simulated and built. The experience thus gained has indicated that: (1) It is possible to include simulation in a development cycle and as a result shorten that cycle, even in a changeable technology; (2) The efficiency of debugging hardware will improve because those testing the system have an opportunity to develop their troubleshooting techniques and to learn the computer system prior to the availability of hardware; (3) Logic problems that might otherwise remain undetected can be detected with three-value simulation; and (4) Future technologies will demand design verification simulators similar to the three-value simulator.

ACKNOWLEDGMENT

The authors wish to acknowledge development of the three-value simulation system by the following individuals: R. W. Butler, D. A. Haynes, J. S. Jephson, J. M. Krein, R. P. McQuarrie, M. T. Talbott, and R. J. Suhocki.

CITED REFERENCE AND FOOTNOTE

1. Correction of design defects.
2. E. B. Eichelberger, "Hazard detection in combinational and sequential switching circuits," *IBM Journal of Research and Development* **9**, No. 2, 90–99 (March 1965).

GENERAL REFERENCES

1. D. E. Muller, "Treatment of transition signals in electronic switching circuits by algebraic methods," *IRE Transactions* **EC–8**, 401 (1959).
2. M. Yoeli and S. Rinon, "Applications of ternary algebra to the study of static hazards," *Journal of the Association for Computing Machinery* **11**, No. 1, 84–97 (January 1964).

An ordering operator that leads to the development of an algorithm for internal sorting is described. An analysis of the algorithm is presented, together with a discussion of the number of comparisons necessary for sorting.

It is shown that the number of comparisons is close to the theoretically obtainable number. The sorting algorithm is a variant of the two-way merge.

Internal sorting with minimal comparing

by L. J. Woodrum

Internal sorting is such a common operation on today's digital computer systems that primitive ordering operations were included in the APL\360 Terminal System.[1-3] The operations, called *grade*, are represented by the symbols ⍋ and ⍒ for ascending and descending ordering, respectively, and they produce ordering permutations as results. An ordering permutation can then be used as a subscript on the original vector to order the data. In APL, subscripts are allowed to be vectors as well as scalars. Either zero-origin or one-origin indexing is allowed; in zero-origin indexing the first element of a vector **A** is $A[0]$, in one-origin indexing the first element is $A[1]$. All indexing used in this paper is zero-origin.

As an example of the use of a vector as a subscript, suppose **A** is the vector 5, 3, 9, 1, and **P** is the vector 3, 1, 0, 2. Then $A[P]$ is 1, 3, 5, 9; $A[0, 1]$ is the vector 5, 3; $A[2, 1, 0]$ is the vector 9, 3, 5; and $A[2, 3, 0, 0]$ is the vector 9, 1, 5, 5. When **P** is a permutation vector and is the same length as **A**, $A[P]$ is a rearrangement of the elements of **A**. When **P** is a permutation vector such that $A[P]$ arranges the elements of **A** in ascending or descending order, **P** is called an ordering permutation.

The APL grade operator ⍋ A produces the ascending ordering permutation for a vector **A**, and ⍒ A produces the descending ordering permutation. The grade operators are also defined for arbitrary multidimensional arrays **A**, and produce ordering permutations for the elements along a selected coordinate. If **A** is a matrix, then $P \leftarrow$ ⍋ A produces a matrix **P** such that $P[i;]$ is the ordering permutation for row i of **A**, i.e., $A[i; P[i;]]$ is row i of **A** arranged in ascending order. To operate on the columns of **A**, the operator is subscripted to indicate that the operation is to be performed along the first coordinate. If $P \leftarrow$ ⍋ $[c]A$, where $c = 0$ in zero-origin indexing, and $c = 1$ in one-origin indexing, then **P** is a matrix the same size as **A**, such that $A[P[; i]; i]$ is column i of **A** arranged in ascending order. For the multidimensional case, ⍋ $[c]A$ produces an array the same size as **A**, which contains ordering permutations along coordinate c.

In the selection of a sorting algorithm for the grade operators, the following requirements must be met:

- The algorithm should produce the result without using much more space than needed to contain the result.
- Extension to sorting along coordinates of multidimensional arrays must be easy.
- The algorithm should be efficient in its use of comparing, since comparing is a fairly expensive operation in the context of the interpreter.
- The original array (the input) must not be disturbed.
- The original order of equals must be preserved.

A variant of the conventional two-way merge, using a chaining technique for merging to avoid the use of two areas, satisfies these requirements. We develop the algorithm discussed here by determining the theoretical lower limit on the number of comparisons needed to sort n numbers, then finding the best achievable two-way merge sort, based on the average number of comparisons. The efficiency of this two-way merge is examined in the light of the theoretical limits on comparing in internal sorts in general. A mathematical model of comparing in two-way merge sorting is developed, and the algorithm is derived from this model. The algorithm so obtained is a recursive program, which is then expressed nonrecursively, using two stacks instead of the recursion. The one-dimensional case is considered first and then generalized to the multidimensional case.

Analysis of sorting algorithms

Three parameters are usually used to measure the performance of a sorting algorithm. They are the number of comparisons, exchanges, and working space required for the sort. For this application, it is desired that no more working space be required than whatever is necessary to hold the result—the ordering permutation vector **P**. An exchange is defined as the swapping, moving, or exchanging of two numbers in the data being sorted, or of two addresses or indices of numbers in that data. The number of exchanges required in an internal sort is usually closely related to the number of comparisons, and for that reason, it is valid to consider internal sorting algorithms based on the number of comparisons required. For certain internal sorting algorithms, the number of exchanges, or moves, required far exceeds the number of comparisons required, e.g., the binary insertion technique.[4] Such techniques are not desirable and are not considered.

comparisons

Given any internal sorting algorithm based on comparing, the sorting of a set of n numbers requires a number of comparisons, say c. This number c will take on various values, based on the particular set of n numbers. However, the smallest value that c can have is $n - 1$. For a given algorithm, c is a random variable that depends on the input or the set of numbers to be sorted. In analyzing an algorithm, several items are of interest: the smallest value c can assume, the largest value c can assume (the worst case), and the expected value of c, $E(c)$, which is the average number of comparisons to sort n numbers. Also of interest is the probability that c will exceed $E(c)$ by a large amount, i.e., that the algorithm will encounter one of its "bad" cases.

Of the entire possible set of internal sorting algorithms, there is at least one algorithm that has $E(c)$ less than or equal to the $E(c)$ for any other algorithm. For such an algorithm, $E(c)$ is greater than or equal to $\log_2(!n)$, that is, $\log_2(!n)$ is a lower limit on the average number of comparisons taken by any internal sorting algorithm (see Section 3 of the Appendix). An internal sort can thus be evaluated by seeing how close its $E(c)$ is to the theoretical minimum.

Of the entire set of possible sorting algorithms, there is also at least one where the maximum value that c can assume is less than or equal to the maximum value of c assumed for any other sorting algorithm. That is, there must be an algorithm whose worst case is less than the worst case for any other algorithm. Whatever the maximum value of c is for such an algorithm, it is greater than or equal to the ceiling of $\log_2(!n)$ where the ceiling, "$\lceil$," of x denotes the smallest integer that is not less than x. For at least some values of n, it is possible to always sort with no more than the ceiling of $\log_2(!n)$ comparisons.[5] It is interesting to note that the limit of the average comparison and the limit of the worst-case comparison are so close to each other that they differ by less than one for all n. The worst-case limit number of comparisons, the ceiling of $\log_2(!n)$, is listed in Table 1 for values of n up to 1000.

natural sequencing

Another criterion for evaluating sorting algorithms is whether or not the algorithm takes advantage of "natural sequencing" present in the data. To use this criterion in evaluating a sort, we must be able to measure the amount of "natural sequencing" present in a set of data. One way to measure the sequencing is to see if the data are correlated with the ordered set of data; the correlation coefficient for this can be computed in the usual way.[6] This method is not a very useful general measure of sequencing

Table 1 Theoretical lower limit of the number of comparisons required to sort n numbers

⌈2⍟!N

	0	1	2	3	4	5	6	7	8	9
0--	0	0	1	3	5	7	10	13	16	19
1--	22	26	29	33	37	41	45	49	53	57
2--	62	66	70	75	80	84	89	94	98	103
3--	108	113	118	123	128	133	139	144	149	154
4--	160	165	170	176	181	187	192	198	203	209
5--	215	220	226	232	238	243	249	255	261	267
6--	273	279	285	290	296	303	309	315	321	327
7--	333	339	345	351	358	364	370	376	383	389
8--	395	402	408	414	421	427	434	440	447	453
9--	459	466	473	479	486	492	499	505	512	519
10--	525	532	539	545	552	559	565	572	579	586
11--	592	599	606	613	620	627	633	640	647	654
12--	661	668	675	682	689	696	703	710	717	724
13--	731	738	745	752	759	766	773	780	787	794
14--	802	809	816	823	830	837	845	852	859	866
15--	873	881	888	895	902	910	917	924	932	939
16--	946	953	961	968	976	983	990	998	1005	1012
17--	1020	1027	1035	1042	1050	1057	1065	1072	1079	1087
18--	1094	1102	1109	1117	1124	1132	1140	1147	1155	1162
19--	1170	1177	1185	1193	1200	1208	1215	1223	1231	1238
20--	1246	1254	1261	1269	1277	1284	1292	1300	1307	1315
21--	1323	1330	1338	1346	1354	1361	1369	1377	1385	1392
22--	1400	1408	1416	1424	1431	1439	1447	1455	1463	1471
23--	1478	1486	1494	1502	1510	1518	1526	1533	1541	1549
24--	1557	1565	1573	1581	1589	1597	1605	1613	1621	1629
25--	1637	1645	1653	1661	1669	1676	1684	1693	1701	1709
26--	1717	1725	1733	1741	1749	1757	1765	1773	1781	1789
27--	1797	1805	1813	1821	1829	1838	1846	1854	1862	1870
28--	1878	1886	1894	1903	1911	1919	1927	1935	1943	1952
29--	1960	1968	1976	1984	1992	2001	2009	2017	2025	2034
30--	2042	2050	2058	2066	2075	2083	2091	2100	2108	2116
31--	2124	2133	2141	2149	2157	2166	2174	2182	2191	2199
32--	2207	2216	2224	2232	2241	2249	2257	2266	2274	2282
33--	2291	2299	2308	2316	2324	2333	2341	2349	2358	2366
34--	2375	2383	2392	2400	2408	2417	2425	2434	2442	2451
35--	2459	2467	2476	2484	2493	2501	2510	2518	2527	2535
36--	2544	2552	2561	2569	2578	2586	2595	2603	2612	2620
37--	2629	2637	2646	2655	2663	2672	2680	2689	2697	2706
38--	2714	2723	2732	2740	2749	2757	2766	2775	2783	2792
39--	2800	2809	2818	2826	2835	2843	2852	2861	2869	2878
40--	2887	2895	2904	2913	2921	2930	2939	2947	2956	2965
41--	2973	2982	2991	2999	3008	3017	3025	3034	3043	3052
42--	3060	3069	3078	3086	3095	3104	3113	3121	3130	3139
43--	3148	3156	3165	3174	3183	3191	3200	3209	3218	3226
44--	3235	3244	3253	3262	3270	3279	3288	3297	3306	3314
45--	3323	3332	3341	3350	3359	3367	3376	3385	3394	3403
46--	3412	3420	3429	3438	3447	3456	3465	3474	3482	3491
47--	3500	3509	3518	3527	3536	3545	3553	3562	3571	3580
48--	3589	3598	3607	3616	3625	3634	3643	3652	3660	3669
49--	3678	3687	3696	3705	3714	3723	3732	3741	3750	3759
50--	3768	3777	3786	3795	3804	3813	3822	3831	3840	3849
51--	3858	3867	3876	3885	3894	3903	3912	3921	3930	3939
52--	3948	3957	3966	3975	3984	3993	4002	4011	4020	4029
53--	4038	4047	4056	4065	4074	4083	4092	4102	4111	4120
54--	4129	4138	4147	4156	4165	4174	4183	4192	4201	4211
55--	4220	4229	4238	4247	4256	4265	4274	4283	4293	4302
56--	4311	4320	4329	4338	4347	4357	4366	4375	4384	4393
57--	4402	4411	4421	4430	4439	4448	4457	4466	4476	4485
58--	4494	4503	4512	4522	4531	4540	4549	4558	4568	4577
59--	4586	4595	4604	4614	4623	4632	4641	4650	4660	4669
60--	4678	4687	4697	4706	4715	4724	4734	4743	4752	4761
61--	4771	4780	4789	4798	4808	4817	4826	4835	4845	4854
62--	4863	4872	4882	4891	4900	4910	4919	4928	4937	4947
63--	4956	4965	4975	4984	4993	5003	5012	5021	5031	5040
64--	5049	5059	5068	5077	5087	5096	5105	5115	5124	5133
65--	5143	5152	5161	5171	5180	5189	5199	5208	5217	5227
66--	5236	5245	5255	5264	5274	5283	5292	5302	5311	5320
67--	5330	5339	5349	5358	5367	5377	5386	5396	5405	5414
68--	5424	5433	5443	5452	5462	5471	5480	5490	5499	5509
69--	5518	5528	5537	5546	5556	5565	5575	5584	5594	5603
70--	5613	5622	5631	5641	5650	5660	5669	5679	5688	5698
71--	5707	5717	5726	5736	5745	5755	5764	5773	5783	5792
72--	5802	5811	5821	5830	5840	5849	5859	5868	5878	5887
73--	5897	5907	5916	5926	5935	5945	5954	5964	5973	5983
74--	5992	6002	6011	6021	6030	6040	6049	6059	6069	6078
75--	6088	6097	6107	6116	6126	6135	6145	6155	6164	6174
76--	6183	6193	6202	6212	6222	6231	6241	6250	6260	6269
77--	6279	6289	6298	6308	6317	6327	6337	6346	6356	6365
78--	6375	6385	6394	6404	6413	6423	6433	6442	6452	6462
79--	6471	6481	6490	6500	6510	6519	6529	6539	6548	6558
80--	6568	6577	6587	6597	6606	6616	6625	6635	6645	6654
81--	6664	6674	6683	6693	6703	6712	6722	6732	6741	6751
82--	6761	6771	6780	6790	6800	6809	6819	6829	6838	6848
83--	6858	6867	6877	6887	6897	6906	6916	6926	6935	6945
84--	6955	6965	6974	6984	6994	7003	7013	7023	7033	7042
85--	7052	7062	7071	7081	7091	7101	7110	7120	7130	7140
86--	7149	7159	7169	7179	7188	7198	7208	7218	7227	7237
87--	7247	7257	7267	7276	7286	7296	7306	7315	7325	7335
88--	7345	7355	7364	7374	7384	7394	7403	7413	7423	7433
89--	7443	7452	7462	7472	7482	7492	7501	7511	7521	7531
90--	7541	7551	7560	7570	7580	7590	7600	7609	7619	7629
91--	7639	7649	7659	7668	7678	7688	7698	7708	7718	7727
92--	7737	7747	7757	7767	7777	7787	7796	7806	7816	7826
93--	7836	7846	7856	7865	7875	7885	7895	7905	7915	7925
94--	7935	7944	7954	7964	7974	7984	7994	8004	8014	8024
95--	8033	8043	8053	8063	8073	8083	8093	8103	8113	8123
96--	8132	8142	8152	8162	8172	8182	8192	8202	8212	8222
97--	8232	8241	8251	8261	8271	8281	8291	8301	8311	8321
98--	8331	8341	8351	8361	8371	8381	8391	8400	8410	8420
99--	8430	8440	8450	8460	8470	8480	8490	8500	8510	8520

(⌈2⍟!1000)=8530

because it depends on the particular values of the data, i.e., it is sensitive to the distribution from which the data were drawn.

To obtain a distribution-independent measure of order, create a set of indices by replacing each number with the index of its position in the ordered set. Then compute the sample correlation coefficient of this set of indices with the ordered set of indices, $0, 1, 2, \ldots, n - 1$. The procedure for computing the sample correlation coefficient for this special case is simpler than the general case and is given in the Appendix.

For data which are already in ascending sequence, this correlation coefficient is 1, for inverse sequences the coefficient is -1, and for "random" sequences this measure is zero. When sorting experts talk about "random" data, they mean that this correlation coefficient is zero, or that they assume it is zero.

To determine if a sorting algorithm takes advantage of natural sequencing in the data then means examining the number of comparisons taken, assuming that the correlation coefficient is some fixed value, or is in some given range. This kind of analysis of sorting algorithms has rarely, if ever, appeared in the literature, except for coefficients of -1, 0, and 1 corresponding to inversely ordered files, "random" files, and in-sequence files. In keeping with this tradition, the algorithm in this paper is evaluated for coefficients of -1, 0, and 1.

Analysis of two-way merging

worst-case number

Certain modifications permit the two-way merge to be especially suitable for the implementation of the APL\360 grade operations. In any internal sort by two-way merging, there is a final merge of two sequences, say of lengths a and b respectively, to form the ordered sequence of length n. The number of comparisons to do this merge is, at most, $n - 1$. Thus the worst-case number of comparisons is $n - 1$ plus the number of comparisons required (worst cases) to obtain the ordered sequences of lengths a and b. These lengths depend on the particular algorithm, and may even be chosen in some random fashion. The function for the worst-case number of comparisons for any internal two-way merge sort is easily defined recursively, as follows:

1. $w(1) = 0$;
2. $w(n) = (n - 1)\, w(a) + w(b)$, where $n = a + b$;
3. n, a, and b are positive nonzero integers.

To minimize the worst case, a and b should be chosen so that $w(n)$ is as small as possible. It follows from results presented by Glicksman[7] that $w(n)$ is minimized when a is chosen to be the ceiling of one half of n, and b is chosen to be the floor of one half

Table 2 Worst-case number of comparisons needed to sort n numbers using two-way merging

+/⌈2⍟1+⍳N

	0	1	2	3	4	5	6	7	8	9
0--	0	0	1	3	5	8	11	14	17	21
1--	25	29	33	37	41	45	49	54	59	64
2--	69	74	79	84	89	94	99	104	109	114
3--	119	124	129	135	141	147	153	159	165	171
4--	177	183	189	195	201	207	213	219	225	231
5--	237	243	249	255	261	267	273	279	285	291
6--	297	303	309	315	321	328	335	342	349	356
7--	363	370	377	384	391	398	405	412	419	426
8--	433	440	447	454	461	468	475	482	489	496
9--	503	510	517	524	531	538	545	552	559	566
10--	573	580	587	594	601	608	615	622	629	636
11--	643	650	657	664	671	678	685	692	699	706
12--	713	720	727	734	741	748	755	762	769	777
13--	785	793	801	809	817	825	833	841	849	857
14--	865	873	881	889	897	905	913	921	929	937
15--	945	953	961	969	977	985	993	1001	1009	1017
16--	1025	1033	1041	1049	1057	1065	1073	1081	1089	1097
17--	1105	1113	1121	1129	1137	1145	1153	1161	1169	1177
18--	1185	1193	1201	1209	1217	1225	1233	1241	1249	1257
19--	1265	1273	1281	1289	1297	1305	1313	1321	1329	1337
20--	1345	1353	1361	1369	1377	1385	1393	1401	1409	1417
21--	1425	1433	1441	1449	1457	1465	1473	1481	1489	1497
22--	1505	1513	1521	1529	1537	1545	1553	1561	1569	1577
23--	1585	1593	1601	1609	1617	1625	1633	1641	1649	1657
24--	1665	1673	1681	1689	1697	1705	1713	1721	1729	1737
25--	1745	1753	1761	1769	1777	1785	1793	1802	1811	1820
26--	1829	1838	1847	1856	1865	1874	1883	1892	1901	1910
27--	1919	1928	1937	1946	1955	1964	1973	1982	1991	2000
28--	2009	2018	2027	2036	2045	2054	2063	2072	2081	2090
29--	2099	2108	2117	2126	2135	2144	2153	2162	2171	2180
30--	2189	2198	2207	2216	2225	2234	2243	2252	2261	2270
31--	2279	2288	2297	2306	2315	2324	2333	2342	2351	2360
32--	2369	2378	2387	2396	2405	2414	2423	2432	2441	2450
33--	2459	2468	2477	2486	2495	2504	2513	2522	2531	2540
34--	2549	2558	2567	2576	2585	2594	2603	2612	2621	2630
35--	2639	2648	2657	2666	2675	2684	2693	2702	2711	2720
36--	2729	2738	2747	2756	2765	2774	2783	2792	2801	2810
37--	2819	2828	2837	2846	2855	2864	2873	2882	2891	2900
38--	2909	2918	2927	2936	2945	2954	2963	2972	2981	2990
39--	2999	3008	3017	3026	3035	3044	3053	3062	3071	3080
40--	3089	3098	3107	3116	3125	3134	3143	3152	3161	3170
41--	3179	3188	3197	3206	3215	3224	3233	3242	3251	3260
42--	3269	3278	3287	3296	3305	3314	3323	3332	3341	3350
43--	3359	3368	3377	3386	3395	3404	3413	3422	3431	3440
44--	3449	3458	3467	3476	3485	3494	3503	3512	3521	3530
45--	3539	3548	3557	3566	3575	3584	3593	3602	3611	3620
46--	3629	3638	3647	3656	3665	3674	3683	3692	3701	3710
47--	3719	3728	3737	3746	3755	3764	3773	3782	3791	3800
48--	3809	3818	3827	3836	3845	3854	3863	3872	3881	3890
49--	3899	3908	3917	3926	3935	3944	3953	3962	3971	3980
50--	3989	3998	4007	4016	4025	4034	4043	4052	4061	4070
51--	4079	4088	4097	4107	4117	4127	4137	4147	4157	4167
52--	4177	4187	4197	4207	4217	4227	4237	4247	4257	4267
53--	4277	4287	4297	4307	4317	4327	4337	4347	4357	4367
54--	4377	4387	4397	4407	4417	4427	4437	4447	4457	4467
55--	4477	4487	4497	4507	4517	4527	4537	4547	4557	4567
56--	4577	4587	4597	4607	4617	4627	4637	4647	4657	4667
57--	4677	4687	4697	4707	4717	4727	4737	4747	4757	4767
58--	4777	4787	4797	4807	4817	4827	4837	4847	4857	4867
59--	4877	4887	4897	4907	4917	4927	4937	4947	4957	4967
60--	4977	4987	4997	5007	5017	5027	5037	5047	5057	5067
61--	5077	5087	5097	5107	5117	5127	5137	5147	5157	5167
62--	5177	5187	5197	5207	5217	5227	5237	5247	5257	5267
63--	5277	5287	5297	5307	5317	5327	5337	5347	5357	5367
64--	5377	5387	5397	5407	5417	5427	5437	5447	5457	5467
65--	5477	5487	5497	5507	5517	5527	5537	5547	5557	5567
66--	5577	5587	5597	5607	5617	5627	5637	5647	5657	5667
67--	5677	5687	5697	5707	5717	5727	5737	5747	5757	5767
68--	5777	5787	5797	5807	5817	5827	5837	5847	5857	5867
69--	5877	5887	5897	5907	5917	5927	5937	5947	5957	5967
70--	5977	5987	5997	6007	6017	6027	6037	6047	6057	6067
71--	6077	6087	6097	6107	6117	6127	6137	6147	6157	6167
72--	6177	6187	6197	6207	6217	6227	6237	6247	6257	6267
73--	6277	6287	6297	6307	6317	6327	6337	6347	6357	6367
74--	6377	6387	6397	6407	6417	6427	6437	6447	6457	6467
75--	6477	6487	6497	6507	6517	6527	6537	6547	6557	6567
76--	6577	6587	6597	6607	6617	6627	6637	6647	6657	6667
77--	6677	6687	6697	6707	6717	6727	6737	6747	6757	6767
78--	6777	6787	6797	6807	6817	6827	6837	6847	6857	6867
79--	6877	6887	6897	6907	6917	6927	6937	6947	6957	6967
80--	6977	6987	6997	7007	7017	7027	7037	7047	7057	7067
81--	7077	7087	7097	7107	7117	7127	7137	7147	7157	7167
82--	7177	7187	7197	7207	7217	7227	7237	7247	7257	7267
83--	7277	7287	7297	7307	7317	7327	7337	7347	7357	7367
84--	7377	7387	7397	7407	7417	7427	7437	7447	7457	7467
85--	7477	7487	7497	7507	7517	7527	7537	7547	7557	7567
86--	7577	7587	7597	7607	7617	7627	7637	7647	7657	7667
87--	7677	7687	7697	7707	7717	7727	7737	7747	7757	7767
88--	7777	7787	7797	7807	7817	7827	7837	7847	7857	7867
89--	7877	7887	7897	7907	7917	7927	7937	7947	7957	7967
90--	7977	7987	7997	8007	8017	8027	8037	8047	8057	8067
91--	8077	8087	8097	8107	8117	8127	8137	8147	8157	8167
92--	8177	8187	8197	8207	8217	8227	8237	8247	8257	8267
93--	8277	8287	8297	8307	8317	8327	8337	8347	8357	8367
94--	8377	8387	8397	8407	8417	8427	8437	8447	8457	8467
95--	8477	8487	8497	8507	8517	8527	8537	8547	8557	8567
96--	8577	8587	8597	8607	8617	8627	8637	8647	8657	8667
97--	8677	8687	8697	8707	8717	8727	8737	8747	8757	8767
98--	8777	8787	8797	8807	8817	8827	8837	8847	8857	8867
99--	8877	8887	8897	8907	8917	8927	8937	8947	8957	8967

(+/⌈2⍟1+⍳1000)=8977

of n, where the floor, "$\lfloor$," of x is the largest integer not greater than x. If a and b are chosen as above, it is a mathematical curiosity that $w(n)$ is given by the following two formulas:

$$w(n) = 1 + 2^i(i - 1) + i(n - 2^i)$$

where $i = \lfloor \log_2 n$ and

$$w(n) = \sum_{i=1}^{n} \lceil \log_2 i$$

The result $w(n)$ is listed for values of n up to 1000 in Table 2. It is fairly well approximated by n ($\log_2$ $(.5n)$), an approximation that has long been used for the average number of comparisons for sorting by two-way merging. However, $w(n)$ is not the average value of c for this sort, but its worst case. Before giving the algorithm, let's look at the expected value of c for such an algorithm.

The expected value of c, $E(c)$, can also be computed by the following recursive function:

1. $e(1) = 0$;
2. $e(n) = e(a) + e(b) + m(a; b)$, where $a = \lceil .5n$, $b = \lfloor .5n$, and $m(a; b)$ is a function of a and b that gives the expected value of the number of comparisons to merge two sequences of lengths a and b, respectively.

The m function is given in the Appendix. The e function has been evaluated for values of n up to 1000 in Table 3. From a comparison of the entries in Tables 1 and 3, it is apparent that $E(c)$ for this algorithm does not differ very much from the theoretical limit for $E(c)$ of any algorithm. For $n = 1000$, the difference is 177 comparisons, a small number compared to 8530, the theoretical limit.

A sort based on the above mathematical model will achieve its minimum value for c when the data are already in ascending order, assuming that the sequence of length b is created first, and that numbers are picked up from the input in the order they occur, i.e., $A[0]$ is the first number used, followed by $A[1]$, etc. The minimum value of c can be calculated by replacing the "$n - 1$" in function w by "b," since if the numbers in the b sequence all precede the numbers in the a sequence, it will take exactly b comparisons to merge the two. When the correlation coefficient is -1 (the data are in inverse order), then the number of comparisons taken will be close to the number taken if the coefficient is $+1$ (the data are in ascending order). An examination of this minimum shows that the algorithm does take advantage of "natural sequencing" present in the data. For example, to sort an ascending input of one hundred numbers, it takes 316 comparisons as opposed to the average of 542 comparisons. Note that if an algorithm could always sort with a worst case equal to the ceiling of $\log_2(!n)$, it could not take advantage of natural sequencing occurring in the data.

Table 3 Expected value of the number of comparisons taken to sort n numbers using two-way merging

	0	1	2	3	4	5	6	7	8	9
0	.0	.0	1.0	2.7	4.7	7.2	9.8	12.7	15.7	19.2
1	22.7	26.3	30.0	33.8	37.7	41.7	45.7	50.1	54.5	59.0
2	63.5	68.1	72.7	77.4	82.1	86.9	91.8	36.7	101.6	106.5
3	111.5	116.5	121.5	126.9	132.3	137.7	143.2	148.6	154.1	159.6
4	165.1	170.7	176.3	181.9	187.6	193.2	198.9	204.5	210.2	216.0
5	221.9	227.8	233.6	239.5	245.4	251.3	257.2	263.2	269.1	275.1
6	281.1	287.1	293.1	299.1	305.1	311.5	317.9	324.3	330.7	337.1
7	343.5	350.0	356.4	362.9	369.3	375.8	382.3	388.8	395.3	401.8
8	408.3	414.9	421.5	428.1	434.7	441.3	447.9	454.6	461.2	467.8
9	474.5	481.1	487.8	494.4	501.1	507.8	514.4	521.3	528.1	535.0
10	541.8	548.7	555.6	562.4	569.3	576.2	583.1	590.0	596.8	603.7
11	610.6	617.5	624.4	631.4	638.4	645.3	652.3	659.3	666.2	673.2
12	680.2	687.2	694.2	701.1	708.1	715.1	722.1	729.1	736.1	743.5
13	750.9	758.3	765.7	773.2	780.6	788.0	795.4	802.8	810.2	817.7
14	825.1	832.5	840.0	847.4	854.8	862.3	869.8	877.2	884.7	892.2
15	899.7	907.2	914.6	922.1	929.6	937.1	944.6	952.1	959.6	967.1
16	974.6	982.2	989.8	997.4	1005.0	1012.6	1020.2	1027.8	1035.4	1043.0
17	1050.7	1058.3	1065.9	1073.5	1081.1	1088.7	1096.4	1104.0	1111.7	1119.3
18	1127.0	1134.6	1142.3	1149.9	1157.6	1165.2	1172.9	1180.6	1188.2	1195.9
19	1203.6	1211.2	1218.9	1226.7	1234.6	1242.4	1250.3	1258.1	1266.0	1273.8
20	1281.7	1289.6	1297.4	1305.3	1313.1	1321.0	1328.9	1336.7	1344.6	1352.5
21	1360.4	1368.3	1376.1	1384.0	1391.9	1399.8	1407.7	1415.6	1423.5	1431.4
22	1439.3	1447.2	1455.1	1463.0	1470.9	1478.9	1486.8	1494.8	1502.7	1510.7
23	1518.7	1526.6	1534.6	1542.6	1550.5	1558.5	1566.5	1574.4	1582.4	1590.4
24	1598.4	1606.4	1614.3	1622.3	1630.3	1638.3	1646.3	1654.3	1662.3	1670.3
25	1678.3	1686.3	1694.3	1702.3	1710.3	1718.3	1726.3	1734.7	1743.1	1751.5
26	1759.9	1768.3	1776.7	1785.1	1793.5	1801.9	1810.3	1818.7	1827.1	1835.6
27	1844.0	1852.4	1860.8	1869.2	1877.6	1886.1	1894.5	1902.9	1911.3	1919.8
28	1928.2	1936.6	1945.1	1953.5	1961.9	1970.4	1978.8	1987.2	1995.7	2004.1
29	2012.6	2021.1	2029.6	2038.0	2046.5	2055.0	2063.5	2071.9	2080.4	2088.9
30	2097.4	2105.9	2114.3	2122.8	2131.3	2139.8	2148.3	2156.8	2165.3	2173.8
31	2182.3	2190.8	2199.3	2207.8	2216.3	2224.8	2233.3	2241.8	2250.3	2258.8
32	2267.3	2275.9	2284.5	2293.1	2301.7	2310.3	2318.9	2327.5	2336.1	2344.7
33	2353.3	2361.9	2370.5	2379.1	2387.7	2396.3	2404.9	2413.5	2422.1	2430.7
34	2439.3	2447.9	2456.6	2465.2	2473.8	2482.4	2491.0	2499.6	2508.3	2516.9
35	2525.5	2534.1	2542.7	2551.4	2560.0	2568.7	2577.3	2586.0	2594.6	2603.3
36	2611.9	2620.6	2629.2	2637.9	2646.5	2655.2	2663.9	2672.5	2681.2	2689.8
37	2698.5	2707.1	2715.8	2724.5	2733.1	2741.8	2750.5	2759.1	2767.8	2776.5
38	2785.1	2793.8	2802.5	2811.1	2819.8	2828.6	2837.5	2846.3	2855.2	2864.0
39	2872.9	2881.7	2890.6	2899.4	2908.3	2917.1	2926.0	2934.9	2943.7	2952.6
40	2961.4	2970.3	2979.1	2988.0	2996.9	3005.7	3014.6	3023.4	3032.3	3041.2
41	3050.0	3058.9	3067.8	3076.6	3085.5	3094.3	3103.2	3112.1	3121.0	3129.9
42	3138.8	3147.6	3156.5	3165.4	3174.3	3183.2	3192.1	3201.0	3209.9	3218.8
43	3227.6	3236.5	3245.4	3254.3	3263.2	3272.1	3281.0	3289.9	3298.8	3307.7
44	3316.6	3325.5	3334.4	3343.3	3352.2	3361.1	3370.0	3378.9	3387.8	3396.8
45	3405.7	3414.7	3423.6	3432.6	3441.6	3450.5	3459.5	3468.5	3477.4	3486.4
46	3495.3	3504.3	3513.3	3522.2	3531.2	3540.2	3549.1	3558.1	3567.1	3576.0
47	3585.0	3594.0	3603.0	3611.9	3620.9	3629.9	3638.8	3647.8	3656.8	3665.8
48	3674.7	3683.7	3692.7	3701.7	3710.7	3719.7	3728.7	3737.7	3746.7	3755.6
49	3764.6	3773.6	3782.6	3791.6	3800.6	3809.6	3818.6	3827.6	3836.6	3845.6
50	3854.6	3863.6	3872.6	3881.6	3890.6	3899.6	3908.6	3917.6	3926.6	3935.6
51	3944.6	3953.6	3962.6	3972.0	3981.4	3990.8	4000.2	4009.6	4019.0	4028.4
52	4037.8	4047.2	4056.6	4066.0	4075.4	4084.8	4094.2	4103.6	4113.0	4122.4
53	4131.8	4141.3	4150.7	4160.1	4169.5	4178.9	4188.3	4197.7	4207.1	4216.5
54	4225.9	4235.4	4244.8	4254.2	4263.6	4273.0	4282.4	4291.9	4301.3	4310.7
55	4320.1	4329.6	4339.0	4348.4	4357.8	4367.3	4376.7	4386.1	4395.6	4405.0
56	4414.4	4423.8	4433.3	4442.7	4452.1	4461.6	4471.0	4480.4	4489.9	4499.3
57	4508.7	4518.2	4527.6	4537.0	4546.5	4555.9	4565.3	4574.8	4584.3	4593.8
58	4603.2	4612.7	4622.2	4631.7	4641.1	4650.6	4660.1	4669.6	4679.0	4688.5
59	4698.0	4707.5	4716.9	4726.4	4735.9	4745.4	4754.9	4764.3	4773.8	4783.3
60	4792.8	4802.3	4811.7	4821.2	4830.7	4840.2	4849.7	4859.1	4868.6	4878.1
61	4887.6	4897.1	4906.6	4916.1	4925.6	4935.1	4944.6	4954.1	4963.6	4973.1
62	4982.6	4992.1	5001.6	5011.0	5020.5	5030.0	5039.5	5049.0	5058.5	5068.0
63	5077.5	5087.0	5096.5	5106.0	5115.5	5125.0	5134.5	5144.0	5153.5	5163.0
64	5172.5	5182.1	5191.7	5201.3	5210.9	5220.5	5230.1	5239.7	5249.3	5258.9
65	5268.5	5278.1	5287.7	5297.3	5306.9	5316.5	5326.1	5335.7	5345.3	5354.9
66	5364.5	5374.1	5383.7	5393.3	5402.9	5412.5	5422.1	5431.7	5441.3	5450.9
67	5460.6	5470.2	5479.8	5489.4	5499.0	5508.6	5518.2	5527.8	5537.4	5547.1
68	5556.7	5566.3	5575.9	5585.5	5595.1	5604.7	5614.4	5624.0	5633.6	5643.2
69	5652.8	5662.4	5672.1	5681.7	5691.3	5700.9	5710.5	5720.2	5729.8	5739.4
70	5749.0	5758.6	5768.2	5777.9	5787.5	5797.1	5806.8	5816.4	5826.1	5835.7
71	5845.4	5855.0	5864.7	5874.3	5884.0	5893.6	5903.3	5912.9	5922.6	5932.2
72	5941.9	5951.5	5961.2	5970.8	5980.5	5990.1	5999.8	6009.4	6019.1	6028.8
73	6038.4	6048.1	6057.7	6067.4	6077.0	6086.7	6096.3	6106.0	6115.7	6125.3
74	6135.0	6144.6	6154.3	6164.0	6173.6	6183.3	6193.0	6202.6	6212.3	6221.9
75	6231.6	6241.3	6250.9	6260.6	6270.3	6279.9	6289.6	6299.3	6308.9	6318.6
76	6328.3	6337.9	6347.6	6357.3	6366.9	6376.6	6386.3	6395.9	6405.6	6415.4
77	6425.3	6435.1	6445.0	6454.8	6464.7	6474.5	6484.4	6494.2	6504.1	6513.9
78	6523.8	6533.6	6543.5	6553.3	6563.2	6573.0	6582.9	6592.7	6602.6	6612.4
79	6622.3	6632.2	6642.0	6651.9	6661.7	6671.6	6681.4	6691.3	6701.1	6711.0
80	6720.8	6730.7	6740.6	6750.4	6760.3	6770.1	6780.0	6789.9	6799.7	6809.6
81	6819.4	6829.3	6839.2	6849.0	6858.9	6868.7	6878.6	6888.5	6898.3	6908.2
82	6918.1	6927.9	6937.8	6947.6	6957.5	6967.4	6977.2	6987.1	6997.0	7006.8
83	7016.7	7026.6	7036.4	7046.3	7056.2	7066.1	7076.0	7085.9	7095.7	7105.6
84	7115.5	7125.4	7135.3	7145.2	7155.1	7165.0	7174.8	7184.7	7194.6	7204.5
85	7214.4	7224.3	7234.2	7244.1	7254.0	7263.9	7273.7	7283.6	7293.5	7303.4
86	7313.3	7323.2	7333.1	7343.0	7352.9	7362.8	7372.7	7382.6	7392.5	7402.3
87	7412.2	7422.1	7432.0	7441.9	7451.8	7461.7	7471.6	7481.5	7491.4	7501.3
88	7511.2	7521.1	7531.0	7540.9	7550.8	7560.7	7570.6	7580.5	7590.4	7600.3
89	7610.2	7620.1	7630.0	7639.9	7649.8	7659.7	7669.6	7679.6	7689.5	7699.5
90	7709.5	7719.4	7729.4	7739.3	7749.3	7759.3	7769.2	7779.2	7789.1	7799.1
91	7809.1	7819.0	7829.0	7839.0	7848.9	7858.9	7868.8	7878.8	7888.8	7898.7
92	7908.7	7918.7	7928.6	7938.6	7948.6	7958.5	7968.5	7978.4	7988.4	7998.4
93	8008.3	8018.3	8028.3	8038.3	8048.2	8058.2	8068.2	8078.1	8088.1	8098.1
94	8108.0	8118.0	8128.0	8138.0	8147.9	8157.9	8167.9	8177.8	8187.8	8197.8
95	8207.7	8217.7	8227.7	8237.7	8247.6	8257.6	8267.6	8277.6	8287.5	8297.5
96	8307.5	8317.5	8327.5	8337.4	8347.4	8357.4	8367.4	8377.4	8387.4	8397.4
97	8407.4	8417.4	8427.4	8437.3	8447.3	8457.3	8467.3	8477.3	8487.3	8497.3
98	8507.3	8517.3	8527.3	8537.3	8547.3	8557.2	8567.2	8577.2	8587.2	8597.2
99	8607.2	8617.2	8627.2	8637.2	8647.2	8657.2	8667.2	8677.2	8687.2	8697.2
100	8707.2									

Obtaining the algorithm

Having analyzed the efficiency of two-way merging in terms of comparisons, we now develop the algorithm. Recalling the constraint that no more space should be used than whatever is necessary to hold the result, we see that the conventional two-way merge technique, which requires two areas, is inappropriate. Instead, merging can be done by a chaining technique in the single area that is used to hold the result. With this area labeled as the vector **P**, an ordered sequence can be represented in **P** as a chained list of indices. In any sequence, there is a first, or lowest, element. Call the index of this number the head of the chain. If i is the head of the chain, then let $P[i]$ be the index of the next number in the sequence. That position in **P** contains the index of the next number in the sequence, and so on to the last number in the sequence. The last position in the chain contains its own index. The following example illustrates these concepts:

Index or position number:	0	1	2	3	4	5	6	7	8	9
The vector **A**:	11	9	3	6	8	5	0	7	4	2
The vector **P**:	0	0	3	4	1	7	5	7	8	8

chain example

There are three chains contained in the vector **P**, one representing the sequence of numbers $A[2\ 3\ 4\ 1\ 0]$, one representing the sequence $A[6\ 5\ 7]$, and one representing the sequence $A[9\ 8]$. The first chain is stored in **P** in positions 2, 3, 4, 1, and 0. The head of the first chain is 2, the index of the lowest number in the sequence 3 6 8 9 11. $P[2]$ is 3, the index of the next number in the sequence, and $P[3]$ is 4, the index of the number 8 in **A**, the next number in the sequence. $P[4]$ is 1, the index of the 9 in **A**, and $P[1]$ is 0, the index of the 11 in **A**. Since $A[0]$,11, is the last number in this sequence, $P[0]$ contains 0 to indicate this. The last position in the chain always contains its own index. The other two sequences represented in **P** start in positions 6 and 9, respectively, and are represented in the same chained fashion. If we have any one of the three starting positions, 2, 6, or 9, we can examine the numbers in any one of the three sequences in the proper order. Given the heads of the chains representing any two sequences, we can merge them to form a new sequence, represented by rechaining the indices in **P**, and the starting position of the new chain can be recorded. If this is done with the last two sequences in the example, **P** becomes 0, 0, 3, 4, 1, 7, 9, 7, 5, 8. The head of the new chain is 6; the end of the new chain is 7.

Let M2 be a function to merge two sequences represented by chains in this way. M2, shown in Figure 1 as an APL function, accepts the heads of two chains, i and j, and returns the head of the single chain as a result.

The M2 function can now be used to repeatedly merge sequences of appropriate lengths, until a single sequence is obtained. A recursive sorting algorithm, MP, is given in Figure 2 as an APL function. MP creates a chained representation in **P** of a sequence of n numbers and gives the head of the chain as a result.

Figure 1 APL program of the M2 function

```
      ∇ IP;I;J;K;L
[1]     P←¯1-P
[2]     I←ρP
[3]     →ι0>L←I←I-1
[4]     →3 IF 0≤J←P[I]
[5]     K←P[J←¯1-J]
[6]     P[J]←L
[7]     L←J
[8]     J←K
[9]     →5 IF J<0
[10]    →3
      ∇
```

Figure 2 APL program of the MP function (uses M2)

```
      ∇ Z←MP N
[1]     →4 IF 1=N
[2]     Z←(MP⌈0.5×N) M2 MP⌊0.5×N
[3]     →0
[4]     NEX←1+P[Z]←Z←NEX
      ∇
```

The MP program also has all the comparison properties implied by the previous mathematical model. If $n = 1$, then the head of a chain of length one is needed, which can easily be obtained by getting the index of the next unexamined number in the input vector **A** and storing that number in its own position in **P**, the signal for the end of the chain. The head of the one-element chain is the result of executing MP when n is 1. However, to get the index of the next unexamined number in **A**, a nonlocal variable, NEX, is needed to record this information. After using NEX, by adding one to it, we see that it is all set for the next request. In MP, NEX is assumed to have been set to zero before the first execution of MP.

MP function

If n is not one in the MP function, then a chain must be created by merging two other sequences, neither of which has been created yet. The function MP is used recursively to create these other two sequences first, and then the merging function, M2, is used to produce the head of the single sequence of length n. When n is not one, the result of MP is set to the output of M2, where M2 is used to merge the two sequences formed by asking for MP($\lfloor .5n$) and MP($\lceil .5n$), respectively. If the vectors **A** and **P** are defined, NEX is set to zero, and MP n is executed, then MP produces the head of a single chain representing a sequence of n elements in ascending order. Suppose, for example, that **A** is the vector 8, 23, 11, 5, 3, 4, 23. The value of n is 7, the number of elements in the vector **A**. After NEX is set to zero, and the function MP 7 is executed, the value returned by MP is 4, the index of the lowest element of **A**. The vector **P** is then 2, 6, 1, 0, 5, 3, 6. Note that the last index in the chain is 6, and that the original order of equals is preserved.

the desired permutation vector

Having obtained a chained sequence in **P**, we find that it is still not the desired permutation vector. Then, what must be done to arrive at the desired permutation? As a first step, trace the chain from beginning to end, replacing the links in the chain with their relative positions in the sequence. The head of the chain thus becomes zero, the second element of the chain becomes a one, etc., until the entire chain is traced. When this procedure is applied to **P** in the above example, it becomes 3, 5, 4, 2, 0, 1, 6. A number in **P**, say $P[x]$, is now the number of elements in **A** which come before $A[x]$ in the final ordering of **A**. $P[0]$ is a 3, thus the element $A[0]$ will be in position 3 in the final result, since there are three elements of **A** preceding it. If we now set $A[P]$ to **A**, the elements of **A** will be in ascending order. Also, if we set $P[P]$ to $0, 1, 2, \cdots, n - 1$, **P** would be the desired permutation, since $A[P]$ would be **A** in ascending order.

Let **P** be some permutation vector of the same length as **A**, and let $iv(P)$ be the inverse of **P**. $P[iv(P)]$ is the identity permutation, $0, 1, 2, 3, \cdots$, etc. Let $X = A[P]$. Then $X[iv(P)] = A$. Since the vector **P** obtained above is such that $X[P] = A$, where X is the elements of **A** in ascending order, then $X = A[iv(P)]$. It is then necessary to change **P** into $iv(P)$ without using another area to do so. This problem is the same as that faced in the sorting method called reserved-seat sorting, and essentially involves inverting a permutation in its own space. Algorithms that do this efficiently have been known for at least the last ten years, and one such algorithm to perform this inversion (IP) is shown in Figure 3.

Figure 3 An algorithm for inverting a permutation

```
      ∇ Z←I M2 J;T
[1]     →4 IF A[I]<A[J]
[2]     Z←J
[3]     →8
[4]     Z←I
[5]     →12
[6]     →11 IF A[I]<A[J]
[7]     P[T]←J
[8]     →6 IF T≠J←P[T←J]
[9]     P[T]←I
[10]    →0
[11]    P[T]←I
[12]    →6 IF T≠I←P[T←I]
[13]    P[T]←J
      ∇
```

With the above results, the function MERGESORT1, shown in Figure 4, creates the permutation **P**, given **A**, by using the recursive function MP and the function IP to do so. Line 1 sets NEX to zero, the index of the first element of **A**. NEX is incremented by one each time it is used in the MP function to incorporate the elements of **A** into the sort one at a time. Line 2 sets **P** to a vector that is the same length as **A**, containing all zeros. The zeros are not used, and **P** could be set to anything, as long as it is the same length as **A**. Line 2 then executes MP, giving it the length of **P** for an argument. A chained representation of a single sequence involving all the elements of **A** is created in **P**, and the head of this chain is returned as a result. I is set to the head of this chain.

Figure 4 Function MERGESORT1 (uses MP, IP)

```
      ∇ P←MERGESORT1 A;NEX;I;J;K
[1]     NEX←0
[2]     I←MPρP←(ρA)ρ0
[3]     K←¯1
[4]     J←P[I]
[5]     P[I]←K←K+1
[6]     I←J
[7]     →4 IF K<¯1+ρP
[8]     IP
      ∇
```

Lines 3 through 7 trace out this chain, replacing each index in the chain with the index of its position in the final ordering. A permutation is now obtained in **P** that is the inverse of the permutation desired. The permutation inversion function is then used, on line 8, to invert **P** in its own space, yielding the final result.

nonrecursive algorithm

Looking at the recursive sorting algorithm, we see that the equivalent nonrecursive algorithm is obtained and extended to multidimensional arrays. The new algorithm, shown in Figure 5 as GRADE, uses two stacks to implement the recursion. In Figure 5, C is the coordinate along which ordering permutations are created, **Z** is the resulting array containing the ordering permutations, and **A** is the input array. The two stacks are the vectors **P** and **R**. **P** contains the lengths of the sequences needed at various stages of the sort, whereas **R** contains the heads of chains representing merged sequences. The lengths of **P** and **R** must be at least one more than $\log_2(n)$, where n is the number of elements to be sorted. This is reflected on line 6 of GRADE, where **P** and **R** are initially set to zeros.

Lines 1 through 6 of GRADE perform the initial housekeeping and are executed only once. Lines 7 through 10 are executed once for each ordering permutation produced in the output array. The variable I serves the same purpose as NEX does in MP, i.e., it is used to pick up the next unexamined element of the data being sorted. J and K are indices for accessing the two stacks **P** and **R**, respectively.

Lines 11 through 13 examine the **P** stack, determining the length of the next sequence required. When a sequence of length one is required, line 12 creates it. When a sequence of length greater than one is needed, lines 14 through 29 create it by merging two chains and store the head of the resulting chain at **R**[K].

Figure 5 Nonrecursive sorting algorithm

```
     ∇ Z←C GRADE A;B;E;H;I;J;K;L;N;T;W;Y;S;P;Q;R
[1]    E←⍴Z←,(H←⍴A)⍴0
[2]    A←,A
[3]    L←H[C]
[4]    Y←L×W←H⊥C=⍳⍴H
[5]    N←0
[6]    P←R←(1+⌈2⌈.⍟1⌈H)⍴0
[7]    B←0
[8]    P[0]←L
[9]    I←N-W
[10]   K←¯1+J←0
[11]   →11 IF 1≠P[J←J+1]←⌊0.5×|P[J]-P[J]<0
[12]   R[K←K+1]←Z[I]←I←I+W
[13]   →11 IF 0>P[J]←-P[J←J-1]
[14]   S←R[K]
[15]   Q←R[K←K-1]
[16]   →19 IF A[S]<A[Q]
[17]   R[K]←Q
[18]   →23
[19]   R[K]←S
[20]   →27
[21]   →26 IF A[S]<A[Q]
[22]   Z[T]←Q
[23]   →21 IF T≠Q←Z[T←Q]
[24]   Z[T]←S
[25]   →29
[26]   Z[T]←S
[27]   →21 IF T≠S←Z[T←S]
[28]   Z[T]←Q
[29]   →13 IF J≠0
[30]   I←R[K]
[31]   K←N
[32]   J←Z[I]
[33]   Z[I]←-K←K+W
[34]   I←J
[35]   →32 IF K<N+Y
[36]   →44 IF N>S←K←K-W
[37]   →36 IF 0≤J←Z[K]
[38]   I←Z[J←-W+J]
[39]   Z[J]←L|⌊S÷W
[40]   S←J
[41]   J←I
[42]   →38 IF J<0
[43]   →36
[44]   N←N+1
[45]   →8 IF W>B←B+1
[46]   N←N+Y-W
[47]   →7 IF N<E
[48]   Z←H⍴Z
     ∇

     ∇ Z←A IF B
[1]    Z←A⌈⍳B
     ∇
```

When a chain of the right length for one of the ordering permutations is obtained, lines 30 through 35 trace it, replacing its elements with the indices of their positions in the final ordering. Then lines 36 through 43 invert the permutation in place. Lines 44 through 47 check for completion of the entire process. Line 48 causes the result, **Z**, to be an array of the same size that **A** is originally.

Summary comment

The machine language implementation was derived from the APL function GRADE directly. The modeling process used to obtain the final machine language program eliminated a substantial number of defects before a machine language program even existed and gave an approximate idea of the performance of the machine language program. If an algorithm is to be implemented in machine language, it is recommended that it first be modeled in the fashion used in this paper, starting with a mathematical analysis of the algorithm, then obtaining successive algorithms at more detailed levels of description until the machine language program has been obtained.

Appendix

Section 1. The usual sample correlation coefficient is

$$\hat{\rho} = \frac{\sum_{i=0}^{n-1} (x_i - \bar{x})(y_i - \bar{y})}{\left[\sum_{i=0}^{n-1} (x_i - \bar{x})^2 \sum_{i=0}^{n-1} (y_i - \bar{y})^2\right]^{1/2}}$$

If x and y are zero-origin permutation vectors of length n, then

$$\hat{\rho} = \frac{\left[12 \sum_{i=0}^{n-1} x_i y_i\right] - 3n(n-1)^2}{n^3 - n}$$

When y is the identity permutation, then

$$x_i y_i = i x_i$$

Section 2. The expected value of the number of comparisons taken to merge two sequences of lengths x and y, where the sequences have been formed by ordering two sets of independent identically distributed random variables, is given by:

$$m(x; y) = \sum_{i=1}^{x \lceil y} \frac{x^{(i)} y + y^{(i)} x}{(x + y)^{(i)}} = \frac{xy(x + y + 2)}{(x + 1)(y + 1)}$$

where $x^{(i)}$ and $(x + y)^{(i)}$ denote falling factorials, $x(x - 1)(x - 2) \cdots (x + 1 - i)$, i terms in the product, and $x^{(0)} = 1$. The term $x \lceil y$ is the larger of either x or y.

Section 3. For any sorting algorithm based on comparing, $E(C) \geq \log_2 (!n)$, provided that the n numbers are distinct, or provided that tests for equality are *not* made, and the $!n$ arrangements of the n numbers are equally likely.

Proof: Let **X** be the vector of n numbers to be sorted, and let **Y** be the vector of the n numbers sorted in ascending order. Then

$\mathbf{Y}[\mathbf{P}] \equiv \mathbf{X}$, where $\mathbf{P}$ is a permutation vector, and $\mathbf{P}$ is unique because of the requirement that $\mathbf{X}$ contains distinct numbers. When two numbers X_i and X_j are compared, if $X_i < X_j$, then $P_i < P_j$, and if $X_i > X_j$, then $P_i > P_j$. Initially the permutation $\mathbf{P}$ can be any one of the $!n$ permutations. The comparison restricts $\mathbf{P}$ to one of two subsets of $!n$ permutations, either the subset where $P_i < P_j$, or the subset where $P_i > P_j$. The next comparison similarly partitions one of these two subsets into two mutually exclusive sets, one containing $\mathbf{P}$, and the other not. This process continues until a one-element set containing $\mathbf{P}$ has been determined.

Now, suppose that, when a sort is performed where $\mathbf{P}$ is the ordering permutation, the results of all the comparisons that the program performed while sorting, i.e., determining $\mathbf{P}$, were recorded in the order the comparisons were done. When two numbers X_i and X_j are compared, record a zero if $X_i < X_j$, and record a 1 if $X_i > X_j$, where $i < j$. The result of the comparison, i.e., a zero or a one, determines which subset of the possible permutations contains $\mathbf{P}$.

Lemma. The sequence of zeros and ones recorded for a given permutation $\mathbf{P}$ is not identical with the sequence of zeros and ones which would have been recorded for a permutation $\mathbf{Q} \neq \mathbf{P}$.

Proof: Each time a comparison is done, a set of permutations is partitioned into two subsets, one containing $\mathbf{P}$ and the other not. If $\mathbf{Q}$ is not the same as $\mathbf{P}$, and if $\mathbf{Q}$ had been the permutation to be determined, then if $\mathbf{Q}$ was in the set not containing $\mathbf{P}$, the result of the comparison would have been different, i.e., a zero would have been recorded instead of a one, or a one instead of a zero. In this case, the sequences recorded for $\mathbf{P}$ and $\mathbf{Q}$ are different. The only way for the sequences to be the same is for $\mathbf{Q}$ to always be in the set containing $\mathbf{Q}$. But eventually a one-element set containing $\mathbf{P}$ alone is obtained; hence, $\mathbf{Q}$ must have been in a set not containing $\mathbf{P}$ at some point. Thus, the sequences for $\mathbf{P}$ and $\mathbf{Q}$ are different somewhere.

Since $\mathbf{P}$ and $\mathbf{Q}$ are arbitrary permutations, then the sequences recorded for any pair of permutations are different. This means that the sequence for a given permutation $\mathbf{P}$ is unique. Since there are $!n$ permutations, there are $!n$ distinct sequences of zeros and ones. In any system of C distinct equally likely codes, an optimal encoding of the system using zeros and ones must have an average of at least $\log_2 C$ bits in a code. Hence, the average number of zeros and ones in a recorded sequence is at least $\log_2(!n)$, or the average number of comparisons to sort is at least $\log_2(!n)$.

The above result also applies to any method of sorting where a series of tests is applied, and where each test has only two possible outcomes. The extension to tests where more than two outcomes can result is similar.

CITED REFERENCES

1. K. E. Iverson, *A Programming Language*, John Wiley & Sons, New York, New York (1962).
2. K. E. Iverson, *Elementary Functions: An Algorithmic Treatment*, Science Research Associates, Chicago, Illinois (1966).
3. A. D. Falkoff and K. E. Iverson, *APL\360 User's Manual*, International Business Machines Corporation, Thomas J. Watson Research Center, Yorktown Heights, New York (August 1968). Although not formally supported by IBM, the APL\360 program may be obtained through any IBM branch office.
4. H. Steinhaus, *Mathematical Snapshots*, Oxford University Press, New York, New York, 30–40 (1950).
5. R. G. Busacher and T. L. Saaty, *Finite Graphs and Networks*, McGraw-Hill Book Company, New York, New York, 228–231 (1965).
6. G. J. Hahn and S. S. Shapiro, *Statistical Models in Engineering*, John Wiley & Sons, New York, New York, 64 (1967).
7. S. Glicksman, "Concerning the merging of equal length tape files," *Journal of the Association for Computing Machinery*, **12**, No. 2, 254–258 (April 1965).