowledge support received from the C.S.I.R. during the first year of this project.

My thanks go to Mr. Kingwill and Mrs. Snyman of the C.S.I.R. Library for introducing me to this field; to Miss Taylor and Mrs. Freislich at the University of Cape Town Library for explaining to me so clearly the problems of the library; to Professor Troskie, my supervisor and Head of the Department of Mathematical Statistics, for his interest and enthusiasm in this project. He has already applied the best of these techniques, the address sort, to his own research.

Finally I would like to thank Mrs. Cousins for typing the manuscript - especially for her patience with my handwriting and her careful setting out of the figures.

I acknowledge support received from the C.S.I.R. during the first year of this project.

CHAPTER 1

SECTION 1 BASIC SELECTION SORT

Procedure

There are two lists to be considered:

I: the input list

O: the output list.

Examine the input list, searching for the minimum item. When this "pass" is complete, place the minimum item at the first cell in the output list. After this transfer, we wish to look for the next smallest item in the input list. The minimum item is still in this list and will turn up during the second list examination. To eliminate this difficulty we replace the minimum item with some marker (an impossible value or maximum number). Note that this implies that, during the pass, we must have in temporary storage both the minimum item so far and its position in the input list.

During the second scan of the list the minimum item is absent, its place being occupied by the marker. The minimum selected on this pass is hence the next-to-minimum of the original input list. It is placed in the second cell of the output list and the place it occupied in the input list is overwritten with the marker.

The third scan produces the third item for the output list.

Continue until the number of items in the output list equals the number in the input list.

Comment

This is highly inefficient.

(a) It occupies a lot of space. Space has to be provided for both the input list, I, and the output list, O.

(b) A *complete* review of the input list is required each time an item is selected.

(c) There is a need for a marker to be inserted after each item is selected.

SECTION 2 SELECTION AND EXCHANGE SORT

Procedure

There is only one list to be considered. The input list, on completion of the sort, is the ordered list, ready for output.

As in the basic selection sort we scan the list to find its minimum.

Instead of placing this item in a new list, we exchange it with the first item in the list, i.e. we place the minimum item in the first cell and the contents of the first cell into the cell where the minimum occurred.

We now consider the sublist formed by excluding the first element. Scanning this sublist to find its minimum produces the next-to-minimum item since the minimum has been removed. Exchange the minimum of the sublist with the second cell's contents.

In general, after i operations on an input list with n items, we have an unordered sublist which contains n-i items. Its complement contains i ordered items. At each stage we find the minimum of the sublist and exchange it with the top item of the sublist.

Comment

In comparison with the Selection Sort, this technique halves both the storage space needed for data and the number of comparisons required.

SECTION 3 COUNTING SORT

Procedure

There are four lists to be considered:-

I: the original input list.

L: an unsorted list which is made up as each item from I is examined.

C: the set of counters which ranks each item in L.

S: the sorted output list.

(I and L are the same, and are only considered separately for explanation — in the program there is only one list of unsorted data.)

The procedure is to bring in successive items from I and place them in L. When an item is brought over from I to L it is compared with all previous items in L. A new counter records the rank of the item, and the other counters are adjusted to indicate their new rank taking into account the item just added.

Example

Refer to figure 1.

Clear all counters in C. (It is convenient in FORTRAN to set them all to 1.)

Bring the first item in I to L_1. (The subscript denotes the position in the vector L.) No counting or examination is done (figure 1(a)).

The next item is brought into L_2. As this is done, all cells above L_2 are reviewed. There is only one such cell, L_1.

We now need a rule for adjusting the counters as we examine the contents of L.

Rule:- If the latest item examined is larger than the earlier item, add 1 to the counter corresponding to the latest item.

If the latest item examined is smaller than the earlier item, add 1 to the counter corresponding to the earlier item.

Briefly, in any comparison add one to the counter corresponding to the larger item.

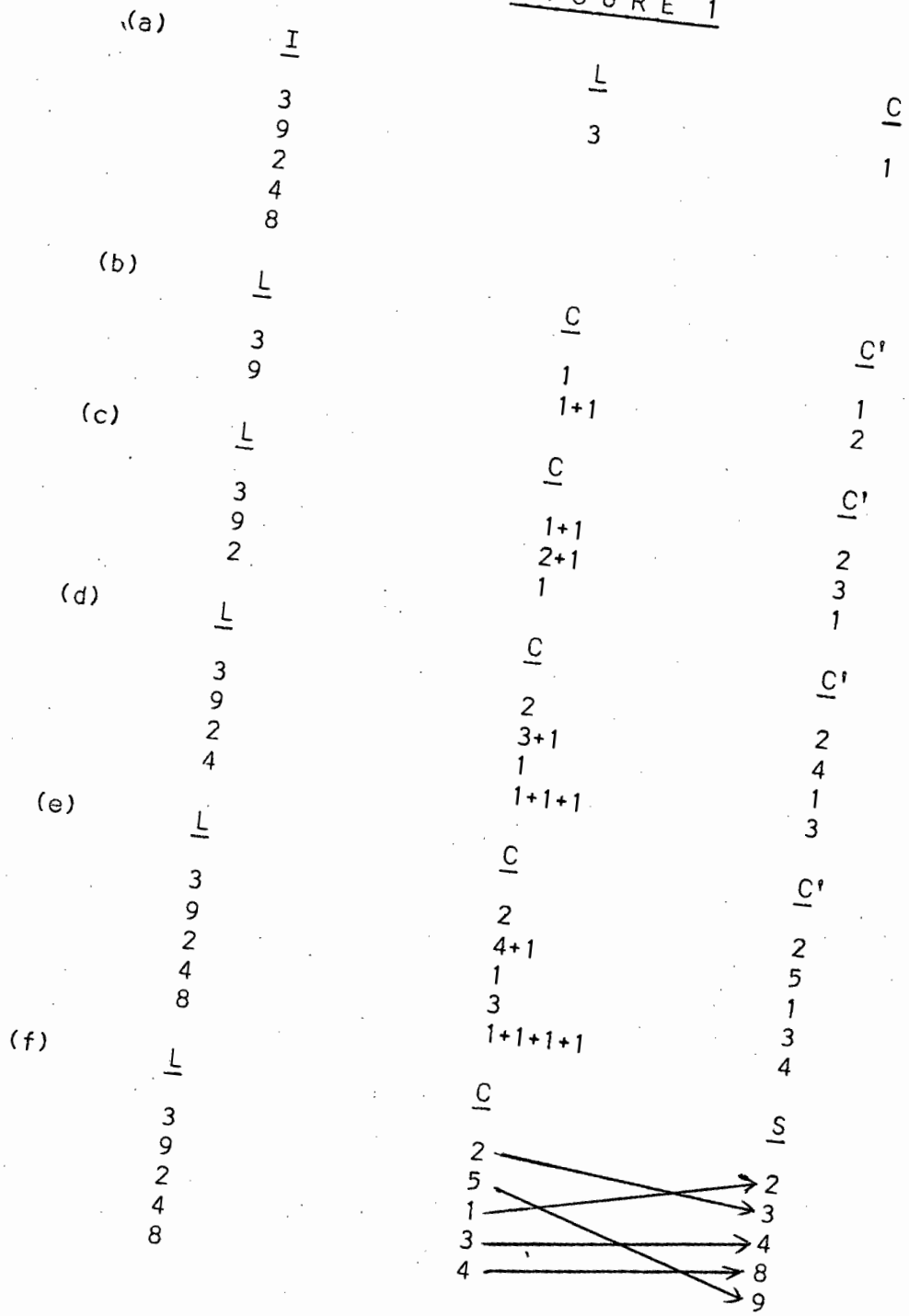
So if L_1 is greater than L_2, add 1 to C_1; if L_2 is greater than L_1, add 1 to C_2. (figure 1(b)).

Enter the next item at L_3. Compare L_3 with L_1 and L_2, and adjust the counters C_1, C_2 (figure 1(c)) and C_3 according the rule above.

Carry on until the list I is exhausted (figure 1(d) and (e)). Finally to make the sorted output list, transfer each item to the position

indicated by the counter. (figure 1(f)).

FIGURE 1



CHAPTER 2

SECTION 1 INSERTION SORTING

Basic Insertion Sort

In the various insertion sorts items are dealt with one by one and inserted into an ordered partial list in the position where they belong. When a new item arrives, we are sure that all items so far have been placed in order.

In the basic insertion sort we look at the ordered partial list and search item by item from the bottom of the list to find where the new item belongs. (The "bottom" of the list is the end which has the lower sort keys). This place will be occupied unless the item is greater than any item in the partial list and goes at the top. Move the item occupying the place where the new item belongs and all subsequent items one place up the list to make room for the insertion of the new item.

The partial list has grown by one, but is still in order.

Comment

The total number of comparisons required using this kind of linear search technique is excessive. For a list of n items it can be shown (see Appendix 1) that the expected number of comparisons is $\frac{1}{4}(n^2-n)$. The position of the item to be inserted can be found more efficiently using a binary search technique.

SECTION 2 IMPROVEMENT TO INSERTION SORTING — THE BINARY SEARCH

We are given an ordered list L and an item a to be inserted. Binary search prescribes that we split the list in two by finding the middle ("fence") item, compare a with the fence item and determine in which section a belongs.

We next take that half and find its fence item. We compare this item with a to determine in which quarter of the original list a should be inserted.

We continue in this way, comparing a with the fence items at closer and closer intervals, narrowing down the search until we find a fence item exactly equal to a, or until we find between which two items a lies. This second alternative occurs when we find that a is larger than one fence item, but smaller than the next, and the fences are one item apart in the original list.

Comment

The number of comparisons is now approximately $\log_2 n!$ (Appendix 1). For a list of 1 000 items this means roughly 8 500 comparisons as opposed to about 250 000 comparisons with the linear search.

SECTION 3 SECOND IMPROVEMENT TO INSERTION SORTING — THE UP/DOWN SHIFT

If the new item belongs to the first (bottom) half of the list, then instead of moving all larger items one position up the list, move all smaller items one position down the list.

The decision to move up or down depends only on the first fence item.

This technique halves the expected number of shifts required to insert the new item.

The first item is placed in the middle position of the output list so that there is room to move both up and down.

CHAPTER 3

SECTION 1 QUADRATIC SELECTION

Procedure

Given a list L, the procedure is outlined as follows.

(1) Divide the list L into sublists ${}^{i}L$.

(2) Find the smallest element of ${}^{i}L$ for each i, and enter it into cell L'_i of the auxiliary list L'.

(3) Find the smallest element of L', L'_k say, and store it for output.

(4) Refill L'_k with the next-to-minimum item from ${}^{k}L$.

(5) Repeat (3) and (4) until the number of items in the output list equals the number of items in L.

The optimal number of sublists, N, is given by

$$N = \sqrt{n} \quad \text{if} \quad N^2 = n$$

where n is the number of items.

Proof

Suppose n is a perfect square.

$$\text{Put} \quad N^2 = n.$$

Suppose that k is the number of items in each sublist, and that ℓ is the number of sublists.

$$\text{Then} \quad k\ell = n$$

$$\ell = n/k$$

The number of comparisons, C, needed for operations (3) and (4) are given by

$$C = (k-1) + (\ell-1)$$

$$= k-1 + n/k-1$$

$$\frac{\partial C}{\partial k} = 1-n/k^2$$

Setting this equal to zero to find the minimum number of comparisons yields

$$k^2 = n.$$

Hence $k = \sqrt{n} = N$ and also $\ell = N$.

If n is not a perfect square, take N to be the integer whose square is either nearest n or is the first square greater than n. Adjust the number of items allocated to each sublist so that their lengths are as nearly equal as possible.

At step (4), to replenish cell L'_k we review ${}^kL$, but we do not wish to choose the item chosen earlier. This is avoided by replacing an item chosen from a sublist by some maximum item denoted by z. Eventually some of the sublists will contain only z's. As soon as this occurs, we fill the cell in the auxiliary list corresponding to the exhausted sublist with z.

We now restate step (4) in more detail.

After the new item has been selected from L' for the *sorted* list

(i) review the sublist from which the item was drawn to find the minimum,

(ii) place the minimum in the cell of L' corresponding to that sublist.

(iii) Since the cell in the sublist from which the item was withdrawn still contains that item, we overwrite that item with z.

Example

Figure 2 depicts a quadratic selection sort for a list L of nine items. Figure 2(a) shows the subdivision of L into sublists ^1L, the formation of the auxiliary list L', and the selection of the first element for output.

Figure 2(b) shows the items chosen for the auxiliary list at the first stage replaced by maximum sort key z, the refilling of the auxiliary list and the output of the second item.

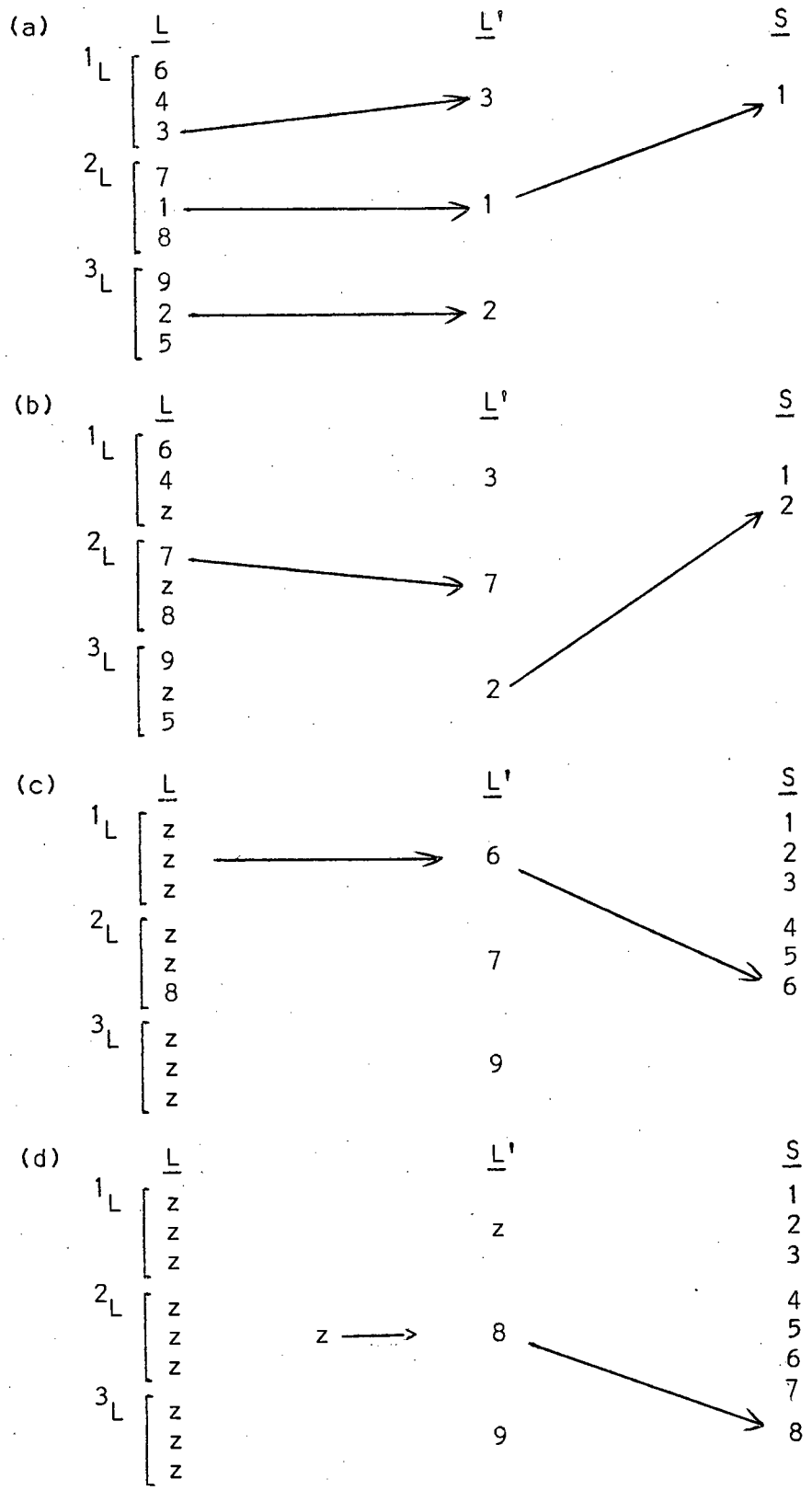
Figure 2(c) shows the sort at the stage when the first sublist is exhausted and a z is entered in L'. After the 6 is stored in the output list, the minimum item in ^1L is a z which is brought into L'.

Figure 2(d) shows a stage near the end of the sort when all the sublists are exhausted.

Comment

In this sort each of the items of a sublist is reviewed each time the corresponding auxiliary list item is chosen for output. This happens for each item of the sublist. Each sublist review amounts to a selection sort, a highly inefficient sorting technique.

F.I G U R E 2


(a)
L
L'
S
1L
6
4
3
2L
7
1
8
3L
9
2
5
3
1
1
2
(b)
L
L'
S
1L
6
4
z
2L
7
z
8
3L
9
z
5
3
7
2
1
2
(c)
L
L'
S
1L
z
z
z
2L
z
z
8
3L
z
z
z
6
7
9
1
2
3
4
5
6
(d)
L
L'
S
1L
z
z
z
2L
z
z
z
3L
z
z
z
z
8
9
1
2
3
4
5
6
7
8

SECTION 2 QUADRATIC SELECTION WITH PRESORT

Introduction

The quadratic selection sort can be vastly improved by arranging each sublist with a more efficient sort and then applying a modified quadratic sort procedure to the sorted sublist.

Suppose each sublist ${}^{i}L$ is presorted by some technique. To fill the auxiliary list L' simply take the bottom (smallest) item from each sublist and place it in the corresponding cell in L', i.e. move ${}^{i}L_1$ to L'_i for each i.

During the sort proper when we select the smallest item in the current auxiliary list L'_k, say, we go back to sublist ${}^{k}L$ and take the next element and place it in L'_k. This implies that we need a set of sublist counters, one for each sublist, which point to the next candidate for promotion to the auxiliary list for each sublist.

Procedure

(1) Divide L into N sublists ${}^{i}L$. These sublists are ordered by one of the techniques discussed earlier.

(2) The smallest item of each sublist is entered into the corresponding location in the auxiliary list L'. Since the sublists are ordered, the first item will be the smallest.

(3) Set the sublist counters, C_i $i = 1,\ldots,N$, equal to 2. These counters will be used to point to the next item to be taken to the auxiliary list from the ordered sublists.

(4) Select the smallest item from the auxiliary list and place it in the output list. Suppose L'_k is selected.

(5) The counter C_k points to the next item in sublist ${}^kL$ to be transfered to L', i.e. move ${}^kL_{C_k}$ to L'_k. This is done provided the sublist is not exhausted. If the sublist is exhausted enter a maximum number z into L' at position k.

(6) Increment C_k by 1, so that it contains the position of the next item for sublist ${}^kL$.

(7) Repeat (4), (5) and (6) until the number of items in the output list equals the number of items in L.

CHAPTER 4

TOURNAMENT SORT

Philosophy

The name is derived from the fact that in this sort items are matched against each other pairwise, the "winners" going through to the next round, getting paired off again, and so forth.

The winner of the final pair is the first item for the sorted list.

We then review the list again - it is however not necessary to review the entire list - to find the next "winner", the second item for the output list.

Procedure

The procedure is best illustrated by considering a simple example. List L in figure 3(a) contains sixteen numbers to be sorted. Sublists L^1 to L^4 contain the winners of the matched pairs in each round. The 2 occupying the first cell in L^1 is the winner (the minimum) of the comparison between the first pair of items in L, the 2 and the 6. The eventual overall winner is the 1 which came from the sixth pair in L.

We now replace the 1 where it appears in L by a maximum item z and adjust our lists by reconsidering only the sublist cells which contained the ultimate winner. The alterations to the lists are underlined in figure 3(b). Note that only four comparisons had to be made; it was not necessary to redo the entire table.

FIGURE 3

(a)

L	L^1	L^2	L^3	L^4
2				
6	2			
3		2		
9	3			
11			2	
16	11			
13		4		
4	4			
5				1
8	5			
1		1		
15	1			
7			1	
12	7			
14		7		
10	10			

(b)

L	L^1	L^2	L^3	L^4
2				
6	2			
3		2		
9	3			
11			2	
16	11			
13		4		
4	4			
5				2
8	5			
z		5		
15	15			
7			5	
12	7			
14		7		
10	10			

Now replace the 2 in L by z and go up its line making adjustments. When both items in a pair in any list or sublist have been transferred to the output list, a z is passed through to the next round.

Difficulties

A problem arises when the number of items is not a power of two. This can be overcome by adopting the following policy for the original list and all sublists:-

(i) If the number of items is even, pair off the items.

(ii) If the number of items is odd, add a maximum item z to the end of the list and pair off.

Figure 4 illustrates how this policy is implemented in a list of ten items. L can be paired off as it stands, but L^1 and L^2 require the addition of a z. L^3 contains only two items.

The advantages gained by this policy become evident at a later stage when we string out the sublists into a single list.

To replace the winning item with a z requires a knowledge of its original position. This problem is solved by storing the locations of the winning items rather than the items themselves. In figure 5 the cells of the list L are numbered - the sublists refer back to the main list. For convenience the maximum item z is called item 1, and the list proper starts in the second position.

Figure 5 contains all the information of figure 4. But in addition there is no need to search through L to find the minimum - we go directly to the seventh item, the 10, and replace it with z.

FIGURE 4

L	L^1	L^2	L^3	L^4
71				
93	71			
39		39		
84	39			
41			10	
10	10			
67		10		10
105	67			
53				
27	27			
		27		
	z		27	
		z		

FIGURE 5

	L	L^1	L^2	L^3	L^4
1	z				
2	71	2			
3	93		4		
4	39	4			
5	84			7	
6	41	7			
7	10		7		
8	67	8			7
9	105				
10	53	11			
11	27		11		
		1		11	
			1		

FIGURE 6

L	1	z	z
	2	71	71
	3	93	93
	4	39	39
	5	84	84
	6	41	41
	7	10	z
	8	67	67
	9	105	105
	10	53	53
	11	27	27
L^1	12	2	2
	13	4	4
	14	7	6
	15	8	8
	16	11	11
	17	1	1
L^2	18	4	4
	19	7	6
	20	11	11
	21	1	1
L^3	22	7	4
	23	11	11
L^4	24	7	11

FIGURE 7

	T
1	12
2	18
3	22
4	24

After replacing the 10 with a z, a new difficulty arises. Do we compare the seventh item with the sixth or with the eighth? The rule to follow here, allowing for the z in cell 1, is:-

(i) If the item number is even, compare it with the next item.

(ii) If the number is odd, compare with the previous item.

If lists L^1 L^2 L^3 and L^4 are strung out as in figure 6 it will be seen that each sublist starts at an even-numbered cell and this rule works for these sublists as well.

The mode of storage in figure 6 is the one adopted for storage of the sublists in the computer. However this introduces a further problem. The second column of figure 6 is easy to fill. The procedure is to compare items 2 and 3, and place the cell number of the smaller at the first even-numbered cell after the end of the list, in this case, cell 12. The smaller of items 4 and 5 is placed at cell 13, and so on. Make sure that each list starts on an even-numbered cell, even if the previous list finished at an even number. (L^1 is complete at position 16 - enter a 1 at cell 17 and start L^2 at 18. At this cell enter the smaller of the items cells 12 and 13 point to, i.e. the smaller of the second and fourth items. The 39 is the smaller, hence a 4 is entered at cell 18.)

In list L of the third column of figure 6 the minimum item of the original list, the seventh, has been replaced by a z. Following the rule above, we compare the seventh item with the sixth. The difficulty is to know where to place the winner of this comparison. To solve this problem it proves essential to store one more list, a table T in which T_i points

to the beginning of list L^i. This table, for the example in figure 6, is given in figure 7.

The rule for determining where in the next list the winner of the pair currently under consideration is to be entered is now simple.

If the overall winner, the item transferred to the output list, was item n, the position in L^1 for the winner of the first comparison is given by

$$T_1 + n_1^*$$

where $$n_1^* = \left[\frac{n}{2}\right] - 1$$

and the square brackets denote drop the fractional part.

The place in subsequent lists is given by

$$T_i + n_i^*$$

where $$n_i^* = \left[\frac{n_{i-1}^*}{2}\right]$$

In the situation in figure 6 where the seventh item was the winner, the position in L^1 is given by

$$12 + \left[\frac{7}{2}\right] - 1 = 12 + 2 = 14$$

In the subsequent lists the positions are given by

$$L_2: \quad 18 + \left[\frac{2}{2}\right] = 18 + 1 = 19$$

$$L_3: \quad 22 + \left[\frac{1}{2}\right] = 22 + 0 = 22$$

$$L_4: \quad 24 + \left[\frac{0}{2}\right] = 24$$

CHAPTER 5

NATURAL TWO-WAY MERGE

Introduction

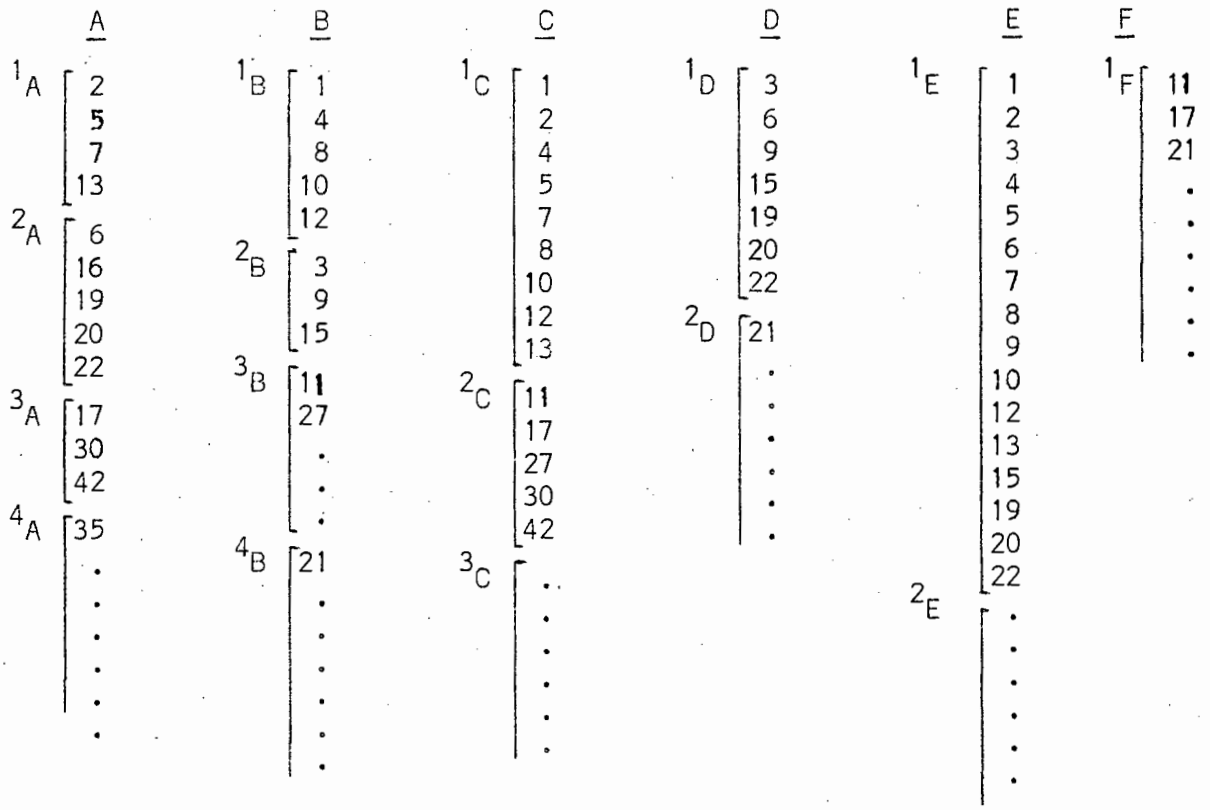
The sorting techniques considered so far have all commenced with a single list of unordered data. By contrast, to merge implies the existence of more than one list. If the data is in one list, it will be necessary as preparation for this merge to split it into two parts according to some rule. The simplest techniques would be to assign alternate items to alternate lists, or simply to split the original list at its midpoint.

Merging can best be explained by an example. In figure 8 we commence with lists A and B. These lists consist of a number of ordered sublists, ${}^{1}A, {}^{2}A, {}^{3}A, \ldots, {}^{1}B, {}^{2}B, {}^{3}B, \ldots$.

We merge together the first sublists from each list, ${}^{1}A$ and ${}^{1}B$ to form a single ordered sublist which we call ${}^{1}C$. ${}^{2}A$ and ${}^{2}B$ are merged and called ${}^{1}D$. Carry on merging corresponding sublists from A and B, placing the resulting longer strings alternatively in C and D. When lists A and B are exhausted, ${}^{1}C$ and ${}^{1}D$ are merged to form ${}^{1}E$, ${}^{2}C$ and ${}^{2}D$ form ${}^{2}F$ and so on. Clearly the sublists are getting fewer and eventually there is only one ordered list.

There is no need to know where each sublist begins. This is particularly valuable at the commencement of the sort, as it enables us to make use of any inherent ordering of the data. Hence the description "natural" in the name of this sorting technique.

FIGURE 8


A
1A 2 5 7 13
2A 6 16 19 20 22
3A 17 30 42
4A 35
B
1B 1 4 8 10 12
2B 3 9 15
3B 11 27
4B 21
C
1C 1 2 4 5 7 8 10 12 13
2C 11 17 27 30 42
3C
D
1D 3 6 9 15 19 20 22
2D 21
E
1E 1 2 3 4 5 6 7 8 9 10 12 13 15 19 20 22
2E
F
1F 11 17 21

Detailed Procedure

In figure 8, we look at the first items of lists A and B (the "input" lists) and choose the smaller - a 1 from B - and transfer it to C (an "output" list). We now compare the 2 from A with the 4 from B. Again we choose the smaller, the 2. We are now in a position to formulate our basic rule of approach.

RULE I

Compare the top items in each list; choose the smaller as the item to be transferred; replace it with the next item from the input list from which it was taken.

This rule works in the example in the figure until we choose the 12 from ^{1}B. The rule then tells us to compare the 13 from ^{1}A with the 3 from ^{2}B. However the 3 is to be ignored as it comes from ^{2}B. We take the 13 from ^{1}A as the next item for output. Lists ^{1}A and ^{1}B are now exhausted and we start using rule I again on the first elements of ^{2}A and ^{2}B. The 3 is the smaller of these two items and it is placed in the second output list, D, where it becomes the first element of sublist ^{1}D.

Two further rules are needed to determine the choice and placing of items when either or both sublists become exhausted. These rules are contained below in the sections outlining the procedure for dealing with the conditions known as single and double stepdown.

After each comparison and transfer, the lists are in one of three conditions.

I NO STEP DOWN

When the items from both lists are larger than or equal to the last item transferred, we have a *no stepdown condition.*

We choose the lesser as the item to be transferred as described in rule I.

II SINGLE STEP DOWN

When one of the pair of ordered sublists has been exhausted, the next item is smaller than the previous item on that sublist.

A *single stepdown condition* occurs when the next item on one list is *smaller* than the last item transferred *and* the next item on the other list is *larger* than the last item transferred.

This condition arose in figure 8 when we compared 16 from ^{2}A with 15 from ^{2}B and placed 15 in ^{1}D. This reveals 11 in the next cell in B. We have the condition for single stepdown since $11 < 15$ and $16 > 15$.

RULE II

Choose the larger item for output as long as the single stepdown condition prevails.

III DOUBLE STEP DOWN

In the example, figure 8, 22 is eventually placed into ^{1}D. We then find a 17 in A, and we have a 11 from B. Both 11 and 17 are smaller than 22, the last item transferred. A *double stepdown* condition has occurred.

This condition occurs when the next item on both lists is smaller than the last item transferred.

RULE III

When double stepdown occurs, start a new ordered sublist in the other output list. Carry on as for no stepdown.

In figure 8, rule III ensures that the ordered sublists are placed alternatively in output lists C and D.

There is one more eventuality.

ROLLOUT

Since the lists may contain an unequal number of sublists and since these are matched off in pairs during the merge, one of the lists may become exhausted before the other. In this case the remaining items on the unexhausted list are transferred to the output lists. Each ordered sublist is assigned to the output lists alternatively.

It is the policy of this library to sort each day's issues using the accession number as sort key.

We assume that the accession numbers were given to the books sequentially - they will be four-digit numbers.

Suppose further that each book has an equal chance of being issued on a given day. (This assumption is unlikely to be valid since newer books tend to be issued more frequently.) Statistically we are assuming that the sort keys are uniformly distributed random integers on the interval

$$[1 \quad 10\,000]$$

Now under these circumstances, it seems reasonable to estimate that the book with accession number 5318 should occupy roughly 53rd place in the final sort of the day's 100 issues.

Suppose the 53rd place is, at this stage, still unoccupied. We immediately place this record here, and deal with the next.

Suppose 5327 crops up later on - as is not impossible. This should also occupy roughly 53rd place. But this place is already occupied with a record which has lower sort key. We therefore check if the 54th place is occupied - if not, we place the incoming record here - if it is, we have to perform further rearrangements to fit in the new record.

The various situations which can arise, and the procedures adopted to carry out the rearrangements are described in detail in the next section.

In this example, the algorithm to convert the sort key into an "address" is clearly to divide by 100 and drop the fraction. Intuitively, this takes into account the rectangular distribution of the sort keys.

The address calculation algorithm is discussed in detail in Chapter 7.

SECTION TWO DESCRIPTION OF REARRANGEMENTS

Once the item has been transformed by the address calculation algorithm, there are five possible situations. For convenience we make use of the algorithm in the example above to explain these five cases. Case 1 deals with the simple and highly desirable situation when the calculated address points to an empty cell. Cases 2 to 5 handle the various situations when the calculated address is occupied. When this happens it is necessary to move either the item at this address, or the incoming item. If adjacent cells are also occupied this will necessitate moving several items, as well as tests to determine how many items need to be moved and in which direction these moves should take place.

CASE 1 CELL EMPTY

The incoming sort key is 1940. This number is mapped to address 19.

The state of the output list at this stage is given in figure 9 column S_1.

FIGURE 9

		S_1	S_2
	16	-	-
	17	1752	1752
	18	1890	1890
New item	19	-	1940
= 1940	20	-	-
	21	2188	2188
	22	-	-

The calculated address is empty. Place the incoming item at this position, giving S_2.

In a strict sense a cell in core is never "empty" - it *must* contain *something*. Before the sort commences the output list is initialized to some value which cannot occur in the list to be sorted. To test if an address is empty is to test if the cell contains this chosen value.

CASE 2 - CELL FULL - ROOM AT THE TOP

Figure 10 illustrates this case. The incoming sort key is 1988. Again this is mapped to address 19. But this address is already occupied by an item with key *less* than that of the incoming item. But the next cell is empty. Therefore place 1988 at address 20.

F I G U R E 10

		S_1	S_2
	16	-	-
	17	1752	1752
New Item	18	1890	1890
= 1988	19	1940	1940
	20	-	1988
	21	2189	2189
	22	-	-

In general, there may be a number of occupied cells between the occupied calculated cell and the next empty cell, but for CASE 2, all of these cells must contain items with sort keys less than that of the incoming item.

CASE 3 - CELL FULL - ROOM AT THE BOTTOM

The incoming item has sort key 2703. In figure 11 the calculated address, 27, is occupied by an item with key *greater* than that of the incoming item. Therefore the incoming item belongs at a lower address. But the cells 26 and 25 are both occupied by items with keys greater

than that of the incoming item. Cell 24 is vacant. So 2703 goes in here.

FIGURE 11

		S_1	S_2
	22	2219	2219
	23	-	-
New item	24	-	2703
= 2703	25	2730	2730
	26	2754	2754
	27	2781	2781
	28	-	-
	29	2953	2953

CASE 4 - SQUEEZE THE ITEM IN ABOVE

Consider the situation in figure 12. Fit in 3571.

FIGURE 12

		S_1	S_2	S_3
	33	3308	3308	3308
	34	3351	3351	3351
New item	35	3473	3473	3473
= 3571	36	3550	3550	3550
	37	3685	(3685)	3571
	38	3891	3685	3685
	39	-	3891	3891
	40	4068	4068	4068

Address 35 is occupied. Clearly the item must be placed between addresses 36 and 37, which are both already occupied. The nearest empty cell higher than the calculated address is cell 39.

The procedure is therefore to move the contents of cells 37 and 38 to cells 38 and 39 respectively, as in S_2, and to shift the incoming item into cell 37, S_3.

CASE 5 - SQUEEZE THE ITEM IN BELOW

In this case, illustrated in figure 13, the incoming item, 2137, is mapped to an address, 21, which is already occupied by an item with a higher key. Several lower addresses are also occupied - some with items having keys greater than that of the new item, some with lower keys.

(i) Find where the item belongs.

(ii) Find the next vacant cell with lower address.

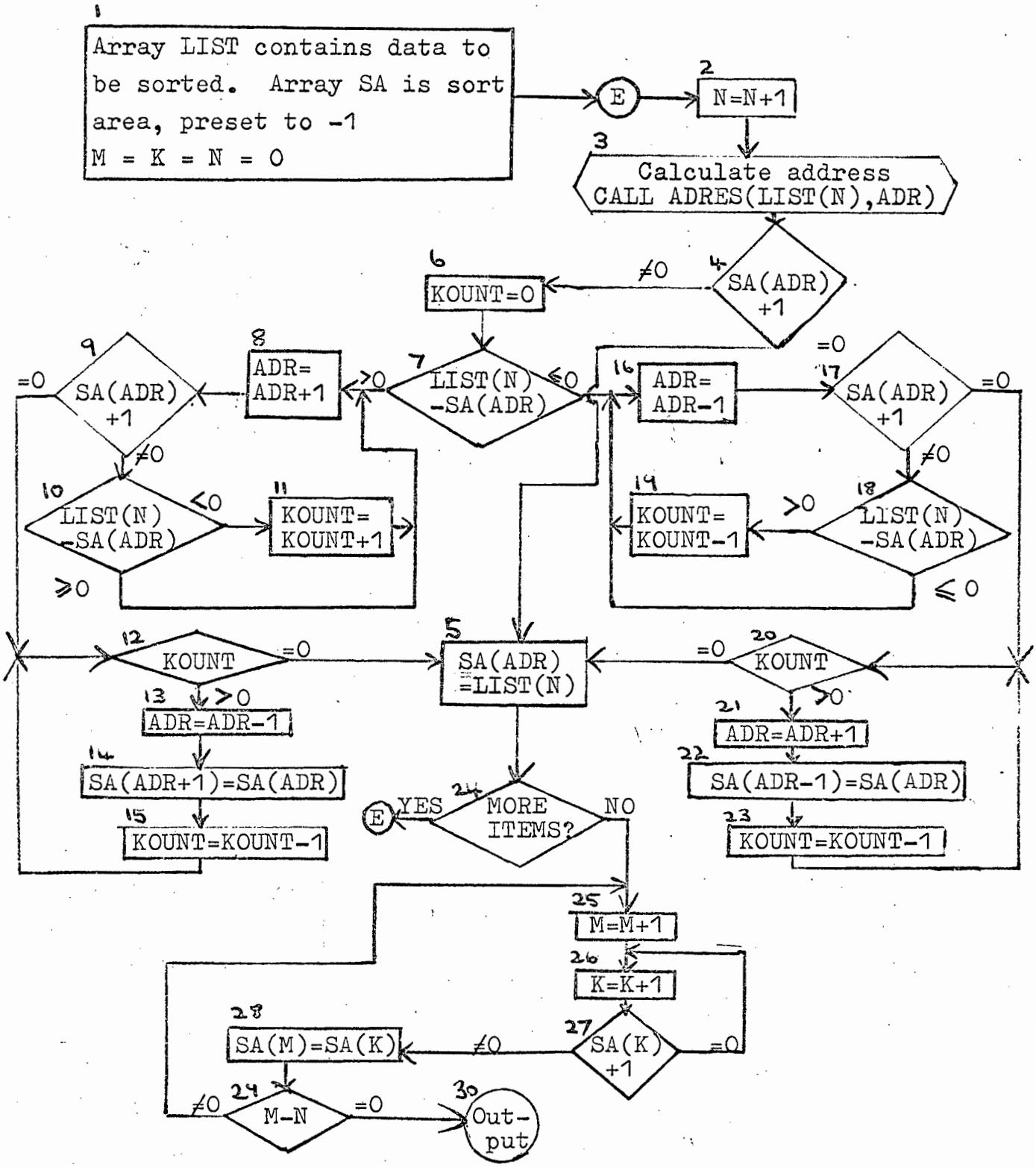
(iii) Starting by shifting one item into this vacant cell, move back by one space the items up to and including the cell where the incoming item belongs (S_2).

(iv) Now insert the incoming item into its proper position (S_3).

F I G U R E 13

		S_1	S_2	S_3
	16	-	1752	1752
	17	1752	1890	1890
New item	18	1890	1940	1940
= 2137	19	1940	1988	1988
	20	1988	(1988)	2137
	21	2189	2189	2189
	22	-	-	-

FIGURE 14


1
Array LIST contains data to be sorted. Array SA is sort area, preset to -1
M = K = N = 0
E
2
N=N+1
3
Calculate address
CALL ADRES(LIST(N),ADR)
4
SA(ADR) +1
≠0
6
KOUNT=0
=0
9
SA(ADR) +1
≠0
10
LIST(N) -SA(ADR)
<0
11
KOUNT= KOUNT+1
≥0
8
ADR= ADR+1
>0
7
LIST(N) -SA(ADR)
≤0
16
ADR= ADR-1
17
SA(ADR) +1
=0
≠0
19
KOUNT= KOUNT-1
>0
18
LIST(N) -SA(ADR)
≤0
5
SA(ADR) =LIST(N)
12
KOUNT
=0
13
ADR=ADR-1
>0
14
SA(ADR+1)=SA(ADR)
15
KOUNT=KOUNT-1
20
KOUNT
=0
21
ADR=ADR+1
>0
22
SA(ADR-1)=SA(ADR)
23
KOUNT=KOUNT-1
24
MORE ITEMS?
YES
NO
25
M=M+1
26
K=K+1
27
SA(K) +1
≠0
=0
28
SA(M)=SA(K)
24
M-N
≠0
=0
30
Out-put

SECTION 3 FLOW CHART

The flow chart for the address sort is given in figure 14. At (1) we initialize the sort area, in this case setting all cells in this array to -1. At (3) we call a subroutine which calculates the address, ADR, for the item currently under consideration, LIST(N).

The desirable properties of this subroutine are discussed in the next chapter. Suffice it to say now, the better the subroutine, the better the sort. We check (4) to determine if the calculated address is empty. (We assume that -1 cannot be a sort key.) If it is empty, we place the item at this address (5). If the calculated cell is occupied, we set a counter, KOUNT, (6) for later use. Next we compare the item with the contents of the calculated address to find out whether the new item should be placed below or above the calculated cell. We consider the cases where the comparison (7) yields *greater than* - i.e. we will be dealing with cases 2 and 4. Cases 3 and 5 are dealt with similarly (16 to 23).

At (8) and (9) we increment the calculated address by one and check if this address is occupied.

If it is empty we go to (12) and start moving items, if necessary. If it is occupied, we check its contents against the incoming item (10). If this item is greater than the contents we potentially have Case 2; no items need to be moved and we return to (8). If the incoming item is less than the contents at the present address, we have Case 4 which requires movement of items. We use the counter to determine how many

items need to be moved. Each time we go around this loop we add one to the counter.

Eventually we find an empty cell (9). We test the counter (12). If the counter is zero, no items need to be moved (Case 2) and we place the item at the cell the address is currently pointing to at (5). If the counter is not zero, the loop (12 to 15) moves the proper number of items up one position in the output area.

When these moves have been completed, we continue to (5) as before. If there are more items, (24), we calculate the address of the next item (2) and (3).

CHAPTER 7

SECTION 1 THE ADDRESS CALCULATION ALGORITHM

Introduction

The development of effective algorithms to map sort keys into addresses is crucial to the success of this sorting technique. The only algorithm we have considered so far is a simple one to deal with data sampled from a rectangular distribution. There are of course many other sort key distributions ranging from those which are nearly rectangular, through those which are normal, through skew distributions to those which merely have empirical forms - alphabetic sorts on bibliographic data in a library catalogue are an example of this last kind. The statistical distribution of the sort keys is one of two main considerations over which we have little or no control.

The second consideration, dictated to us by computer limitations, is that of core space available for use as working storage. As will become clear when we deal with fullness ratios the sort becomes more efficient when there are a great deal more cells available for the output array than there are items to be sorted.

In mathematical terms, we are given a population universe, Ω, containing all the sort keys which may possibly occur. The sample we are to sort, denoted by L, is a finite subset of Ω.

We are looking for a mapping, w, which maps Ω into the output list S, i.e.

$$w: \Omega \rightarrow S$$

S is a finite subset of the integers taking the form $\{1,2,\ldots,n\}$ where n is the number of cells available as working storage for the sort.

We consider fully the implications of the relationship between the population universe, Ω, and the sample to be sorted, L, i.e. the distribution, in a statistical sense, of L in Ω, in Section 2 of this chapter.

Fullness Ratio

We introduce now the concept of *fullness ratio*. The fullness ratio R is defined as $R = \frac{|S|}{|L|}$ where the vertical lines denote the cardinalities of the sets S and L.

In practical computer terms, this means the ratio between the number of items to be sorted, and the core available for the working storage of the output list.

In the example we considered earlier the fullness ratio was 1. This will be very inefficient because towards the end of the sort the nearest empty cell to a calculated address may be a considerable distance away.

If the number of locations in the output list is at least twice that of the number of items to be sorted then, given a good address calculation algorithm, this sorting technique is very efficient.

If on the other hand there are not many more output locations than

input items, the sort becomes progressively more inefficient. This inefficiency arises when the address calculated for an incoming item points to a location which is already occupied, forcing us to go through Cases 2 - 5. These cases are however unavoidable. If w were a one-to-one transformation there would be no collisions, but the working area would need as many locations as Ω has elements - and Ω will frequently be a continuum.

With a fullness ratio of, for example, two, half the cells in the sort area will still be unoccupied when the last item has been dealt with. The next stage is then to eliminate these empty cells by collecting up the occupied cells. Generally speaking it is possible to use the sort area itself for this process, as is shown in blocks (25 to 29) of figure 14.

SECTION 2 PROPERTIES OF THE ALGORITHM

Property 1

The address calculation formula must be a non-decreasing function.

An algorithm that maps sort key 3,08 to address 90, 5,74 to 65 and 7,30 to 138 is clearly not going to work!

Property 2

There may be no ambiguities. A single sort key may not be mapped to different addresses at different stages during the sort.

This precludes the possibility of "improving" the formula as one goes along. It is not possible to incorporate some technique to spread items in a section of the sort area which is turning out to be "overcrowded".

This emphasizes the importance of having a shrewd idea of the distribution of the sort key before commencing.

Property 3

To minimise the number of collisions, it is clear that each element of the output list must have an equal probability of occurring. If addresses in a short section of the output array keep cropping up while others are virtually unused, we clearly have a very unsatisfactory algorithm. This situation would arise if we used the address formula of the example in the introduction on normally distributed data with a mean of 5 000 and standard deviation of 250. We would then expect the addresses between 42 and 57 to be used repeatedly while the rest would be very rarely calculated.

To satisfy this property of equal probabilities for each address we are looking for a transformation that will map the distribution our data is sampled from into a rectangular distribution on $[0 , n]$ where n is the number of cells available for the output list.

Fortunately, this transformation is easy to find. If $f(x)$ is a density function then the distribution function $F(x) = \int_{-\infty}^{x} f(t)dt$ has the rectangular distribution on the interval $[0 , 1]$.

Proof

Consider the random variable X with density function $f(x)$ and distribution function $F(x)$ as defined above. Let Y be the random variable such that

$$Y = F(X)$$

Then the distribution function of Y,

$$H(y) = P\{Y < y\}$$

$$= P\{F(X) < y\} = \begin{cases} 1 & y > 1 \\ y & 0 \leq y \leq 1 \\ 0 & y < 0 \end{cases}$$

$$H(y) = P\{F(X) < y\} = P\{X < F^{-1}(y)\}$$

$$= F(F^{-1}(y)) = y \quad \text{for} \quad 0 \leq y \leq 1$$

Therefore $h(y) = \frac{dH(y)}{dy} = \begin{cases} \frac{d}{dy}\ 1 = 0 & \text{for} \quad y > 1 \\ \frac{d}{dy}\ y = 1 & 0 \leq y \leq 1 \\ \frac{d}{dy}\ 0 = 0 & y < 0 \end{cases}$

By following this with a scalar transformation, we can get a uniform distribution on any interval we desire.

Consider the random variable X with distribution function $F(x)$. Then for data sampled from this distribution

$$[n\ F(x)] + 1$$

will be an address calculation algorithm satisfying properties (1), (2) and (3). The square brackets denote "take the integral part" and n is the number of cells in the output away.

Proof

(1) and (2) are trivial. (3) Suppose x_i is the ith item in the array to be sorted. Let a_i be the calculated address for this item,

$$a_i = [n\ F(x_i)] + 1$$

Let k be an integer $1 \leq k \leq n$.

Then $P\{a_i \leq k\}$

$= P\{[n\ F(x_i)] + 1 \leq k\}$

$= P\{[n\ F(x_i)] \leq k-1\}$

$= P\{ n\ F(x_i) < k\}$

$= P\{F(x_i) < k/n\}$

$= k/n$

since $F(X)$ has the uniform distribution on $[0\ ,\ 1]$.

Similarly $P\{a_i \leq k-1\} = (k-1)/n$.

Hence $P\{a_i = k\} = P\{a_i \leq k\} - P\{a_i \leq k-1\} = k/n - (k-1)/n = 1/n$.

This is independent of k.

Property 4

An efficient address calculation formula must be speedily computable. It must be fast - a formula that involves lengthy calculations of complicated integrals is going to be time consuming.

Unless the distribution function has a simple closed form, properties (3) and (4) are in conflict. A balance must be struck between a precise formula which gives a perfect transformation to the rectangular distribution and the practical considerations of computer time. A useful compromise is to provide a table of values of the integral and to interpolate linearly - a table of 100 values or fewer is generally adequate.

In fact, since the data itself is never quite perfect the number of collisions with some good approximation to the algorithm is not likely to be many more than the number which the perfect algorithm would give. The time manipulating these extra collisions would be small compared with

the time spent calculating a marginally better address for every item.

Two Finer Points

1) A Special Difficulty

Difficulties arise at the ends of the sort table.

In the example in the introduction, towards the end of the sort we may have the situation shown in figure 15.

FIGURE 15

1	108
2	283
3	345
4	371
5	-

The next item to be sorted has key 332. Then the calculated address is 3, which is occupied by a cell of higher key. 332 should be allocated to address 2 and the contents of cells 1 and 2 be shifted to make room for it. The contents of cell 1 gets shifted out of the sort area - disastrous for a FORTRAN DIMENSION statement.

This difficulty is avoided by providing a safety zone of "padding" space at either end of the sort area.

Five percent of the sort area - two and a half percent at either end - is sufficient safety if the algorithm is used for the distribution it was designed for. If the Programmer fears, for example, that the sort he has written for normal data will be used to sort data from a lognormal distribution he would be wise to leave a larger margin of safety!

With a safety zone of p cells at either end of the n cells in the

sort array the address calculation formula becomes

$$[(n-2p)F^{*}(x)] + p$$

where $F^{*}(x)$ is either the distribution function itself, or some approximation to it.

2) Record or Record Index

In the sort area only the record index need be stored.

In the earlier example if the twenty first record had sort key 5318, a 21 would be placed at address 53, assuming cell 53 is still unoccupied. Any subsequent comparisons with this item which may be necessary due to collisions are made by comparing items with the sort key of the twenty first record and not with the actual contents of the fifty third address.

This has two advantages

(i) It saves space. Only the record index, and not the whole record is stored.

(ii) It is not necessary to move the whole record when collisions occur - only move the record index.

An extra instruction is needed before each comparison to look up the sort key of the record index.

CHAPTER 8

SOME PRACTICAL ADDRESS CALCULATION ALGORITHMS

Address formulae are considered here for the following types of data.

(i) Rectangular

(ii) Normal

(iii) Exponential

(iv) A general formula for data with a central tendency, whether skew or not.

These illustrate the principles involved in developing algorithms. The rectangular and exponential distribution have simple distribution functions whilst the normal distribution shows the table-based approach.

(i) The Rectangular Distribution

For the rectangular distribution on the interval $[A,B]$ the distribution function $F(x)$ is given by

$$F(x) = \frac{x-A}{B-A}$$

Making use of the formula

$$a_i = [(n-2p)F(x_i)] + p$$

derived earlier, we have

$$a_i = \left[\frac{(n-2p)(x_i-A)}{B-A}\right] + p$$

$$\approx \left[\frac{(n-2p)x_i}{B-A}\right] + \left(p - \frac{(n-2p)A}{B-A}\right)$$

$$= n^* x_i + p^*$$

n^* and p^* are constants determined before the commencement of the sort.

The algorithm is simply a linear transformation of the sort key.

With NSTAR and PSTAR predetermined constants the FORTRAN coding for this algorithm reduces to the single statement

```
IADRS = NSTAR * X(I) + PSTAR
```

(ii) The Normal Distribution

For this distribution the precise address calculation formula is

$$a_i = \left[(n-2p) \int_{-\infty}^{x_i} (2\pi\sigma^2)^{-\frac{1}{2}} \exp(-(2\sigma^2)^{-1}(t-\mu)^2)dt\right] + p$$

There are no ambiguities, it is non-decreasing and each address in the range $[p, n-p]$ has an equal chance of occurring. However it does not meet the requirements of speedy calculation! To overcome this difficulty we approximate by providing a table of values of the normal integral and by interpolating linearly.

The amount of care necessary for a satisfactory table together with the coding to perform the look-up operation is far less than the care required for a numerical integration subroutine. (The Scientific Subroutine Package Program QATR, integration of a function by trapezoidal rule, requires 386 words of core on the IBM 1130.)

The table shown in figure 16, has proved itself in practice. It consists of 61 points giving the area under the N(0 , 1) curve to the left of points at intervals of ,1 between -3 and 3.

Figure 17 illustrates the use of this table. Suppose we have 1 000 records with sort keys from N(12 , 16). Suppose n = 3 000 and we decide to let p = 250.

FIGURE 16

COLUMN 1		COLUMN 2		COLUMN 1		COLUMN 2	
-3.0	0.0000	1	250	0.1	0.5398	32	1599
-2.9	0.0019	2	254	0.2	0.5793	33	1698
-2.8	0.0026	3	256	0.3	0.6179	34	1794
-2.7	0.0035	4	258	0.4	0.6554	35	1888
-2.6	0.0047	5	261	0.5	0.6915	36	1978
-2.5	0.0062	6	265	0.6	0.7257	37	2064
-2.4	0.0082	7	270	0.7	0.7580	38	2144
-2.3	0.0107	8	276	0.8	0.7881	39	2220
-2.2	0.0139	9	284	0.9	0.8159	40	2289
-2.1	0.0179	10	294	1.0	0.8413	41	2353
-2.0	0.0227	11	306	1.1	0.8643	42	2410
-1.9	0.0287	12	321	1.2	0.8849	43	2462
-1.8	0.0359	13	339	1.3	0.9032	44	2507
-1.7	0.0446	14	361	1.4	0.9192	45	2547
-1.6	0.0548	15	387	1.5	0.9332	46	2582
-1.5	0.0668	16	417	1.6	0.9452	47	2612
-1.4	0.0808	17	452	1.7	0.9554	48	2638
-1.3	0.0968	18	492	1.8	0.9641	49	2660
-1.2	0.1151	19	537	1.9	0.9713	50	2678
-1.1	0.1357	20	589	2.0	0.9773	51	2693
-1.0	0.1587	21	646	2.1	0.9821	52	2705
-0.9	0.1841	22	710	2.2	0.9861	53	2715
-0.8	0.2119	23	779	2.3	0.9893	54	2723
-0.7	0.2420	24	855	2.4	0.9918	55	2729
-0.6	0.2743	25	935	2.5	0.9938	56	2734
-0.5	0.3085	26	1021	2.6	0.9953	57	2738
-0.4	0.3446	27	1111	2.7	0.9965	58	2741
-0.3	0.3821	28	1205	2.8	0.9974	59	2743
-0.2	0.4207	29	1301	2.9	0.9981	60	2745
-0.1	0.4602	30	1400	3.0	1.0000	61	2750
0.0	0.5000	31	1500				

F I G U R E 17

x_i	$\frac{x_i-\mu}{\sigma}$	x_{i_1}, x_{i_2}	$F(x_{i_1}), F(x_{i_2})$	e_i	$F^*(x_i)$	a_i
0,1	-2,975	-3,0 -2,9	0,0000 0,0019	0,0005	0,0005	251
0,2	-2,950	-3,0 -2,9	0,0000 0,0019	0,0010	0,0010	252
6,1	-1,475	-1,5 -1,4	0,0668 0,0808	0,0035	0,0703	425
6,2	-1,450	-1,5 -1,4	0,0668 0,0808	0,0070	0,0738	434
12,1	0,025	0,0 0,1	0,5000 0,5398	0,0099	0,5099	1524
12,2	0,050	0,0 0,1	0,5000 0,5398	0,0199	0,5199	1549

The first column gives the sort key, the second the sort key transformed to N(0 , 1). The third column shows which values in column 1 of figure 17 to look up. Column four shows the results of this operation.

Column five shows the linear interpolation according to the formula $e_i = 10(F(x_{i_2}) - F(x_{i_1}))(x_i - x_{i_1})$. Column six gives the approximation to the integral:

$$F^*(x_i) = F(x_{i_1}) + e_i$$

In the seventh column is the final address calculated by the formula

$$a_i = [(n-2p)\ F^*(x)] + p$$
$$= [2\ 500\ F^*(x)] + 250$$

The sort keys have been chosen to illustrate how more space is made available around the mean.

The probability that the transformed point lies outside the range of

the table, i.e. outside of $[-3 , 3]$ is about 0,0024 (one point in 350). Points like 24,27 -16,19 and others more than three standard deviations from the mean are given integral values of 1,0 and 0,0 respectively and their addresses are calculated from these values.

On the computer the actual technique used is not to convert sort keys to N(0 , 1) but to convert then to N(31 , 100).

The integral part of the transformed item is then the first of the two consecutive table values needed, and the fractional part is the proportion of the difference of these two table values needed for the linear interpolation.

If the integral part is less than one it is not equal to one and the fractional part is set to zero. If the integral part is greater than sixty it is set equal to sixty and the fractional part is set to 1,0.

As the size of the sort area, n, and the amount of padding, p, are predetermined, they can be incorporated into the table. Such a table with n = 2 500 and p = 250, is shown in column 2 of figure 16.

The FORTRAN statements needed to calculate the address of normally distributed data is shown in figure 18. The first instruction converts the item to N(31 , 100).

Then follows a check to prevent the looking up of values that lie beyond the table.

Finally the interpolation is performed, using the table in column 2 of figure 16, into which the size of the sort area and the padding space

has been incorporated.

If the mean and variance are unknown, they are estimated by evaluating the sample mean and variance before commencing the sort.

FIGURE 18

```
      SSDEV = 10./SDEV

      NORO1 = (X(J) - AMEAN) * SSDEV + 31.0
      INT = IFIX (NORO1)
      FRAC = NORO1 - INT
      IF (INT)    18 , 18 , 19
18    INT = 1
19     IF (INT - 60)  20 , 20 , 21
21    INT = 60
20    IADRS = TABLE(INT) + FRAC*(TABLE(INT + 1) - TABLE (INT))
```

(iii) The Exponential Distribution

The exponential distribution,

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

has the particularly simple distribution function

$$1 - e^{-\lambda x}$$

So a suitable algorithm would be

$$a_i = [(n-2p)(1-e^{-\lambda x_i})] + p$$
$$\approx n-p-[(n-2p)(e^{-\lambda x_i})]$$

(iv) A general formula for data with a central tendency, whether symmetric or skew

This is a very powerful address generating formula. It effectively handles data with a wide variety of distributions. It is most efficient when the sort keys are normal or near-normal. It is, however, sufficiently

robust to deal with even rectangularly distributed sort keys without breaking down.

There are three passes through the data, the first and second determining parameters used in the final pass which calculates the addresses.

The first pass determines the mean of the sort keys. The second pass yields two parameters which are estimates of the dispersion to the left and right of the mean.

If we have n sort keys $x_1, x_2, \ldots, x_n$ and mean $\bar{x} = \Sigma_{i=1}^{n} x_i$ then these parameters are given by

$$s_\ell^2 = 1/n_\ell \; \Sigma_{i=1}^{n} (x_i - \bar{x})^2 \; \chi_\ell(x_i)$$

where $$\chi_\ell(x) = \begin{cases} 1 & \text{for } x \leq \bar{x} \\ 0 & \text{for } x > \bar{x} \end{cases}$$

and $$n_\ell = \Sigma_{i=1}^{n} \chi_\ell(x_i)$$

and $$s_r^2 = 1/n_r \; \Sigma_{i=1}^{n} (x_i - \bar{x})^2 \; \chi_r(x_i)$$

where $$\chi_r(x) = \begin{cases} 0 & \text{for } x \leq \bar{x} \\ 1 & \text{for } x > \bar{x} \end{cases}$$

and $$n_r = \Sigma_{i=1}^{n} \chi_r(x_i)$$

s_ℓ^2 and s_r^2 are used for the normalization of the sort keys, according to formula

$$\frac{x_i - \bar{x}}{\sqrt{s^*(x_i)}}$$

where $$s^*(x_i) = \begin{cases} s_\ell^2 & x_i \leq \bar{x} \\ s_r^2 & x_i > \bar{x} \end{cases}$$

The third pass through the data performs this normalization and

calculates the addresses from the same table as is used for the sorting of normally distributed data.

Thus, the addresses are calculated as if the data to the left of the mean had been sampled from a normal distribution with mean $\bar{x}$ and variance s_ℓ^2, and the data to the right of the mean is treated as if it came from $N(\bar{x},s_r^2)$.

The final program in Appendix II contains a FORTRAN listing of this algorithm.

Figure 19 gives an indication of how this technique fits data derived from a number of situations. In these graphs the dotted line represents the empirical distribution of the data. The solid line is the distribution function by which the address sort approximates the empirical distribution. The graphs are all plotted from three "left deviations" (s_ℓ) below the mean to three "right deviations" above it. They have been scaled so that the horizontal axes are of equal length.

In figure 19(a) the data are 500 sort keys generated by the scientific subroutine GAUSS, which produces normally distributed random numbers with a given mean and deviation.

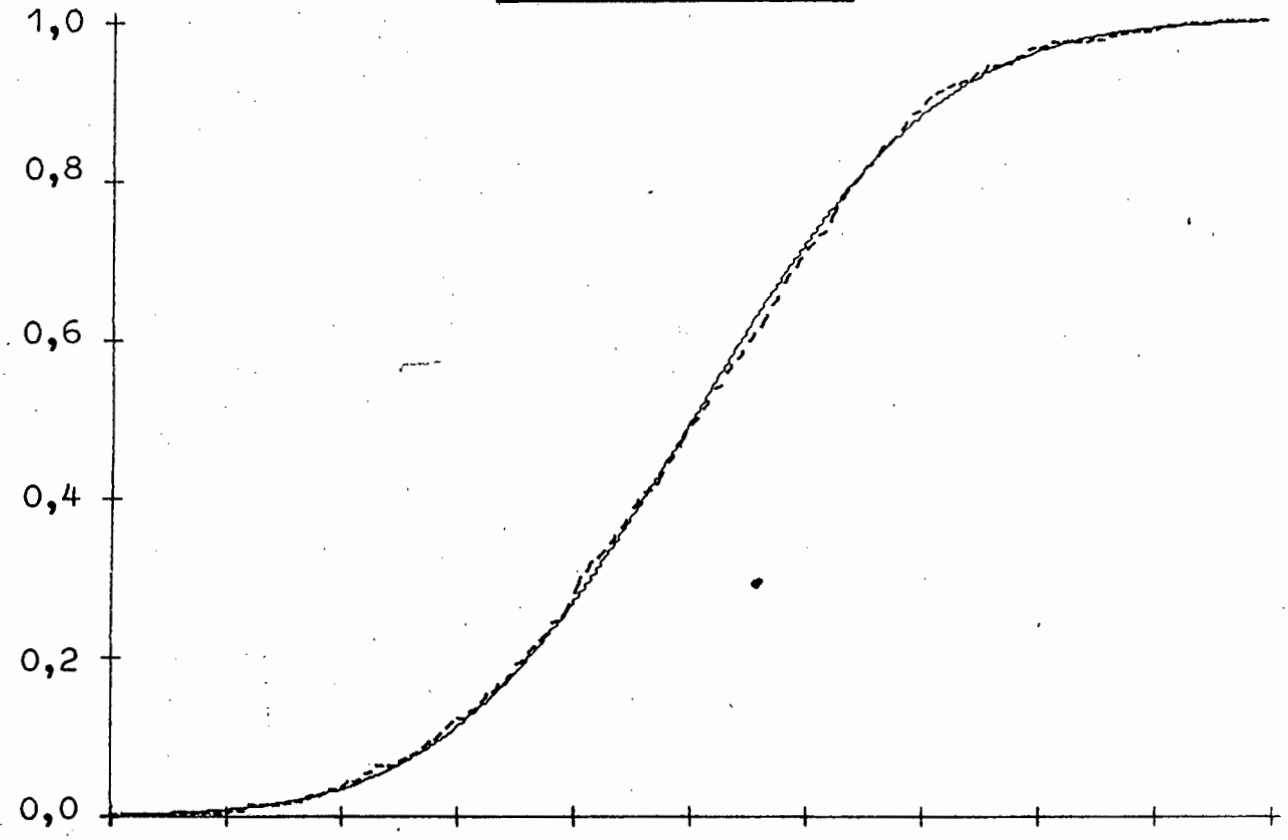
In figure 19(b) the data are the appreciations since the beginning of 1970 to March 1971 of 738 share prices on the Johannesburg Stock Exchange. (Appreciation = 100 Move/Last Effective Price.)

Figure 19(c) shows how the technique deals with data with a lognormal distribution.

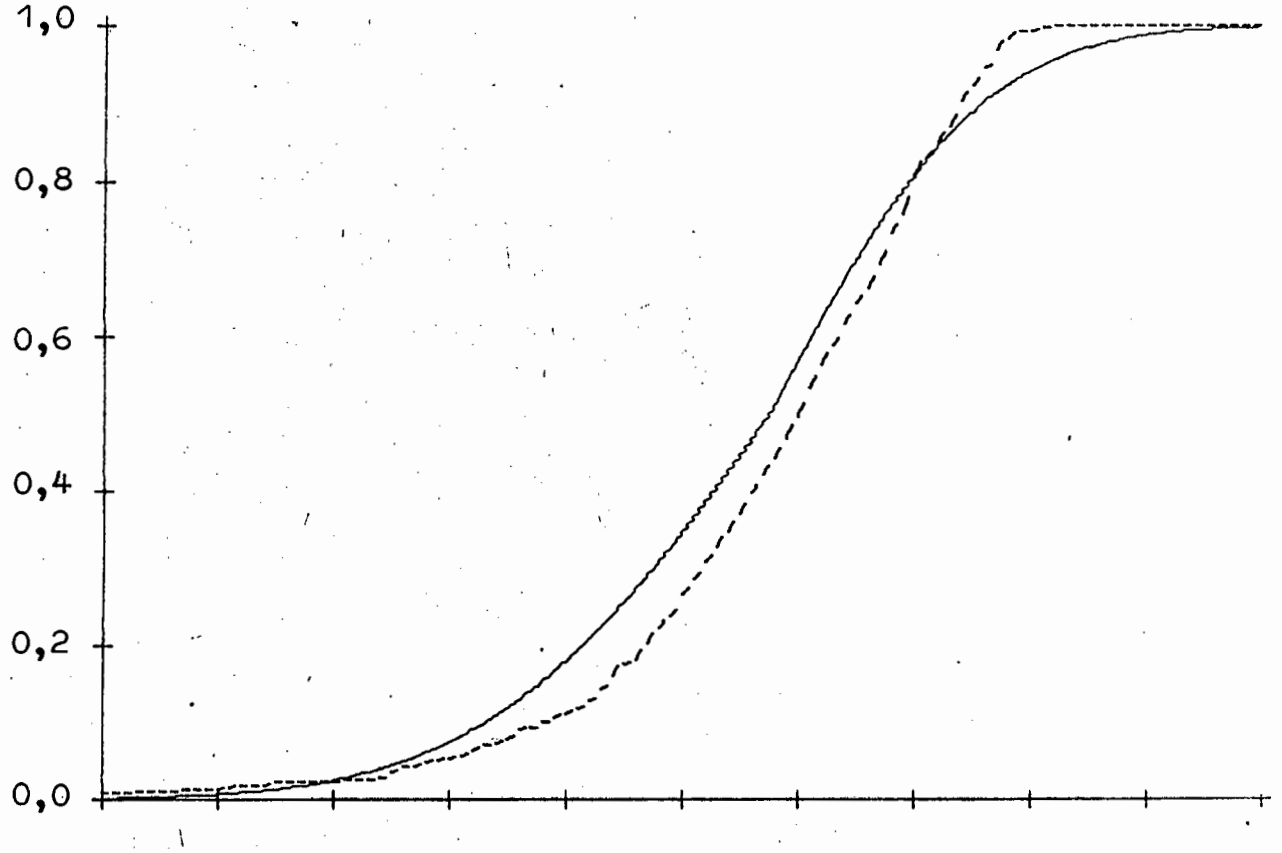
In figure 19(d) the data are the volume density of shares traded on the Johannesburg Stock Exchange during a week.
(Volume density = 100 Volume traded/Total volume issued.)

Although the fit in graphs (c) and (d) is rather poor, the increase in sorting time is very small.

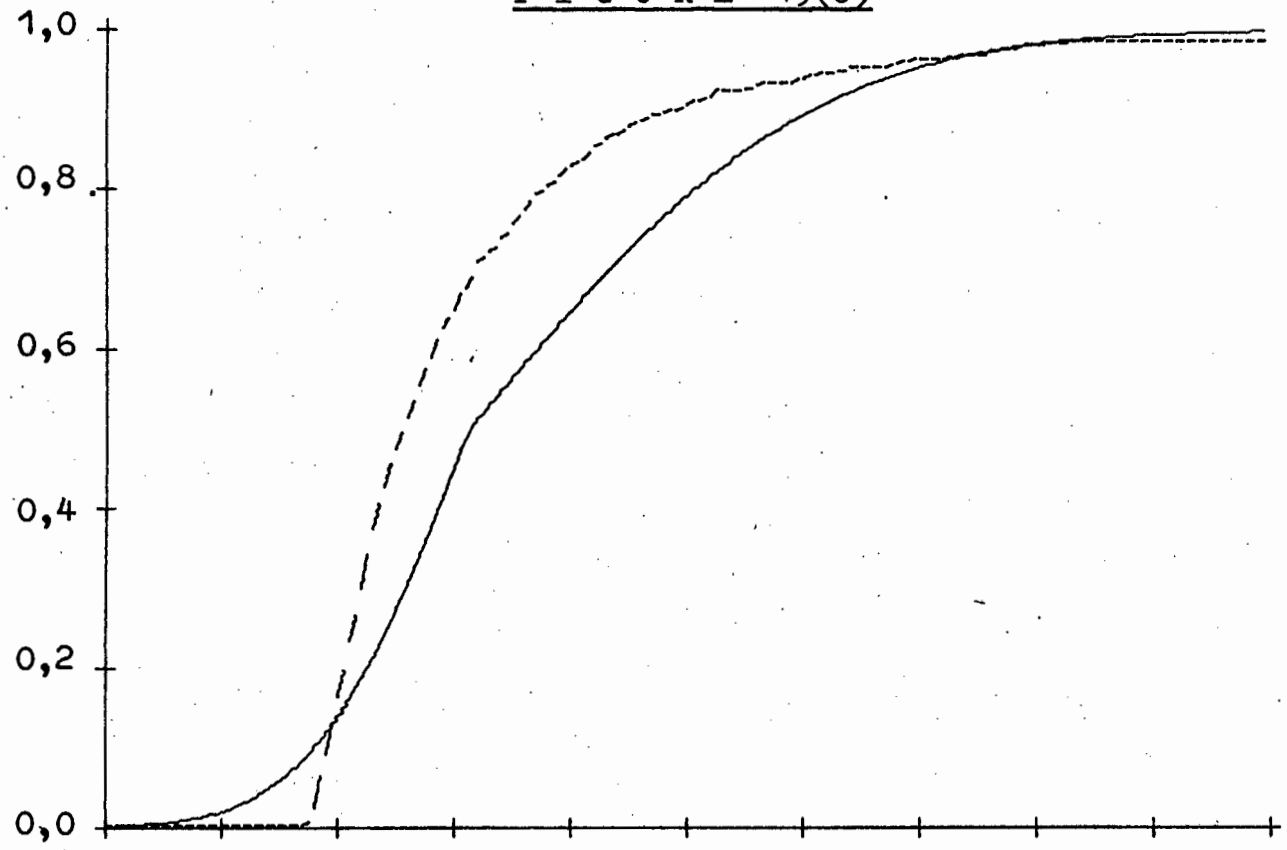
F I G U R E 19(a)



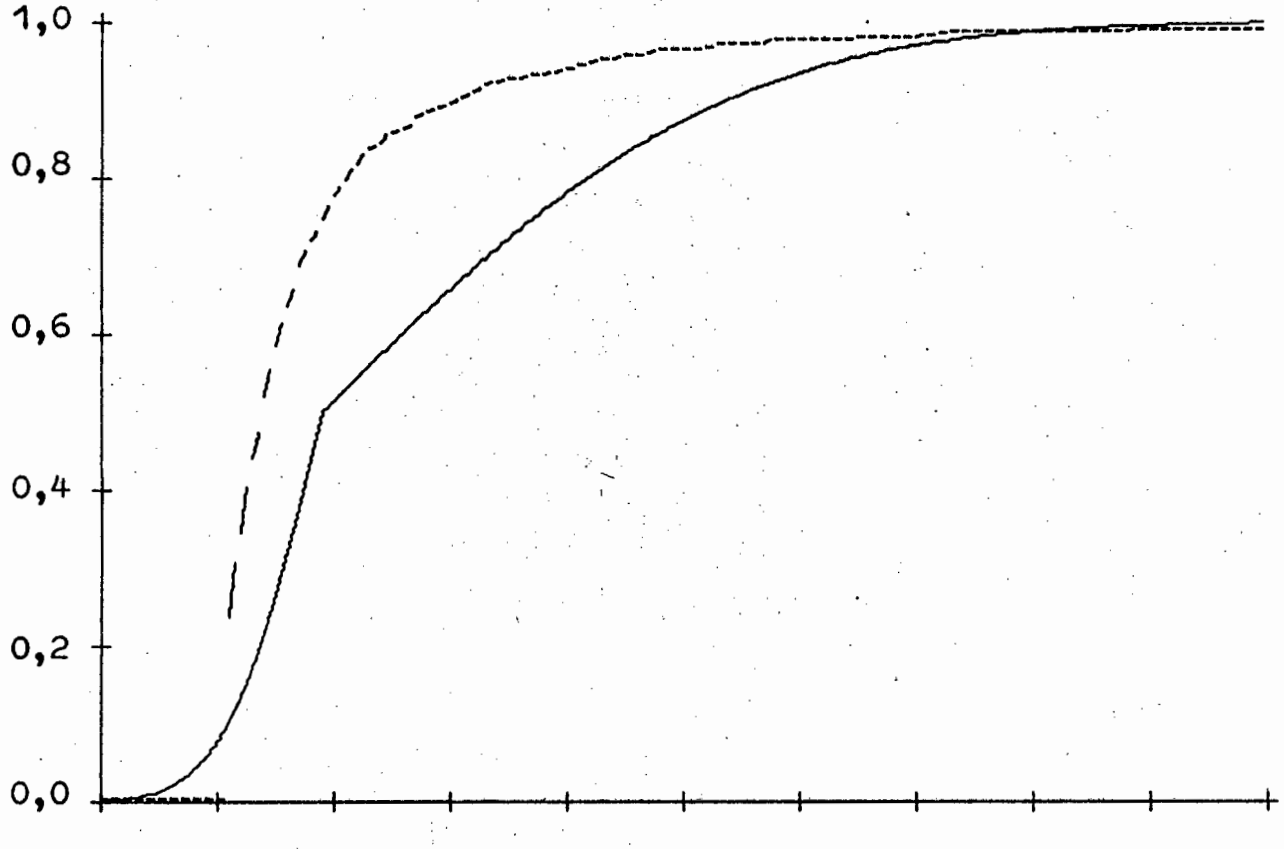
F I G U R E 19(b)



F I G U R E 19(c)



F I G U R E 19(d)



A P P E N D I X I

Figure 20 gives a comparison of the various sorting techniques described earlier. Brief derivations of core requirements and the theoretical number of comparisons for each technique follow.

The sort time given in the final column was the time taken to sort 1 000 random integers. These were generated by the scientific subroutine RANDU. The programs were timed by instructing the computer to print a line of asterisks immediately preceding and following the actual sort instructions.

1. SELECTION

Core is required for both input and output lists.

N-1 comparison needed for each item, giving a total of N(N-1) comparisons.

2. SELECTION AND EXCHANGE

Only one array in core is required.

When the list is of length n, n-1 comparisons are needed to find the next item. Hence total number of comparisons is

$$\Sigma_{n=1}^{N-1} n = \tfrac{1}{2} N(N-1)$$

3. COUNTING

Core is required for the input and output lists and the counters.

Comparisons derived as in 2.

4. BASIC INSERTION

Only one array in core is required. As the sort proceeds this list is split into an ordered and unordered sublist.

When the ordered sublist has length n, the expected number of comparisons is $n/2$. Hence total number of comparisons is

$$\tfrac{1}{2} \Sigma_{n=1}^{N-1} n = \tfrac{1}{4}N(N-1)$$

5. BINARY INSERTION

Core requirements as in 4.

When the ordered sublist has length n we expect

$$[\log_2 n] + 1$$

comparisons to locate the position for new item.

Hence total number of comparisons is approximately

$$\Sigma_{n=1}^{N} \log_2 n = \log_2 N!$$

6. BINARY INSERTION WITH UP/DOWN SHIFT

Core required for original list. Output list requires $2N$ cells of core since first item is place and centre of list.

Comparisons are as for 5, but number of shifts is halved.

7. QUADRATIC SELECTION

$N + \sqrt{N}$ cells needed for sublists and auxiliary list.

N cells needed for output list. $(\sqrt{N}-1)$ comparisons needed to select item from each sublist for auxiliary list. Hence $\sqrt{N}(\sqrt{N}-1)$ comparisons are needed to form auxiliary list. To output the first item

requires a further $\sqrt{N}-1$ comparisons. For each of the remaining N-1 items $2(\sqrt{N}-1)$ comparisons are needed. The overall total is

$$(N-1)(2\sqrt{N}-1).$$

8. TOURNAMENT

N cells of core required for input list and for output list. $N/2^{k+1}$ cells are needed for kth sublist. Hence total cells for sublists

$$\begin{aligned} & N/2 + N/4 + \ldots + 2 \\ \approx\ & N(\tfrac{1}{2} + \tfrac{1}{4} + \ldots) \\ =\ & N \end{aligned}$$

Thus total core requirements are approximately 3N.

Number of comparisons to fill the sublists is N. For each of the remaining N-1 items $2(\lceil \log_2 N \rceil + 1)$ comparisons are made.

Total number of comparisons is

$$N + 2(N-1)(\lceil \log_2 N \rceil + 1)$$

9. NATURAL TWO-WAY MERGE

Since it is impossible to predetermine which list will eventually contain all the tiems, it is necessary that each of the four lists has N cells allocated to it.

For each item during each pass, three comparisons are required. At worst, if the data are in reverse order, $\lfloor \log_2 N \rfloor + 1$ passes will be required.

Hence number of comparisons will be less than

$$3N(\lceil \log_2 N \rceil + 1)$$

10. ADDRESS CALCULATION

Core is required for the input list and for the sort area. With fullness ratio R, the total core requirements are thus

$$(N+1)R$$

The probability of a collision when there are n occupied cells in the sort area is n/RN.

The expected number of collisions is thus $(RN)^{-1} \sum_{n=1}^{N-1} n = (N-1)/2R$

With $R = 3{,}0$ and $N = 1\ 000$ this yields 167 collisions.

For each item there is one comparison to test if the calculated address is occupied. For each collision there are at least three extra comparisons. Hence the total number of comparisons will exceed

$$N + 3(N-1)/2R$$

For values of R greater than about 2,5 we can expect that the total number of comparisons will not be much larger since the probability of adjacent cells being occupied is small.

FIGURE 20

SORTING TECHNIQUE	CORE REQUIREMENTS FORMULA	CORE REQUIREMENTS N = 1000	COMPARISONS FORMULA	COMPARISONS N = 1000	TIME N = 1000
					min:secs.
SELECTION	2N	2000	N^2-N	999000	5:43
SELECTION AND EXCHANGE	N	1000	$\frac{1}{2}(N^2-N)$	499500	2:54
COUNTING	3N	3000	$\frac{1}{2}(N^2-N)$	499500	4:16
BASIC INSERTION	N	1000	$\frac{1}{4}(N^2-N)$	249750	4:17
BINARY INSERTION	N	1000	$\log_2(N!)$	8500	2:49
BINARY INSERTION WITH UP/DOWN SHIFT	3N	3000	$\log_2(N!)$	8500	1:31
QUADRATIC SELECTION	$2N+\sqrt{N}$	1035	$(N-1)(2\sqrt{N}-1)$	65000	:37
TOURNAMENT	3N	3000	$N+2N(\lceil\log_2 N\rceil+1)$	19000	:27
NATURAL TWO WAY MERGE	4N	4000	less than $3N(\lceil\log_2 N\rceil+1)$	27000	:27
ADDRESS	(R+1)N	(R+1)1000	greater than $N+3(N-1)/2R$	1500 -	:07 for R=3 :13 for R=1

A P P E N D I X II

A listing of the FORTRAN coding for each of the sorting techniques described earlier (with the exception of Quadratic Selection with Presort) follows.

Comment cards describe each section of the programs - if these are read in conjunction with the detailed descriptions, it should be possible to follow the coding.

These programs have been carefully tested. The timings given in Appendix I were obtained from them when a thousand items of data were sorted.

The final listing is that of the currently most advanced version of the Address Sort. It is written as a subroutine and returns to the mainline program the ranking of the data to be sorted.

```
                    L.G. UNDERHILL    ***   SELECTION SORT   VERSION  4   ***

      INTEGER   INPUT(1000),OUTPT(1000)                                  LES  SS
      DEFINE FILE 2 (1000,1,U,KREC)                                      LES  SS
      DATA  IPRNT/5/                                                     LES  SS
  101 FORMAT(' ************************************************************') LES  SS
  100 FORMAT  (1X,12I10)                                                 LES  SS
      WRITE(IPRNT,101)                                                   LES  SS
      N  =  1000                                                         LES  SS
                                                                         LES  SS
C*****READ LIST TO BE SORTED                                            LES  SS
      READ (2'1) INPUT                                                   LES  SS
                                                                         LES  SS
                                                                         LES  SS
C***** N  PASSES THROUGH DATA                                           LES  SS
                                                                         LES  SS
      DO  2  J=1,N                                                       LES  SS
                                                                         LES  SS
C*****SET  MIN  TO FIRST ITEM ON LIST                                   LES  SS
C*****SET  LEAST  TO POINT TO THIS ITEM                                 LES  SS
      MIN = INPUT(1)                                                     LES  SS
      LEAST = 1                                                          LES  SS
                                                                         LES  SS
                                                                         LES  SS
C*****TO C***1 DETERMINES MINIMUM ITEM ON LIST AND ITS LOCATION         LES  SS
                                                                         LES  SS
      DO  4  K=2,N                                                       LES  SS
      IF ( INPUT(K) - MIN)              3, 4, 4                          LES  SS
    3 LEAST = K                                                          LES  SS
      MIN = INPUT(K)                                                     LES  SS
    4 CONTINUE                                                           LES  SS
                                                                         LES  SS
C***1                                                                    LES  SS
                                                                         LES  SS
                                                                         LES  SS
C*****TO C***2 STORES MINIMUM ITEM AND REPLACES IT BY MAXIMUM INTEGER    LES  SS
C***** 1130 CAN HANDLE                                                   LES  SS
                                                                         LES  SS
      OUTPT(J) = MIN                                                     LES  SS
    2 INPUT(LEAST)=32767                                                 LES  SS
                                                                         LES  SS
C***2                                                                    LES  SS
                                                                         LES  SS
                                                                         LES  SS
      WRITE(IPRNT,101)                                                   LES  SS
      WRITE(IPRNT,100)   OUTPT                                           LES  SS
      CALL EXIT                                                          LES  SS
      END                                                                LES  SS

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    2018  PROGRAM     168

 END OF COMPILATION

// XEQ          1                                                       LES  SS

*FILES(2,RNDMO)                                                         LES  SS
```

L.G. UNDERHILL *** SELECTION AND EXCHANGE SORT VERSION 2 ***

```
      INTEGER  INOUT(1000)                                                  LGU  SES
      DATA  IPRNT/5/                                                        LGU  SES
      DEFINE FILE2 (1000,1,U,KREC)                                          LGU  SES
  100 FORMAT  (1X,12I10)                                                    LGU  SES
  101 FORMAT(' ************************************************************') LGU  SES
      N  =  1000                                                            LGU  SES
      READ(2'1) (INOUT(I), I= 1,N)                                          LGU  SES
      WRITE(IPRNT,101)                                                      LGU  SES
                                                                            LGU  SES
                                                                            LGU  SES
C***** N-1 PASSES THROUGH DATA (ON (N-1)TH PASS FINAL PAIR OF ITEMS        LGU  SES
C*****ORDERED)                                                             LGU  SES
                                                                            LGU  SES
      DO  1  J = 2,N                                                        LGU  SES
      J1  =  J-1                                                            LGU  SES
                                                                            LGU  SES
C*****SET  MIN  TO FIRST ITEM OF UNSORTED SUBLIST                          LGU  SES
C*****SET  LEAST  TO POINT TO THIS ITEM                                    LGU  SES
      MIN  =  INOUT(J1)                                                     LGU  SES
      LEAST  =  J1                                                          LGU  SES
                                                                            LGU  SES
                                                                            LGU  SES
C*****TO C***1 DETERMINES MINIMUM ITEM AND ITS LOCATION IN SUBLIST         LGU  SES
      DO  4  K  =  J,N                                                      LGU  SES
      IF  (INOUT(K)-MIN)      3, 4, 4                                       LGU  SES
    3 MIN  = INOUT(K)                                                       LGU  SES
      LEAST=K                                                               LGU  SES
    4 CONTINUE                                                              LGU  SES
                                                                            LGU  SES
C***1                                                                       LGU  SES
                                                                            LGU  SES
                                                                            LGU  SES
C*****TO C***2 EXCHANGES MINIMUM ITEM WITH FIRST ITEM OF SUBLIST           LGU  SES
C*****SORTED SUBLIST GROWS BY ONE ITEM, UNSORTED SUBLIST SHRINKS ONE       LGU  SES
C*****ITEM                                                                 LGU  SES
                                                                            LGU  SES
      ISAVE  =  INOUT(J1)                                                   LGU  SES
      INOUT(J1)  =  INOUT(LEAST)                                            LGU  SES
    1 INOUT(LEAST)=ISAVE                                                    LGU  SES
                                                                            LGU  SES
C***2                                                                       LGU  SES
                                                                            LGU  SES
                                                                            LGU  SES
      WRITE(IPRNT,101)                                                      LGU  SES
      WRITE(IPRNT,100)  INOUT                                               LGU  SES
      CALL EXIT                                                             LGU  SES
      END                                                                   LGU  SES

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    1020  PROGRAM     200

 END OF COMPILATION

// XEQ          1                                                         LGU  SES
```

L.G. UNDERHILL *** SORTING BY COUNTING VERSION 3 ***

```
      INTEGER ISORT(1000)                                              LGU  CS
      INTEGER  LIST(1000),KOUNT(1000)                                  LGU  CS
      DATA  KOUNT,IPRNT / 1000*1,5/                                    LGU  CS
      DEFINE FILE  1(1000,1,U,KREC)                                    LGU  CS
  101 FORMAT(' **********************************************************') LGU  CS
  100 FORMAT  (1X,12I10)                                               LGU  CS
      READ(1'1) LIST                                                   LGU  CS
      WRITE (IPRNT,101)                                                LGU  CS
      N  =  1000                                                       LGU  CS
                                                                       LGU  CS
C*****ALL COUNTERS IN ARRAY  KOUNT  PRESET TO ONE BY DATA STATEMENT     LGU  CS
                                                                       LGU  CS
                                                                       LGU  CS
C***** N-1 PASSES  (FIRST PASS DEALS WITH TWO ITEMS)                   LGU  CS
      DO  5  I=2,N                                                     LGU  CS
                                                                       LGU  CS
C*****THESE INSTRUCTIONS SAVE THE COMPUTER REFERENCING THE SAME        LGU  CS
C*****DIMENSIONED ITEM REPEATEDLY  -  A TIME CONSUMING OPERATION ON     LGU  CS
C*****THE IBM 1130                                                     LGU  CS
      LISTI = LIST(I)                                                  LGU  CS
      KOUNI = KOUNT(I)                                                 LGU  CS
                                                                       LGU  CS
C*****TO C***1 COMPARES  I TH ITEM OF LIST WITH ALL PREVIOUS ITEMS AND LGU  CS
C*****INCREMENTS COUNTERS IN  KOUNT  ACCORDING TO THE RULE  -  ADD ONE LGU  CS
C*****TO THE COUNTER CORRESPONDING TO THE LARGER ITEM                   LGU  CS
      I1=I-1                                                           LGU  CS
      DO  2  J=1,I1                                                    LGU  CS
      IF(LISTI-LIST(J))       1, 3, 3                                  LGU  CS
    3 KOUNI   =KOUNI+1                                                 LGU  CS
      GO TO 2                                                          LGU  CS
    1 KOUNT(J)=KOUNT(J)+1                                              LGU  CS
    2 CONTINUE                                                         LGU  CS
    5 KOUNT(I)  =  KOUNI                                               LGU  CS
                                                                       LGU  CS
C***1                                                                  LGU  CS
                                                                       LGU  CS
                                                                       LGU  CS
C*****TO C***2 USES FINAL VALUES OF THE COUNTERS TO ORDER THE LIST     LGU  CS
      DO 4  J=1,N                                                      LGU  CS
      KNTJ = KOUNT(J)                                                  LGU  CS
    4 ISORT(KNTJ) = LIST(J)                                            LGU  CS
                                                                       LGU  CS
C***2                                                                  LGU  CS
                                                                       LGU  CS
                                                                       LGU  CS
      WRITE(IPRNT,101)                                                 LGU  CS
      WRITE (IPRNT,100) ISORT                                          LGU  CS
      CALL EXIT                                                        LGU  CS
      END                                                              LGU  CS

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    3020  PROGRAM     210

 END OF COMPILATION
```

```
                        L.G. UNDERHILL     ***  BASIC  INSERTION  SORT  ***

      INTEGER  LIST(1000),  SHIFT                                          LGU  IS
      DEFINE FILE  1 (1000,1,U,KREC)                                       LGU  IS
      DATA   IPRNT /5/                                                     LGU  IS
  100 FORMAT (' ',12I10)                                                   LGU  IS
  101 FORMAT(' ************************************************************') LGU  IS
      READ  (1'1)  LIST                                                    LGU  IS
      WRITE (IPRNT,101)                                                    LGU  IS
      DO  5   J=2,1000                                                     LGU  IS
      SHIFT  =  0                                                          LGU  IS
      LISTJ  =  LIST(J)                                                    LGU  IS
      J1    =J-1                                                           LGU  IS
      DO  1   K=1,J1                                                       LGU  IS
                                                                           LGU  IS
C*****WHILE  SHIFT  IS ZERO WE ARE STILL SEARCHING FOR POSITION OF NEW     LGU  IS
C*****ITEM.  IF  SHIFT  IS NON ZERO WE ARE IN SHIFTING PHASE               LGU  IS
      IF(SHIFT)                4, 3, 4                                     LGU  IS
                                                                           LGU  IS
C*****IF NEW ITEM IS LARGER, EXAMINE NEXT ITEM                             LGU  IS
C*****IF NEW ITEM IS SMALLER, START SHIFTING                               LGU  IS
    3 IF(LISTJ - LIST(K))      2, 1, 1                                     LGU  IS
                                                                           LGU  IS
C***** M  MARKS POSITION AT WHICH NEW ITEM IS TO BE INSERTED               LGU  IS
    2 M=K                                                                  LGU  IS
      SHIFT  =  1                                                          LGU  IS
                                                                           LGU  IS
C*****GO TO BOTTOM OF SORTED SUBLIST, START SHIFTING THERE AND WORK        LGU  IS
C*****BACKWARDS TO POSITION MARKER  M                                      LGU  IS
    4 MJK  =M+J-K                                                          LGU  IS
      LIST(MJK)=LIST(MJK-1)                                                LGU  IS
                                                                           LGU  IS
    1 CONTINUE                                                             LGU  IS
                                                                           LGU  IS
C*****IF AT THIS STAGE  SHIFT  IS STILL ZERO, NEW ITEM IS ALREADY IN       LGU  IS
C*****CORRECT POSITION                                                     LGU  IS
      IF  (SHIFT)              6, 5, 6                                     LGU  IS
    6 LIST(M)  =  LISTJ                                                    LGU  IS
                                                                           LGU  IS
    5 CONTINUE                                                             LGU  IS
                                                                           LGU  IS
      WRITE (IPRNT,101)                                                    LGU  IS
      WRITE(IPRNT,100)  LIST                                               LGU  IS
      CALL EXIT                                                            LGU  IS
      END                                                                  LGU  IS

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    1020  PROGRAM     192

 END OF COMPILATION

// XEQ          1                                                         LGU  IS

*FILES(1,RNDMO)                                                          LGU  IS

************************************************************
************************************************************
```

```
                    L.G. UNDERHILL   ***   BINARY INSERTION SORT   ***

      INTEGER LIST(1000),P,SAVE                                          LGU  BIS
      DEFINE FILE  1(1000,1,U,KREC)                                      LGU  BIS
      DATA  IPRNT/5/                                                     LGU  BIS
  100 FORMAT(' ',12I10)                                                  LGU  BIS
  101 FORMAT(' ************************************************************')LGU  BIS
      WRITE(IPRNT,101)                                                   LGU  BIS
      N  =  1000                                                         LGU  BIS
                                                                         LGU  BIS
C*****READ ITEMS TO BE SORTED                                            LGU  BIS
      READ(1'1)(LIST(J) ,J=1,N)                                          LGU  BIS
                                                                         LGU  BIS
                                                                         LGU  BIS
C-----THESE COMMENT DESCRIBE THE PROCEDURE BETWEEN C*****2 AND C*****3   LGU  BIS
                                                                         LGU  BIS
C***** L  IS HALVED IN EACH LOOP BETWEEN C*****2 AND C*****3             LGU  BIS
                                                                         LGU  BIS
C***** K  POINTS TO CELL IN THE SORTED SUBLIST WITH WHICH NEW ITEM IS     LGU  BIS
C*****BEING COMPARED                                                     LGU  BIS
                                                                         LGU  BIS
C*****THUS  L*P  (WHERE  P  IS  +1  OR  -1) IS THE FACTOR WHICH MUST     LGU  BIS
C*****BE ADDED TO  K  SO THAT IT POINTS TO WHERE THE NEXT COMPARISON      LGU  BIS
C*****WILL TAKE PLACE                                                    LGU  BIS
                                                                         LGU  BIS
C***** P  IS SET TO  +1  IF NEW ITEM HAS GREATER SORT KEY THAN  LIST(K)  LGU  BIS
C***** P  IS SET TO  -1  IF NEW ITEM HAS SMALLER SORT KEY THAN  LIST(K)  LGU  BIS
C*****IF NEW ITEM AND  LIST(K)  HAVE EQUAL KEYS, SET  P  TO  +1  AND     LGU  BIS
C*****LEAVE INNER LOOP                                                   LGU  BIS
                                                                         LGU  BIS
C*****WHEN  L  BECOMES EQUAL TO  1  WE HAVE DETERMINED BETWEEN WHICH     LGU  BIS
C*****PAIR OF ITEMS ON SORTED SUBLIST NEW ITEM IS TO BE INSERTED AND      LGU  BIS
C*****WE LEAVE INNER LOOP                                                LGU  BIS
                                                                         LGU  BIS
                                                                         LGU  BIS
C*****1                                                                  LGU  BIS
                                                                         LGU  BIS
      DO  1  J=2,N                                                       LGU  BIS
                                                                         LGU  BIS
C***** K ,  P  AND  L  ARE SET TO  0 ,  +1  AND  J  RESPECTIVELY AT THE  LGU  BIS
C*****BEGINNING OF EACH OUTER LOOP (C*****1 TO C*****4)                  LGU  BIS
      K  =  0                                                            LGU  BIS
      P=1                                                                LGU  BIS
      L = J                                                              LGU  BIS
                                                                         LGU  BIS
C***** SAVE  CONTAINS THE NEW ITEM TO BE INSERTED                        LGU  BIS
      SAVE  =  LIST(J)                                                   LGU  BIS
                                                                         LGU  BIS
                                                                         LGU  BIS
C*****2                                                                  LGU  BIS
                                                                         LGU  BIS
    3 L=(L+1)/2                                                          LGU  BIS
      K=K+L*P                                                            LGU  BIS
      IF  (K)                  5, 5,12                                   LGU  BIS
   12 IF  (K-J)               14, 7, 7                                   LGU  BIS
   14 IF (LIST(K)-SAVE)        5, 6, 7                                   LGU  BIS
    5 P=1                                                                LGU  BIS
      GO TO 4                                                            LGU  BIS
    7 P=-1                                                               LGU  BIS
    4 IF  (L-1)                2, 2, 3                                   LGU  BIS
```

```
                    L.G. UNDERHILL   ***   BINARY INSERTION SORT   ***

    6 P=1                                                                LGU  BIS
                                                                         LGU  BIS
C*****3                                                                  LGU  BIS
                                                                         LGU  BIS
                                                                         LGU  BIS
C******THIS FORMULA CALCULATES THE NUMBER OF ITEMS THAT REQUIRE SHIFTING LGU  BIS
C******IT IS A FUNCTION OF  J  THE CURRENT LENGTH OF THE SUBLIST, THE     LGU  BIS
C*****POSITION  K  AT WHICH THE LAST COMPARISON OCCURRED AND  P  WHICH   LGU  BIS
C*****MAKES THE ADJUSTMENT FOR PLACING THE NEW ITEM ABOVE OR BELOW       LGU  BIS
C***** LIST(K)                                                           LGU  BIS
                                                                         LGU  BIS
    2 MOVES=J-K-(P+1)/2                                                  LGU  BIS
                                                                         LGU  BIS
                                                                         LGU  BIS
C*****TREAT SPECIAL CASE WHEN NUMBER OF MOVES IS ZERO SEPARATELY         LGU  BIS
C*****OTHERWISE START SHIFTING AT TOP OF SORTED SUBLIST AND FINALLY      LGU  BIS
C*****MOVE THE NEW ITEM STORED IN  SAVE  INTO ITS CORRECT POSITION       LGU  BIS
                                                                         LGU  BIS
      IF (MOVES)              9, 9,10                                    LGU  BIS
    9 LIST(J)=SAVE                                                       LGU  BIS
      GO TO 1                                                            LGU  BIS
                                                                         LGU  BIS
   10 DO  8  M=1,MOVES                                                   LGU  BIS
      JM= J-M                                                            LGU  BIS
    8 LIST(JM+1)=LIST(JM)                                                LGU  BIS
      KP=K+(P+1)/2                                                       LGU  BIS
      LIST(KP)=SAVE                                                      LGU  BIS
    1 CONTINUE                                                           LGU  BIS
                                                                         LGU  BIS
C*****4                                                                  LGU  BIS
                                                                         LGU  BIS
                                                                         LGU  BIS
      WRITE(IPRNT,101)                                                   LGU  BIS
      WRITE(IPRNT,100) (LIST(J),J=1,N)                                   LGU  BIS
      CALL EXIT                                                          LGU  BIS
      END                                                                LGU  BIS

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    1024  PROGRAM     310

 END OF COMPILATION

// XEQ          1                                                       LGU  BIS

*FILES(1,RNDMO)                                                         LGU  BIS

************************************************************
************************************************************
        11        35        43        73        75        91       131       187       195
       353       355       385       403       435       483       523       563       579
       729       747       771       779       849       899       923       953       961
      1051      1099      1131      1145      1169      1171      1225      1227      1241
      1353      1403      1497      1505      1539      1547      1705      1707      1745
      1811      1819      1849      1851      1859      1889      2059      2129      2131
```

```
                    L.G. UNDERHILL * BINARY INSERTION SORT WITH RETARD/ADVANCE *

      INTEGER  LIST(1000),SORT(2000),ZIP,P                                  LES BIRA
      DEFINE FILE  1(1000,1,U,KREC)                                         LES BIRA
      DATA  IPRNT/5/                                                        LES BIRA
  100 FORMAT(' ',12I10)                                                     LES BIRA
  101 FORMAT(' **********************************************************')LES BIRA
      WRITE(IPRNT,101)                                                      LES BIRA
      N  =  1000                                                            LES BIRA
      READ(1'1) ( LIST(J),  J = 1,N)                                        LES BIRA
                                                                            LES BIRA
C*****PLACE FIRST ITEM OF ARRAY TO BE SORTED AT MIDDLE OF ARRAY  SORT       LES BIRA
      SORT(1000)=LIST(1)                                                    LES BIRA
                                                                            LES BIRA
C***** ITOP  AND  IBOT  MARK THE LIMITS OF SORTED SUBLIST IN ARRAY  SORTLES BIRA
      ITOP  =  1000                                                         LES BIRA
      IBOT   =  1000                                                        LES BIRA
                                                                            LES BIRA
      DO  1   J=2,N                                                         LES BIRA
      LISTJ =LIST(J)                                                        LES BIRA
                                                                            LES BIRA
C***** K ,  P  AND  L  ARE AS BINARY INSERTION SORT                         LES BIRA
                                                                            LES BIRA
      K=IBOT - 1                                                            LES BIRA
      P=1                                                                   LES BIRA
      ZIP  =  1                                                             LES BIRA
      ITEST  =  1                                                           LES BIRA
      L=J                                                                   LES BIRA
    4 IF  (L-1)                2, 2, 3                                    LES BIRA
    3 L = (L+1)/2                                                           LES BIRA
      K=K+L*P                                                               LES BIRA
      IF  (K - IBOT)           5,17,17                                    LES BIRA
   17 IF  (K - ITOP)          18,18, 7                                    LES BIRA
                                                                            LES BIRA
   18 IF (SORT(K)-LISTJ)       5, 6, 7                                    LES BIRA
    5 P=1                                                                   LES BIRA
      GO TO 12                                                              LES BIRA
    7 P=-1                                                                  LES BIRA
   12 IF  (ITEST)              4, 4,11                                    LES BIRA
   11 ITEST  =  0                                                           LES BIRA
C***** ZIP  DETERMINES WHETHER WE RETARD OR ADVANCE                         LES BIRA
C*****AFTER THE FIRST COMPARISON  ZIP  IS SET EQUAL TO THE CURRENT          LES BIRA
C*****VALUE OF  P                                                          LES BIRA
      ZIP = P                                                               LES BIRA
      GO TO 4                                                               LES BIRA
    6 P=1                                                                   LES BIRA
                                                                            LES BIRA
C*****ADJUSTS LIMITS OT SORTED ARRAY                                        LES BIRA
    2 IBOT  =  IBOT-(1-ZIP)/2                                               LES BIRA
      ITOP  =  ITOP+(1+ZIP)/2                                               LES BIRA
                                                                            LES BIRA
C*****COMPUTES NUMBER OF SHIFTS UP OR DOWN REQUIRED                         LES BIRA
      MOVES = (1-ZIP)*(K-IBOT-(1-P)/2)/2                                    LES BIRA
     1      + (ZIP+1)*(ITOP-K-(P+1)/2)/2                                    LES BIRA
                                                                            LES BIRA
C*****TREAT SPECIAL CASE OF NO MOVES (ITEM GOES AT LIMITS) SEPARATELY       LES BIRA
      IF  (MOVES)                9, 9,10                                    LES BIRA
                                                                            LES BIRA
C*****TREAT SHIFTS UP OR DOWN SEPARATELY                                    LES BIRA
   10 IF  (ZIP)                 14,15,15                                   LES BIRA
```

L.G. UNDERHILL * BINARY INSERTION SORT WITH RETARD/ADVANCE *

```
   14 DO 16    M=1,MOVES                                          LES BIRA
      MBOT  =  IBOT+M                                           LES BIRA
   16 SORT(MBOT-1)  =  SORT(MBOT)                               LES BIRA
      GO TO 9                                                   LES BIRA
   15 DO 8    M=1,MOVES                                         LES BIRA
      MTOP  =  ITOP-M                                           LES BIRA
    8 SORT(MTOP+1)  =  SORT(MTOP)                               LES BIRA
                                                                LES BIRA
C*****COMPUTES POSITION AT WHICH NEW ITEM MUST BE INSERTED      LES BIRA
    9 KP  =  K+(P+ZIP)/2                                        LES BIRA
    1 SORT(KP)  =  LISTJ                                        LES BIRA
                                                                LES BIRA
      WRITE(IPRNT,101)                                          LES BIRA
      WRITE(IPRNT,100)  (SORT(J),  J=IBOT,ITOP)                 LES BIRA
      CALL EXIT                                                 LES BIRA
      END                                                       LES BIRA

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    3038  PROGRAM     448

 END OF COMPILATION

// XEQ          1                                              LES BIRA

*FILES(1,RNDMO)                                                LES BIRA

************************************************************
************************************************************
        11        35        43        73        75        91       131       187       195
       353       355       385       403       435       483       523       563       579
       729       747       771       779       849       899       923       953       961
      1051      1099      1131      1145      1169      1171      1225      1227      1241
      1353      1403      1497      1505      1539      1547      1705      1707      1745
      1811      1819      1849      1851      1859      1889      2059      2129      2131
      2169      2227      2235      2315      2355      2419      2441      2505      2515
      2579      2625      2633      2659      2673      2675      2713      2731      2753
      2817      2827      2841      2921      2961      2971      3025      3041      3083
      3185      3225      3257      3313      3315      3339      3347      3353      3425
      3491      3497      3571      3601      3713      3737      3747      3755      3785
      3899      3913      3931      3961      4003      4017      4019      4033      4099
      4273      4305      4361      4529      4539      4571      4577      4585      4603
      4779      4899      4947      4961      4963      4985      5001      5065      5091
      5289      5331      5393      5425      5475      5561      5625      5633      5681
      5811      5889      5931      5937      5939      5979      6001      6003      6019
      6203      6259      6345      6363      6441      6473      6513      6529      6545
      6665      6683      6777      6779      6787      6803      6825      6851      6899
      6993      7105      7147      7155      7185      7267      7273      7305      7395
      7563      7571      7603      7625      7731      7763      7771      7779      7801
      7889      7931      7979      8041      8059      8075      8089      8131      8147
      8217      8233      8251      8331      8337      8339      8353      8355      8411
      8601      8603      8611      8617      8705      8729      8731      8793      8843
      8937      8977      8979      9049      9099      9113      9121      9225      9241
      9345      9347      9355      9395      9411      9457      9507      9523      9561
      9601      9611      9643      9699      9747      9763      9785      9795      9817
```

```
                    L.G. UNDERHILL  ** QUADRATIC SELECTION WITHOUT PRESORT  VERSION 3  **

      INTEGER  LIST(33,32),SORT(1000),PLACE,AUX,ITMIN(33)                    LES QS
      DEFINE FILE  1(1000,1,U,KREC)                                          LES QS
      DATA                PLACE/1/,IMP/0/,IPRNT/5/                           LES QS
      DATA     MAX/32767/                                                    LES QS
  100 FORMAT(' ',12I10)                                                      LES QS
  101 FORMAT(' ************************************************************')LES QS
C***** ITEMS  TOTAL NUMBER OF ELEMENTS TO BE SORTED                          LES QS
      ITEMS  =  1000                                                         LES QS
      ITEMS  =  800                                                          LES QS
      WRITE (IPRNT,101)                                                      LES QS
                                                                             LES QS
                                                                             LES QS
C*****DETERMINE NUMBER OF ITEMS IN EACH SUBLIST                              LES QS
C*****SUBLISTS HAVE UP TO TWO ELEMENTS LESS THAN THE ROUNDING OFF TO         LES QS
C*****THE NEXT INTEGER OF THE SQUARE ROOT OF  ITEMS                         LES QS
C***** ITMIN(J)  NUMBER OF ITEMS IN  J TH  SUBLIST                          LES QS
                                                                             LES QS
      LISTS = IFIX(  SQRT(FLOAT(ITEMS))+1.000001)                            LES QS
      IF ((LISTS-1)**2-ITEMS)      19,18,16                                  LES QS
   18 LISTS = LISTS-1                                                        LES QS
      LOVER = 0                                                              LES QS
      GO TO 10                                                               LES QS
   19 LOVER = LISTS**2-ITEMS                                                 LES QS
      IF (LOVER-LISTS)        10,10,11                                       LES QS
   11 IMP   = 1                                                              LES QS
   10 LIMP  = LISTS-IMP                                                      LES QS
      NOVER = LOVER-IMP*LISTS                                                LES QS
      DO  1   J=1,LISTS                                                      LES QS
    1 ITMIN(J) = LIMP                                                        LES QS
      IF  (NOVER)             16,21,20                                       LES QS
   20 LIMP  =  LIMP-1                                                        LGU QS
      DO 2    J=1,NOVER                                                      LES QS
      DO 2    J=1,NOVER                                                      LES QS
    2 ITMIN(J) = LIMP                                                        LGU QS
                                                                             LES QS
                                                                             LES QS
C*****FORM SUBLISTS                                                          LGU QS
                                                                             LES QS
                                                                             LES QS
   21 READ (1'1)(SORT(J),J=1,ITEMS)                                          LES QS
      IMP   =   1                                                            LES QS
      DO 23  J=1,LISTS                                                       LES QS
      ITEMJ=ITMIN(J)                                                         LES QS
      DO  22  K=1,ITEMJ                                                      LES QS
      LIST(J,K)=SORT(IMP)                                                    LES QS
   22 IMP  = IMP+1                                                           LES QS
   23 CONTINUE                                                               LES QS
                                                                             LES QS
                                                                             LES QS
C*****FORM AUXILIARY LIST - FIND MINIMAL ELEMENT IN EACH SUBLIST             LES QS
                                                                             LES QS
                                                                             LES QS
      AUX  =  LISTS+1                                                        LES QS
      ITMIN(AUX) = LISTS                                                     LES QS
      DO 5  K=1,LISTS                                                        LES QS
      GO TO 3                                                                LES QS
    8 LIST(AUX,K) = MIN                                                      LES QS
    5 LIST(K,LEAST)=MAX                                                      LES QS
```

L.G. UNDERHILL ** QUADRATIC SELECTION WITHOUT PRESORT VERSION 3 **

```
                                                                      LES QS
                                                                      LES QS
C*****FIND MINIMUM OF AUXILIARY LIST FOR OUTPUT                       LES QS
                                                                      LES QS
                                                                      LES QS
      IMP   = 1                                                       LES QS
   15 PLACE = 2                                                       LES QS
      K     = AUX                                                     LES QS
      GO TO 3                                                         LES QS
    9 IF (MIN-MAX)              17,16,16                              LES QS
   17 SORT(IMP) = MIN                                                 LES QS
      IMP = IMP+1                                                     LES QS
                                                                      LES QS
                                                                      LES QS
C*****REPLACE ITEM CHOSEN FROM AUXILIARY LIST WITH SMALLEST ITEM FROM  LES QS
C*****CORRESPONDING SUBLIST                                           LES QS
                                                                      LES QS
                                                                      LES QS
      K  =   LEAST                                                    LES QS
      PLACE =  3                                                      LES QS
      GO TO  3                                                        LES QS
   12 LIST(AUX,K) = MIN                                               LES QS
      LIST(K,LEAST) = MAX                                             LES QS
      GO TO 15                                                        LES QS
                                                                      LES QS
                                                                      LES QS
C*****ROUTINE TO FIND SMALLEST ITEM  -  STRAIGHT SELECTION            LES QS
                                                                      LES QS
                                                                      LES QS
    3 LEAST = 1                                                       LES QS
      MIN  =  LIST(K,1)                                               LES QS
      ITEMK = ITMIN(K)                                                LES QS
      DO  4  J=2 ,ITEMK                                               LES QS
      IF (LIST(K,J)-MIN)              6, 4, 4                         LES QS
    6 LEAST  =  J                                                     LES QS
      MIN = LIST(K,J)                                                 LES QS
    4 CONTINUE                                                        LES QS
      GO TO  (8,9,12),  PLACE                                         LES QS
                                                                      LES QS
                                                                      LES QS
   16 WRITE(IPRNT,101)                                                LES QS
      WRITE(IPRNT,100) SORT                                           LES QS
      CALL EXIT                                                       LES QS
      END                                                             LES QS

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES   2120  PROGRAM    510

 END OF COMPILATION

// XEQ          1                                                    LES QS

*FILES(1,RNDMO,1CEC)                                                  LES QS
```

L.G. UNDERHILL *** TOURNAMENT SORT ***

```
      INTEGER  LIST(2500),SORT(1000),TABLE(15)                          LGU  TS
      DATA IPRNT,M,MAX /5,2,32767/                                      LGU  TS
      DEFINE FILE 1(1000,1,U,KREC)                                      LGU  TS
  100 FORMAT(' ',12I10)                                                 LGU  TS
  101 FORMAT(' ************************************************************')LGU  TS
      WRITE (IPRNT,101)                                                 LGU  TS
                                                                        LGU  TS
                                                                        LGU  TS
                                                                        LGU  TS
                                                                        LGU  TS
C-----THESE COMMENTS ARE A GENERAL DESCRIPTION OF THE PROGRAM           LGU  TS
C-----BETWEEN C*****1 AND C*****2                                       LGU  TS
                                                                        LGU  TS
                                                                        LGU  TS
C***** TABLE(M)  STORES THE LOCATION OF THE START OF THE  M TH  SUBLIST LGU  TS
C*****IT IS ALWAYS EVEN                                                 LGU  TS
                                                                        LGU  TS
C***** J  POINTS TO POSITION WHERE CURRENT COMPARISON IS TAKING PLACE   LGU  TS
C*****IN  (M-1)TH  SUBLIST                                              LGU  TS
                                                                        LGU  TS
C***** JDASH  POINTS TO POSITION IN  M TH  SUBLIST WHERE THE WINNER OF  LGU  TS
C*****THIS COMPARISON IS STORED                                         LGU  TS
                                                                        LGU  TS
C***** J  INCREASES IN STEPS OF TWO.  IT IS ALWAYS AN EVEN NUMBER.  IT  LGU  TS
C*****ALWAYS POINTS TO THE FIRST OF THE TWO ITEMS BEING COMPARED        LGU  TS
                                                                        LGU  TS
C***** JDASH  IS INCREMENTED BY ONE                                     LGU  TS
                                                                        LGU  TS
C*****WHEN  J  BECOMES EQUAL TO  TABLE(M)  THE  (M-1) TH  SUBLIST IS    LGU  TS
C*****EXHAUSTED,  THE  M TH  SUBLIST IS COMPLETE, AND THE FIRST         LGU  TS
C*****EVEN-NUMBERED CELL AFTER  JDASH  BECOMES  TABLE(M+1)              LGU  TS
                                                                        LGU  TS
C*****NOTE THAT THE  M TH  SUBLIST IS APPROXIMATELY HALF THE LENGTH OF  LGU  TS
C*****THE  (M-1) TH  SUBLIST                                            LGU  TS
                                                                        LGU  TS
C*****NOTE THAT IN ALL SUBLISTS (OTHER THAN THE FIRST) THE ADDRESSES    LGU  TS
C*****ARE STORED, AND NOT THE KEYS THEMSELVES                           LGU  TS
                                                                        LGU  TS
C*****FIRST ITEM SET GREATER THAN MAXIMUM POSSIBLE SORT KEY             LGU  TS
      LIST(1) = MAX                                                     LGU  TS
                                                                        LGU  TS
      N  = 1000                                                         LGU  TS
      N1  =  N+1                                                        LGU  TS
      READ(1'1) (LIST(J),J=2,N1)                                        LGU  TS
      J=2                                                               LGU  TS
                                                                        LGU  TS
C*****INITIAL SETTING OF  JDASH  TO FIRST EVEN NUMBER AFTER  N          LGU  TS
      LIST(N+2)  =  MAX                                                 LGU  TS
      JDASH  = 2*((N+3)/2)                                              LGU  TS
                                                                        LGU  TS
                                                                        LGU  TS
C*****1                                                                 LGU  TS
                                                                        LGU  TS
    7 TABLE(M) = JDASH                                                  LGU  TS
                                                                        LGU  TS
C*****SINCE FIRST SUBLIST CONTAINS KEYS IT IS TREATED SEPARATELY        LGU  TS
   20 IF (M-2)                 8, 8, 9                                  LGU  TS
    8 KO = J                                                            LGU  TS
```

L.G. UNDERHILL *** TOURNAMENT SORT ***

```
      K1 = J+1                                                          LGU  TS
      GO TO 10                                                          LGU  TS
                                                                        LGU  TS
    9 KO = LIST(J)                                                      LGU  TS
      K1 = LIST(J+1)                                                    LGU  TS
                                                                        LGU  TS
   10 IF (LIST(KO)-LIST(K1))  1, 1, 2                                   LGU  TS
    1 LIST(JDASH) = KO                                                  LGU  TS
      GO TO 3                                                           LGU  TS
    2 LIST(JDASH) =K1                                                   LGU  TS
                                                                        LGU  TS
    3 J=J+2                                                             LGU  TS
      JDASH = JDASH + 1                                                 LGU  TS
                                                                        LGU  TS
      IF (J-TABLE(M))         20,5 ,5                                   LGU  TS
                                                                        LGU  TS
C*****THE SUBLISTS ARE COMPLETED WHEN THIS CONDITION IS MET             LGU  TS
    5 IF (JDASH-J-1)           6,6,4                                    LGU  TS
                                                                        LGU  TS
C*****IF  JDASH  IS ODD THESE INSTRUCTIONS MAKE  LIST(JDASH)  REFERENCE LGU  TS
C*****  LIST(1) , AND CONVERT  JDASH TO THE NEXT EVEN NUMBER            LGU  TS
C*****IF  JDASH  IS EVEN THESE INSTRUCTIONS HAVE NO EFFECT              LGU  TS
    4 LIST(JDASH)  =  1                                                 LGU  TS
      JDASH  =  2*((JDASH+1)/2)                                         LGU  TS
                                                                        LGU  TS
      M=M+1                                                             LGU  TS
      GO TO  7                                                          LGU  TS
                                                                        LGU  TS
                                                                        LGU  TS
C*****2                                                                 LGU  TS
                                                                        LGU  TS
                                                                        LGU  TS
    6 JDASH = JDASH - 1                                                 LGU  TS
                                                                        LGU  TS
      DO  11  K=1,N                                                     LGU  TS
                                                                        LGU  TS
C*****OUTPUT THE WINNING ITEM AND REPLACE IT WITH MAXIMUM SORT KEY       LGU  TS
      KO  =  LIST(JDASH)                                                LGU  TS
      SORT(K) = LIST(KO)                                                LGU  TS
      LIST(KO) = MAX                                                    LGU  TS
                                                                        LGU  TS
C*****FIND POSITION THIS ITEM OCCUPIED IN SECOND SUBLIST                LGU  TS
                                                                        LGU  TS
      MM =  2                                                           LGU  TS
      K1  =  (KO-2)/2                                                   LGU  TS
      JDASH = TABLE(MM)+K1                                              LGU  TS
      IF (KO-2*K1-2)          12,12,13                                  LGU  TS
   13 KO =KO-2                                                          LGU  TS
                                                                        LGU  TS
C*****REPLACE IT IN SECOND SUBLIST WITH ITEM IT WAS ORIGINALLY          LGU  TS
C*****COMPARED WITH                                                    LGU  TS
   12 LIST(JDASH) = KO+1                                                LGU  TS
                                                                        LGU  TS
C*****FIND POSITION WHICH MUST BE ADJUSTED IN THIRD AND SUBSEQUENT      LGU  TS
C*****SUBLISTS                                                         LGU  TS
                                                                        LGU  TS
   14 MM = MM+1                                                         LGU  TS
```

L.G. UNDERHILL *** TOURNAMENT SORT ***

```
      IF(M-MM+1)              11,11,15                                  LGU  TS
                                                                        LGU  TS
C***** KO  GIVES POSITION IN  (MM-1) TH  SUBLIST RELATIVE TO TABLE(MM-1)LGU  TS
C***** K1  GIVES POSITION IN  M TH  SUBLIST RELATIVE TO  TABLE(MM)      LGU  TS
                                                                        LGU  TS
   15 KO =K1                                                            LGU  TS
      K1 = KO/2                                                         LGU  TS
      IF (KO-2*K1)            16,16,17                                  LGU  TS
   17 KO = KO-1                                                         LGU  TS
                                                                        LGU  TS
C***** JDASH  NOW GIVES POSITION IN  (MM-1) TH  SUBLIST RELATIVE TO     LGU  TS
C*****START OF LIST                                                     LGU  TS
   16 JDASH = TABLE(MM-1)+KO                                            LGU  TS
                                                                        LGU  TS
C***** J  AND  J1  REFER TO LOCATIONS IN FIRST SUBLIST WHERE THE SORT   LGU  TS
C*****KEYS THEMSELVES ARE STORED                                        LGU  TS
      J = LIST(JDASH)                                                   LGU  TS
      J1  =  LIST(JDASH+1)                                              LGU  TS
                                                                        LGU  TS
C***** JDASH  NOW GIVES POSITION IN  MM TH SUBLIST RELATIVE TO START     LGU  TS
C*****OF LIST                                                           LGU  TS
      JDASH = TABLE(MM) + K1                                            LGU  TS
                                                                        LGU  TS
C*****COMPARISON TO SEE WHICH ITEM GOES FORWARD TO  MM TH SUBLIST        LGU  TS
      IF (LIST(J)-LIST(J1))  18,18,19                                   LGU  TS
   18 LIST(JDASH) = J                                                   LGU  TS
      GO TO 14                                                          LGU  TS
   19 LIST(JDASH) = J1                                                  LGU  TS
      GO TO 14                                                          LGU  TS
                                                                        LGU  TS
   11 CONTINUE                                                          LGU  TS
      WRITE (IPRNT,101)                                                 LGU  TS
      WRITE (IPRNT,100) SORT                                            LGU  TS
      CALL EXIT                                                         LGU  TS
      END                                                               LGU  TS

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    3540  PROGRAM     516

 END OF COMPILATION

// XEQ          1                                                       LGU  TS

*FILES(1,RNOMO)                                                         LGU  TS

************************************************************
************************************************************
        11         35         43         73         75         91        131        187        195
       353        355        385        403        435        483        523        563        579
       729        747        771        779        849        899        923        953        961
      1051       1099       1131       1145       1169       1171       1225       1227       1241
      1353       1403       1497       1505       1539       1547       1705       1707       1745
      1811       1819       1849       1851       1859       1889       2059       2129       2131
      2169       2227       2235       2315       2355       2419       2441       2505       2515
      2579       2625       2633       2659       2673       2675       2713       2731       2753
```

```
                    L.G. UNDERHILL   ***   NATURAL TWO-WAY MERGE   ***

      INTEGER LIST(4,1001) ,OUT1,OUT2                                    LGU N2WM
      DATA  ISTAR /32767/                                                LGU N2WM
      DATA   ITE /500/                                                   LGU N2WM
      DATA              NIP,OUT1,OUT2/3,1,2/,  IPRNT /5/                 LGU N2WM
      DEFINE FILE  1(1000,1,U,KREC)                                      LGU N2WM
  100 FORMAT (' ',12I10)                                                 LGU N2WM
  101 FORMAT(' ************************************************************')LGU N2WM
      WRITE (IPRNT,101)                                                  LGU N2WM
                                                                         LGU N2WM
C*****SPREADS DATA OVER TWO LISTS                                        LGU N2WM
      READ(1'1) ((LIST(I,J),J=1,ITE) ,I=1,2)                             LGU N2WM
                                                                         LGU N2WM
C*****PLACES END OF FILE MARK AT END OF LISTS                            LGU N2WM
      ITE = ITE + 1                                                      LGU N2WM
      DO 20 I=1,2                                                        LGU N2WM
   20 LIST(I,ITE) = ISTAR                                                LGU N2WM
                                                                         LGU N2WM
C***** IA  AND  IB  MARK POSITIONS IN THE LISTS  OUT1  AND  OUT2  FROM   LGU N2WM
C*****WHICH DATA IS BEING TAKEN                                          LGU N2WM
   18 IA = 1                                                             LGU N2WM
      IB= 1                                                              LGU N2WM
                                                                         LGU N2WM
C***** IX  MARKS THE POSITION IN THE LIST INTO WHICH DATA IS CURRENTLY    LGU N2WM
C*****BEING PLACED                                                       LGU N2WM
      IX = 1                                                             LGU N2WM
                                                                         LGU N2WM
C***** IY  MARKS POSITION IN ALTERNATE OUTPUT LIST                       LGU N2WM
      IY = 1                                                             LGU N2WM
                                                                         LGU N2WM
      IP = 1                                                             LGU N2WM
      IE = 1                                                             LGU N2WM
                                                                         LGU N2WM
C***** NIP  TELLS WHICH PAIR OF LISTS ARE CURRENTLY IN USE FOR OUTPUT     LGU N2WM
      IN = NIP                                                           LGU N2WM
                                                                         LGU N2WM
                                                                         LGU N2WM
C*****NO STEPDOWN  -  TESTS WHICH ITEM SHOULD BE OUTPUTTED               LGU N2WM
                                                                         LGU N2WM
    4 IF (LIST(OUT1,IA)-LIST(OUT2,IB))              1, 3, 3              LGU N2WM
    1 LIST (IN,IX)  =  LIST(OUT1,IA)                                     LGU N2WM
      IA = IA + 1                                                        LGU N2WM
      IX = IX + 1                                                        LGU N2WM
                                                                         LGU N2WM
                                                                         LGU N2WM
C*****TESTS FOR END OF LIST AND ROLLOUT                                  LGU N2WM
                                                                         LGU N2WM
      IF (LIST(OUT1,IA)-ISTAR)                   2,10, 2                 LGU N2WM
                                                                         LGU N2WM
C*****TESTS FOR SINGLE STEPDOWN                                          LGU N2WM
    2 IF (LIST(OUT1,IA) - LIST(OUT1,IA-1))      11, 4, 4                 LGU N2WM
                                                                         LGU N2WM
                                                                         LGU N2WM
    3 LIST(IN,IX) = LIST(OUT2,IB)                                        LGU N2WM
      IB = IB + 1                                                        LGU N2WM
      IX = IX + 1                                                        LGU N2WM
      IF (LIST(OUT2,IB) - ISTAR)                 5,12, 5                 LGU N2WM
    5 IF (LIST(OUT2,IB)- LIST(OUT2,IB-1))       13, 4, 4                 LGU N2WM
```

L.G. UNDERHILL *** NATURAL TWO-WAY MERGE ***

```
   13 LIST(IN,IX) = LIST(OUT1,IA)                                    LGU N2WM
      IA = IA + 1                                                    LGU N2WM
      IX = IX + 1                                                    LGU N2WM
      IF (LIST(OUT1,IA) - LIST(OUT1,IA-1))        14, 6, 6          LGU N2WM
    6 IF (LIST(OUT1,IA) - ISTAR)                  13,10,13          LGU N2WM
   10 IE = 2                                                         LGU N2WM
      GO TO 15                                                       LGU N2WM
   11 LIST(IN,IX) = LIST(OUT2,IB)                                    LGU N2WM
      IB = IB + 1                                                    LGU N2WM
      IX = IX + 1                                                    LGU N2WM
                                                                     LGU N2WM
                                                                     LGU N2WM
C*****TESTS FOR DOUBLE STEPDOWN                                      LGU N2WM
                                                                     LGU N. WM
      IF (LIST(OUT2,IB) - LIST(OUT2,IB-1))        14, 7, 7          LGU N2WM
    7 IF (LIST(OUT2,IB) - ISTAR)                  11,12,11          LGU N2WM
   12 IE = 3                                                         LGU N2WM
      GO TO 16                                                       LGU N2WM
                                                                     LGU N2WM
                                                                     LGU N2WM
C*****CHANGE TO OTHER OUTPUT LIST                                    LGU N2WM
                                                                     LGU N2WM
   14 IN = IN + IP                                                   LGU N2WM
      IP = -IP                                                       LGU N2WM
      ISAVE = IX                                                     LGU N2WM
      IX = IY                                                        LGU N2WM
      IY = ISAVE                                                     LGU N2WM
                                                                     LGU N2WM
C*****  IE  =  1   UNTIL AN END OF FILE MARK ENCOUNTERED             LGU N2WM
C*****  IE  =  2   ROLLOUT FROM LIST  OUT2                          LGU N2WM
C*****  IE  =  3   ROLLOUT FROM LIST  OUT1                          LGU N2WM
      GO TO (4,8,9),    IE                                         LGU N2WM
                                                                     LGU N2WM
                                                                     LGU N2WM
C*****ROLLOUT FROM LIST  OUT2                                        LGU N2WM
                                                                     LGU N2WM
    8 LIST(IN,IX)  =  LIST(OUT2,IB)                                  LGU N2WM
      IB = IB + 1                                                    LGU N2WM
      IX = IX + 1                                                    LGU N2WM
                                                                     LGU N2WM
C*****TEST FOR END OF LIST  OUT2                                     LGU N2WM
      IF (LIST(OUT2,IB) - ISTAR)                  15,17,15           LGU N2WM
                                                                     LGU N2WM
                                                                     LGU N2WM
C*****ASSIGN SORTED STRINGS TO ALTERNATE OUTPUT LISTS                LGU N2WM
   15 IF (LIST(OUT2,IB) - LIST(OUT2,IB-1))        14, 8,8           LGU N2WM
                                                                     LGU N2WM
    9 LIST(IN,IX) = LIST(OUT1,IA)                                    LGU N2WM
      IA = IA + 1                                                    LGU N2WM
      IX = IX + 1                                                    LGU N2WM
      IF (LIST(OUT1,IA) - ISTAR)                  16,17,16           LGU N2WM
   16 IF (LIST(OUT1,IA) -LIST(OUT1,IA-1))         14, 9, 9          LGU N2WM
                                                                     LGU N2WM
C*****PLACE END OF FILE MARKERS ON OUTPUT LISTS                      LGU N2WM
   17 LIST(IN,IX)  =  ISTAR                                          LGU N2WM
      IN  =  IN + IP                                                 LGU N2WM
      LIST(IN,IY)  =  ISTAR                                          LGU N2WM
                                                                     LGU N2WM
C*****SWOP INPUT LISTS FOR OUTPUT LISTS                              LGU N2WM
```

L.G. UNDERHILL *** NATURAL TWO-WAY MERGE ***

```
      ISAVE = OUT1                                                    LGU N2WM
      OUT1 = NIP                                                      LGU N2WM
      NIP = ISAVE                                                     LGU N2WM
      OUT2 = OUT1 + 1                                                 LGU N2WM
                                                                      LGU N2WM
C*****TEST IF ALL ITEMS ARE ON ONE OUTPUT LIST                        LGU N2WM
      IF (IX - 2*ITE + 1)                18,19,19                     LGU N2WM
                                                                      LGU N2WM
                                                                      LGU N2WM
   19 WRITE (IPRNT,101)                                               LGU N2WM
      WRITE (IPRNT,100) (LIST(OUT1,K),K=1,1000)                       LGU N2WM
      CALL EXIT                                                       LGU N2WM
      END                                                             LGU N2WM

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    4032  PROGRAM     738

 END OF COMPILATION

// XEQ          1                                                    LGU N2WM

*FILES(1,RNDMO)                                                      LGU N2WM

************************************************************
************************************************************
```

11	35	43	73	75	91	131	187	195
353	355	385	403	435	483	523	563	579
729	747	771	779	849	899	923	953	961
1051	1099	1131	1145	1169	1171	1225	1227	1241
1353	1403	1497	1505	1539	1547	1705	1707	1745
1811	1819	1849	1851	1859	1889	2059	2129	2131
2169	2227	2235	2315	2355	2419	2441	2505	2515.
2579	2625	2633	2659	2673	2675	2713	2731	2753
2817	2827	2841	2921	2961	2971	3025	3041	3083
3185	3225	3257	3313	3315	3339	3347	3353	3425
3491	3497	3571	3601	3713	3737	3747	3755	3785
3899	3913	3931	3961	4003	4017	4019	4033	4099
4273	4305	4361	4529	4539	4571	4577	4585	4603
4779	4899	4947	4961	4963	4985	5001	5065	5091
5289	5331	5393	5425	5475	5561	5625	5633	5681
5811	5889	5931	5937	5939	5979	6001	6003	6019
6203	6259	6345	6363	6441	6473	6513	6529	6545
6665	6683	6777	6779	6787	6803	6825	6851	6899
6993	7105	7147	7155	7185	7267	7273	7305	7395
7563	7571	7603	7625	7731	7763	7771	7779	7801
7889	7931	7979	8041	8059	8075	8089	8131	8147
8217	8233	8251	8331	8337	8339	8353	8355	8411
8601	8603	8611	8617	8705	8729	8731	8793	8843
8937	8977	8979	9049	9099	9113	9121	9225	9241
9345	9347	9355	9395	9411	9457	9507	9523	9561
9601	9611	9643	9699	9747	9763	9785	9795	9817
9865	9867	9889	9987	10025	10049	10083	10107	10123
10379	10409	10417	10483	10555	10577	10579	10603	10635
10673	10691	10739	10769	10891	10897	10929	10963	10977

L.G. UNDERHILL *** ADDRESS SORT ***

```
      INTEGER LIST(1000)                                                    LES ADS1
      INTEGER ADRES, PLACE(3200)                                            LES ADS1
      DEFINE FILE  2(1000,1 ,U,KREC)                                        LES ADS1
      DATA  PLACE/3200*-1/, IPRNT/5/ ,K/0/                                  LES ADS1
  101 FORMAT(' ************************************************************') LES ADS1
  102 FORMAT  (1X,12I10)                                                    LES ADS1
      WRITE(IPRNT,101)                                                      LES ADS1
      READ  (2'1)  LIST                                                     LES ADS1
      DO  1  J =1,1000                                                      LES ADS1
                                                                            LES ADS1
C*****ADDRESS CALCULATION ALGORITHM                                         LES ADS1
      ADRES  =  3000.*LIST(J)/32767. + 100                                  LES ADS1
      ITEM  =  LIST(J)                                                      LES ADS1
                                                                            LES ADS1
C*****ARRAY  PLACE  IS INITIALISED TO  -1                                   LES ADS1
C*****TESTS TO SEE IF  PLACE(ADRES)  IS OCCUPIED                            LES ADS1
      IF (PLACE(ADRES)+1)     2, 1, 2                                       LES ADS1
                                                                            LES ADS1
C*****REFER TO DESCRIPTION AND FLOWCHART                                    LES ADS1
    2 KOUNT=0                                                               LES ADS1
      IF (PLACE(ADRES)-ITEM)  4, 4, 3                                       LES ADS1
    4 ADRES=ADRES+1                                                         LES ADS1
      IF (PLACE(ADRES)+1)     5, 6, 5                                       LES ADS1
    5 IF (PLACE(ADRES)-ITEM)  4, 4, 8                                       LES ADS1
    8 KOUNT=KOUNT+1                                                         LES ADS1
      GO TO 4                                                               LES ADS1
    6 IF (KOUNT)              1, 1, 9                                       LES ADS1
    9 ADRES=ADRES-1                                                         LES ADS1
      PLACE(ADRES+1)=PLACE(ADRES)                                           LES ADS1
      KOUNT=KOUNT-1                                                         LES ADS1
      GO TO 6                                                               LES ADS1
    3 ADRES=ADRES-1                                                         LES ADS1
      IF (PLACE(ADRES)+1)    11,12,11                                       LES ADS1
   11 IF (PLACE(ADRES)-ITEM)  7, 3, 3                                       LES ADS1
    7 KOUNT=KOUNT+1                                                         LES ADS1
      GO TO 3                                                               LES ADS1
   12 IF (KOUNT)              1, 1,10                                       LES ADS1
   10 ADRES=ADRES+1                                                         LES ADS1
      PLACE(ADRES-1)=PLACE(ADRES)                                           LES ADS1
      KOUNT=KOUNT-1                                                         LES ADS1
      GO TO 12                                                              LES ADS1
    1 PLACE(ADRES)=ITEM                                                     LES ADS1
                                                                            LES ADS1
      DO  13  J = 1,3200                                                    LES ADS1
      IF (PLACE(J))          13,13,14                                       LES ADS1
   14 K=K+1                                                                 LES ADS1
      PLACE(K)=PLACE(J)                                                     LES ADS1
   13 CONTINUE                                                              LES ADS1
                                                                            LES ADS1
      WRITE(IPRNT,101)                                                      LES ADS1
      WRITE(IPRNT,102)   (PLACE(J),J=1,1000)                                LES ADS1
      CALL EXIT                                                             LES ADS1
      END                                                                   LES ADS1

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    4220  PROGRAM     368
```

L.G. UNDERHILL * ADDRESS SORT FOR DATA WITH CENTRAL TENDENCY *

```
      SUBROUTINE  ADSOR(TABLE,N,A,R)                                    LEZ ADSR
                                                                        LEZ ADSR
C*****ARRAY  TABLE  CONTAINS VALUES RELATED TO NORMAL DISTRIBUTION.      LEZ ADSR
C*****  N  NUMBER OF ITEMS                                              LEZ ADSR
C*****ARRAY  A  CONTAINS ITEMS TO BE SORTED                             LEZ ADSR
C*****ARRAY  R  RETURNS RANKINGS OF ITEMS IN ARRAY  A                   LEZ ADSR
                                                                        LEZ ADSR
      INTEGER  SAVE,R(1000),TABLE(61),ADRES,PLACE(3000),TABZP           LEZ ADSR
      REAL  LDEV,LLDEV, MEAN, NOR01, ITEM, A(1000)                      LEZ ADSR
      EQUIVALENCE (AMEAN,NOR01),(SAVE,ADRES)                            LEZ ADSR
                                                                        LEZ ADSR
C-----SECTION ONE - PRELIMINARY CALCULATIONS                            LEZ ADSR
                                                                        LEZ ADSR
C*****INITIALISE ARRAY  PLACE  TO ZERO                                  LEZ ADSR
      DO 45 J=1,3000                                                    LEZ ADSR
   45 PLACE(J)=0                                                        LEZ ADSR
                                                                        LEZ ADSR
C*****CALCULATE MEAN OF DATA TO BE SORTED                               LEZ ADSR
                                                                        LEZ ADSR
      MEAN = 0.0                                                        LEZ ADSR
      DO  71  I  =  1,N                                                 LEZ ADSR
   71 MEAN = MEAN + A(I)                                                LEZ ADSR
      MEAN = MEAN/FLOAT(N)                                              LEZ ADSR
                                                                        LEZ ADSR
C*****CALCULATE APPROXIMATION TO DEVIATIONS TO LEFT AND RIGHT OF THE     LEZ ADSR
C*****MEAN                                                              LEZ ADSR
                                                                        LEZ ADSR
      LDEV = 0.                                                         LEZ ADSR
      RDEV = 0.                                                         LEZ ADSR
      DO  72  I = 1,N                                                   LEZ ADSR
      ITEM  =  A(I)                                                     LEZ ADSR
      IF  (ITEM)              78,72,78                                  LEZ ADSR
   78 AMEAN  =  ITEM - MEAN                                             LEZ ADSR
      IF  (AMEAN)             73,72,74                                  LEZ ADSR
   73 LDEV=LDEV+AMEAN*AMEAN                                             LEZ ADSR
      GOTO  72                                                          LEZ ADSR
   74 RDEV=RDEV+AMEAN*AMEAN                                             LEZ ADSR
   72 CONTINUE                                                          LEZ ADSR
      RRDEV  =  SQRT(FLOAT(50*N)/RDEV)                                  LEZ ADSR
      LLDEV  =  SQRT(FLOAT(50*N)/LDEV)                                  LEZ ADSR
                                                                        LEZ ADSR
C-----SECTION TWO - ADDRESS CALCULATION AND DISTRIBUTION OF ITEM NUMBERSLEZ ADSR
C-----IN ARRAYS  PLACE  AND  R                                          LEZ ADSR
                                                                        LEZ ADSR
      NORTS  =  0                                                       LEZ ADSR
      NONEG  =  0                                                       LEZ ADSR
      DO 111 J=1,N                                                      LEZ ADSR
      ITEM  =  A(J)                                                     LEZ ADSR
      IF  (ITEM)              41,42,43                                  LEZ ADSR
                                                                        LEZ ADSR
C*****COUNT NUMBER OF NEGATIVE ITEMS.  DEAL WITH ZERO'S SEPARATELY      LEZ ADSR
C*****BY ENTERING THEIR ITEM NUMBERS IN ARRAY  R                        LEZ ADSR
                                                                        LEZ ADSR
   41 NONEG  =  NONEG + 1                                               LEZ ADSR
      GO TO 43                                                          LEZ ADSR
   42 NORTS  =  NORTS  +  1                                             LEZ ADSR
      R(NORTS)  =  J                                                    LEZ ADSR
```

```
      GO TO 111                                                          LEZ ADSR
                                                                         LEZ ADSR
C*****DETERMINE WHETHER ITEM LIES TO LEFT OR RIGHT OF MEAN               LEZ ADSR
                                                                         LEZ ADSR
   43 NOR01  =  ITEM-MEAN                                                LEZ ADSR
      IF  (NOR01)              75,76,76                                  LEZ ADSR
                                                                         LEZ ADSR
C*****NORMALIZE ITEM                                                     LEZ ADSR
                                                                         LEZ ADSR
   75 NOR01  =  NOR01  *  LLDEV + 31.                                    LEZ ADSR
      GO TO 77                                                           LEZ ADSR
   76 NOR01  =  NOR01  *  RRDEV + 31.                                    LEZ ADSR
                                                                         LEZ ADSR
C*****DETERMINE WHICH VALUE IN ARRAY TABLE TO LOOKUP.  NOR01 BECOMES     LEZ ADSR
C*****FRACTIONAL REMAINDER FOR INTERPOLATION PURPOSES.                   LEZ ADSR
                                                                         LEZ ADSR
   77 IZAP  =  IFIX(NOR01)                                               LEZ ADSR
      NOR01  =  NOR01 - IZAP                                             LEZ ADSR
                                                                         LEZ ADSR
C*****CHECK THAT THIS VALUE LIES BETWEEN 1 AND 60.  ADJUST REMAINDER.    LEZ ADSR
                                                                         LEZ ADSR
      IF (IZAP)                    18,18,19                              LEZ ADSR
   18 IZAP = 1                                                           LEZ ADSR
      NOR01   =  0.                                                      LEZ ADSR
   19 IF (IZAP - 60)   20,20,21                                          LEZ ADSR
   21 IZAP = 60                                                          LEZ ADSR
      NOR01 = 1                                                          LEZ ADSR
                                                                         LEZ ADSR
C*****CALCULATE ADDRESS.  ADDRESS LIES BETWEEN TABLE(IZAP) AND           LEZ ADSR
C*****TABLE(IZAP+1)                                                      LEZ ADSR
                                                                         LEZ ADSR
   20 TABZP = TABLE(IZAP)                                                LEZ ADSR
      ADRES  =  TABZP+IFIX(NOR01*FLOAT(TABLE(IZAP+1)-TABZP))             LEZ ADSR
                                                                         LEZ ADSR
C*****TEST IF THIS ADDRESS VALUE IS OCCUPIED.  IF EMPTY ENTER ITEM       LEZ ADSR
C*****NUMBER HERE.  IF OCCUPIED MOVE ITEM NUMBERS SO THAT NEW ITEM       LEZ ADSR
C*****NUMBER IS SQUEEZED IN AT THE RIGHT PLACE                           LEZ ADSR
                                                                         LEZ ADSR
      IF (PLACE(ADRES))        2, 1, 2                                   LEZ ADSR
    2 KOUNT=0                                                            LEZ ADSR
      L = PLACE(ADRES)                                                   LEZ ADSR
      IF  (A(L) - ITEM)              4,4,3                               LEZ ADSR
    4 ADRES=ADRES+1                                                      LEZ ADSR
      IF (PLACE(ADRES))        5, 6, 5                                   LEZ ADSR
    5 L = PLACE(ADRES)                                                   LEZ ADSR
      IF  (A(L) - ITEM)               4,4,8                              LEZ ADSR
    8 KOUNT=KOUNT+1                                                      LEZ ADSR
      GO TO 4                                                            LEZ ADSR
    6 IF (KOUNT)               1, 1, 9                                   LEZ ADSR
    9 ADRES=ADRES-1                                                      LEZ ADSR
      PLACE(ADRES+1)=PLACE(ADRES)                                        LEZ ADSR
      KOUNT=KOUNT-1                                                      LEZ ADSR
      GO TO 6                                                            LEZ ADSR
    3 ADRES=ADRES-1                                                      LEZ ADSR
      IF (PLACE(ADRES))       11,12,11                                   LEZ ADSR
   11 L = PLACE(ADRES)                                                   LEZ ADSR
      IF   (A(L)- ITEM)            7,3,3                                 LEZ ADSR
    7 KOUNT=KOUNT+1                                                      LEZ ADSR
```

```
                    L.G. UNDERHILL  *  ADDRESS SORT FOR DATA WITH CENTRAL TENDENCY  *

      GO TO 3                                                                    LEZ ADSR
   12 IF (KOUNT)              1, 1,10                                            LEZ ADSR
   10 ADRES=ADRES+1                                                              LEZ ADSR
      PLACE(ADRES-1)=PLACE(ADRES)                                                LEZ ADSR
      KOUNT=KOUNT-1                                                              LEZ ADSR
      GO TO 12                                                                   LEZ ADSR
    1 PLACE(ADRES) = J                                                           LEZ ADSR
  111 CONTINUE                                                                   LEZ ADSR
                                                                                 LEZ ADSR
C-----SECTION THREE - CALCULATE RANKINGS                                         LEZ ADSR
                                                                                 LEZ ADSR
C*****FORM AN ARRAY IN R WITH ITEM NUMBERS IN CORRECT ORDER AND WITH NO          LEZ ADSR
C*****UNOCCUPIED CELLS IN BETWEEN                                                LEZ ADSR
                                                                                 LEZ ADSR
      J=0                                                                        LEZ ADSR
      IF (NONEG)              37,46,37                                           LEZ ADSR
   37 IF (NORTS)              38,30,38                                           LEZ ADSR
   38 KK  =  NORTS+1                                                             LEZ ADSR
      K = NONEG + KK                                                             LEZ ADSR
                                                                                 LEZ ADSR
C*****SHIFT ITEM NUMBERS OF ZERO VALUED ITEMS IN ARRAY  R  SO THAT THERE         LEZ ADSR
C*****IS ROOM FOR THE ITEM NUMBERS OF NEGATIVE VALUED ITEMS                      LEZ ADSR
      DO 47  L=1,NORTS                                                           LEZ ADSR
      K  =  K-1                                                                  LEZ ADSR
      KK = KK-1                                                                  LEZ ADSR
   47 R(K)  =  R(KK)                                                             LEZ ADSR
                                                                                 LEZ ADSR
C*****MOVE INTO ARRAY  R  ITEM NUMBERS OF NON-ZERO ITEMS ON EITHER SIDE         LEZ ADSR
C*****OF SECTION CONTAINING THE ITEM NUMBERS OF ZERO ITEMS                       LEZ ADSR
   30 NI  =  NONEG                                                               LEZ ADSR
      K=0                                                                        LEZ ADSR
   16 J=J+1                                                                      LEZ ADSR
      IF  (PLACE(J))          16,16,14                                           LEZ ADSR
   14 K=K+1                                                                      LEZ ADSR
      R(K)  =  PLACE(J)                                                          LEZ ADSR
   13 IF  (K-NI)              16,22,22                                           LEZ ADSR
   22 IF  (NI -N)              46,44,46                                          LEZ ADSR
   46 NI  =  N                                                                   LEZ ADSR
      K  =  NONEG + NORTS                                                        LEZ ADSR
      GO TO 13                                                                   LEZ ADSR
   44 IZAP = 1                                                                   LEZ ADSR
                                                                                 LEZ ADSR
C*****SHIFT THIS ARRAY TO PLACE                                                  LEZ ADSR
                                                                                 LEZ ADSR
      DO  48  J=1,N                                                              LEZ ADSR
   48 PLACE(J)=R(J)                                                              LEZ ADSR
                                                                                 LEZ ADSR
C*****THIS SECTION PERFORMS RANKING                                              LEZ ADSR
                                                                                 LEZ ADSR
      PLACE(N+1)  =  N+1                                                         LEZ ADSR
      A(N+1)  =  .1E38                                                           LEZ ADSR
      M  =  PLACE(1)                                                             LEZ ADSR
      DO  40  K = 1,N                                                            LEZ ADSR
      L  =  M                                                                    LEZ ADSR
      M  =  PLACE(K+1)                                                           LEZ ADSR
                                                                                 LEZ ADSR
C*****TEST IF NEXT ITEM IS BIGGER. IF IT IS, CALCULATE RANKING.  IF IT           LEZ ADSR
```

```
                    L.G. UNDERHILL  *  ADDRESS SORT FOR DATA WITH CENTRAL TENDENCY  *
C*****IS NOT, WAIT UNTIL ONE IS AND CALCULATE AVERAGE RANKING          LEZ ADSR
                                                                      LEZ ADSR
      IF  (A(L)-A(M) )       33,34,34                                 LEZ ADSR
   34 GO TO  (35,40), IZAP                                           LEZ ADSR
   35 IZAP = 2                                                       LEZ ADSR
      SAVE  =  K                                                     LEZ ADSR
      GO TO 40                                                       LEZ ADSR
   33 GO TO (31,32), IZAP                                            LEZ ADSR
   31 IPOSN = PLACE(K)                                               LEZ ADSR
      R(IPOSN)  =  K                                                 LEZ ADSR
      GO TO 40                                                       LEZ ADSR
   32 IZAP  =  1                                                     LEZ ADSR
      AMEAN = (SAVE+K)*.5                                            LEZ ADSR
      DO 39 L=SAVE,K                                                 LEZ ADSR
      IPOSN  =  PLACE(L)                                             LEZ ADSR
   39 R(IPOSN)=  AMEAN                                               LEZ ADSR
   40 CONTINUE                                                       LEZ ADSR
      RETURN                                                         LEZ ADSR
      END                                                            LEZ ADSR

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR ADSOR
 COMMON       0  VARIABLES   3036  PROGRAM     898

 END OF COMPILATION
```

B I B L I O G R A P H Y

The publications which proved most useful to this project in its narrowest sense, that of computer sorting, are listed. Many other books and journal articles dealing with problems in library computerisation were studied. It would be inappropriate to list them here.

BORK, A.M. *Using the IBM 1130*. Addison Wesley, 1968.

FLORES, I. *Computer Sorting*. Prentice Hall, 1969.

FLORES, I. *Computer Organization*. Prentice Hall, 1969.

HUGHES, J.K. *Programming the IBM 1130*. John Wiley and Sons, 1969.

INTERNATIONAL BUSINESS MACHINES. *1130 Scientific Subroutine Package - Programmers Manual*.