Algorithm ① Input ② Definiteness
Output ③ effective
④ unambiguous
⑤ steps must remain same i.e generic approach

concurrency V/S Parallelism

Speedup

A+B take 1 unit time

adding 100 nos. take 99 unit of time

A → add 50 nos = 49 units of time } 1 unit for sum of both
B → add 50 nos = 49 units —

⇒ 50 units of time

$$\text{Speedup} = \frac{S(n)}{T(n,p)}$$

if $\frac{S(n)}{T(n,p)} = \theta(p)$ ⇒ linear speedup

sort n heaps ~~unit~~ using heap sort algo → $\theta(n \log n)$

A → n processor ⇒ $\theta(\log n)$

B → n^2 processor ⇒ $\theta(\log n)$

speedup of $A = \frac{\theta(n \log n)}{\theta(\log n)} = \theta(n)$ ⇒ linear speedup

Speedup of $B = \frac{\Theta(n \log n)}{\Theta(\log n)} = \Theta(n) \neq \Theta(n^2)$

$\therefore$ not linear speedup

Total Work done $= p \cdot T(n, p)$

Efficiency = speedup of algo ÷ no. of processors used

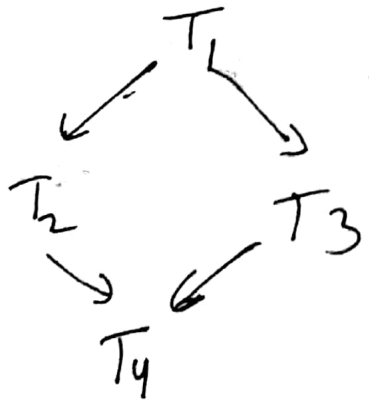
$$= \frac{S(n)}{p \cdot T(n, p)}$$

if efficiency $= \Theta(1)$ then we can say parallel algorithm is work-optimal

Efficiency of $A = \Theta(1)$ or $\frac{\Theta(n)}{n}$ while for $B = \Theta\left(\frac{1}{n}\right)$

Amdhal's law →

if we try to use all processors, divide the problem is different parts.



max speed up $= \frac{1}{f + \frac{1-f}{n}}$

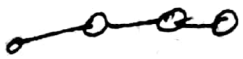
$f \rightarrow$ parallel code

$p = 10 \quad f = 0.5$

$= \frac{1}{0.5 + \frac{0.5}{10}}$

$p=10 \quad f=0.5$

$$f = \frac{1}{0.5 + \frac{0.5}{10}} = 1.8$$



4 Processors P_i $(i = 1 \text{ to } 4)$

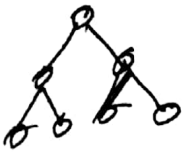
Read M[1]

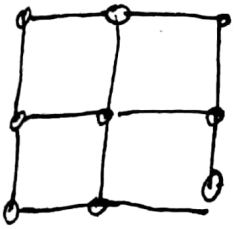
CRCW

CREW model requires 1 unit time

CREW → 1

EREW → 4 unit





write M[1]

CRCW → 1 unit

CREW → 4 unit

EREW → 4 unit

23 Jan '20

A is a no. having n bits

Boolean OR

n bits

$A = A[1, 2, \ldots n]$

$A[0] = A[1] \,||\, A[2] \,||\, \ldots \,||\, A[n]$

$A = 1\,1\,0\,1\,0\,0$

$1\;1\;1\;1\;1\;1$

on a single processor machine

$T(n) = O(n)$

for n-processor machine we use CRCW

$P_1 \rightarrow A[1]$

$P_2 \rightarrow A[2]$

$\vdots$

$P_n \rightarrow A[n]$

in CRCW

① arbitrary
③ priority
② common

Processor i (in parallel $1 \le i \le n$) does

for i^{th} processor if $(A[i] = 1)$ then $A[0] = A[i]$

$T(n) = O(1)$

for EREW
EREW $O(n)$

slow down lemma

$I/p \to n$

processor $\to n$

(in real scenario processors $< n$)

for p- processor machine
taking time T

for p' processor machine
$(p' < p)$

time $= \frac{pT}{p'}$

Prefix computation problem

suppose a problem Π

we want to compute prefix of given numbers x, y, z

$x, y, z \in \Sigma$

$\oplus$ is representing any operator

condition

① $\left.\begin{matrix} x \oplus y \\ x \oplus z \\ y \oplus z \end{matrix}\right\} \in \Sigma$

② they should follow the associativity property

$((x \oplus y) \oplus z) = (x \oplus (y \oplus z))$

I/p: $x_1, x_2, x_3 \ldots x_n$

o/p: $x, x_1 \oplus x_2, x_1 \oplus x_2 \oplus x_3, \ldots, x_1 \oplus x_2 \oplus \ldots x_n$

I/p: 2, 4, −6, −1, 7, 9 $\quad T(n) = 2T(\frac{n}{2}) + O(1)$

⊕ addition

o/p: 2, 6, 0, −1, 6, 15

⊕ multiplication

o/p: 2, 8, −48, 48, 336, 3024

⊕ min

o/p: 2, 2, −6, −6, −6, −6

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

$n=8$ $p=8$ processors

p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8

This is divide & conquer approach

⊕ addition

12, 15, 3, 6, 8 | 11, 4, 5, 7

$T(n) = T(\frac{n}{2}) + O(1)$

12, 15, 21, 29 | 11, 15, 20, 27

40, 49, 56

at 1 time max 4 processors can be used

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

$$T(n) = O(\log n)$$

Prefix computation in $(\log n)$ time with n processors

Step 0: If $n=1$, one processor output x_1

Step 1: Let the first $n/2$ processors recursively compute the prefixes of $x_1, x_2 \ldots x_{n/2}$ & let $y_1, y_2 \ldots y_{n/2}$ be the result. At the same time let the rest of the processors recursively compute the prefixes of $x_{n/2+1}, x_{n/2+2}, \ldots x_n$ & let $y_{n/2+1}, y_{n/2+2} \ldots$ y_n be the output.

Step 3: Note that the first half of the final answer is the same as $y_1, y_2 \cdots y_{n/2}$

the second half of the answer is $y_{n/2} \oplus y_{n/2+1}$,

$y_{n/2} \oplus y_{n/2+2}$, $y_{n/2} \oplus y_{n/2+3} \cdots y_{n/2} \oplus y_n$

$$T(1) = O(1)$$

for a single processor, $O(n)$ is sequential runtime & parallel runtime is $O(\log n)$ for n processor

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

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

28 Jan 2020

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

$\neq O(1)$

∴ this algo is not work optimal

X——X——X——X——X

I/p: 2, -1, 3, 4, -2, 5

⊕ Multiplications

2, -2, -6, -24, 48, 240

I/p: 5, 8, -2, 7, -11, 12 $n=8$ $p=8$

⊕ Minimum

5, 5, -2, -2, -11, -11

P_1 P_2	P_3 P_4	P_5 P_6	P_7 P_8
12 3	6 8	11 4	5 7

12 15 6 14 11 15 5 12 1 unit time

12 15 21 29 11 15 20 27 2 unit time

12 15 21 29 40 44 4956 3 unit time

$T(8) = \log_2 8 = \log 3$

work-optimal logarithmic time prefix computation using $\frac{n}{\log n}$ processors

Processor i $\left(i \in 1, 2, \ldots \frac{n}{\log n}\right)$

each processor have $\log n$ elements

$x_{(i-1)\log n+1}$

$x_{(i-1)\log n+2}$

⋮

$x_{i \log n}$

Step 1: Processor i ($i = 1, 2, \ldots \frac{n}{\log n}$) in parallel computes the prefixes of its $\log n$ assigned elements $x_{i-1 \log n+1}, x_{i-1 \log n+2}, \ldots, x_{i \log n}$. This takes $O(\log n)$ time.

Let the results be $z_{(i-1)\log n+1}, z_{(i-1)\log n+2}, \ldots, z_{i \log n}$. $\rightarrow O(\log n)$

Step 2: a total of $\frac{n}{\log n}$ processors collectively employ Algorithm 13.2 to compute the prefixes of the $\frac{n}{\log n}$ elements $z_{\log n}, z_{2\log n}, z_{3\log n}, \ldots z_{n}$

Let $w_{\log n}, w_{2\log n}, w_{3\log n}, \ldots, w_{n}$ be the result. $\rightarrow O(\log \frac{n}{\log n})$

step 3: Each processor updates the prefixes it computed in step 1 as follows. Processor i computes and outputs $w_{(i-1)\log n} \oplus z_{(i-1)\log n+1}$, $w_{(i-1)\log n} \oplus z_{(i-1)\log n+2}, \ldots, w_{(i-1)\log n} \oplus z_{i \log n}$.

for $i = 1, 2, 3, \ldots \frac{n}{\log n}$. Processor i outputs

$z_1, z_2, \ldots z_{\log n}$ without any modifications.

$\rightarrow O(\log n)$

$\log n + \log\left(\frac{n}{\log n}\right) + \log n$

$= O(\log n)$

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

$n = 16$

⊕ addition

no of processors $= \frac{16}{\log 16} = \frac{16}{4} = 4$

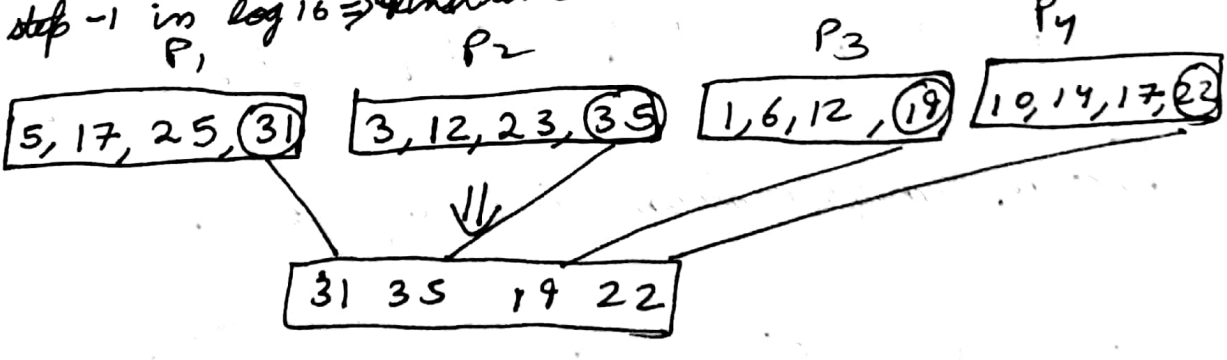
Processor 1 5, 12, 8, 6

P_2 3, 9, 11, 12

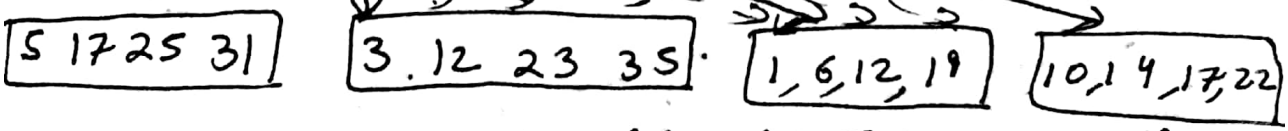
P_3 1, 5, 6, 7

P_4 10, 4, 3, 5

step -1 in $\log 16 \Rightarrow 4$ unit time

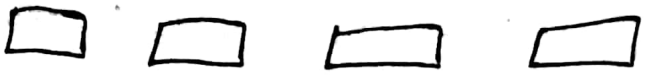


31 66 85 107



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

List Ranking Problem.

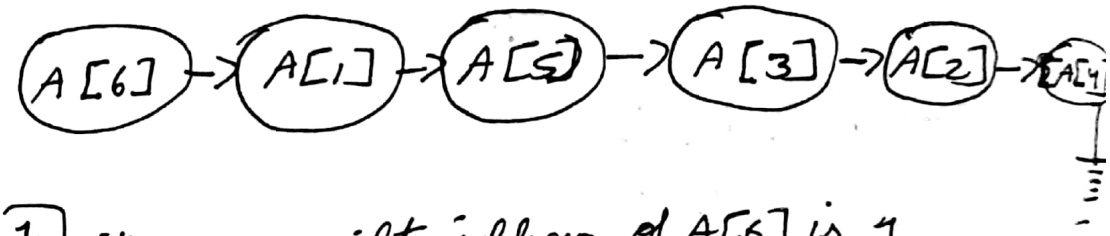


list of nodes representing an array.

find the rank of nodes

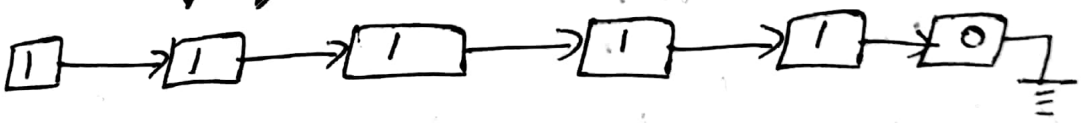
rank → no. of elements present is right of it

5	4	2	0	3	1
A[1]	A[2]	A[3]	A[4]	A[5]	A[6]

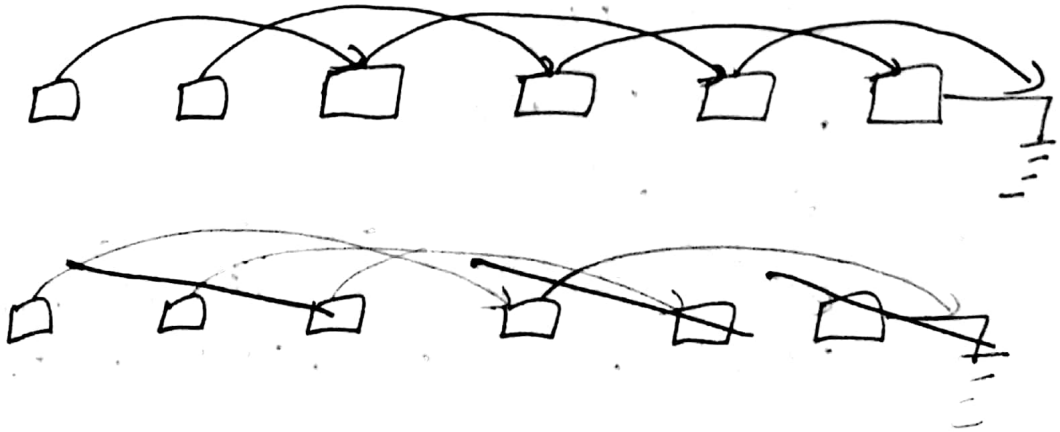


it means right neighbour of A[6] is 1

Pointer jumping



$\lceil \log n \rceil$ times, operation will be performed



An $O(n \log n)$ 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]];
        }
```

Neighbor

5	4	2	0	3	1
A[1]	A[2]	A[3]	A[4]	A[5]	A[6]

Rank

1	1	1	0	1	1
1	2	3	4	5	6

beginning stage

Step-1 $q=1$

3	0	4	0	2	5
1	2	3	4	5	6

2	1	2	0	2	2
1	2	3	4	5	6

$q=2$

4	0	0	0	0	2

4	1	2	0	3	4
2+2	1+0	2+0	0	2+1	2+2

$q=3$

0	0	0	0	0	0

4	1	2	0	3	5
4+0					4+2

Not a work optimal algorithm

4 Feb '20

The Problem of Selection

n keys (distinct) $k_1, k_2, \ldots k_n$ are given. find i^{th} smallest key where $1 \le i \le n$

maximum selection with n^2 processors. $\rightarrow O(1)$ time

step 0: if $n=1$ output the key

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

step 2: The n^2 processors are grouped into n groups $G_1, G_2, G_3 \ldots G_n$ where G_i $(1 \le i \le n)$ consists of the processors $P_{i1}, P_{i2}, P_{i3} \ldots P_{in}$.

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 processors P_{i1} output k_i as the answer.

example: $n=5$ then $p=25$

$(p = n^2 = 5^2 = 25)$

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

G_1:
- $P_{11} = x_{11} = k_1 < k_1 = 0$
- $P_{12} = x_{12} = k_1 < k_2 = 1$
- $P_{13} = x_{13} = k_1 < k_3 = 1$
- $P_{14} = x_{14} = 1$
- $P_{15} = x_{15} = 1$

G_2:
- $P_{21} = 0$
- $P_{22} = 0$
- $P_{23} = 1$
- $P_{24} = 0$
- $P_{25} = x_{25} = k_2 < k_5 = 1$

G_3:
- $P_{31} = 4 < 1 = 0$
- $P_{32} = 4 < 3 = 0$
- $P_{33} = 4 < 4 = 0$
- $P_{34} = 4 < 2 = 0$
- $P_{35} = 4 < 5 = 1$

$$\begin{cases} p_{41} = 2<1 = 0 \\ p_{42} = 2<3 = 1 \\ p_{43} = 2<4 = 1 \\ p_{44} = 2<2 = 0 \\ p_{45} = 2<5 = 1 \end{cases} \quad G_5 \quad \begin{cases} p_{51} = 5<1 = 0 \\ p_{52} = 5<3 = 0 \\ p_{53} = 5<4 = 0 \\ p_{54} = 5<2 = 0 \\ p_{55} = 5<5 = 0 \end{cases}$$

$G_1 = p_{11} | p_{22} | p_{13} | p_{14} | p_{15}$

$= 0|1|1|1|1 = 1$

$G_1 = 1$

$G_2 = 1$

$G_3 = 1$

$G_4 = 1$

$G_5 = 0 \Rightarrow k_5$ is maximum

$$\text{speedup} = \frac{O(n)}{O(1)} = O(n)$$

$$\text{efficiency} = \frac{O(n)}{n^2} = O\left(\frac{1}{n}\right)$$

Now finding the maximum using 'n' processors. → $O(\log\log n)$ time

Step 0: if $n=1$ return k_1.

Step 1: Partition the input keys into $\sqrt{n}$ parts $K_1, K_2 \ldots K_{\sqrt{n}}$ → $\lceil\sqrt{n}\rceil$ → ceil

where K_i consists of $k_{(i-1)\sqrt{n}+1}, k_{(i-1)\sqrt{n}+2} \ldots k_{i\sqrt{n}}$

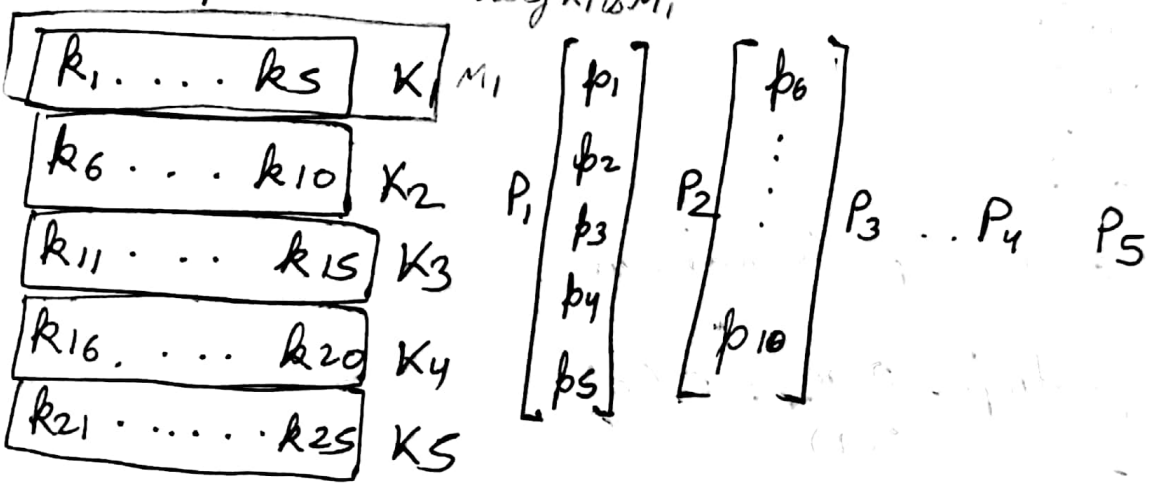
Similarly partition the processors so that P_i $(1 \le i \le \sqrt{n})$ consists of the processors $p_{(i-1)\sqrt{n}+1}$, $p_{(i-1)\sqrt{n}+2} \ldots p_{i\sqrt{n}}$. Let p_i find the maximum of K_i recursively for $(1 \le i \le \sqrt{n})$.

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 processor.

example $n = 25$

$\sqrt{n} = 5$

max of K_1 is M_1



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

$$O(\log \log n)$$

Maximal Selection among Integers (represented in binary)

010100 -> key
001 010
.
.
.

n keys

for (i = 1 to 2c) do
{

Step 1: find the maximum of all the alive keys with respect to their i^{th} parts. Let M be the

maximum.

Step 2 : Delete each alive key whose i^{th} part is < m.

}

output one of the alive keys.

example :- $n = 4$

$k_1 = 1010$ numbers lie b/w $[0, n^c]$

$k_2 = 1101$

$k_3 = 0110$

$k_4 = 1100$ $16 = 4^2$ therefore $c = 2$.

$\log n = \log 4 = 2$

$c \log n = 2 \times 2 = 4$

if $k_3 = 110$ append 0 & make $k_3 = 0110$

$i=1$

$i=1$	$i=2c$
1	0 1 0
1	1 0 1
0	1 1 0
1	1 0 0

delete k_3

$i=2$

k_1	1	0 1 0
k_2	1	1 0 1
k_4	1	1 0 0

delete k_1

$i=3$

k_2	1	0 1
k_4	1	0 0

$i=4$

k_2	1 1 0	1
k_4	1 1 0	0

delete k_4

output $\Rightarrow k_2 = 1101$

11 Feb '2020

$S = \{S_1, S_2, S_3, \ldots S_n\}$

find k^{th} smallest element using 'p' processors with $\log n$ complexity

where $p < n$.

$[n^x]$ or $[S^\alpha]$ elements P_i

$p = [n^{1-x}]$

every P_i have $[S^x]$ elements.

eg:-

4, 11, 7, 6, 5, 17, 21, 38, 35, 33, 18, 39, 40, 15,

25, 29, 34, 33, 23

$n = 19$ $P = 4$ find $k = 14^{th}$ smallest element

5^x elements for each processor

$P = n^{1-x}$

$4 = (19)^{1-x}$

$\log 4 = (1-x)\log 19$

$1 - x = 0.4708$

$x = 0.5292$

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

$M/2$ in even numbers

$M/2+1$ in odd numbers

$P_1 = \{4, 11, 7, 6, 5\} = \{4, 5, 6, 7, 11\} = 6$ median

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

$P_3 = \{18, 39, 40, 15, 25\} = 25$

$P_4 = \{29, 34, 33, 23\} = 29$

M | 6 | 33 | 25 | 29 | = 25

L =	4	11	7	6	5	17	21	18	15	23	|L| = 10
E =	25										1
G =	38	35	33	39	40	29	34	33			8

Select (S, k)

① $|L| \geq k$ Select (L, k) $10 \geq 14$ false

② $|L| + |E| \geq k$ return M $10+1 \geq 14$ false

③ Select (G, k-(|L| + |E|))

Select (G, 14-(10+1)) = Select (G, 3)

~~8 5 2 9 3~~

n^{x} ~~8 30~~ $8^{0.33} = 2$ $n^{0.33} = 2$

$p = n^{1-x}$ $4 = 8^{1-x}$ $\log 4 = (1-x)\log 8$

$\frac{2}{3} = 1-x$

$x = 1 - \frac{2}{3} = 0.33$

	M
P_1 = 38, 35	35
P_2 = 33, 39	33
P_3 = 40, 29	29
P_4 = 34, 33	33

35	33	29	33

M = 33

~~A = 38, 35, 39, 40~~ new k = 3

L = 29 |L| = 1 $1 \geq 3$ false

E = 33 33 1+2 = 3 $\geq$ 3 true

G = 38 35 39 40

33

b Merging — m is an integral power of 2

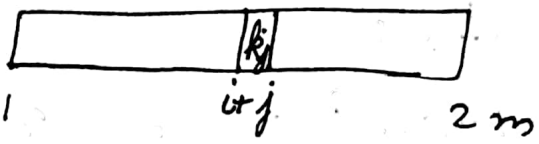
$x_1 = k_1, k_2, k_3 \ldots k_j \ldots k_m$

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

2m processors are given

find number of elements in x_2 smaller than j — $O(\log m)$ time

let it be i



Odd-Even merge $O(\log m)$

Step 0: if $m = 1$, merge the sequences with one comparison $O(1)$

Step 1: Partition X_1 & X_2 into their odd & 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} & X_2^{even}. $O(1)$

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} & X_2^{even} using other m processors toget

$L_2 = l_{m+1}, l_{m+2} \ldots l_{2m}$.

Step 3: Shuffle L_1 & 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}) & interchang

them if they are out of order. Output the resulting sequence.

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

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

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

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

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

$X_1'^{odd} = 2, 13$ $\quad X_1'^{even} = 8, 21$

$X_2'^{odd} = 4, 23$ $\quad X_2'^{even} = 12, 31$

$X_1''^{odd} = 2$ $\quad X_1''^{even} = 13$ $\quad X_1'' \; 8 \quad X_1'' \; 21$

$X_2'' = 4$ $\quad X_2''^{even} = 23$ $\quad X_2'' \; 12 \quad X_2'' \; 31$

$L_1'' = 2 \; 4 \quad L_2'' = 13 \; 23$

$L_1' = 2, 13, 4, 23$

$L_1' = 2, 4, 13, 23 \quad L_2' = 8, 12, 21, 31$

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

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

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