Bubble Sort Algorithm

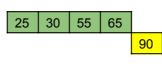| raw data	65	90	30	25	55

65	90	30	25	55
65	90	30	25	55
30	90	65	25	55
30	90	65	25	55
25	90	65	30	55
25	90	65	30	55

25

90	65	30	55
65	90	30	55
65	90	30	55
30	90	65	55
30	90	65	55

25	30

90	65	55
65	90	55
65	90	55
55	90	65

25	30	55

90	65
65	90

25	30	55	65

90

25	30	55	65	90

Rank Sort Algorithm

Count, for each list element x, how many list elements y fulfill x>=y

raw data	65	90	30	25	55	35	80	60	75	15	85	50	70	45	20	40

rank	11	16	4	3	9	5	14	10	13	1	15	8	12	7	2	6
pos[rank]	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

for rank from 1 to 16: print raw_data[pos[rank]]:

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
pos[rank]	10	15	4	3	6	16	14	12	5	8	1	13	9	7	11	2
	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90

Rank Sort Algorithm

raw data	65	90	30	25	55	35	80	60	75	15	85	50	70	45	20	40

rank	11	16	4	3	9	5	14	10	13	1	15	8	12	7	2	6

Parallelization of the process (for 4 processes):

We devide the list into 4 chunks
Each process receives the complete list from the master.
Each process determines the ranks of the number corresponding to its chunk.
Each process send the ranks of the numbers in its chunk to the master.

Process 0

65	90	30	25	55	35	80	60	75	15	85	50	70	45	20	40
65	90	30	25												
11	16	4	3												

Process 1

65	90	30	25	55	35	80	60	75	15	85	50	70	45	20	40
				55	35	80	60								
				9	5	14	10								

Process 2

65	90	30	25	55	35	80	60	75	15	85	50	70	45	20	40
								75	15	85	50				
								13	1	15	8				

Process 3

65	90	30	25	55	35	80	60	75	15	85	50	70	45	20	40
												70	45	20	40
												12	7	2	6

Process 0 The master collects the chunks of ranks it received.

11	16	4	3	9	5	14	10	13	1	15	8	12	7	2	6

The master presents the ordered list in the sequence of the ranks

65	90	30	25	55	35	80	60	75	15	85	50	70	45	20	40
11	16	4	3	9	5	14	10	13	1	15	8	12	7	2	6

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90