# CSE 6220 Homework 4

Karl Hiner, @khiner6

## 1

Consider the following modification to the *Sample Sort* algorithm described in class. Suppose instead of choosing $p-1$ local splitters per processor we choose $kp-1$ local splitters per processor. Prove that this modification guarantees the number of elements received by any processor is bounded by $\frac{n}{p}\left(1+\frac{1}{k}\right)$.

We can calculate the maximum number of elements received by any single processor by observing that the elements received will necessarily contain local splitters originating from other processors $P_i$.

The *total* number of local splitters contained in the elements received by each processor will be $kp-1$. Since each processor contains $\frac{n}{p}$ elements, each local split will subdivide

$$\frac{\frac{n}{p}}{kp-1} = \frac{n}{p}\cdot\frac{1}{kp-1}$$

elements. Thus, if a processor receives $s_i$ local splitters that originally came from $P_i$, we can bound the number of received elements $E_i$ that originated in $P_i$ by

$$E_i < (s_i+1)\left(\frac{n}{p}\cdot\frac{1}{kp-1}\right).$$

Finally, the total number of elements $E$ received by any processor will be bounded by

$$\begin{aligned}
E &< \sum_{i=0}^{p-1}(s_i+1)\left(\frac{n}{p}\cdot\frac{1}{kp-1}\right) \\
&= \left(\sum_{i=0}^{p-1}(s_i+1)\right)\left(\frac{n}{p}\cdot\frac{1}{kp-1}\right) && \text{(separate terms w/o } i\text{)} \\
&= \left(\sum_{i=0}^{p-1}s_i+p\right)\left(\frac{n}{p}\cdot\frac{1}{kp-1}\right) && \text{(evaluate } +1 \text{ in sum)} \\
&= (kp-1+p)\left(\frac{n}{p}\cdot\frac{1}{kp-1}\right) && \text{(by def. of } s_i,\ \sum_{i=0}^{p-1}s_i = kp-1\text{)} \\
&= \frac{n}{p}\cdot\frac{kp-1+p}{kp-1} && \text{(rearrange)} \\
&= \frac{n}{p}\left(1+\frac{p}{kp-1}\right) \approx \frac{n}{p}\left(1+\frac{p}{kp}\right) && \text{(neglecting the } -1 \text{ in the denominator)} \\
&< \frac{n}{p}\left(1+\frac{1}{k}\right). && \text{(QED)}
\end{aligned}$$

## 2

Show that a $p$-processor ring can be embedded into a $p$-processor array with load = 1 and dilation = 2. Specify the mapping function from ring to array that works for arbitrary values of $p$.

For load = 1, we must not have multiple nodes in the ring (source) mapped to a single node in the array (target). Since we are given that the source and target have the same number of nodes, having load = 1 implies that we must have a one-to-one mapping between nodes.

For dilation = 2, every edge in the ring must map to at most two edges in the array. Put another way, we must be able to travel between any two nodes that were originally neighbors in the ring with at most two hops in the array.

We can achieve this by interleaving the first half of the ring's nodes with the *reversed* second half of its nodes. Then, the ring's first node will have a direct link to its last node, the second node will have a direct link to the second-to-last node, and so on.

Here is an example:

```
Ring:  0-1-2-3-4-5-6-7-8-9 (9-0)
Array: 0-9-1-8-2-7-3-6-4-5
```

Here is the mapping `rtoa` from processor rank $r$ in the ring to its corresponding rank in the array:

$$\texttt{rtoa}(r) = \begin{cases} 2r & \text{if } 2r < p \\ 2(p-r)-1 & \text{otherwise} \end{cases}$$

## 3

A three dimensional torus of size $16 \times 16 \times 8$ is embedded in a 2048-processor parallel computer that can route hypercubic permutations.

(a) Determine rank of the processor to which (9,13,6) is mapped.

$$\begin{aligned} (9,13,6) &= \texttt{btog}(9_{10}) \| \texttt{btog}(13_{10}) \| \texttt{btog}(6_{10}) \\ &= \texttt{btog}(1001_2) \| \texttt{btog}(1101_2) \| \texttt{btog}(110_2) \qquad \text{(using (4,4,3) bits)} \\ &= \texttt{1101} \| \texttt{1011} \| \texttt{101} \\ &= \texttt{11011011101}_2 = 1757_{10} \end{aligned}$$

Thus, processor (9,13,6) has hypercube rank 1757.

(b) What is the torus rank of processor 532?

We first convert 532 to binary, and then split it into segments of lengths (4,4,3) to find the gray codes of the three dimensions.

$$532_{10} = \texttt{01000010100}_2 = \texttt{0100} \| \texttt{0010} \| \texttt{100},$$

and so we have gray codes $x = \texttt{0100}$, $y = \texttt{0010}$, and $z = \texttt{100}$.

$$\begin{aligned} (\texttt{gtob}(x), \texttt{gtob}(y), \texttt{gtob}(z)) &= (\texttt{gtob}(\texttt{0100}), \texttt{gtob}(\texttt{0010}), \texttt{gtob}(\texttt{100})) \\ &= (\texttt{0111}_2, \texttt{0011}_2, \texttt{111}_2) \\ &= (7_{10}, 3_{10}, 7_{10}). \end{aligned}$$

Thus, the torus rank of processor 532 is (7,3,7).

## 4

Consider the embedding of a 16-leaf complete binary tree into a 16-node hypercube assuming only one tree level at a time is involved computationally. Consider processor with rank 12 in the hypercube.

(a) Which levels of the tree does this processor participate in?

Let $j$ denote the number of trailing zeros in the binary representation of the processor rank $r$. Then, the processor with rank $r$ participates in levels $\log p, \ldots, \log p - j$.

For $p = 16, j = \text{TrailingZeros}(\texttt{1100}) = 2$, we have

$$\begin{aligned} &\log p, \ldots, \log p - j \\ &\log 16, \ldots, \log 16 - 2 \\ &4, \ldots, 4 - 2 \\ &4,3,2. \end{aligned}$$

Thus, the processor with rank 12 participates in levels 4, 3, and 2.

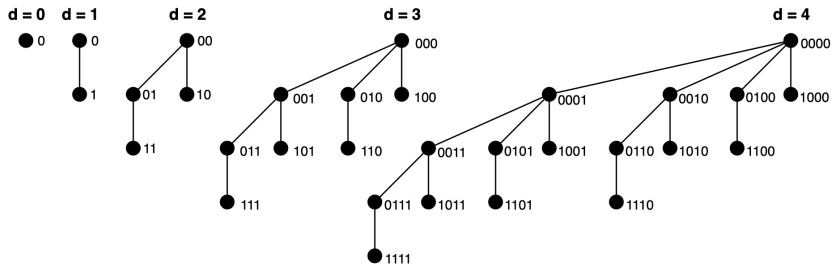
(b) For each level above, compute the tree rank of the node, and the hypercube rank of its parent, left child, and right child.

- Level 4:
  - **Tree rank: (4,12)**. The rank within the leaf level is the same as the rank of the processor in the hypercube.
  - **Parent hypercube rank: 12**. With $k$ denoting the level, change $(\log p - k)^{th}$ bit to 0. $\implies$ Zero $(4 - 4 = 0)^{th}$ bit, which is already 0.
  - **Left/right child hypercube ranks: Leaf nodes have no children.**
- Level 3:
  - **Tree rank: (3,6)**. Leading $k = 3$ bits of hypercube rank of $\texttt{1100} = \texttt{110}_2 = 6_{10}$.
  - **Parent hypercube rank: 12**. Zero bit $4 - 3 = 1$, which is already 0.
  - **Left/right child HC ranks: 12 and 13**. Left child is always self. For right child, change $(\log p - k - 1 = 4 - 3 - 1 = 0)^{th}$ bit to 1.
- Level 2:
  - **Tree rank: (2,3)**. Leading $k = 2$ bits of hypercube rank of $\texttt{1100} = \texttt{11}_2 = 3_{10}$.
  - **Parent hypercube rank: 8**. Zero bit $4 - 2 = 2$ of $\texttt{1100} \rightarrow \texttt{1000}_2 = 8_{10}$
  - **Left/right child HC ranks: 12 and 14**. For right child, change bit $4 - 2 - 1 = 1$ to 1: $\texttt{1100} \rightarrow \texttt{1110}_2 = 14_{10}$.

## 5

A Binomial tree of height $d$, termed $B(d)$ is defined as follows: If $d = 0$, then $B(0)$ is a single node. Otherwise, $B(d)$ is constructed by taking two binomial trees of height $d - 1$ and making the root of one tree the child of the root of the other tree. Show that $B(d)$ can be embedded in a $d$-dimensional hypercube.

Here is an algorithm for constructing a mapping from $B(D)$ to a $D$-dimensional hypercube:




- **Construct** $B(0)$**:** $B(0)$ is a single rootnode, so we can embed it in a 0-dimensional hypercube by mapping the node to hypercube processor 0.
- **Iterate over** $0 < d \leq D$**:**
  - Append 0 to the binary representation of the hypercube rank of all nodes in $B(d-1)$ until the length of the binary representation is $d$. (For $d = 1$, this is a no-op. For all other $d$, this is a left shift.)
  - Create a copy of the tree with these $d$-dimensional ranks, add 1 to the ranks of all nodes in the copy, and add the resulting tree as a child of the root.