CSE6512: Randomization in Computing
Lecture 19, Nov 1st 2011

Notes by Anas Al-Okaily

Selection:

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

Output: the ith smallest element of X.

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

$\Rightarrow$ *selection can be done in* $O(\log n)$ *time using* $\frac{n^2}{\log n}$ 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;
$N := n$; $P = \frac{n}{\log n}$

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 $s = N^{\frac{1}{3}}$.
   This step takes O(1) time.

2) Sort the sample and pick two elements l_1 and l_2, so that
   Rank $(l_1, S) = i \frac{s}{N} - \delta$

   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 ith 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 $\tilde{O}\left(\frac{N}{\sqrt{s}}\sqrt{\log N}\right) = \tilde{O}\left(\frac{N}{\sqrt{N^{\frac{1}{3}}}}\sqrt{\log N}\right) \Rightarrow \tilde{O}(N^{0.9})$.

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

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

Sorting:

Authors	Model	Processors	Time	Rand/Det	Years
BATCHER	Butterfly	n	$\frac{1}{2}\log^2 n$	Deterministic	1961
PREPARATA	CRCW PRAM	$n \log n$	O(log n)	Deterministic	1971
AKS	Sorting network	n	O(log n)	Deterministic	1981
REISCHüK	CRCW PRAM	n	$\tilde{O}(\log n)$	Randomized	1981
COLE	EREW PRAM	n	O(log n)	Deterministic	1984
RAJASEKARAN & REIF	CRCW PRAM	$n(\log n)^{\epsilon}$ $0<\epsilon<1$	$\tilde{O}\left(\frac{\log n}{\log\log n}\right)$	Randomized	1987
COLE	CRCW PRAM	$n(\log n)^{\epsilon}$, $0<\epsilon<1$	$O\left(\frac{\log n}{\log\log\log n}\right)$	Deterministic	1989

(ALON & AZAR 1985)

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

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

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