1 Integer sort

1.1 Coarse Sort

In Coarse Sort we have to sort $n$ integers in the range $[1, \frac{n}{(\log n)^3}]$. Let $D = \frac{n}{(\log n)^3}$. Let the input be $X = k_1, k_2, ..., k_n$. Let $I(k) = \{i : k_i = k\}$, $1 \leq k \leq D$. The Coarse Sort algorithm works as follows.

1. Compute $N_1, N_2, ..., N_D$ such that $N_i \geq |I(i)|$ for $1 \leq i \leq D$ and $\sum_{i=1}^{D} N_i = O(n)$.

   a) For $1 \leq i \leq D \log n$ in parallel do: processor $i$ picks randomly a key from $X$.

   b) Sort the sample picked in a) using Preparata’s algorithm. This can be done in $O(\log n)$ time using $\frac{n}{\log n}$ processors. Let $I_s(k) = \{i : k_i is in the sample and k_i = k\}$.

   c) For $1 \leq i \leq D$ in parallel do: Processor $i$ computes $N_i = d \alpha \log^2 n \times$ max${|I_s(k)|, \log n}$ where $d$ is a constant. This is done in $O(1)$ time.

2. Using the $N_i$’s and the assignment algorithm, rearrange the keys. The group # of any key is its value.

1.2 Analysis

1. Case 1. If $|I(k)| < d \alpha \log^3 n$ then $N_k \geq I(k)$

2. Case 2. $|I(k)| \geq d \alpha \log^3 n$:

   Note that $I_S(k)$ is binomial with parameters ($\frac{n}{\log^2 n}, \frac{|I(k)|}{n}$). Using Chernoff bounds:

   \[
   \text{Prob}[|I_S(k)| < (1 - \epsilon)\frac{|I(k)|}{\log^2 n}] \leq \exp(-\frac{\epsilon^2|I(k)|}{2 \log^2 n})
   \]

   Let $\epsilon = \frac{1}{2} \Rightarrow \text{RHS} \leq \exp(-\frac{|I(k)|}{8 \log^2 n})$

   If $d \geq 8$, RHS $\leq n^{-\alpha}$
⇒ $N_k \geq I(k)$ with probability $\geq (1 - n^{-\alpha})$.

3. $\sum_{k=1}^{D} N_k = \sum_{k=1}^{D} d\alpha \log^2 n \times \max\{|I_s(k)|, \log n\}$

$$\leq \sum_{k=1}^{D} d\alpha \log^2 n \{|I_s(k)| + \log n\}$$

$$\leq d\alpha \log^2 n \sum_{k=1}^{D} |I_s(k)| + \sum_{k=1}^{D} d\alpha \log^3 n$$

$$= 2d\alpha n = O(n)$$

Note that $\sum_{k=1}^{D} d\alpha \log^3 n = d\alpha n$,

$$\sum_{k=1}^{D} |I_s(k)| = \frac{n}{\log^2 n},$$

and $d\alpha \log^2 n \sum_{k=1}^{D} |I_s(k)| = d\alpha n$.

2 Sub-logarithmic time algorithms

2.1 Solving the assignment problem in $O\left(\frac{\log n}{\log \log n}\right)$ time

Input: $X = k_1, k_2, ..., k_n$

Group #’s: $g_1, g_2, ..., g_n$. Assume that the group number is an integer in the range $[1, q]$.

Upper bounds on the group sizes: $N_1, N_2, ..., N_q$

Output: A rearrangement of $X$ based the groups #’s.
**Theorem:** We can solve the above problem in \( O \left( \frac{\log n}{\log \log n} \right) \) time using \( n \log n (\log \log n)^2 \) arbitrary CRCW PRAM processors.

**Proof:** Here is an algorithm: Let \( P = \frac{n}{\log n} (\log \log n)^2 \).

1. Using a prefix sums computation on \( 2N_1, 2N_2, \ldots, 2N_q \), identify the boundaries of the buckets. This takes \( O \left( \frac{\log n}{\log \log n} \right) \) time.

2. For \( 1 \leq i \leq n \) in parallel do:
   
   Make \( \log \log n \) attempts to place the key \( k_i \) in its right bucket. This can be done using \( \frac{n}{\log n} (\log \log n)^2 \) processors in \( O \left( \frac{\log n}{\log \log n} \right) \) time.

3. Do a prefix computation to identify the number \( Z \) of elements that have not been placed yet.
   
   \( \Rightarrow \) This takes \( O \left( \frac{\log n}{\log \log n} \right) \) time.

4. Assign \( \left( \frac{P}{Z} \right) \) processors to each such element. The processors associated with any such element attempt in parallel to place the element in its bucket. A total of \( O \left( \frac{\log n}{\log \log n} \right) \) time is allocated.

5. Even if there is a single unsuccessful element, start all over again (from step 2).

**Analysis:** in the next lecture.