28.01.20

Prefix computation

To compute n nos. on 1 processor, complexity = $O(n)$

Suppose nos. $n = 8$, processors $p = 8$.

I/p : 12, 3, 6, 8, 11, 4, 5, 7.

$\{P_1 \; P_2\}$ $\{P_3 \; P_4\}$ $\{P_5 \; P_6\}$ $\{P_7 \; P_8\}$

12 3 6 8 11 4 5 7.

clubbing two processors in each step:

12 (5) 6 14 11 (15) 5 12

12 15 21 (29) 11 15 20 27

12 15 21 29 40 44 49 56

Now time = 3 units.

∴ Total time = $\log n$.

We are reducing the time from n to $\log n$ when we are using n no. of processors.

$$\text{Speedup} = \frac{O(n)}{O(\log n)}$$

$n \times O(\log n)$

Total work done = $O(n \log n)$

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{No. of processors used}} = \frac{O(n)}{n O(\log n)} = \frac{O(n)}{O(n \log n)}$$

$$= O\left(\frac{1}{\log n}\right)$$

So, this algorithm is not work optimal.

For work optimal algorithms, efficiency = $O(1)$ we use $n/\log n$ processors for this

We need to assign log n elements to each processor.

Work optimal logarithmic time prefix computation

Step 1. Processor $i = 1, 2, \ldots n/\log n$

$x_{(i-1)\log n+1}, x_{(i-1)\log n+2}, \ldots, x_{i\log n}$ $\longrightarrow O(\log n)$

Results = $z_{(i-1)\log n+1}, z_{(i-1)\log n+2}, \ldots, z_{i\log n}$

Step 2. Total of $n/\log n$ processors employ to compute prefix $x_{\log n}, x_{2\log n}, x_{3\log n}, \ldots x_n$. Let $w_{\log n}, w_{2\log n}, w_{3\log n}, \ldots w_n$ be the result. $\longrightarrow O\left(\log\left(\frac{n}{\log n}\right)\right)$

Step 3 Outputs! $w_{(i-1)\log n} \oplus x_{(i-1)\log n+1}, w_{(i-1)\log n} \oplus x_{(i-1)}$

$\longrightarrow O(\log n)$

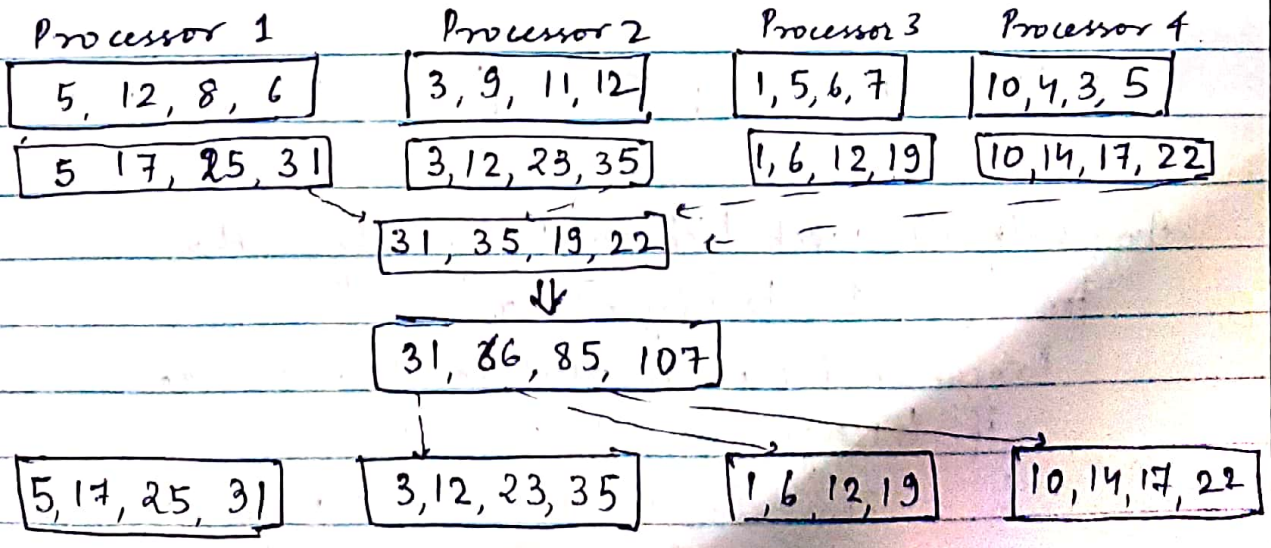
$\therefore$ Total = $O(\log n)$

Eg. 5, 12, 8, 6, 3, 9, 11, 12, 1, 5, 6, 7, 10, 4, 3, 5

$n = 16$ $\quad \oplus \Rightarrow$ Addition

$p = \frac{n}{\log n} = \frac{16}{4} = 4$

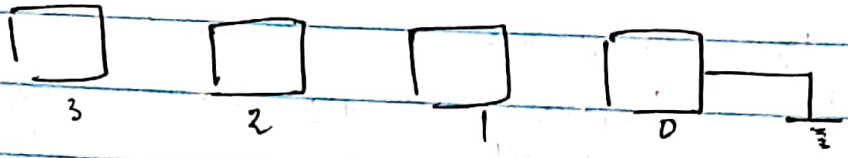
$\therefore n = 16, \quad p = 4$



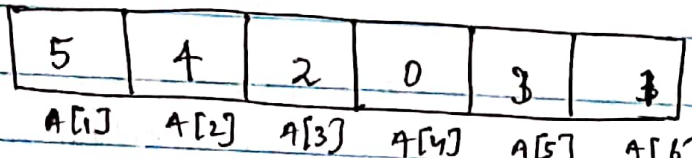
5, 17, 25, 31 | 34, 43, 54, 66 | 67, 72, 78, 85 | 95, 99, 102, 107

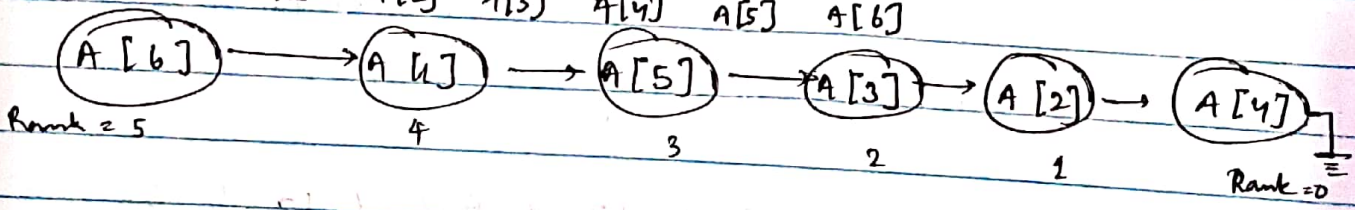
List ranking problem

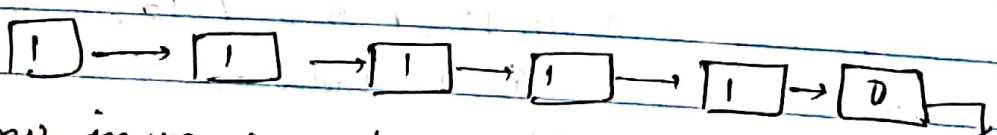
Problem is to find the rank of a node in the list.



Maintain info abt the neighbour

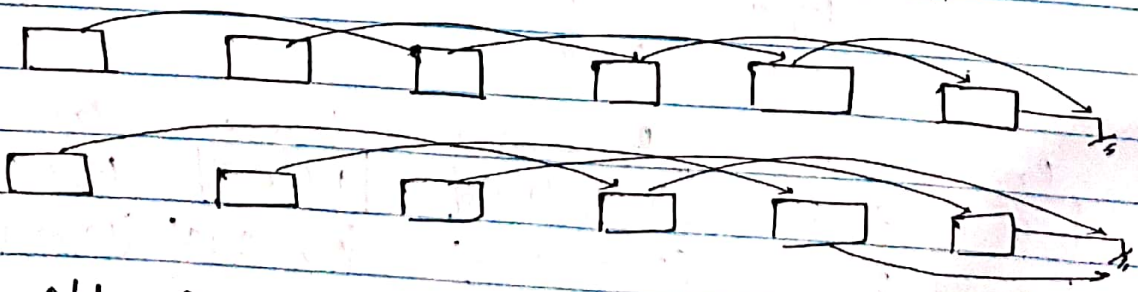






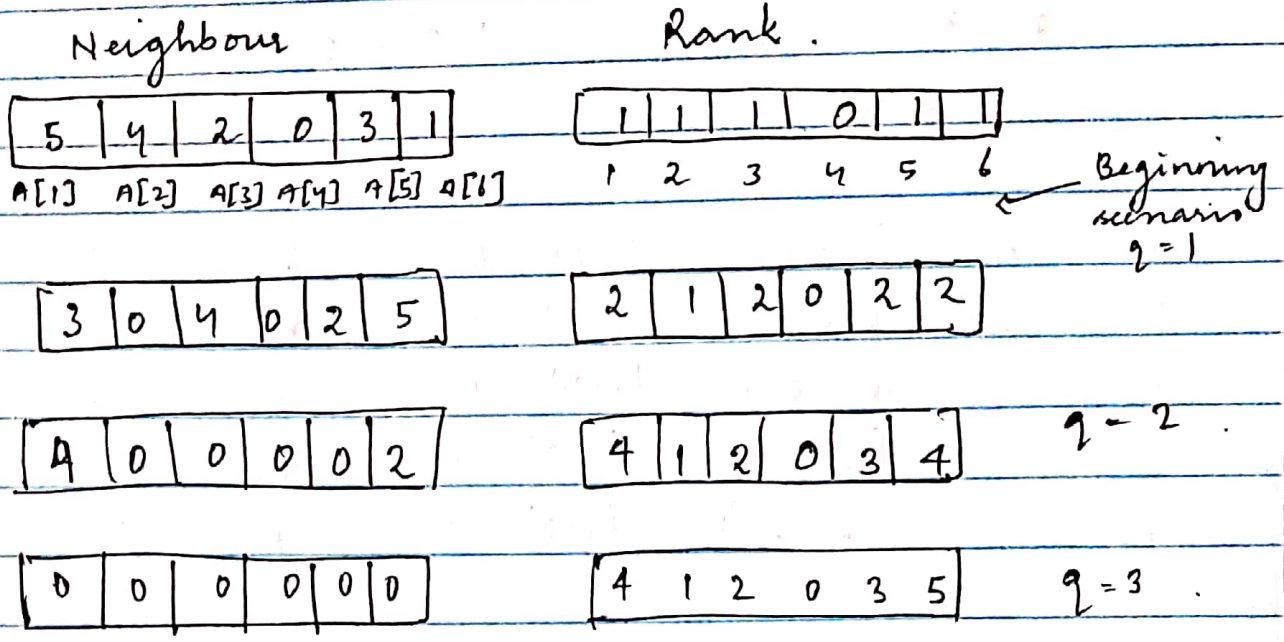
log n
2^1
2^2

Jump in powers of 2 i.e. $2^1, 2^2, 2^3, \ldots$



An O(log n) work list ranking algorithm.

```
for q := 1 to [log n] do
  Processor i ( in parallel for 1 ≤ i ≤ n) does:
    if ( Neighbour[i] ≠ 0 ) then
    {   Rank[i] := Rank[i] + Rank[Neighbour[i]];
        Neighbour[i] := Neighbour[Neighbour[i]];
    }
```



This is also not a work optimal algorithm.

04.02.20

The problem of selection.

Maximum selection with n^2 processors $\rightarrow O(1)$

n keys $\quad 1 \le i \le n$

$k_1, k_2, \ldots k_n$ ith smallest key.

Step 0: If $n = 1$, output the key

Step 1: Processor p_{ij} (for each $1 \le i, j \le n$ in parallel) compute $x_{ij} = (k_i < k_j)$

Step 2: The n^2 processors are grouped into n groups $G_1, G_2, \ldots G_n$ where G_i $(1 \le i \le n)$ consists of the processors $p_{i1}, p_{i2}, \ldots p_{in}$. Each group G_i computes the boolean OR of $x_{i1}, x_{i2}, \ldots x_{in}$

Step 3: If G_i computes a zero in Step 2, then processor p_{i1} outputs k_i as the answer.

Algorithm for finding the maximum in $O(1)$ time.

$p = n^2 = 5^2 = 25$.

$p = 25$.

1	2	3	4	5
1,	3,	4,	2,	5
k_1	k_2	k_3	k_4	k_5

p_{ij}

$p_{11} = x_{11} = (k_1 < k_1) = (1<1) = 0$

$p_{12} = x_{12} = (k_1 < k_2) = (1<3) = 1$

$p_{13} = x_{13} = (k_1 < k_3) = (1<4) = 1$

$p_{14} = x_{14} = (k_1 < k_4) = (1<2) = 1$

$p_{15} = x_{15} = (k_1 < k_5) = (1<5) = 1$

$p_{21} = (3<1) = 0$

$p_{22} = (3<3) = 0$

$p_{23} = (3<4) = 1$

$p_{24} = (3<2) = 0$

$p_{25} = (3<5) = 1$

$p_{31} = (4<1) = 0$

$p_{32} = (4<3) = 0$

$p_{33} = (4<4) = 0$

$p_{34} = (4<2) = 0$

$p_{35} = (4<5) = 1$

$p_{41} = (2<1) = 0$

$p_{42} = (2<3) = 1$

$p_{43} = (2<4) = 1$

$p_{44} = (2<2) = 0$

$p_{45} = (2<5) = 1$

$p_{51} = (5<1) = 0$

$p_{52} = (5<3) = 0$

$p_{53} = (5<4) = 0$

$p_{54} = (5<2) = 0$

$p_{55} = (5<5) = 0$

$G_1 = 1, G_2 = 1, G_3 = 1, G_4 = 1, G_5 = 0$

5 is the answer.

Speedup $= \frac{O(n)}{O(1)} = O(n)$ Efficiency $= \frac{O(n)}{n^2} = O\left(\frac{1}{n}\right)$.

It is not a work optimal algorithm.

finding the maximum using n processors.

n keys, n procesors — how to calculate maximum.

Step 0 : If $n=1$, return k_1

Step 1 : Partition the input keys into $\sqrt{n}$ parts $k_1, k_2, \ldots k_{\sqrt{n}}$ where k_i consists of $k_{(i-1)\sqrt{n}+1}, k_{(i-1)\sqrt{n}+2}, \ldots k_{i\sqrt{n}}$. Similarly partition the

processors so that P_j $(1 \le i \le \sqrt{n})$ consists of the processors $P_{(i-1)\sqrt{n}+1}$, $P_{(i-1)\sqrt{n}+2}$, ... $P_{i\sqrt{n}}$. Let P_i find the maximum of k_i recursively (for $1 \le i \le n$).

Step 2: If $M_1, M_2, \ldots M_{\sqrt{n}}$ are the group maxima, find and output the maximum of these maxima employing the algorithm which computes maximum with n^2 processors.

$$T(n) = T(\sqrt{n}) + O(1)$$
$$= O(\log \log n)$$

If n is not a perfect square, we take $\lceil \sqrt{n} \rceil$ (ceil)

Maximal selection among Integers

for $i = 1$ to $2c$ do

{ Step 1: Find the maximum of all the alive keys with respect to their ith parts. Let M be the maximum.

Step 2: Delete each alive key whose ith part is $< M$

}

Output one of the alive keys.

Eg. $k_1 = 1010$, $k_2 = 1101$, $k_3 = 0110$, $k_4 = 1100$

$n = 4$, $c = 2$ $\quad [0, n^c]$ $\quad 16 = 4^2$

$\log n = 2$

Each no. will be represented by $c \log n$ bits

If k_3 would be 110, we would append 0, so that k_3 0110.

k_1 1010	$\times k_1$ 1010	k_2 1101	
k_2 1101	$\rightarrow k_2$ 1101	$\rightarrow \times k_4$ 1100	$\rightarrow k_2 = 1101$
$\times k_3$ 0110	k_4 1100	Since same, move to next bit	
k_4 1100			

11.02.20

$S = \{S_1, S_2, S_3, \ldots S_n\}$ $[n^x]$ elements P_i

Now $p < n$ and not $n^x = p$ $p = [n^{(1-x)}]$

The array s is unsorted and we need to find kth element.

4, 11, 7, 6, 5, 17, 21, 38, 35, 33, 18, 39, 40, 15, 25, 29, 34, 33, 23

Given $n = 19$, $p = 4$ Find k=14th smallest element.

n^x elements to each processor

$p = n^{1-x} \Rightarrow 4 = 19^{1-x} \Rightarrow \log 4 = (1-x) \log 19$

$\Rightarrow x = 0.52918$

$\Rightarrow 19^{0.529} = 4.747 \approx 5$

$P_1 \rightarrow \{4, 11, 7, 6, 5\} \Rightarrow \{4, 5, 6, 7, 11\}$: 6 (median)

$P_2 = \{17, 21, 38, 35, 33\}$: 33

$P_3 = \{18, 39, 40, 15, 25\}$: 25 M | 6 | 33 | 25 | 29 |

$P_4 = \{29, 34, 33, 23\}$: 29 M = 25

L = | 4 | 11 | 7 | 6 | 5 | 17 | 21 | 18 | 15 | 23 | = 10 (|L|)

E = | 25 | = 1

G = | 38 | 35 | 33 | 39 | 40 | 29 | 34 | 33 | = 8

(i) $|L| \geq k$ select (L, k) $10 \geq 14$ ✗

(ii) $|L| + |E| \geq k$ then return M $10+1 \geq 14$ ✗

(iii) select (G, k − (|L| + |E|)) select (G, 3)

Again, repeating the same.

$n = n^{1-x} \Rightarrow 4 = 8^{1-x} \Rightarrow 4 = 8^{1-x} \Rightarrow x = 0.33$

$0.33^{0.33}$

No. allocated to each processor will be : $8^{0.33} = 1.99 \approx 2$

$P_1 = \{38, 35\}$: 35

$P_2 = \{33, 39\}$: 33 M = | 35 | 33 | 29 | 33 |

$P_3 = \{40, 29\}$: 29 m = 33 Now, k = 3

$P_4 = \{34, 33\}$: 33

L = 29 = 1 $1 \geq 3$ ✗

E = 33, 33 = 2 $1+2 \geq 3$ ✓

4 = 38, 35, 39, 40, 34 = 5

∴ k = 33.

Merging

$x_1 = k_1, k_2, \ldots k_m$

$x_2 = k_{m+1}, k_{m+2}, \ldots k_{2m}$

We can find qth element in x_2 in log m time and then place it at its position.

Odd - Even merge — m is an integral power of 2. 2m processors are given.

Step 0: If m = 1, merge the sequences with one comparison

Step 1: Partition x_1 and x_2 into their odd and even parts. That is, partition x_1 into $x_1^{odd} = k_1, k_3 \ldots k_{m-1}$ and $x_1^{even} = k_2, k_4 \ldots k_m$. Similarly partition x_2 into x_2^{odd} and x_2^{even}

Step 2: Recursively merge x_1^{odd} with x_2^{odd} using m processors. Let $L_1 = l_1, l_2, \ldots l_m$ be the result. Here, $x_1^{odd}, x_1^{even}, x_2^{odd}, x_2^{even}$ are in sorted order. At the same time merge x_1^{even} and x_2^{even} using the other m processors to get $L_2 = l_{m+1}, l_{m+2}, \ldots l_{2m}$

Step 3: Shuffle L_1 and L_2 i.e. $L = l_1, l_{m+1}, l_2, l_{m+2}, \ldots l_m, l_{2m}$. Compare (l_{m+i}, l_{i+1}) and interchange them if they are out of order. Output the resultant sequence.

$$T(n) = O(1) + T\left(\frac{m}{2}\right) + O(1)$$
$$= O(\log_2 m)$$

$X_1 = 2, 5, 8, 11, 13, 16, 21, 25$

$X_2 = 4, 9, 12, 18, 23, 27, 31, 34$

$X_1^{odd} = 2, 8, 13, 21$ $\quad X_1^{even} = 5, 11, 16, 25$

$X_2^{odd} = 4, 12, 23, 31$ $\quad X_2^{even} = 9, 18, 27, 34$

$L_1 = \{2, 4, 8, 12, 13, 21, 23, 31\}$ $\quad L_2 = \{5, 9, 11, 16, 18, 25, 27, 34\}$

2, 5, 4, 9, 8, 11, 12, 16, 13, 18, 21, 25, 23, 27, 31, 34

2, 4, 5, 8, 9, 11, 12, 13, 16, 18, 21, 23, 25, 27, 31, 34

18.02.20

Work-optimal algorithm for merging.

$X_1 = k_1, k_2 \ldots\ldots k_m$ $\quad p = \frac{m}{\log m}$ processors

$X_2 = k_{m+1}, k_{m+2}, \ldots\ldots k_{2m}$

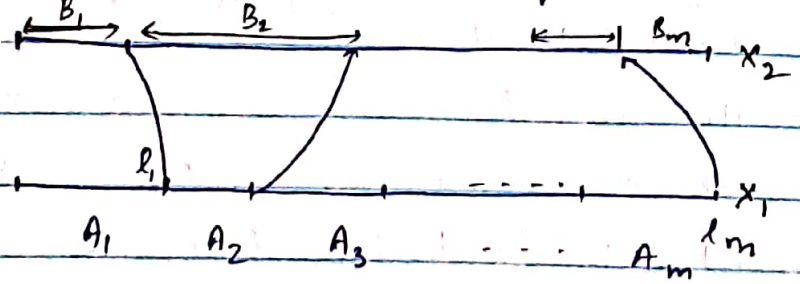
X_1 : (A_1 / l_1 log m) (A_2 / l_2 log m) (A_3 log m) ..... (A_m / l_m m/log m)

X_2 : B_1 B_2 B_3 B_m.

$M = \frac{m}{\log m}$

l_1 is largest element of A_1.

If we find no. of elements which are smaller than l_1 in B_1, then we can find rank of l_1.



We can now merge A_i and B_i.

B_i can be very small or very large. so we divide B_i into $\left\lceil \frac{|B_i|}{\log m} \right\rceil$

Complexity of this algorithm = $O(\log m)$.

Another algorithm.

Complexity = $O(\log \log m)$.

Now we divide m elements into $\sqrt{m}$ parts.

∴ We have $\sqrt{m}$ parts with $\sqrt{m}$ elements.

Each $\sqrt{m}$ subproblem will be solved by the previous method.

Odd–even merge sort.

$k_1, k_2, \ldots k_n$ — n processors

Divide this unsorted list into two parts and then solve recursively by merging the two parts with odd–even merge.

$k_1, k_2, \ldots k_{n/2}$ — $n/2$ — $n/4$ — $n/4$

$k_{n/2+1}, k_{n/2+2}, \ldots k_n$ — $n/2$ — $n/4$ — $n/4$

$x = 25, 21, 8, 5, 2, 13, 11, 16, 23, 31, 9, 4, 18, 12, 27, 34$

$p = 16$.

$x_1 = 25, 21, 8, 5, 2, 13, 11, 16$.

$x_2 = 23, 31, 9, 4, 18, 12, 27, 34$

Recursively sort x_1 and x_2

$x_1' = 2, 5, 8, 11, 13, 16, 21, 25$

$x_2' = 4, 9, 12, 18, 23, 27, 31, 34$

Merge x_1' and x_2'.

$\underbrace{25, 21}_{21, 25}, \underbrace{8, 5}_{5, 8}, \underline{2, 13}, \underline{11, 16}$

$5, 8, 21, 25$

Complexity : $T(n) = O(\log^2 n)$

$T(n) = O(1) + T(n/2) + O(\log n)$

Parallel quicksort $p = 4$

75, 91, 15, 64, 21, 8, 88, 54 P_0

50, 12, 47, 72, 65, 54, 66, 22 P_1

83, 66, 67, 0, 70, 98, 99, 82 P_2

20, 40, 89, 47, 19, 61, 86, 85 P_3

lower half

upper half

(pivot element)

Choose a random value from P_0 and broadcast it to P_1, P_2, P_3

All elements which are smaller than the pivot element in P_0 and P_2 are kept in P_0 and larger elements are kept in P_2. Similarly with P_1 and P_3

75, 15, 64, 21, 8, 54, 66, 67, 0, 70 P_0

50, 12, 47, 72, 65, 54, 66, 22, 20, 40, 47, 19, 61 P_1

83, 98, 99, 82, 91, 88 P_2

89, 86, 85 P_3

Now this can be divided into two subproblems choosing pivot elements as 21 and 91 in both.

∴ There will be total log p steps

15, 21, 8, 0, 12, 20, 19 P_0

75, 64, 54, 66, 67, 70, 50, 47, 72, 65, 54, 66, 22, 40, 47, 61 P_1

83, 82, 91, 88, 89, 86, 85 P_2

98, 99, P_3

Now we sort these 4 unsorted lists using sequential quicksort.

Sorting elements of each processor using quicksort:

P_0 : 0, 8, 12, 15, 19, 20, 21

Now we can directly combine all the elements.

Hyperquicksort

75, 91, 15, 64, 21, 8, 88, 54 — 8, 15, 21, (54), 64, 75, 88, 91 — P_0

50, 12, 47, 72, 65, 54, 66, 22 — 12, 22, 47, 50, 54, 65, 66, 72 — P_1

83, 66, 67, 0, 70, 98, 99, 82 — 0, 66, 67, 70, 82, 83, 98, 99 — P_2

20, 40, 89, 47, 19, 61, 86, 85 — 19, 20, 40, 47, 61, 85, 86, 89 — P_3

L.H. — quick sort — U.H.

Sort the list of each processor using quicksort

P_0 chooses a pivot element which is the median element and then broadcasts it to P_0, P_1, P_2, P_3

Median n: even $\Rightarrow n/2$; n: odd $\Rightarrow n/2 + 1$

Combine P_0 and P_2 as in the previous method.

Merge and then apply quick sort

0, 8, (15), 21, 54 — P_0

L.H. 12, 19, 20, 22, 40, 47, 47, 50, 54 — P_1

64, 66, 67, 70, 75, (82), 83, 88, 91, 98, 99 — P_2

U.H. 61, 65, 66, 72, 85, 86, 89 — P_3

0, 8, 12, 15

19, 20, 21, 22, 40, 47, 47, 50, 54, 54

61, 64, 65, 66, 66, 67, 70, 72, 75, 82

83, 85, 86, 88, 89, 91, 98, 99

Now combine all of these sorted lists.

Complexity : $O\left(\frac{n}{p} \log \frac{n}{p}\right)$.

Parallel Sorting by Regular Sampling (PSRS)