The Bitonic Sort Algorithm

Mark Greenstreet

CpSc 418 – October 17, 2018

Outline

- Review: The 0-1 principle
- Bitonic Sort
  - Let's take another look at Merge Sort
  - If we use a sorting network, we only have to consider 0s and 1s
  - The easy cases
  - The general case
  - The whole algorithm

Review: The 0-1 Principle

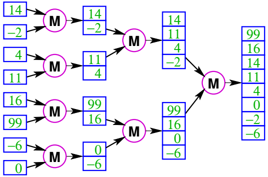
- Statement: **If a sorting network sorts all inputs consisting of 0s and 1s correctly, then it sorts all inputs correctly**.
- Proof (summary):
  - Monotonic functions commute with compare-and-swap
  - Monotonic functions commute with sorting networks
  - Given some input that a sorting network does not sort correctly:
    - ★ Choose a threshold-function that can be applied to the outputs of the sorting network to produce an unsorted output of 0s and 1s.
      - ✾ The threshold function is monotonic.
    - ★ Move the threshold function to the inputs of the sorting network.
    - ★ We now have an input consisting of only 0s and 1s that the sorting network does not sort correctly.
  - $\therefore$ If there is **any** input that the sorting network does not sort correctly, there is an input consisting only of 0s and 1s that it does not sort correctly.
  - **Contrapostive:** If a sorting network sorts all inputs consisting of 0s and 1s correctly, then it sorts all inputs correctly.
  - □

Bitonic Sort: Overview

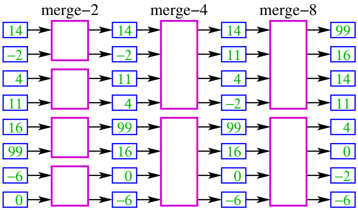
- Merge sort is a great sequential sorting algorithm
  - But, the final merge step(s) is (are) a sequential bottleneck.
- What if we could merge in parallel?
  - We'll use sorting networks. Only need to think about 0s and 1s.
  - We'll see that parallel merge is easy for some special cases.
  - Then generalize to any inputs of 0s and 1s.
- Once we have a parallel merge, then parallel sort is "easy"

Merge Sort

- Sequential



- Parallel



Parallel Merge

- Recursion assumption: input is two, **sorted** vectors of equal length.
- Use the 0-1 principle: Inputs are 0s and 1s.
- Super easy case, the two input vectors have the same number of 1s.
  - How do we merge them?
  - See, we use counting arguments.
- Easy case: the number of 1's in the two input vectors differ by at most 1.
- General case: arbitrary input vectors of the same length, consisting of 0s and 1s.
  - This is where **bitonic sequences** make there appearance.

Monotonic sequences

- A sequence, $X_0, X_1, \ldots, X_{N-1}$ is **monotonically increasing** if

$$X_0 \leq X_1 \leq \cdots \leq X_{N-1}$$

- A sequence, $X_0, X_1, \ldots, X_{N-1}$ is **monotonically decreasing** if

$$X_0 \geq X_1 \geq \cdots \geq X_{N-1}$$

- A sequence is **monotonic** if it is either monotonically increasing or monotonically decreasing.
- A sequence is **strictly** monotonically increasing if

$$X_0 < X_1 < \cdots < X_{N-1}$$

  - Likewise for strictly monotonically decreasing or strictly monotonic.
  - We won't use the "strict" versions very much – they aren't very useful with 0-1 sequences. ☺

A handy lemma

- Let X be a monotonically increasing sequence of 0s and 1s of length N. Let Y be a monotonically decreasing sequence of 0s and 1s of length N.
- Let Z be the sequence of length $2N$ with

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

- Then, either $Z_0, Z_1, \ldots, Z_{N-1}$ are all 0s, or $Z_N, Z_{N+1}, \ldots Z_{2N-1}$ are all 1s.
- Proof (details on the whiteboard):
  - Let $\texttt{zcount}(X)$ denote the number of 0s in X.
  - If $\texttt{zcount}(X) + \texttt{zcount}(Y) \geq N$, then $Z_0, \ldots, Z_{N-1}$ are all 0s.
  - If $\texttt{zcount}(X) + \texttt{zcount}(Y) \leq N$, then $Z_N, \ldots, Z_{2N-1}$ are all 1s.
  - $\square$
- What about the other half?
  - It's either $0^*1^*0^*$ or $1^*0^*1^*$.

Bitonic Sequences

- A sequence is **bitonic** if it consists of a monotonically increasing sequence followed by a monotonically decreasing sequence.
  - Either of those sub-sequences can be empty.
  - We'll also consider a monotonically decreasing followed by monotonically increasing sequence to be bitonic.
- Properties of bitonic sequence
  - Any subsequence of a bitonic sequence is bitonic.
  - Let A be a bitonic sequence consisting of **0s** and **1s**. Let A_0 and A_1 be the even- and odd-indexed subsequences of A.
  - If the length of A is even, then number of **1s** in A_0 and A_1 differ by at most 1.
    - ★ Likewise for the number of 0s.

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 element in the lower half is $\leq$ every element in the upper half.
- Proof (the easy cases):
  - If $X_0, \ldots, X_{\frac{N}{2}-1}$ is all 0s, then $Z = X$, $Z_0, \ldots, Z_{\frac{N}{2}-1}$ is all 0s, and $Z_{\frac{N}{2}}, \ldots, Z_{N-1}$ is bitonic – it's a subsequence of a bitonic sequence.
  - Likewise, if $X_0, \ldots, X_{\frac{N}{2}-1}$ is all 1s, or if $X_{\frac{N}{2}}, \ldots, X_{N-1}$ is all 0s or all 1s.
  - Need to consider the case when both $X_0, \ldots, X_{\frac{N}{2}-1}$ and $X_{\frac{N}{2}}, \ldots, X_{N-1}$ are mixed.

Case: both halves of X are mixed

Consider the case where $X \in 0^*1^*0^*$ – the other case is equivalent.

- Let i be the smallest integer with $0 \leq i < \frac{N}{2}$ such that $X_i = 1$.
- Let j be the smallest integer with $\frac{N}{2} \leq j < N$ such that $X_j = 0$.
- If $j - i \leq \frac{N}{2}$, then
  - $Z_0, \ldots, Z_{\frac{N}{2}-1}$ is all 0s, and
  - $Z_{\frac{N}{2}}, \ldots, Z_{N-1} \in 1^*0^*1^*$.
- If $j - i \geq \frac{N}{2}$, then
  - $Z_0, \ldots, Z_{\frac{N}{2}-1} \in 0^*1^*0^*$.
  - $Z_{\frac{N}{2}}, \ldots, Z_{N-1}$ is all 1s.
- $\square$

Bitonic Merge

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 10 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: $O(N \log^2 N)$.