Cube Root of Two Sort - fast sort

sort

sortoff.zip

If the platform you are using doesn't support double-precision floats, do not use the sort routine shown below - it fails to sort correctly and there are better ones.

I found this sort algorithm years ago on Saxon BBS as a Public Domain (remember that?) contribution.

It was written in Microsoft GW Basic and I converted it to Quick Basic and later VB 5&6. It has proved to be the fastest sort when programmed in Basic in similar fashion - i.e. not using machine code sorts which will inherently be fast.

There are several methods of sorting and two of the most popular are the Bubble sort and the Shell sort. Bubble sorts are simple to remember and always work but they have to go through the entire range over and over again which makes them very slow. Shell sorts are usually four or five times faster than a Bubble sort.

Before choosing a sort algorithm, read this article.

This routine is faster than a shell sort and uses the Cube Root Of Two (1.2599211^3=2) to balance and seed the sort intervals. I have used it time and again and it is fast - even on huge data sets (10 million items - on PCs and servers, not a 'Mite!), even to the point that it'll have you checking because you won't believe it did it.

Caveats: Unfortunately CR2Sort is not immune to caveats, but they are trivial:

- It ignores element zero (i.e. OPTION BASE 0)
- It won't sort data-sets of less than six items. To get around this, you could have CR2 and a Bubble sort in your code and check the size of the data-set to determine which sort to use. This is not as mad as it sounds; Small data-sets sort in “human-imperceptible” times but the advantages of CR2 come in early; at 16 elements, CR2 is already twice as fast as a Bubble with the same data. The code overhead of two routines is small and the time taken to decide which to use is not even worth discussing. Something like this (I'd likely have a single sort Sub which made this choice for me):

```
IF my_array_top >5 THEN
  CR2SORT my_array_top
ELSE
  BBLSORT my_array_top
END IF
```

Attached is a CSV file with the data used to generate the graph below together with the code. The bubble sort was modified to make a fair comparison (it was slowed by use of a sub to perform swaps) but it is essentially the same code as the bubble sort here: Bubble Sort (part of the original MMBasic library). For completeness, here is a Shell Sort in MMBasic.

Assumptions:

```
  Option Base 1
```

```
    Dim xx As Integer
    xx=my_arraysize     'lack of UBOUND() means we have to manually track
the array size
    Dim SP$(xx)
```

Usage:

```
  CR2Sort my_arraysize
```

The Routine: It uses the string array SP$ and you must pass in the number of elements to sort (this is usually the whole array size but you might only want to sort the first 20 or whatever...

```
    Sub CR2SORT(SPtop)
        Local INTEGER i,x,b
        Local FLOAT p,y
        Local STRING a$
        p=1.2599211: x=2: b=1: y=SPtop\2

        While y>2: y=y/p: Wend

        While x>1
            y=y*p
            x=SPtop\y
            For i=1 To SPtop-x
                If SP$(b+i)>SP$(x+i) Then
a$=SP$(b+i):SP$(b+i)=SP$(x+i):SP$(x+i)=a$ 'SWAP SP$(b+i),SP$(x+i)
            Next
        Wend
    End Sub
```

It probably doesn't compare with CSubs that do a sort with machine code but it is a very fast sort routine and if you need to tweak it in any way (to sort multiple arrays etc) then its a great little routine.

I did a comparison between this and the Bubble sort, with a fairly random-ish array of 100 strings (code is attached above if you want to play), it was nearly seven times faster with 100 elements! This is the result of that code running on a 48MHz MicroMite with MMBasic V5.2.

The graph is interesting because beyond the time differences for varying sort sizes, it shows two things: 1. The smooth manner in which CR2 times are fairly linear to data-set size due to performing fewer compares and swaps. 2. The typical huge surges in Bubble sort times when it is “startled” by having to move strings long distances in an already well-sorted data-set.

the original text file

CR2SORT.DOC

```
'
```

```
   'CR2SORT DOCUMENTATION---6/19/1988
'
'
'
' THIS SORT IS A RESULT OF AN INTEREST I DEVELOPED IN SORT PROGRAMS
'AND IS MY ATTEMPT AT WRITING AN EFFICIENT ALGORITHM.
'
' THE VALUE USED TO DETERMINE COMPARE/SWAP INTERVALS IS THE CUBE
'ROOT OF 2, HENCE THE NAME 'CR2SORT'.
'
'THE SORT FUNCTIONS IN THE FOLLOWING MANNER
'
'1. THE LIST TO BE SORTED IS READ INTO AN ARRAY WITH (T) BEING EQUAL TO
'THE TOTAL NUMBER OF ELEMENTS IN THE ARRAY.
'
' 2. THE TOTAL (T) IS DIVIDED BY 2 (INTEGER DIVISION) THEN REPEATEDLY
'DIVIDED BY (P#) THE 'CR2' UNTIL THE RESULTANT (Y#) IS LESS THAN 2.
'THIS IS ACCOMPLISHED BY A WHILE/WEND LOOP. THIS IS THE SEED NUMBER
'WHICH IS USED TO BALANCE THE SORT.
'
' 3. THIS STARTING NUMBER (Y#) IS MULTIPLIED BY (P#) THE 'CR2' AND DIVIDED
'INTO THE TOTAL NO. OF ELEMENTS (T), TO OBTAIN THE VALUE (X) FOR THE
'COMPARE/SWAP INTERVAL.  THIS VALUE IS OBTAINED BY INTEGER DIVISION.
'
' 4. THE INTEGER X IS THEN USED IN A FOR-NEXT LOOP FOR COMPARING B+I
'AND X+I ELEMENTS UNTIL THE (I) COUNT REACHES T-X. THE VALUE FOR X
'DECREASES AS STEP 3 IS REPEATED UNTIL X REACHES 2 AND ADJACENT
'ELEMENTS ARE BEING COMPARED. THIS IS ACCOMPLISED WITH A WHILE/WEND
'LOOP.
'
' BECAUSE THE STARTING NUMBER IS DETERMINED BY THE LENGTH OF THE
'ARRAY, A PERFECT SORT IS OBTAINED WITHOUT SETTING A FLAG TO CHECK IF
'A SWAP HAS BEEN MADE AND MAKING ADDITIONAL LOOPS TO COMPLETE THE SORT.
'
' I HAVE FOUND THIS ALGORITHM TO BE VERY EFFICIENT AND TO HAVE
'EXCELLENT SPEED WITH A MINIMAL NUMBER OF SWAPS PERFORMED.  THIS SORT
'IS 17 OR MORE TIMES FASTER THAN A COMPARABLE BUBBLE SORT AND IS ALSO
'FASTER THAN
'THE SHELL SORT THAT I TESTED IT AGAINST. IT MADE FEWER COMPARES AND
'HAD BETTER RUN TIMES.  ON A TEST LIST OF 5322 STRING ELEMENTS IT MADE
'52 PERCENT AS MANY SWAPS AND RAN IN 29 PERCENT OF THE TIME THAT IT
'TOOK TO SORT THE LIST WITH A VERSION OF THE SHELL SORT THAT I HAVE.
'
' I CHECKED THIS SORT ON MANY LISTS AND FOUND NO ERRORS. I USED TEST
'LISTS OF COMPUTER GENERATED NONSENSE WORDS FROM 2 TO 8 CHARACTERS LONG
'WITH SUCCESIVE NUMBERS FOR THE INTEGER ARRAY.
'
' IT IS A VERY SHORT ROUTINE AND SIMPLE TO WRITE INTO YOUR OWN
'PROGRAMS OR TO USE AS A SUBROUTINE.
'
' THIS PROGRAM MAY BE COPIED FOR PERSONAL USE ONLY AND MAY NOT BE
```

```
'USED AS A PART OF OTHER MARKETABLE PROGRAMS WITHOUT PERMISSION.
'
' IF UPLOADED TO ANOTHER BULLETIN BOARD OR PASSED ALONG TO SOMEONE
'ELSE, PLEASE KEEP CR2SORT.DOC TOGETHER WITH THE CR2SORT PROGRAM IN
'THE CR2SORT.ARC VERSION. (couldn't support defunct archive format - sorry)
'
'DAVID J. HANSON
'1213 SOUTH 21ST STREET
'FARGO, NORTH DAKOTA 58103
'
```

CR2SORT.BAS

```
'
'SUB CR2SORT (W$(1),N%(1),T%) STATIC
'P#=1.2599211: X=2: B=1: Y#=T\2: WHILE Y#>2: Y#=Y#/P#: WEND
'WHILE X>1: Y#=Y#*P#: X=T\Y#
'    FOR I=0 TO T-X
'      IF W$(B+I)>W$(X+I) THEN SWAP W$(B+I),W$(X+I):SWAP N%(B+I),N%(X+I)
'    NEXT I
'WEND
'END SUB
```