Implementing Bitonic Sort

Mark Greenstreet and Ian M. Mitchell

CpSc 418 – February 7, 2018

- Finish deriving the bitonic sort algorithm
- Sorting networks in practice
  - Merging networks
  - Higher-Radix algorithms

The handy lemma, bitonic-version

- Let X be a bitonic sequence of 0s and 1s. Let $N = \texttt{length}(X)$. Let

$$\begin{array}{rcll} Z_i & = & \min(X_i, X_{i+\frac{N}{2}}), & 0 \leq i < \frac{N}{2} \\ Z_i & = & \max(X_{i-\frac{N}{2}}, X_i), & \frac{N}{2} \leq i < N \end{array}$$

  - Then either $Z_0, \ldots, Z_{\frac{N}{2}-1}$ is all 0s or $Z_{\frac{N}{2}}, \ldots, Z_{N-1}$ is all 1s, and
  - The other half of Z is bitonic.
  - Note: this implies that every element in the lower half is $\leq$ every element in the upper half.

Proving the handy lemma, bitonic-version

- Assume $X \in 0^*1^*0^*$. The other case is analagous.
- Let i be the index of the first 1 in X. $i = N$ if X is all 0s.
- Let j be the index of the first 0 in the *second* segment of 0s in X. $j = N$ if $X \in 0^*1^*$.
- Note that $j - i$ is the number of 1s in X.

Proof: case $j - i \leq \frac{N}{2}$

- $X_0, \ldots, X_{i-1}$ are all 0s. Therefore
- $Z_0, \ldots, Z_{i-1}$ are all 0s.
- $X_{i+\frac{N}{2}}, \ldots, X_{N-1}$ are all 0s because $i + \frac{N}{2} \geq j$. Therefore
- $Z_i, \ldots, Z_{\frac{N}{2}-1}$ are all 0s.
- $\therefore$
- $Z_0, \ldots, Z_{\frac{N}{2}-1}$ are all 0s.
- If $i \leq \frac{N}{2}$, then
  - $Z_{\frac{N}{2}}, \ldots Z_{j-1}$ are all 1s,
  - $Z_j, \ldots, Z_{i+\frac{N}{2}-1}$ are all 0s,
  - $Z_{i+\frac{N}{2}}, \ldots Z_{N-1}$ are all 1s,
  - $\therefore$
  - $Z_{\frac{N}{2}}, \ldots, Z_{N-1}$ is bitonic.
- Otherwise, $i \geq \frac{N}{2}$,
  - $X_0, \ldots, X_{\frac{N}{2}-1}$ are all 0s.
  - $Z_{\frac{N}{2}}, \ldots, Z_{N-1}$ is the same sequence as $X_{\frac{N}{2}}, \ldots, X_{N-1}$.
  - $\therefore$
  - $Z_{\frac{N}{2}}, \ldots, Z_{N-1}$ is bitonic.

The proof when $j - i \geq \frac{N}{2}$ is analagous.

Bitonic Merge

To merge N items:

- Perform compare-and-swaps with a stride of $N/2$.
- Now, every element in the top half is greater than every element in the bottom half. Each half is bitonic.
- Continue with a bitonic merge of $N/2$ items for each half.
  - Then four bitonic merges of $N/4$ values;
  - Then eight bitonic merges of $N/8$ values;
  - ...
  - Finally $N/2$ merges of 2 values. The base-case – each merge of two values can be done with a singel compare and swap.
- How many compare and swap operations?
- What is the parallel time?

Bitonic Sort: The big picture

Sort N values

- Divide into two halves of size $\frac{N}{2}$.
  - **Parallel:** sort each half.
  - This is a typical, divide-and-conquer approach.
  - Now, we just need to merge the two halves.
- Combine the two, sorted halves into one **bitonic** sequence of length N.
- Use the method described on slide 2 to create a clean half of length $\frac{N}{2}$ and a bitonic half of length $\frac{N}{2}$.
- Recursively merge the two halves.
  - **Parallel:** merge each half.
  - The recursion works on sequences of length N, $\frac{N}{2}$, $\frac{N}{4}$, $\ldots$, 2.
  - Total parallel time: $\log_2 N$.
  - Total number of compare-and-swaps $\frac{N}{2} \log_2 N$.

Complexity of Bitonic Sort

- The whole algorithm:
  - Use $\frac{N}{2}$ compare-and-swap operations in parallel to sort pairs of elements.
  - Perform a 4-way bitonic merge for each pair of length-2 sorted sequences to obtain a length-4 sorted sequence.
  - Perform a 8-way bitonic merge for each pair of length-4 sorted sequences to obtain a length-8 sorted sequence.
  - ...
  - Perform a N-way bitonic merge for the two length-$\frac{N}{2}$ sorted sequences to obtain the length-N sorted sequence.
- Complexity
- Parallel time:

$$\sum_{k=1} \log_2 Nk = O(\log^2 N)$$

- Total number of compare and swaps: $O(N \log^2 N)$.

Midterm: Feb. 28

This lecture is the cut-off for material that will be covered on the midterm.