Faster than a speeding bubblesort . . . it's Shellsort!

Whilst bubbles just drift along in a sort routine, the Shellsort really speeds things up — as much as five times!

A. Daviel

IF YOU RUN Alphasort as we published recently, you will notice that, though it is very fast indeed for short lists, it takes disproportionately longer for large lists. To be exact the time taken is proportional to N^2 for a random list of N records.

The following program, though somewhat more complex, takes a time proportional to $N\log_2(N)$. To give an idea of what this means, suppose it is necessary to sort a government census file of one million records. If, on a given machine, the Bubblesort and Shellsort each take 20 milliseconds to sort 64 records, the Shellsort would take half an hour to sort the census file. The Bubblesort — four months! On a more down-to-earth application the Shellsort will sort 250 records in one-fifth of the time.

Using the Shellsort

The example program, written in Microsoft BASIC, illustrates how the routine is driven. You will observe that the routine does not actually sort the array, but instead returns the array SP as a pointer into the string array. This may sound unduly complicated at first, but by using this method it is not necessary to move the records around (which may be quite long) more than once. In the example program, the records being sorted consist of a single string, but in general they will consist of mixed string and numeric fields. The list will be sorted according to one of these fields, called the key. A$ in line 8250 of Shellsort would be replaced by this field.

How it works

The general principle is that of merging two sorted sequential files — the method by which files are sorted which will themselves not fit into main store. The next record in the merged file is selected from the input file with the record next in order; see Figure 1. To apply this technique to an array of strings, the array is first split into N lists of one string each. These are merged in pairs to give N/2 lists of two strings, and so on until the whole array has been sorted; see Figure 2. You will note N has to be a power of two, so that in the example program the array is preset to an artificially high value.

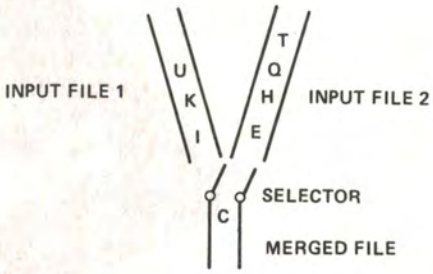

Figure 1. **Using selection to merge files.**

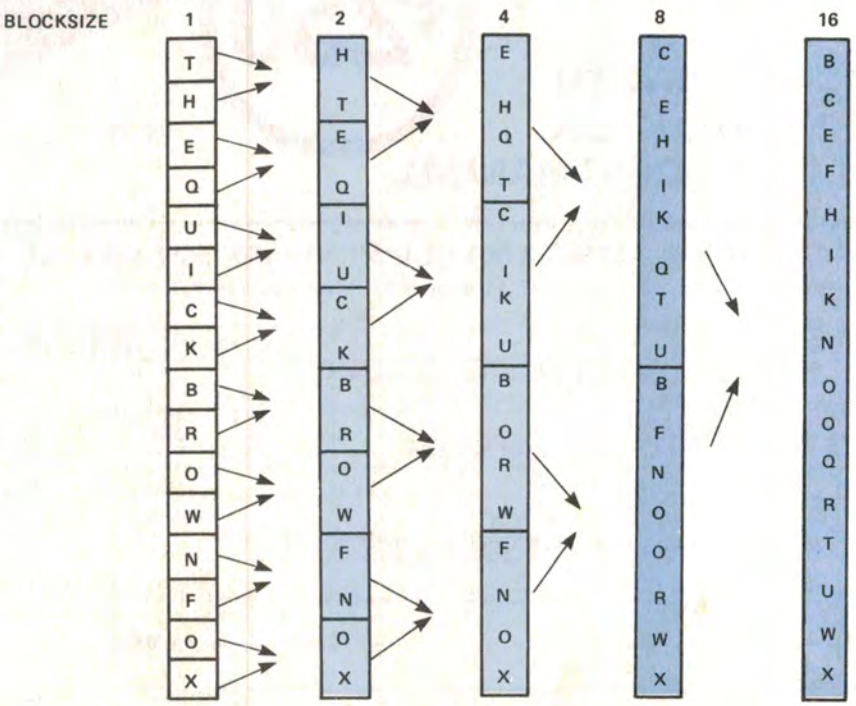

Figure 2. **How the files are sorted and merged.**

Program listing

```
100 PRINT" [CLS ] STRINGSORT"
110 REM INITIALISATION
130 DIM A$(255) : EN = 255 : CT = 0
132 REM PRESET A$ TO MAXIMUM
135 I = 0 TO 255 : A$(I) = CHR$(255) : NEXT
140 PRINT"PLEASE INPUT NAMES,WHEN YOU
    ARE"
150 PRINT"READY TO SORT TYPE'*'"
160 PRINT
170 REM INPUT ROUTINE
180 PRINT"YOU HAVE ROOM FOR ";EN;"MORE EN-
    TRIES"
190 INPUT A$(CT)
200 IF A$(CT) = "*" THEN 250
210 CT = CT + 1 : PRINT" [CLS ]"
220 IF CT > 254 THEN 250
230 EN = 255 – CT : GOTO 180
240 END
245 REM SET NUMBER OF ELEMENTS & CALL
    SHELLSORT
250 SN = CT – 1 : GOSUB 8100
470 REM LINE LOOP OUTPUT
475 FOR KK = 0 TO 9:GET K$:NEXT
480 PRINT"HIT A KEY FOR LIST"
490 GET K$ : IF K$ = " " GOTO 490
510 LP = 0 : SL = 18
520 FOR P = LP TO LP + SL
525 IF P > CT THEN END
527 REM ARRAY SP GIVES POINTER INTO A$
530 PRINT A$(SP(P))
540 NEXT P
545 FOR KK = 0 TO 9:GET K$:NEXT
547 PRINT
550 PRINT" **HIT ANY KEY TO CONTINUE**"
560 PRINT" **'$' WILL BREAK**"
565 PRINT
570 GET K$ : IF K$ = " " THEN 570
580 IF K$ = "$" THEN END
590 LP = LP + SL + 1
600 GOTO 520
8000 REM SHELLSORT ROUTINE
8010 REM THIS ROUTINE WILL SORT A
8020 REM STRING ARRAY A$
8030 REM INTO ASCENDING ORDER.
8035 REM THE ROUTINE RETURNS SP(SN – 1)
8040 REM AS A POINTER ARRAY INTO THE
8045 REM LIST. THE ROUTINE USES
8050 REM VARIABLES PREFIXED 'S'.
8070 REM THE ROUTINE TAKES ABOUT 2 MIN.
8075 REM TO SORT A 256 ELEMENT LIST
8080 REM THE TIME TAKEN IS PROPORTIONAL
8085 REM TO 2*LOG2(N)*N.
8100 SS = (INT(LOG(SN)/LOG(2)) + 1)
8110 SN = 2↑SS:REM SN MUST BE A POWER OF 2
8140 REM SS IS NO OF STEPS
8150 DIM SP(SN) : DIM SQ(SN)
8160 FOR SI = 0 TO SN – 1
8170 SP(SI) = SI : NEXT SI
8180 SB = 1 : SP = 1:REM BLOCKSIZE,STEP NO
8190 IF S > SS THEN RETURN
8195 PRINT" [CLS ]SORTING: BLOCKSIZE = ";SB
8200 SJ = SN/2:SI = 0:SK = 0
8210 SL = SB + SI : SM = SB + SJ
8220 IF(SJ > = SM)AND(SI > = SL) GOTO 8300
8230 IF(SJ > = SM) GOTO 8260
8240 IF(SI > = SL) GOTO 8280
8250 IF A$(SP(SI)) > A$(SP(SJ)) GOTO 8280
8260 SQ(SK) = SP(SI)
8270 SK = SK + 1 : SI = SI + 1 : GOTO 8220
8280 SQ(SK) = SP(SJ)
8290 SK = SK + 1 : SJ = SJ + 1 : GOTO 8220
8300 IF SK > SN – 1 GOTO 8320
8310 SM = SM + SB : SL = SL + SB: GOTO 8220
8320 FOR SI = 0 TO SN – 1
8330 SP(SI) = SQ(SI): NEXT
8340 SB = SB*2 : SP = SP + 1
8350 GOTO 8190
```