THE GROUP SORTING ALGORITHM – THE ALGORITH WHICH PERFORMS NO MORE 49 COMPARISONS FOR 16 ITEMS.

Kirill Ivanov

Brest State Technical University, Moskowskaja 267, Brest 224017 Belarus, inbox2000@inbox.ru

February 28, 2006

Abstract: *Today there is a lot of sorting algorithms which people apply in dataprocess – to order data. The sizes of processed data are huge so it's very important to make this processing more quicly. That's why main goal of sorting algorithms to perform less comparison operations for sorting data. Today the best algorithm performs 60 items for 16 items.*
And I would like to propose the algorithm which performs from 32 to 49 comparisons for 16 items.

Group Sorting Algorithm

Let's touch to the Group Sort with the example on Figure 1. There is a sequence of 5 different integer items (from 1 to 5) at random order. The figure has comments of every performing action. After the figure there is an overview of which steps an iteration consists.

Iteration number	Comparison number	Sorting sequence					Comments
i = 1		5	3	2	4	1	It's how the sequence looks before (sorting) the 1st iteration. The one is divided into groups of 2^i items in each.
1	1	**5**	**3**				The (1st) group is divided into 2 equal parts: left and right. Coparing : the 1st left part item and the 1st right part item.
		3	5	2	4	1	Since the right item is less the left one so the 1st one was inserted before the left item. So the sequence looks like...
1	2			**2**	**4**		The (2nd) group is divided into equal left and right parts. Comparing items: 1st left part with the 1st right part.
		3	5	2	4	1	It's the sequence after sorting the 2nd group where the right item wasn't inserted before 2 (because 4 is NOT less 2).
i = 2		3	5	2	4	1	It's how the sequence looks before the 2nd iteration. The sequence is divided into groups of 2^i items in the group.
2	3	**3**	5	**2**	4		Comparing: the 1st item of the left part with the 1st item of the right part. (2 < 3) so 2 will inserted to the left before 3.
2	4	2	**3**	5	**4**		Since the 1st right item is NOT less the left one (4 !< 3) so the right item (4) will compared with the next left one (5).
2	5	2	3	**5**	**4**		Since our right item is less the next left one (4 < 5) so the right one will be inserted before its greeter left opponent.
		2	3	4	5		It's the sequence after sorting the 1st (lonely) group. *We sort the groups where the number of items is greeter* 2^{i-1} !!!
i = 3		2	3	4	5	1	It's how the sequence looks before the 3rd iteration. The sequence is divided into groups of 2^i *items* in the group.
3	6	**2**	3	4	5	**1**	*Tthe operations for each group are the same*! The sort in group completes when *there is no the next item to compare*
		1	2	3	4	5	It's the sequence after sorting the 1st (lonely) group. The sorting completes when ALL(!!) items are ONE group to sort.

Figure 1. The example of sorting with the group sort algorithm (with commnets)

Each iteration consists of the next steps:

Step1. Group division.

The sequence is divided into groups of 2^i (i – the number of iteration) items in *each* (if it is *the only one* or *last group* of the sequence then the number of items in such group may be less – see at the 6th comparison on Figure 1).

The number of items in the last group = the all items of the sequence ***mod*** 2^i

The number of items in the only one group = the all items of the sequence

Step2. Parts division.

Each group is divided into 2 equal parts: 'left' and 'right'. *Each* part consists of 2^{i-1} items (if it is *the only one* or *last group* of the sequence then the number of items in each part may be less – see at the 6th comparison on Figure 1). Any more: if the number of items in such *last* group is not greeter then 2^i so *it is needn't to sort it* (the 1-6th comparisons on Figure 1 brightly show it). And if it is sorted then the number of items is such group is divided into parts like:

left part = 2^{i-1}*; right part = all the items of the group – left part.*

Step3. Group Sort rule.

The sort of each group begins with the applying the *rule* to the 1st item of the left part and to the 1st items of the right part. And it sounds like:

if the item of right part is less the item of the left part
then the right one insert before the left item.

If the rule antecedent (if part) is *true* then after performing the rule consequent (than part) to apply this rule to *the next* item of the right part (and this item it's always the first in his part) and the same item of the left part.

If the rule antecedent (if part) is *false* then to apply this rule to the same item of the right part and *the next* item of the left part.

Step 3 completes when *it needs to apply the rule to the next item but the current item is the last (so there is not 'the next item' already)*!

Step4. The end of the sort.

The sort of the sequence completes when the iteration has the only one group (see at the 6th comparison on Figure 1!