INFORMATION SCIENCES **22**, 79–98 (1980)

A Modeling Approach to the Evaluation of Internal Sorting Methods

S. SITHARAMA IYENGAR**
and
DALE R. BARRETT*

Department of Computer Science, Jackson State University, Jackson, Mississippi 39217

Communicated by Arne Jensen

ABSTRACT

The purpose of this paper is to report a modeling approach to the evaluation of internal sorting methods. The technique used is a regression modeling technique and has been found to be a very fast statistical method for evaluation which relies on the performance of data collection from the system being evaluated. The parameters considered for evaluation are: (1) number of stages, (2) number of transfers, (3) number of records, (4) sort time, (5) number of comparisons. The empirical model has been developed for sorting time as a function of the number of stages, number of records, number of comparisons, and number of transfers. The correlation coefficient obtained during the process of modeling was an average of 0.96 and has been found statistically significant.

1. INTRODUCTION

There has been much work going on to find efficient methods of sorting, since it is an important part of many large business data-processing problems.

Knuth [1] gives a detailed description of searching and sorting, and the algorithms described by him are simple but authoritative. He notes that sorting can be classified generally into internal sorting, in which the records are kept in the computer's high-speed random-access memory, and external sorting, when there are more records than can be held in memory at once. Internal sorting allows more flexibility in the structuring and accessing of the data, while external sorting shows us how to live with rather stringent accessing categories. The six different sorting techniques for our purpose are insertion sort, shell sort, quick sort, bubble sort, and tree sort. We shall present a little description of each of these sorting methods (for details see Ref. [1]):

*Presently at TRW, Inc. Rodendo Beach, California.

**Presently at Louisiana State University, Baton Rouge, Louisiana 70803.

52 Vanderbilt Ave., New York, NY 10017 0020-0255/80/08079-20$01.75

Insertion sort. The items are considered one at a time, and each new item is inserted into the appropriate position relative to the previously sorted items. (This is the way many bridge players sort their hands, picking up one card at a time.)

Exchange sort. If two items are found to be out of order, they are interchanged. This process is repeated until no more exchanges are necessary.

Selection sort. First the smallest (or perhaps the largest) item is located, and it is somehow separated from the rest; then the next smallest (or the next largest) is selected, and so on.

Merge sort. Merging (or collating) means the combination of two or more ordered files into a single ordered file.

Distribution sort. Readers who are familiar with punched-card equipment are well aware of the efficient procedure used on card sorters, based on the digits of the keys; the same idea can be adapted to computer programming, and it is generally known as "radix sorting," "digital sorting," or "pocket sorting."

According to the above definitions the technique called in the previous report[1] "insertion sorting" is really a selection sort and will be so called in this report. The tree sort is considered a type of selection sort. The technique called selection sort previously is actually the shell-of-diminishing-increment sort. The shell sort procedure is a variation of the insertion technique. The bubble sort is considered one of the exchange procedures, as is the quick sort.

Algorithms for these five different procedures were obtained from D. E. Knuth and also from A. T. Berztiss [5]. Knuth suggests that the quick sort method could be enhanced if "subfiles of M of fewer elements are left unsorted until the very end of the procedure, then a single pass of straight insertion is used to produce the final ordering." The algorithm for straight insertion was therefore used as a separate procedure as well as a procedure to be used in conjunction with quick sort. The remainder of the paper is organized as follows: general description of the model and its design, description of the input data to the model, calibration of the model, discussion of the results, model predictions, and conclusions.

2. GENERAL DESCRIPTION OF THE MODEL AND ITS DESIGN

A regression model [6, 7] is considered as a fast statistical model of system performance which relies on the performance data collected from the system being evaluated. In view of the development of efficient algorithms described

[1] "A Note on Comparison of Internal Sorting Methods," an unpublished paper by S. Sitharama Iyengar and Wendall Ingram.

in Knuth's book [1], an empirical model to evaluate the sort time as a function of number of stages, number of comparisons, number of transfers, and number of records will enhance the process of evaluation of internal sorting methods. Before we go into the formulation of the model, we shall discuss the system parameters: the number of stages, number of comparisons, number of transfers, and number of records. The number of stages is how many times the sort method must cycle before completion. The storage ratio is the ratio of the number of storage locations to the number of elements to be sorted. The number of transfers is an indication of the model's activity.

The model we are proposing in our paper will be of the following form:

$$\theta = f(N_R, D_t, N_s, N_c), \tag{1}$$

where

θ = sort time in sec,

N_R = number of records,

D_t = number of transfers,

N_c = number of comparisons,

N_s = number of stages.

The formulation process of the model is explained in Sec. 3.2.

The above functional relationship can be reduced to the following form after a multiple regression analysis:

$$\theta = C_0 + C_1 N_R + C_2 D_t + C_3 N_s + C_4 N_c. \tag{2}$$

This general model for all types of sort will be obtained after performing multiple regression analysis with θ as a dependent variable and N_R, D_t, N_s, and N_c as independent variables. C_1, C_2, C_3, C_4 are regression coefficients, and C_0 is the intercept obtained after the multiple regression analysis. The model sorting-time equation can be expressed as a linear (log-log) function using the SPSS (Statistical Package for Social Science) program, which will be described in the next section of the paper. Then the equation (2) can be expressed in the following form:

$$\begin{aligned} \log \theta = C_0 + C_1 \log N_R + C_2 \log D_t \\ + C_3 \log N_s + C_4 \log N_c, \end{aligned} \tag{3}$$

or

$$\theta = e^{C_0 + C_1 \log N_R + C_2 \log D_t + C_3 \log N_s + C_4 \log N_c}. \quad (4)$$

This equation can be used to find the sort time for any type of sort provided we know the variables N_s, N_c, D_t, and N_R.

2.1. CALIBRATION OF THE MODEL

The following definition used by Gomma [4] for the calibration of the model can be used in our study.

Calibration is basically an iterative procedure whose purpose is to reduce the difference in behavior between the model and the real system by adjusting the parameters of the model [4]. In the process of the calibration of the model, both stepwise and multiple regression analysis was done for each sort method using the log form of the data from Table 6 (excluding sort time as input).

3. PROGRAM DESCRIPTION AND GENERATION OF DATA

The sorting techniques (straight insertion, straight selection, bubble, shell, tree, and quick) were each written as subroutines in FORTRAN to be run on the MUSIC* system. The arguments for all six subroutines are R and N: the records and the number of records to be sorted.

The subroutine was tested using the program PSORT and printed these numbers. The program then had these numbers sorted by the subroutine being tested and printed by the terminal.

The processing time required by each subroutine to sort 100, 200, 500, 1000, 2000, and 5000 records was determined with program COMSRT. The time was determined using the system subroutine TSTIME before and after calling on the tested subroutine. Unfortunately the system subroutine TSTIME reported time to only 0.01 seconds. This placed one limitation on the determination of the sorting time. The second limitation was the system limit of 180 seconds. Thus, the time for sorting 5000 records could not be obtained for the insertion, selection, and bubble sort subroutines. To obtain sorting statistics the program COMSRT was altered to GMSRT. For details ref. [3].

*McGill University Interactive Computing Operating System implemented on IBM 370/145 system.

TABLE 1

Determination of Optimum Value of M to be Used in the Quick-Sort Subroutine[a]

No. of Records	$M=1$	$M=3$	$M=5$	$M=10$
50	0.00	0.00	0.03	0.00
100	0.06	0.00	0.02	0.08
200	0.02	0.02	0.09	0.05
500	0.22	0.29	0.25	0.23
1000	0.55	0.54	0.54	0.50
2000	1.24	1.35	1.30	1.31
5000	3.36	3.41	3.41	3.17

[a]Average time to sort the number of records shown, in seconds. Each value shown is the average of three determinations.

3.1. DETERMINATION OF OPTIMUM M

The number of elements (M) to be left unsorted at the end of the quick-sort subroutine had to be determined before accumulating any sorting data. To determine the optimum value of M, the sorting time was determined with values of M set at 1, 3, 5, and 10 for array sizes varying from 50 to 5000 records. Results were obtained for three runs at each value of M.

The average values for the three runs at each value of M and array size are reported in Table 1. No advantage in sorting time is evident in Table 1 for any value of M. The value $M=1$ was used for all subsequent runs.

3.2. SORTING TIME AND SORTING STATISTICS

Three runs were made to determine the sorting time for each of the six sorting methods. The averages of the three runs are shown in Table 2.

Three runs were also made to determine the sorting statistics (processing time, number of stages, number of comparisons, and number of transfers) for each of the six sorting methods. The averages of the three runs are given in Tables 2 (sorting and processing time), 3 (number of stages), 4 (number of comparisons), and 5 (number of transfers). Wherever possible in these tables the results obtained by Iyengar and Ingram [2] are given for comparison.

It is evident in Table 2 that the sorting time is increased when sorting statistics are counted in the same run that sorting time is measured. These latter times should not be used as a measure of sorting time. It is also evident that the times reported by Iyengar and Ingram do not check with the results

TABLE 2

Sorting Times[a]

No. of records sorted	50	100	200	500	1000	2000	5000
Insertion sort							
Sorting time	.02	.12	.35	2.27	9.08	36.27	[b]
Sorting statistics	.03	.06	.48	2.93	12.00	47.32	[b]
Selection sort							
Sorting time	.02	.11	.41	2.81	11.32	45.11	[b]
Sorting statistics	.05	.16	.55	3.61	14.18	56.10	[b]
Iyengar & Ingram		.21			20.05		55.33
Bubble sort							
Sorting time	.02	.13	.50	3.48	13.72	54.48	[b]
Sorting statistics	.06	.20	.67	4.64	18.45	74.07	[b]
Iyengar & Ingram		.29			26.17		669.83
Shell Sort							
Sorting time	[c]	.09	.05	.36	.84	1.99	6.07
Sorting statistics	.01	.08	.13	.46	.94	2.15	6.82
Iyengar & Ingram		.05			1.01		7.39
Tree Sort							
Sorting time	.07			.43	.85	2.12	6.03
Sorting statistics	.03			.44	1.04	2.44	6.90
Iyengar & Ingram		.08			.93		5.47
Quick Sort							
Sorting time	.02	.11	.04	.17	.45	1.06	3.05
Sorting statistics	.01	.01	.11	.28	.51	1.36	3.72
Iyengar & Ingram		.04			.47		2.64

[a] Average time in seconds. Each value shown is an average of three determinations.
[b] Exceeds 180 sec.
[c] Less than 0.01 sec.

obtained in this study for the selection, bubble, and shell sort, and are about the same as those obtained for the tree sort, but lower than those obtained for the quick sort.

The averages given in Table 2 are graphed in Fig. 1. Excellent linear (log-log) results are evident for the selection, insertion, and bubble sorts, with practically the same slopes and intercepts for all of them. The graphs for shell and tree sorts are the same and different from the other procedures. The lowest times are evident for the quick sort. Linearity (log-log) is also evident for the shell, tree, and quick sorts, particularly for record sizes of 500 or greater. The poor resolution in sorting time (0.01 seconds) is probably responsible for the poor data obtained for array sizes of 200 and less.

TABLE 3

Number of Stages[a]

No. of records sorted	50	100	200	500	1000	2000	5000
Insertion sort[b]	49	99	199	499	999	1999	—
Selection sort							
This study[b]	49	99	199	499	999	1999	—
Iyengar & Ingram[b]		99			999		4999
Bubble sort							
This study	38	90	187	481	962	1947	—
Iyengar & Ingram		99			999		4999
Shell sort							
This study[c]	5	6	7	8	9	10	12
Iyengar & Ingram		480			7987		51822
Tree sort							
This study[d]	73	148	298	748	1498	2998	7498
Iyengar & Ingram		149			1499		7499
Quick sort							
This study	32	65	134	33		318	3273
Iyengar & Ingram		33					1863

[a]Average of three determinations.
[b]Stages = (number of records) − 1; variance = 0.
[c]Stages = $\log_2$(number of records); variance = 0.
[d]Stages = 1.498(number of records); variance = 0.

TABLE 4

Number of Comparisons[a]

No. of records sorted	50	100	200	500	1000	2000	5000
Insertion sort	686	2370	10389	63585	249776	991621	—
Selection sort							
This study	1225	4950	19900	124750	499500	1999000	—
Iyengar & Ingram		4950			499500		12497500
Bubble sort							
This study	1106	4648	19696	124221	497209	1993960	—
Iyengar & Ingram		4922			499409		12493759
Shell sort							
This study	299	783	1948	6268	14670	33464	104436
Iyengar & Ingram		734			14895		110739
Tree sort							
This study	413	1033	2448	7436	16834	37690	107678
Iyengar & Ingram		1010			16842		107598
Quick sort							
This sort	299	762	1801	5465	11661	27236	74937
Iyengar & Ingram		746			12666		71911

[a]Average of three determinations. Comparisons = $N(N-1)/2$; variance = 0.

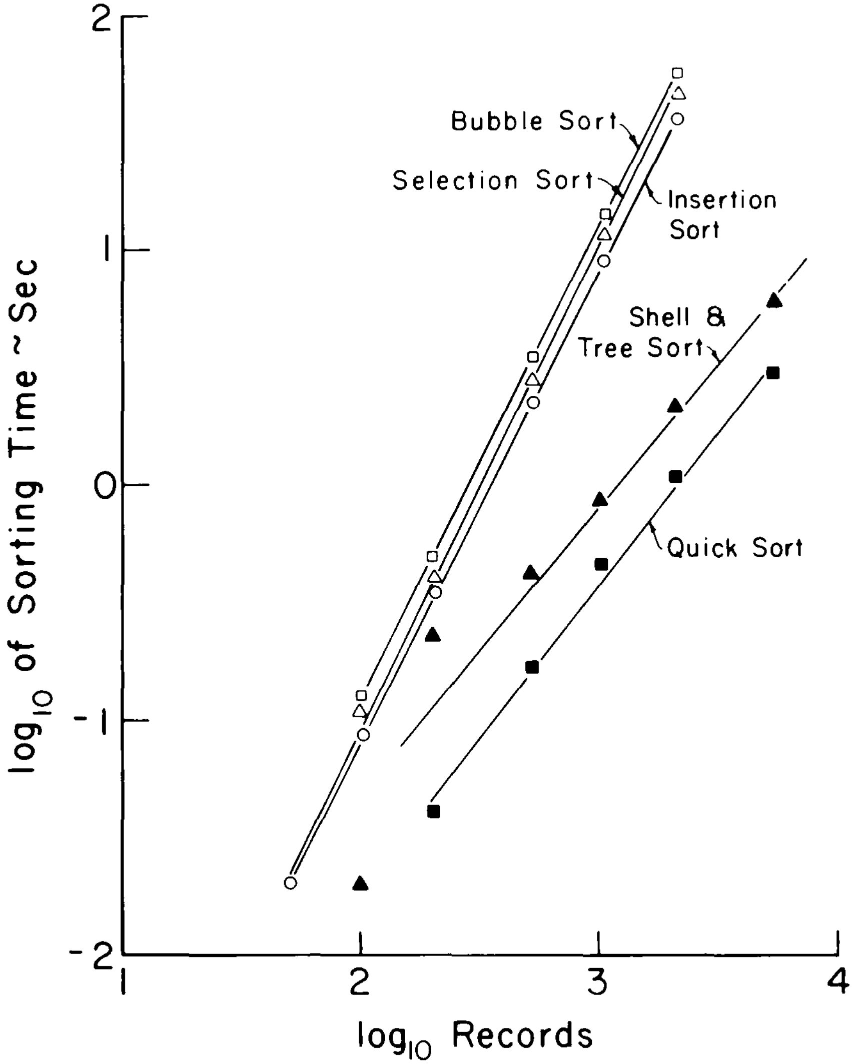

Fig. 1. Sorting time.

TABLE 5

Number of Transfers[a]

No. of records sorted	50	100	200	500	1000	2000	5000
Insertion sort	690	2374	10395	63591	249782	991627	—
Selection sort							
This study	182	423	984	2893	6484	14509	—
Iyengar & Ingram		297			2997		14997
Bubble sort							
This study	586	2393	9785	61689	251831	1005054	—
Iyengar & Ingram		7545			780258		18789441
Shell sort							
This study	13	352	899	3046	7251	16639	55256
Iyengar & Ingram		879			7518		184659
Tree sort							
This study	316	732	1655	4809	10570	23172	64579
Iyengar & Ingram		776			11077		67058
Quick sort							
This study	7	168	364	1099	2419	5284	14808
Iyengar & Ingram		411			5998		35707

[a]Average of three determinations.

TABLE 6

Sample Input Data[a]

N_R Var. 1 Records	θ Var. 2 Time	N_S Var. 3 Stages	N_c Var. 4 Comparisons	D_t Var. 5 Transfers
		Quick Sort		
50	0.02	32	299	7
100	0.11	65	762	168
200	0.04	134	1801	364
500	0.17	333	5465	1099
1000	0.45	659	11661	2419
2000	1.06	1318	27236	5284
5000	3.05	3273	74937	14808
		Insertion sort		
50	0.02	49	686	690
100	0.12	99	2370	2374
200	0.35	199	10389	10395
500	2.27	499	63585	63591
1000	9.08	999	249778	249782
2000	36.27	1999	991621	991627

TABLE 6 *(Continued)*

N_R Var. 1 Records	θ Var. 2 Time	N_S Var. 3 Stages	N_c Var. 4 Comparisons	D_t Var. 5 Transfers
		Selection sort		
50	0.02	49	1225	182
100	0.11	99	4950	423
200	0.41	199	19900	984
500	2.81	499	124750	2893
1000	11.32	999	499500	6484
2000	45.11	1999	1999000	14509
		Bubble sort		
50	0.02	38	1106	586
100	0.13	90	4648	2393
200	0.50	187	19696	9785
500	3.48	481	124221	61689
1000	13.72	962	497209	251831
2000	54.48	1947	1993960	1005054
		Tree sort		
50	0.07	73	413	316
100	0.02	148	1033	732
200	0.22	298	2448	1655
500	0.43	748	7436	4809
1000	0.85	1498	16843	10570
2000	2.12	2998	37690	23172
5000	6.03	7498	107678	64579
		Shell sort		
50	0.00	5	299	13
100	0.09	6	783	352
200	0.05	7	1948	899
500	0.36	8	6268	3046
1000	0.84	9	14670	7251
2000	1.99	10	33464	16639
5000	6.07	12	104436	55256

[a] Each subroutine was tested using the program PSORT. PSORT generated 25 uniformly distributed random numbers with subroutine RANDU and printed these numbers (uniform distribution and normal distribution).

4. INPUT DATA TO THE MODEL

This section describes how the data generated for all the parameters in Sec. 3 are used as the input data to the model. Table 6 shows the average value of three runs for each sorting method and the following parameters:

(1) Number of records sorted
(2) Sort time (in seconds)
(3) Number of stages
(4) Number of comparisons
(5) Number of data transfers

4.1. REGRESSION SUBMODEL TECHNIQUES

As described in the previous sections, it has been assumed that the amount of time required to sort a number of records can be expressed as a linear (log-log) function of the number of records, number of data transfers, number of stages, and number of comparisons. If true, this means that the sort time may be predicted by a linear regression submodel which is assumed to be of the form

$$\log \theta = C_0 + C_1 \log N_R + C_2 \log D_t + C_3 \log N_s + C_4 \log N_c, \tag{5}$$

where θ represents the sort time, N_R the number of records, N_s the number of stages, D_t the number of data transfers, N_c the number of comparisons, C_0 the intercept value, and C_1, C_2, and C_3 the regression coefficients of the parameters.

The linear regression analysis was done, and the regression submodel constructed, by using a standardized program developed for the Statistical Package for the Social Sciences (SPSS) language. SPSS is a packaged program specifically designed to compute those statistics typically used by social scientists. It was designed and developed in the late 1960s and is now one of the most widely used statistical packages.

The particular SPSS statistical subprogram used in this case was the subprogram REGRESSION, which can compute both step wise and multiple regressions. In the process of the calibration of the model, both stepwise and multiple regression analysis was done for each sort method using the log form of the data from Table 6 as input. The resulting submodels are shown in Tables 7–12 and figs. 2–4. Each submodel gives a very good fit to the data: the proportion of the variation explained (R^2) for each is 0.94 or better.

TABLE 7

Regression Submodel of Sort Time (Insertion Sort)[a]

N_R	C_1	−1.69486
	S.E.	27.60607
	F-value	0.004
	R	0.99882
D_t	C_2	27.08349
	S.E.	76.26212
	F-value	0.126
	R	0.99789
N_S	C_3	9.91640
	S.E.	28.32105
	F-value	0.123
	R	0.99887
N_c	C_4	−30.22620
	S.E.	76.65120
	F-value	0.155
	R	0.99791
	C_0	−6.73479
	R^2	0.99999
	S	0.00667
	F	41710.01041

θ	Sort time (sec)
C_0	Intercept
N_R	Number of records
D_t	Numer of data transfers
N_s	Number of stages
N_c	Number of comparisons
C_1, C_2, C_3, C_4	Regression coefficients
S.E.	Standard error of regression coefficients
F-value	Statistic significance of regression coefficient
R	Correlation coefficients
R^2	Proportion of variation explained by model
S	Standard error of model
F	Statistic for significance of regression equation

[a] $\log \theta = C_0 + C_1 \log N_R + C_2 \log D_t + C_3 \log N_s + C_4 \log N_c$.

TABLE 8

Regression Submodel of Sort Time (Selection Sort)[a]

N_R	C_1	−31130.32718
	S.E.	3571.08551
	F-value	75.992
	R	0.99952
D_t	C_2	−10.26504
	S.E.	0.35539
	F-value	834.287
	R	0.99966
N_S	C_3	−30995.76956
	S.E.	3570.16102
	F-value	75.375
	R	0.99957
N_c	C_4	31069.97930
	S.E.	3570.66436
	F-value	75.715
	R	0.99955
	C_0	9351.6018
	R^2	1.00
	S	0.00175
	F	655559.78950

[a] $\log \theta = C_0 + C_1 \log N_R + C_2 \log D_t + C_3 \log N_s + C_4 \log N_c$. The symbols have the same meaning as in Table 7.

TABLE 9

Regression Submodel of Sort Time (Bubble sort)[a]

N_R	C_1	1.84957
	S.E.	6.07267
	F-value	0.093
	R	0.99881
D_t	C_2	1.15970
	S.E.	3.73141
	F-value	0.097
	R	0.99888
N_S	C_3	3.72698
	S.E.	0.58052
	F-value	41.218
	R	0.99987
N_c	C_4	−2.96168
	S.E.	1.55185
	F-value	3.642
	R	0.99907
	C_0	−4.92527
	R^2	0.99997
	S	0.0153
	F	8879.05616
	N	

[a] $\log \theta = C_0 + C_1 \log N_R + C_2 \log D_t + C_3 \log N_s + C_4 \log N_c$. The symbols have the same meaning as in Table 7.

TABLE 10

Regression Submodel of Sort Time (Tree Sort)[a]

N_R	C_1	−403.50562
	S.E.	907.93983
	F-value	0.198
	R	0.94835
D_t	C_2	239.28564
	S.E.	312.52822
	F-value	0.586
	R	0.94698
N_S	C_3	332.73695
	S.E.	877.12420
	F-value	0.144
	R	0.94797
N_c	C_4	−170.98246
	S.E.	256.45680
	F-value	0.445
	R	0.94553
	C_0	−86.55856
	R^2	0.94014
	S	0.36098
	F	7.85257
	N	

[a] $\log \theta = C_0 + C_1 \log N_R + C_2 \log D_t + C_3 \log N_s + C_4 \log N_c$. The symbols have the same meaning as in Table 7.

TABLE 11

Regression Submodel of Sort Time (Shell Sort)[a]

N_R	C_1	9.94651
	S.E.	7.86484
	F-value	1.599
	R	0.67603
D_t	C_2	−0.65336
	S.E.	0.54089
	F-value	1.459
	R	0.46517
N_S	C_3	−24.20037
	S.E.	13.31877
	F-value	3.302
	R	0.61122
N_c	C_4	−3.00225
	S.E.	6.93346
	F-value	0.187
	R	0.66075
	C_0	8.17475
	R^2	0.98160
	S	0.17257
	F	26.67267

[a] $\log \theta = C_0 + C_1 \log N_R + C_2 \log D_t + C_3 \log N_s + C_4 \log N_c$. The symbols have the same meaning as in Table 7.

TABLE 12

Regression Submodel of Sort Time (Quick Sort)[a]

N_R	C_1	56.41420
	S.E.	2.80966
	F-value	403.151
	R	0.95586
D_t	C_2	0.61355
	S.E.	0.06584
	F-value	86.831
	R	0.92273
N_S	C_3	−61.11769
	S.E.	3.77782
	F-value	261.729
	R	0.95421
N_c	C_4	4.22348
	S.E.	1.25641
	F-value	11.3
	R	0.95310
	C_0	−16.52790
	R^2	0.99962
	S	0.02602
	F	1329.97925
	N	

[a] $\log \theta = C_0 + C_1 \log N_R + C_2 \log D_t + C_3 \log N_s + C_4 \log N_c$.

4.2. MODEL PREDICTION

In order to examine the difference between the real system and the models, a comparison was done between the predicted sort time using the model equations and the observed sort time from Table 2. The results of these comparisons can be seen in Table 13. As the proportion of the variation explained by the model (R^2) in each case was very good, we expected to find a close correlation between the actual and predicted times, and this was in fact the case. As can be seen from Table 13, the percentage differences between the actual times from Table 2 and the predicted times are quite small unless they represent a small number (less than 500) of records; and although the correlation increases with the number of records sorted, it is still quite good even at larger numbers of records. For instance, the percentage difference between the predicted time using the quick-sort regression submodel (Table 12) and the actual sort time is 0% at 50 records and increases only to 3.6% for 5000 records. Better results than those obtained for the shell sort, insertion sort, and

TABLE 13

Verification of Model

Type of sort	Number of records	Actual sort time (sec)	Predicted sort time (sec)
Insertion	1000	9.18	9.08
	100	0.12	0.12
Selection	50	0.02	0.02
	200	0.415	0.41
	1000	10.74	11.32
	2000	42.6	45.11
Bubble	50	0.02	0.02
	1000	13.61	13.72
Tree	50	0.06	0.07
	500	0.598	0.43
	2000	2.22	2.12
Quick	50	0.02	0.02
	500	0.165	0.17
	5000	2.94	3.05
	100	0.108	0.11
Shell	50	0.0099	0.00
	500	0.31	0.36
	1000	0.78	0.84
	5000	4.91	6.07

selection sort would no doubt be possible but for the limitations described in Sec. 3 on the measurement of the sort time.

5. NOTE

Needless to say, the regression model presented herein is not complete. Possibly the most critical section is that dealing with model prediction. To our surprise, the model prediction was excellent. This is explained by the fact that the correlation coefficient obtained during the process of modeling was an average of 0.96 or better. It is possible that an empirical model such as the one developed in our paper would be useful in determining the type of sort to be used in data processing.

However, in the meantime we are considering the development of a hybrid model (regression and simulation model) for the classification and evaluation of internal sorting methods.

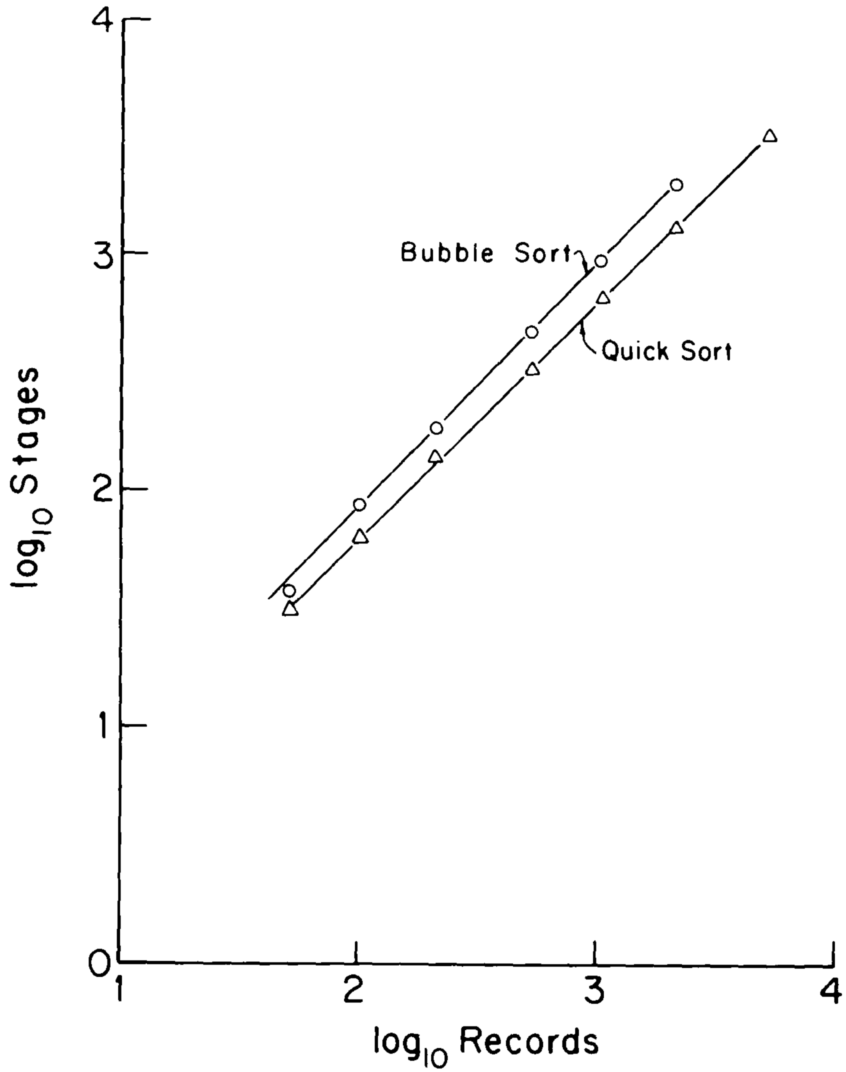

Fig. 2. Number of stages.

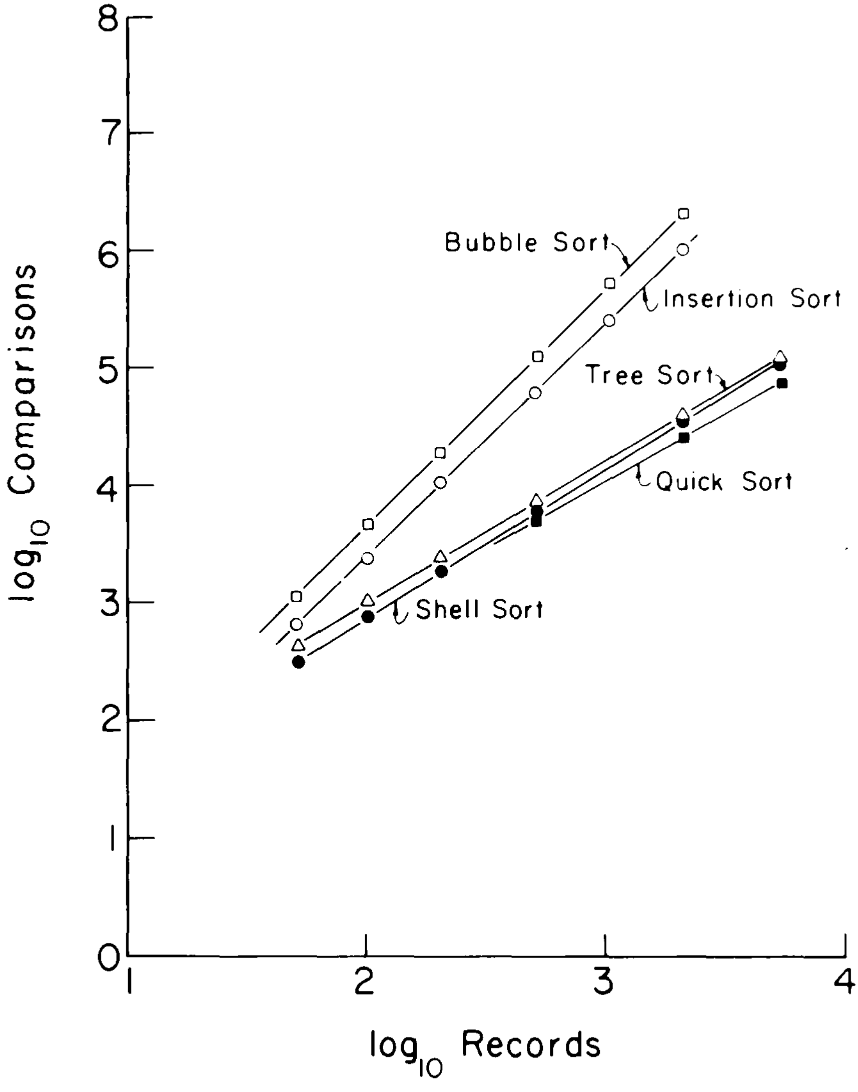

Fig. 3. Number of comparisons.

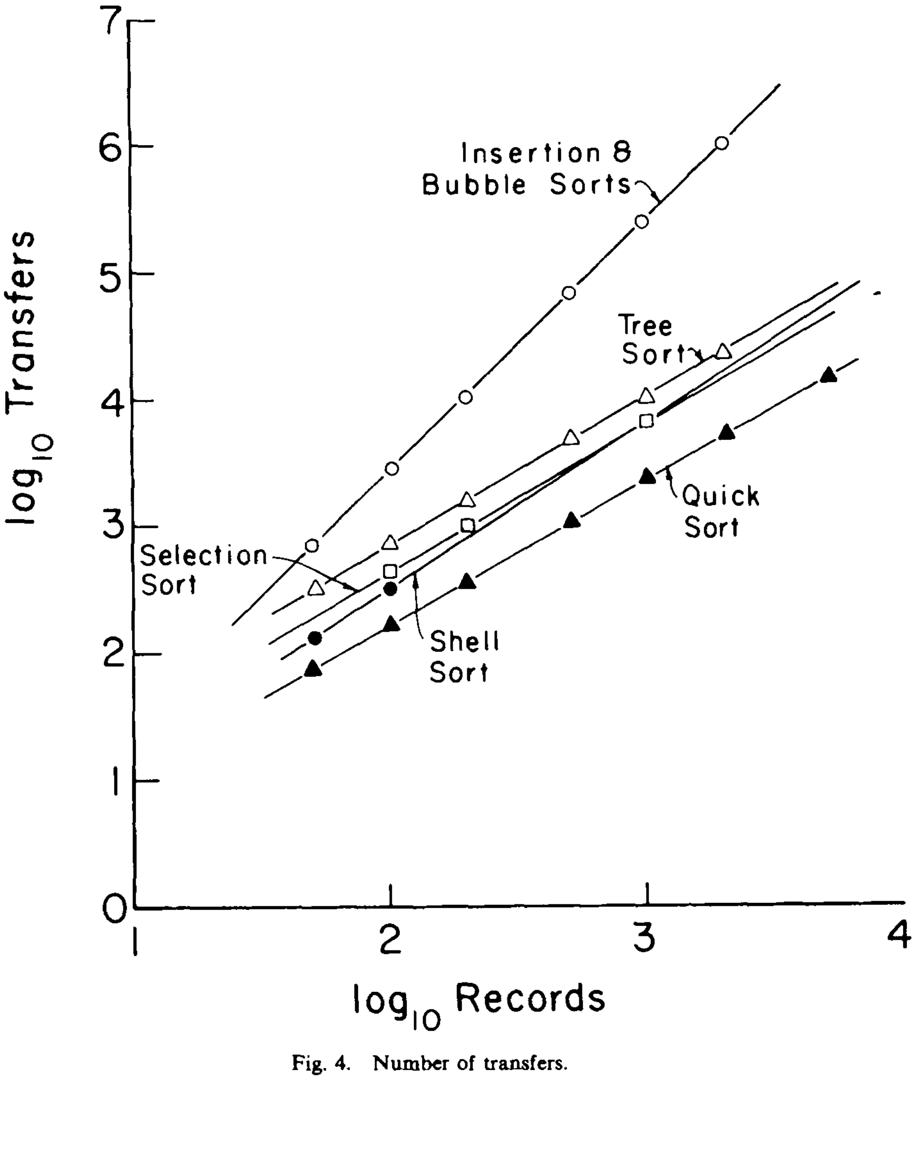

Fig. 4. Number of transfers.

6. CONCLUSIONS

This paper has described how a regression model for an internal sorting system may be constructed. By this means the advantages of regression techniques are exploited. Modeling by regression analysis provides a fast statistical method of modeling a system or a subsystem at a gross level.

A general model to compute the sorting time as a function of the number of stages, number of comparisons, number of transfers, and number of records has been presented in this paper, and a specific model for each of the six sorts (shell sort, insertion sort, tree sort, bubble sort, exchange sort, selection sort) to compute the sorting time has been presented. The correlation coefficient obtained during the process of modeling was an average of 0.96 and the predictions from the model are excellent.

Future endeavors to combine simulation and regression modeling would add greatly to the models' value.

The authors would like to thank Professor Iyengar's graduate students, especially Mr. L. Pepper and Mr. W. Ingram, for contributing input data for the model; Dr. N. R. V. Murthy for his expert guidance on SPSS *analysis of the data; and Dr. Jesse C. Lewis for his encouragement in this project.*

Preliminary funding for this project was provided by the senior author.

REFERENCES

1. D. E. Knuth, *The Art of Computer Programming*, Vol. 3., Addison Wesley Publishing Co, March 1975.
2. S. Iyengar and W. Ingram, A note on nonstandard binary radix method for comparison of internal sorting methods, submitted for publication.
3. S. Iyengar and L. Pepper, Program comparing internal sorting methods, unpublished report, Jackson State Univ., 1977.
4. H. Gomma, A simulation model of a virtual storage system, in *Proceedings of the 12th Annual Simulation Conference*, Tampa, Fla., Mar. 1979 pp. 273–303.
5. A. T. Berzliss, *Data Structure Theory and Practice.*, Academic Press Inc, New York 10003.
6. H. Beillner, The method of good balance procedure, in *7th Annual Symposium on Computer Science and Statistics*, Iowa State Univ., 1973.
7. H. Beillner and G. Waldbaum, Statistical methodology for calibrating a trace-driven simulator of a batch computer system, in *Statistical Computer Performance Evaluation* (W. Treiberger, Ed.), Academic, 1972.

Received September 1979.