CSE 6512 Lecture on Optimal Randomized Sorting

James Lindsay

November 3, 2011

Preparata's Algorithm

Theorem

We can sort any n elements in $O(\log n)$ time using $n \log n$ CREW PRAM processors.

The input is a sequence of k elements denoted by

$$X = \left[k_1, k_2, ..., k_{n/\log n}\right], \left[k_{n/\log n+1}, ..., k_{2n/\log n}\right], ..., [..., k_n]$$

Let S_1 be the subsequence of the first $\frac{n}{\log n}$ elements of X; let S_2 be the subsequence of the next $\frac{n}{\log n}$ elements of X; and so on.

Preparata's Algorithm

The algorithm is detailed as follows

for $1 \leq i \leq \log n$ in $||^l$ **do**
  (1) Recursively sort S_i using n processors
  (2)
  for $1 \leq i, j \leq \log n$ in $||^l$ **do**
    merge S_i with S_j
  end for
  (3)
  for $1 \leq i \leq n$ in $||^l$ **do**
    using $\log n$ processors compute the global rank of k_i using a prefix computation
  end for
  (4)
  for $1 \leq i \leq n$ in $||^l$ **do**
    if $rank(k_i) = r_i$ **then**
      output k_i in cell r_i
    end if
  end for
end for

Analysis

Let $T(n)$ be the run time of Preparata's algorithm to sort n elements using $n \log n$ processors. Then, the various steps take the following times:

1. $T(n/\log n)$
2. $O(\log \log n)$. Here we have used the fact that we can merge two sorted sequences of length n each in $O(\log \log n)$ time using n processors.
3. $O(\log \log n)$
4. $O(1)$

Thus we have:

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

Theorem

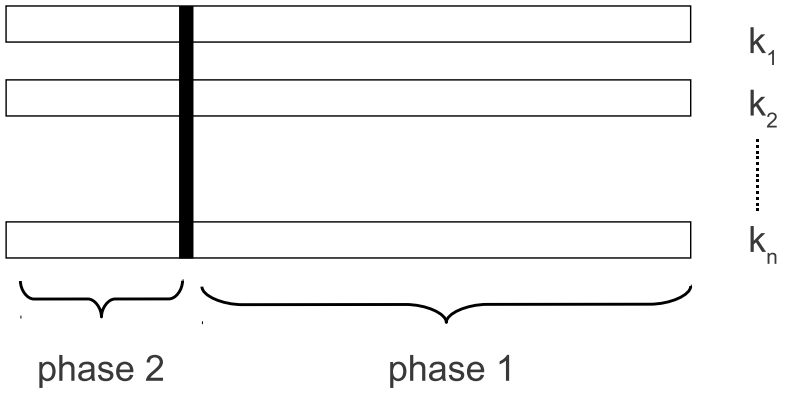
(Rajasekaran and Reif 1989)

We can sort n integers in the range $[1, n(\log n)^c]$ in $\widetilde{O}(\log n)$ time using $\frac{n}{\log n}$ arbitrary CRCW PRAM processors, for any constant c.

Algorithm

The algorithm utilizes bucket sorting techniques using two phases. The first phase sorts a majority of the bits using a non-stable technique while the second phase sorts the remaining bits using a stable technique. Since the first phase of the sorting does not have any previous ordering to maintain it is possible to gain an advantage by using non stable sorting.

We can think of each input key as a binary string of length $\log n + c \log \log n$.


k1
k2
kn
phase 2
phase 1

In the first phase the keys are sorted with respect to their $\log n + c \log\log n$ least significant bits and in the second phase they are sorted with respect to the remaining $(c+3)\log\log n$ bits. More details will be covered in a subsequent lecture.

Optimal Randomized Sorting

Theorem

We can sort n arbitrary elements in $\tilde{O}(\log n)$ time using n arbitrary CRCW PRAM processors.

Algorithm

(1) Let $s = n/\log n$. Pick a random sample S of size s.
(2) Sort the sample using Preparata's algorithm. Let the sorted sample be $L = l_1, l_2, ..., l_s$.
(3)
for $1 \leq i \leq n$ in $||^l$ **do**
  Using 1 processor do a binary search on L to figure out the partition that k_i belongs to. See figure 1.
end for
(4) Sort the keys with respect to their part numbers using Rajasekaran and Reif's Lemma.
(5)
for all $1 \leq i \leq (s+1)$ in $||^l$ **do**
  Sort X_i using Preparata's algorithm using $|X_i|$ processors
end for

Note that the size of $X_i = \tilde{O}(\log^2 n)$. This follows from a random sampling lemma we have proved a while ago. Each X_i can be sorted in $O(\log |X_i|)$ time using $|X_i| \log |X_i|$ processors. This implies (using the slow down lemma) that X_i can also be sorted in $O(\log^2 |X_i|)$ time using X_i processors.

Analysis

1. Step 1 can be done in $O(1)$ time.
2. Step 2 takes $O(\log n)$ time using n processors.
3. Step 3 takes $O(\log n)$ time.
4. This step takes $\widetilde{O}(\log n)$ time.

5. This step takes $O(\max_{i=1}^{s+1} \log^2 |X_i|)$ time using $\sum_{i=1}^{s+1} |X_i| = n$ processors. Since $\max_{i=1}^{s+1} |X_i| = \widetilde{O}(\log^2 n)$, this step takes $\widetilde{O}\left((\log\log n)^2\right) = \widetilde{O}(\log n)$ time.

Thus the entire algorithm takes $\widetilde{O}(\log n)$ time using n processors. $\square$

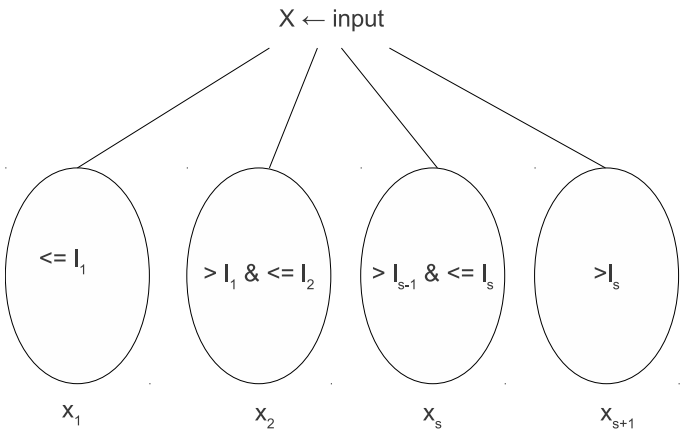


Figure 1: sorting strategy