1 Tournament sort algorithm

This algorithm is obviously one of ways to sort the data. Nevertheless, description here will be given from a little bit different angle. One of the ideas of using this algorithm is to apply it, when we have shared resources that can perform operations in parallel.
This is the situation e.g. for Teradata database technology. Briefly speaking, engine of this database in constructed in such a way, that portions of data are kept and managed across so called AMPs (Access Module Processor). So, if we have one table, it simply means that its rows are distributes across these AMPs.
Hence, if we want to select such table with data sorted, it means that such application has to be done, where all AMPs will make part of the job. Here is where tournament sort comes.
Let us look at following example. Assume we have table with nine elements, from 1 to 9. This data is placed unordered across three AMPs, which looks as follows:

AMP 1	AMP 2	AMP 3
$[1, 7, 4]$	$[2, 8, 5]$	$[9, 6, 3]$

In our tournament algorithm, first step is done in parallel, namely each AMP sorts its own portion of the data, which leads to following result:

AMP 1	AMP 2	AMP 3
$[1, 4, 7]$	$[2, 5, 8]$	$[3, 6, 9]$

In second step, each AMP returns first element to a middle layer, which then takes smallest element ouf of them. In our case this middle layer will receive elements $1, 2, 3$ from which it returns 1 as first (sorted) element.
In subsequents steps, this AMP, whose element was returned, will provide next element from its list.

[1,4,7] [2,5,8] [3,6,9]
$[1, 2, 3] \rightarrow$ $[1]$

[4,7] [2,5,8] [3,6,9]
$[4, 2, 3] \rightarrow$ $[1, 2]$

[4,7] [5,8] [3,6,9]
$[4, 5, 3] \rightarrow$ $[1, 2, 3]$

This goes on till all elements from all AMPs are taken. There are two things that are worth to underline. First one, this is only a general idea, how this approach works. Second, this algorithm can also be implemented in a programming langauge, most commonly with usage of tree structure, where elements are compared in pairs. In this case, complexity of such algorithm would be $O(n \log n)$.