Sorting Routines for Microcomputers

Macmillan Microcomputer Books

General Editor: Ian Birnbaum
(General Adviser (Microelectronics in Education)
Education Department, Humberside County Council)

Advanced Graphics with the Acorn Electron
Ian O. Angell and Brian J. Jones
Advanced Graphics with the BBC Model B Microcomputer
Ian O. Angell and Brian J. Jones
Interfacing the BBC Microcomputer
Brian Bannister and Michael Whitehead
Assembly Language Programming for the Acorn Electron
Ian Birnbaum
Assembly Language Programming for the BBC Microcomputer (second edition)
Ian Birnbaum
Using Your Home Computer (Practical Projects for the Micro Owner)
Garth W. P. Davies
Microchild – Learning through LOGO
Serafim Gascoigne
The Purple Planet – Micro-PROLOG for the Spectrum 48K
Serafim Gascoigne
Turtle Fun – LOGO for the Spectrum 48K
Serafim Gascoigne
A Science Teacher's Companion to the BBC Microcomputer
Philip Hawthorne
Operating the BBC Microcomputer – A Concise Guide
Graham Leah
Sorting Routines for Microcomputers
Keith McLuckie and Angus Barber
Beginning BASIC with the ZX Spectrum
Judith Miller
Using Sound and Speech on the BBC Microcomputer
Martin Phillips
File Handling on the BBC Microcomputer
Brian J. Townsend
Good BASIC Programming on the BBC Microcomputer
Margaret White

Other Books of Related Interest

Advanced Graphics with the IBM Personal Computer
Ian O. Angell
Advanced Graphics with the Sinclair ZX Spectrum
Ian O. Angell and Brian J. Jones
Micro-Maths
Keith Devlin
Beginning BASIC
Peter Gosling
Continuing BASIC
Peter Gosling
Practical BASIC Programming
Peter Gosling
Program Your Microcomputer in BASIC
Peter Gosling
More Real Applications for the ZX81 and ZX Spectrum
Randle Hurley
The Commodore 64 in Action
M. M. Novak
Computer Literacy: A beginners' guide
Vincent Walsh

Sorting Routines for Microcomputers

Keith McLuckie
and
Angus Barber

M
MACMILLAN

© Keith McLuckie and Angus Barber 1986

All rights reserved. No reproduction, copy or transmission of this publication may be made without written permission.

No paragraph of this publication may be reproduced, copied or transmitted save with written permission or in accordance with the provisions of the Copyright Act 1956 (as amended).

Any person who does any unauthorised act in relation to this publication may be liable to criminal prosecution and civil claims for damages.

First published 1986

Published by
MACMILLAN EDUCATION LTD
Houndmills, Basingstoke, Hampshire RG21 2XS
and London
Companies and representatives
throughout the world

Typeset by
TecSet Ltd, Sutton, Surrey

British Library Cataloguing in Publication Data
McLuckie, Keith

Sorting routines for microcomputers.—
(Macmillan microcomputer books)

1. Sorting (Electronic computers) 2. Micro-
computers

I. Title II. Barber, Angus

001.64'25 QA769.S/

ISBN 978-0-333-39587-5

ISBN 978-1-349-08147-9 (eBook)

DOI 10.1007/978-1-349-08147-9

Contents

<i>Preface</i>	<i>vii</i>
<i>Acknowledgements</i>	<i>x</i>
1 Linear Selection Sorts	1
Linear Selection Sort	1
Linear Selection with Counting	9
Linear Selection with Exchange	15
2 Basic Exchange Sorts	23
Pair Exchange Sort	23
Standard Exchange Sort	30
Sifting Sort	36
3 Shell's Sort	44
4 Binary Tree Sort	58
5 Tournament Sort	68
Tournament Sort	68
Heapsort	81
6 Quicksort	87
Quicksort	87
Hibbard's Quicksort	93
7 Improving the Quicksort	100
Hoare's Quicksort	100
Singleton's Quicksort	102
Quickersort	103
Quick Quickersort	106
8 Sorting by Insertion	111
Straight Insertion Sort	111
Binary Insertion Sort	117
Centred Insertion Sort	125

9	Distributive Sorts	132
10	Relative Merits	142
11	Merges	148
12	Sort Systems	165
	<i>Index</i>	<i>169</i>

Preface

This book is the result of an abiding interest on the part of the authors in sorts and their relative performances. Generally the speed of a sort is rarely quoted in quantitative terms, and when it is, then the sort's performance against a number of single characters is given. In the bench tests provided with each sort in this book, the performance of the sort is measured using a 'useful' key length of 16 characters.

The sort is possibly one of the most widely used procedures in computing (one estimate states that 25 per cent of all computing is sorting!), and machines that spend a lot of their 'data-processing' day 'sorting' may not be running as efficiently as they might. It would appear from our investigations that sort-rates of two or three hundred thousand keys per hour are perfectly possible on quite modest microcomputers. Our hope is that this book will provide some practical help to programmers in this direction, as well as explaining the fundamentals of the sort.

The choice of machine upon which to run these tests was not significant. It was in fact an ACT Sirius running under MSDOS. The machine is relatively slow, and in any case the programs were run in interpreted BASIC. The essential requirement was that the times produced by the bench tests were directly comparable.

All timings are given in seconds.

Throughout the book we may be guilty of using technical language. We feel, however, that most of it should be familiar to the readership. A couple of exceptions may be: *COMPERAND*, which means 'the item to be compared with or against' (it is used in much the same way as 'operand' and 'operator' are), and *collation order* or *collating sequence*, which is an attribute based on the ASCII value of the characters used, and is the order used to place elements in their ranked positions. This means that the value of any character is determined by its coded representation and this in turn determines the particular element's ranked position.

We should also say a little about the program listings that are provided as part of each chapter. The first listing the reader will find is written in MICROSOFT basic, and after this is provided a BBC BASIC program and a SPECTRUM BASIC program. Exceptionally there is a PASCAL listing; this is used because the algorithm requires recursion and PASCAL does not place any limit (for practical purposes) on the depth of recursion. The programs have been written primarily for speed and in some cases this may be at the expense of clarity, although we have generally avoided the use of multi-statement lines, in spite of the fact that

such use might have saved a little time. Additional listings provided at the end of chapters are the optimised versions.

In order to execute programs given in the book, the reader should: (a) load data into the array $D\$ (n)$ excluding element $D\$ (0)$, (b) specify a value for $N\%$ in the program (that is, the number of items to be sorted).

The Origins of Sorting

Sorting began as a large-scale operation in the US Census Bureau. Every ten years the United States takes a census, and this began last century, but by 1880 the amount of data that needed to be processed was becoming too great and in fact the 1880 census was never completed. Herman Hollerith, then a 20 year old employee of the Census Bureau, invented an ingenious electric tabulating machine of which about 100 were used to successfully complete the 1890 census. The machine accepted cards that had holes punched in them in positions relating to the information to be stored on them. The machine then placed the cards in pigeon-holes, depending on the position of the holes (that is, it sorted them). In addition to sorting the cards the machine kept a tally of the number of cards in the various pigeon-holes, displayed on clock-like dials. These early card sorters were very successful and one man reportedly processed 19 071 cards in a six and half hour day.

Unfortunately by the 1900 census Hollerith's original tabulators were too slow and he had to design a second machine. The new machine had an automatic card feed and was the predecessor to modern card sorters. Hollerith went on to form a company that subsequently became IBM.

The concept of merging also came from an idea for a card processing machine. The Collator was invented in 1938 and had two feeding stations and could merge two decks of cards in one pass; this type of merge is known as a 'straight merge'.

Computers were soon to arrive and there is evidence that a sorting routine was the first ever program written for a stored program computer. The designers of EDVAC, one of the first mass-produced computers, were especially interested in sorting because it epitomised the potential non-numerical applications of computers. John von Neumann in 1945 prepared programs for internal sorting in order to test the adequacy of the instruction code he was suggesting. The merits of his proposed computer organisation could be compared with the special-purpose sorting machines already on the market.

Because of the limited memory available on early computers a lot of thought went into external sorting as well as internal sorting. It was pointed out in an article published by two members of the EDVAC design team that a computer augmented with magnetic tape could simulate the action of card sorting equipment but achieve a much faster speed. J. W. Mauchly, co-author of the above article, at a special session on Computing at the Moore School in 1946 presented a lecture on 'Sorting

and Collating' and his notes represent the first published discussion of computer sorting.

Shortly after 1946 J. P. Eckert and Mauchly went into business together and their company produced some of the earliest computers: the BINAC (for military use) and UNIVAC (for commercial use). The US Census Bureau received the first UNIVAC; such is population growth! At this time, although computers could sort faster than card machines, they were much costlier and it was not clear whether their use was economical. The UNIVAC programmers therefore put considerable effort into the design of efficient external sort routines. According to their work 100 million 10 word records could be sorted on the UNIVAC in 9000 hours or 375 days!

When the UNIVAC I was released an efficient external sorting routine was developed for it which used six input buffers to provide a perfect interface between the computer and the magnetic tape used as input and output. The culmination of this work was a sort generator, which was the first major software package ever developed for automatic data processing. The user could specify record size and the positions of up to five keys in partial fields of each record, and the sort generator would produce a copyrighted sorting program for one-reel files.

By 1952 many approaches to internal sorting were well known, but little theory had been developed on the subject. Several noteworthy theses were written about this time. These included even a Ph.D thesis written by Howard B. Demuth in 1956 on 'Electronic Data Sorting'. This work helped lay the foundations of computational complexity theory.

After Demuth's paper, although not as a direct result of it, many sorts were written: address calculation (1956), merge insertion (1959), radix exchange (1959), cascade merge (1959), Shell's sort (1959), polyphase merge (1960), tree insertion (1960), oscillating sort (1962), Hoare's quicksort (1962), William's heapsort (1964), and Batcher's merge exchange (1964). After this extensive sort development, the 1960s saw the corresponding development of the theory.

The importance of sorting can be seen by the fact that it has been linked with most of the major developments in computers: the first data processing machine, the first computer, the first stored program, the first software, the first buffering methods, and the first work on algorithmic analysis and computational theory.

On the lighter side, it may interest the reader to know that the Chinese sort names on the number of brush strokes in the surname.

Acknowledgements

The authors would like to thank Longfield Upper School for its provision of facilities. Thanks are also due to Lesley Lee and her husband for their help in providing valuable reference material, to Jean Chaston of Longfield School Maths Department, for her help in establishing the statistical validity of the test data and finally to Eleanor Barber and Deborah McLuckie for their interest and encouragement.