Selection:

Input: \( X = k_1, k_2, \ldots, k_n \) and \( i, 1 \leq i \leq n \)

Output: the \( i \)th smallest element of \( X \).

Fact: let \( y \) be any element. We can compute \( \text{Rank}(y, X) \) in \( O(\log n) \) time. This can be done using a prefix addition using \( \frac{n}{\log n} \) CREW PRAM processors.

\[ \Rightarrow \text{selection can be done in } O(\log n) \text{ time using } \frac{n^2}{\log n} \text{ Processors.} \]

Theorem: we can solve selection in \( \tilde{O}(\log n) \) time using \( \frac{n}{\log n} \) CREW PRAM processors.

An Algorithm:
To begin with, each key is alive; \( N \) is the number of alive keys at any time;
\( N := n; \quad P = \frac{n}{\log n} \)

While \( N > \sqrt{n} \) do

1) Pick a sample \( S \) of size \( s \) keys; the first \( s \) processors pick one sample key each randomly. Here \( s = \frac{N}{3} \).
   This step takes \( O(1) \) time.

2) Sort the sample and pick two elements \( l_1 \) and \( l_2 \), so that \( \text{Rank}(l_1, S) = i \frac{s}{N} - \delta \)
   \( \text{Rank}(l_2, S) = i \frac{s}{N} + \delta \); Where \( \delta = \sqrt{4\alpha \log n} \)
   This step takes \( O(\log n) \) time.

3) Count the number of \( N_1 \) of alive keys that are \( < l_1 \); as well count the number \( N_2 \) of alive keys in the range \( [l_1, l_2] \).
   This step will take \( O(\log n) \) time, since we can use prefix computation.

4) If \( !N_1 < i \leq N_1 + N_2 \) then start over from step 1.
This takes $O(1)$ time.

5) Delete all the keys that are not in the range $[l_1, l_2]$.
   
   $i = i - N_1$;
   $N = N_2$.
   This step takes $O(\log n)$ time.

6) Concentrate the alive keys using a prefix computation.
   This step will take $O(\log n)$ time.

End of while

7) Sort the alive keys using the trivial algorithm and output the $i$th smallest element.
   This step will take $O(\log n)$ time.

Analysis:
According to the sampling lemma, the number of alive keys after each run of the while loop is

\[ \tilde{O} \left( \frac{N}{\sqrt{s}} \sqrt{\log N} \right) = \tilde{O} \left( \frac{N}{\sqrt{\log N}} \right) \Rightarrow \tilde{O} \left( N^{0.9} \right). \]

After a constant number of while loops, the number of keys will be $\tilde{O}(\sqrt{n})$. □

Corollary: we can do the same in $\tilde{O}(\frac{\log n}{\log \log n})$ time using $\frac{n}{\log n} \log \log n$ arbitrary CRCW PRAM processors.

Sorting:

<table>
<thead>
<tr>
<th>Authors</th>
<th>Model</th>
<th>Processors</th>
<th>Time</th>
<th>Rand/Det</th>
<th>Years</th>
</tr>
</thead>
<tbody>
<tr>
<td>BATCHER</td>
<td>Butterfly</td>
<td>$n$</td>
<td>$\frac{1}{2} \log^2 n$</td>
<td>Deterministic</td>
<td>1961</td>
</tr>
<tr>
<td>PREPARATA</td>
<td>CRCW PRAM</td>
<td>$n \log n$</td>
<td>$O(\log n)$</td>
<td>Deterministic</td>
<td>1971</td>
</tr>
<tr>
<td>AKS</td>
<td>Sorting network</td>
<td>$n$</td>
<td>$O(\log n)$</td>
<td>Deterministic</td>
<td>1981</td>
</tr>
<tr>
<td>REISCHÜK</td>
<td>CRCW PRAM</td>
<td>$n$</td>
<td>$\tilde{O}(\log n)$</td>
<td>Randomized</td>
<td>1981</td>
</tr>
<tr>
<td>COLE</td>
<td>EREW PRAM</td>
<td>$n$</td>
<td>$O(\log n)$</td>
<td>Deterministic</td>
<td>1984</td>
</tr>
<tr>
<td>RAJASEKARAN &amp; REIF</td>
<td>CRCW PRAM</td>
<td>$n(\log n)^\epsilon$, $0 &lt; \epsilon &lt; 1$</td>
<td>$\tilde{O}(\frac{\log n}{\log \log \log n})$</td>
<td>Randomized</td>
<td>1987</td>
</tr>
<tr>
<td>COLE</td>
<td>CRCW PRAM</td>
<td>$n(\log n)^\epsilon$, $0 &lt; \epsilon &lt; 1$</td>
<td>$O(\frac{\log n}{\log \log \log n})$</td>
<td>Deterministic</td>
<td>1989</td>
</tr>
</tbody>
</table>
(ALON & AZAR 1985)

**Theorem:** Sorting of $n$ elements using $P$ processors needs $\Omega \left( \frac{\log n}{\log(1 + \frac{1}{P})} \right)$ time on the parallel comparison tree model.

**Theorem:** we can sort $n$ elements in $\tilde{O}(\log n)$ time using $n$ CRCW PRAM processors.

**Proof:** here is an algorithm…. *To be continued in the next lecture.*