Bitonic Sorting, Part 2

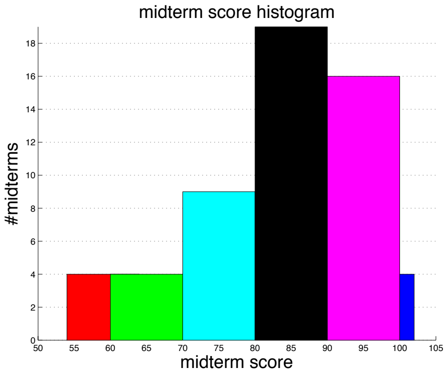
Mark Greenstreet

CpSc 418 – Nov. 12, 2013

Lecture Outline

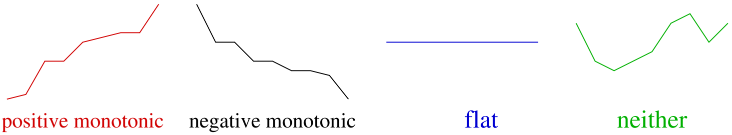
- Recap of Bitonic Sort
- The general version of the algorithm
- An implementation

The Midterm



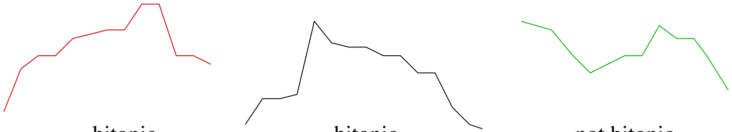
- Will return at the end of class.
- min=54, max=102, median=86, mean=83.43, std=12.54.
- High distribution with a somewhat long tail.
- Everyone passed. ☺
- Question 1 had unexpectedly low scores. ☹
- Question 3 should have been more quantitative and challenging.

Monotonic Sequences



- Let $X_0, X_1, \ldots, X_{N-1}$ be a sequence.
- If $X_0 \leq X_1 \leq \cdots \leq X_{N-1}$, then we say that X is positive-monotonic.
- If $X_0 \geq X_1 \geq \cdots \geq X_{N-1}$, then we say that X is negative-monotonic.
- If $X_0 = X_1 = \cdots = X_{N-1}$, then we say that X is flat.
- We will say that X is monotonic to mean:
  - X is positive-monotonic (the default meaning)
  - X is either positive- or negative-monotonic – sometimes we might use this sense to avoid saying "positive- or negative-monotonic" over and over. If so, I'll make it clear that we are using this more general sense of monotonic.

Bitonic Sequences



bitonic | bitonic | not bitonic

- Let $X_0, X_1, \ldots, X_{N-1}$ be a sequence.
- If there exists I with $0 \le I \le N$ such that

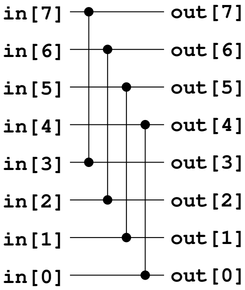
$$
\begin{array}{ll}
 & (\forall 0 \le J < I.\ X_J \le X_{J+1}) \quad \wedge \quad (\forall I \le J < N.\ X_J \ge X + J + 1) \\
\vee & (\forall 0 \le J < I.\ X_J \ge X_{J+1}) \quad \wedge \quad (\forall I \le J < N.\ X_J \le X + J + 1)
\end{array}
$$

  Then we say that X is bitonic.
- When it matters, we'll call the first case "up-down" bitonic and the second case "down-up" bitonic.

Properties of Bitonic Sequences

- Every monotonic sequence is bitonic – just choose $I = 0$ or $I = N$.
- Every flat sequence is monotonic and thus bitonic.
- If X is a bitonic sequence of length N and $0 \leq I_0 \leq I_1 \leq \cdots \leq I_{M-1} < N$, then $X_{I_0}, X_{I_1}, \ldots X_{I_{M-1}}$ is bitonic. In other words, every subsequence of a bitonic sequence is bitonic.
  - X is positive-monotonic (the default meaning)
  - X is either positive- or negative-monotonic – sometimes we might use this sense to avoid saying "positive- or negative-monotonic" over and over. If so, I'll make it clear that we are using this more general sense of monotonic.

Bitonic Merge

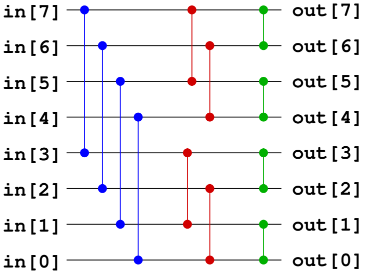


- Assume `in[0]` ... `in[`$N-1$`]` is bitonic.
- wlog, assume `in` is an array of 0s and 1s.
- If `in[0]` ... `in[`$\frac{N}{2}-1$`]` is flat-0:
  - then `out[0]` ... `out[`$\frac{N}{2}-1$`]` is flat-0;
  - `out[`$\frac{N}{2}\cdots(N-1)$`]` is the same as `in[`$\frac{N}{2}\cdots(N-1)$`]` and is therefore bitonic;
  - for every $0 \le i_1 < \frac{N}{2}$ and every $\frac{N}{2} \le i_2 < N$, `out[`i_1`]` $\le$ `out[`i_2`]`.
- If `in[0]` ... `in[`$\frac{N}{2}-1$`]` is flat-1:
  - then `out[`$\frac{N}{2}$`]` ... `out[`$N-1$`]` is flat-1;
  - `out[0]` ... `out[`$\frac{N}{2}-1$`]` is the same as `in[`$\frac{N}{2}\cdots(N-1)$`]` and is therefore bitonic;
  - for every $0 \le i_1 < \frac{N}{2}$ and every $\frac{N}{2} \le i_2 < N$, `out[`i_1`]` $\le$ `out[`i_2`]`.

Bitonic Merge (continued)

- If $\texttt{in}[0 \cdots \frac{N}{2} - 1]$ is monotonic but not flat:
  - then $\texttt{in}[\frac{N}{2} \cdots N - 1]$ must be montonic in the other direction (possibly flat);
  - at least one of $\texttt{out}[0 \cdots \frac{N}{2} - 1]$ or $\texttt{out}[\frac{N}{2} \cdots N - 1]$ must be clean – same argument as for Shear sort;
  - for every $0 \leq i_1 < \frac{N}{2}$ and every $\frac{N}{2} \leq i_2 < N$, $\texttt{out}[i_1] \leq \texttt{out}[i_2]$.
- If $\texttt{in}[0 \cdots \frac{N}{2} - 1]$ is bitonic but not monotonic:
  - then $\texttt{in}[\frac{N}{2} \cdots N - 1]$ must be flat;
  - either $\texttt{out}[0 \cdots \frac{N}{2} - 1]$ or $\texttt{out}[\frac{N}{2} \cdots N - 1]$ is flat,
  - and the other is bitonic;
  - for every $0 \leq i_1 < \frac{N}{2}$ and every $\frac{N}{2} \leq i_2 < N$, $\texttt{out}[i_1] \leq \texttt{out}[i_2]$.
- In all cases:
  - either $\texttt{out}[0 \cdots \frac{N}{2} - 1]$ or $\texttt{out}[\frac{N}{2} \cdots N - 1]$ is flat,
  - and the other is bitonic;
  - for every $0 \leq i_1 < \frac{N}{2}$ and every $\frac{N}{2} \leq i_2 < N$, $\texttt{out}[i_1] \leq \texttt{out}[i_2]$.

Bitonic Merge (continued)



- Each phase of the merge requires:
  - Inputs to the merger (dashed box) must be bitonic.
- Each phase of the merge ensures:
  - All values in upper half greater than or equal to all outputs in lower half.
  - Both halves are bitonic.

- **If the input to the whole merger is bitonic, then, the output is monotonic.**

Bitonic Sort

- We can merge, but can we sort?
  - Arrays of one element are already sorted.
  - We can merge two one-element arrays with a single compare-and swap.
  - Given arrays of length N sorted in opposite directions
    - concatenate them to make a bitonic array of length $2N$.
    - perform a bitonic merge to obtain a sorted array of length $2N$.
- In Erlang:

Bitonic Sort

- Sorting can be done using bitonic merges of width 2, 4, ... N.
  - A merge of width k has depth $log_2 k$ and uses $\frac{k}{2} \log_2 k$ compare-and-swap modules.
  - When sorting N items, we use N/k merges of width k in parallel for a step that requires width-k merging.
  - $\therefore$ bitonic sort has depth $\binom{\log_2 N}{2}$ and uses $\frac{N}{2}\binom{\log_2 N}{2}$ comparators.
  - That's $O(log_2^2 N)$ parallel time and $O(N \log_2^2 N)$ comparisons.

When N is odd

Bandwidth Considerations

The rest of the course

Build on what we've covered to make it solid.
Here are the topics that I have planned. I'm also happy to cover past homework problems in detail or the midterm. I can do some stuff on the current homework, and describe solutions in detail after the due date. What's the due date?

- Nov. 14: Mutual Exclusion
- Nov. 19: Mesh sorting, and distributed Erlang
- Nov. 21: MPI or Map-Reduce
- Nov. 26: GPUs
- Nov. 28: The future, or my research, or course review, or ...

Review

- Let *A* and *B* be positive monotonic sequences of the same length. Show that

  ```
  [ max(X, Y) || {X, Y} <- zip(A, B) ]
  ```

  is positive monotonic.
- Let *A* be a positive monotonic sequence and *B* be a negative monotonic sequence of the same length as *A*. Show that

  ```
  [ max(X, Y) || {X, Y} <- zip(A, B) ]
  ```

  is bitonic.