

Sorting Networks

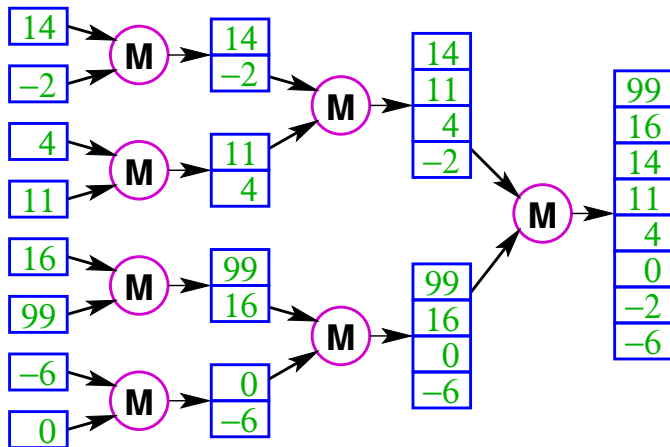
Mark Greenstreet

CpSc 418 – Nov. 15, 2012

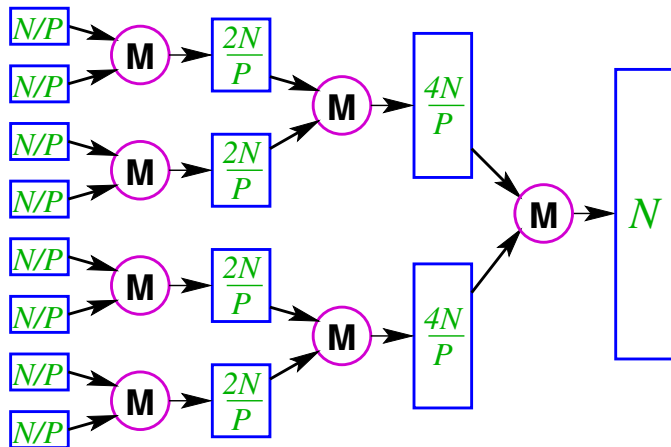
Lecture Outline

- Parallelizing mergesort and/or quicksort
- Sorting Networks
- Bitonic Sorting

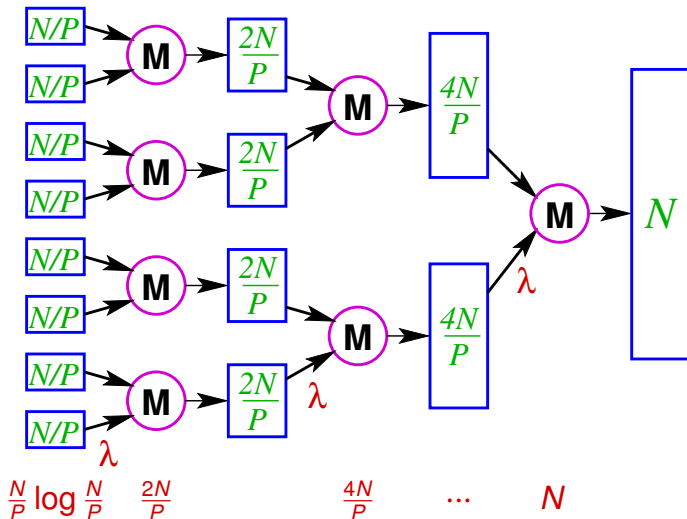
Parallelizing Mergesort



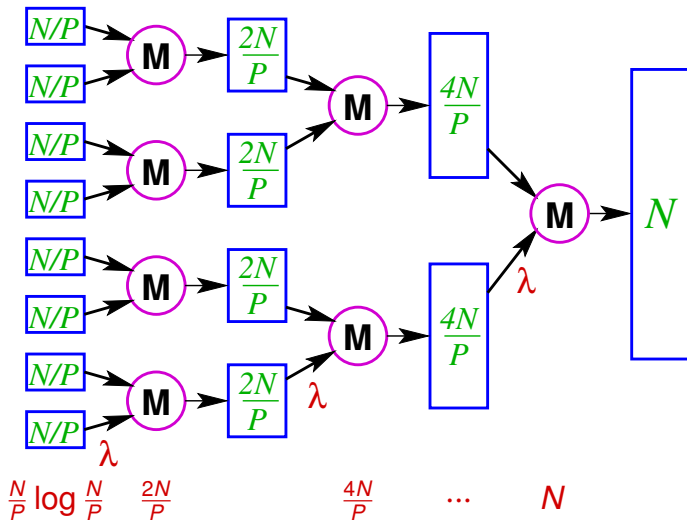
Parallelizing Mergesort



Parallelizing Mergesort



Parallelizing Mergesort

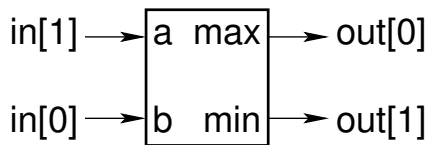


Total time: $\frac{N}{P} (\log N + 2(P - 1) - \log P) + (\log P)\lambda$

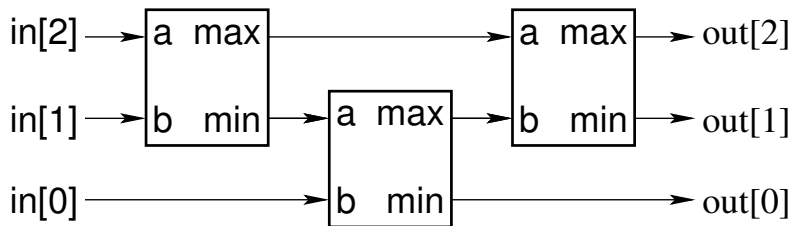
Parallelizing Quicksort

Sorting Networks

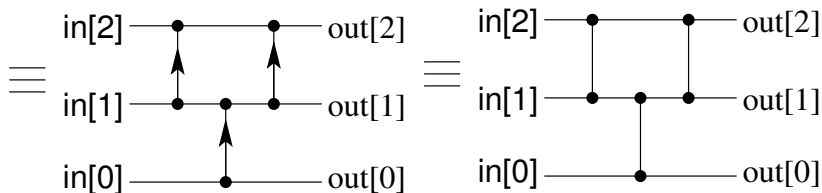
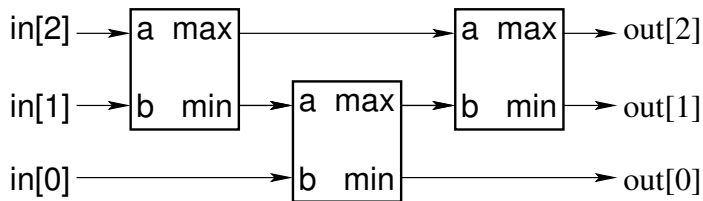
Sorting Network for 2-elements



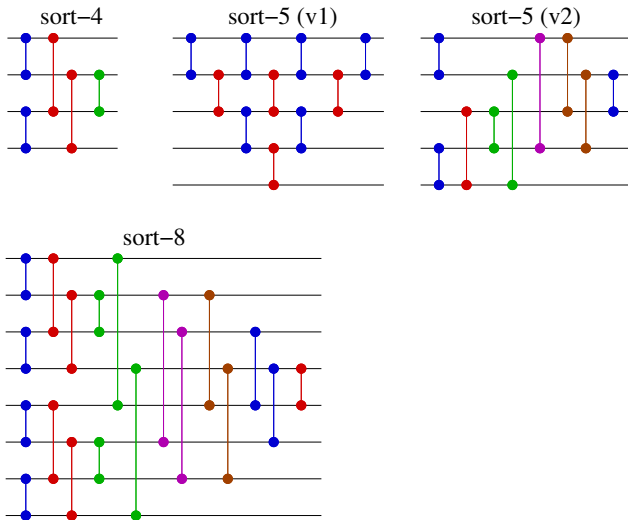
A Sorting Network for 3-elements



Sorting Networks – Drawing



Sorting Networks – Examples



See: <http://pages.ripco.net/~jgamble/nw.html>

Sorting Networks: Definition

Structural version:

An N -input sorting network is either:

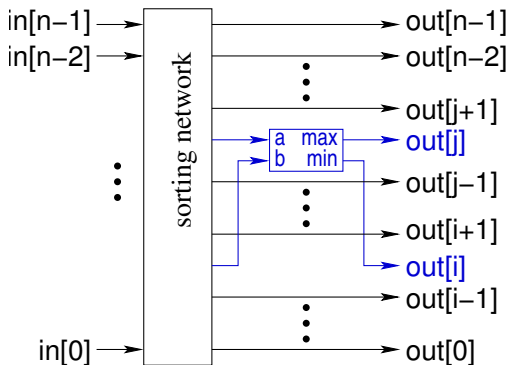
The identity function

$in[n-1] \rightarrow out[n-1]$
 $in[n-2] \rightarrow out[n-2]$

⋮

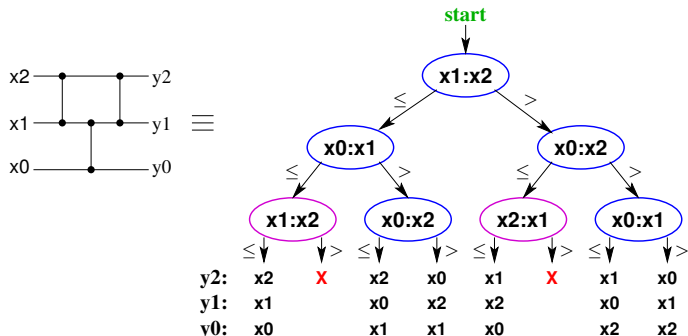
$in[0] \rightarrow out[0]$

A sorting network composed with a compare-and-swap element



Sorting Networks: Definition

Decision-tree version:



- Let v be an arbitrary vertex of a decision tree, and let x_i and x_j be the variables compared at vertex v .
- A decision tree is a sorting network iff for every such vertex, the left subtree is the same as the right subtree with x_i and x_j exchanged.

The 0-1 Principle

If a sorting network correctly sorts all inputs consisting only of 0's and 1's, then it correctly sorts inputs of any values.

Monotonicity Lemma

Lemma: sorting networks commute with monotonic functions.

- Let \mathbb{D} and \mathbb{E} be two domains, each with an ordering relation.
- $f : \mathbb{D} \rightarrow \mathbb{E}$ is monotonic iff

$$\forall x, y \in \mathbb{D}. x \leq y \rightarrow f(x) \leq f(y)$$

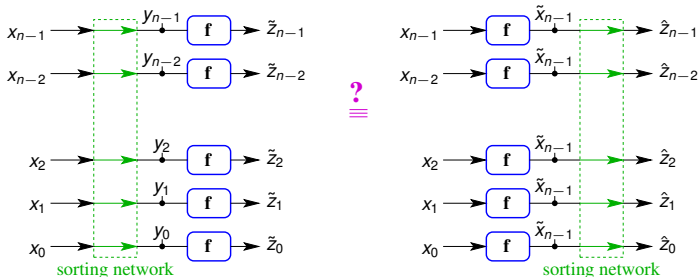
- We extend f element-wise to vectors:

$$f([x_0, x_1, \dots, x_{n-1}]) = [f(x_0), f(x_1), \dots, f(x_{n-1})]$$

- We can view an n -input sorting network, S as a function on vectors of length n .
- The monotonicity lemma states that $f \bullet S \equiv S \bullet f$.
- We prove the monotonicity lemma by induction on the structure of the sorting network (next slide).

Monotonicity Lemma (proof)

By induction. Base case:

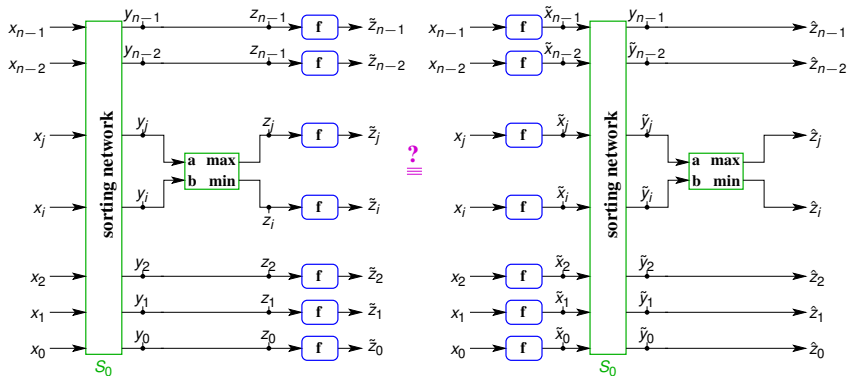


The sorting network, S , is the identity function.

$$f \bullet S = f \bullet \text{ident} = f = \text{ident} \bullet f = S \bullet f$$

Monotonicity Lemma (proof)

Induction step: Let S_0 be a sorting network, and append a compare-and-swap to outputs i and j .



Monotonicity Lemma (proof)

Definitions:

- S_0 is a sorting network, and
- $\text{cas}_{i,j}$ is a compare-and-swap unit that compares the i^{th} and j^{th} outputs of S_0 to produce the i^{th} and j^{th} outputs of S .
- Without loss of generality, assume that the smaller value is output to the the i^{th} output of S .
- Let x denote any input vector to $S \bullet f$ (or $f \bullet S$).
- Let $y = S_0(x)$, $z = S(x)$, $\tilde{x} = f(x)$, $\tilde{y} = f(y)$, and $\tilde{z} = f(z)$, and $\hat{z} = \text{cas}_{i,j}(S_0(f(x)))$.
- We need to show that $\hat{z} = \tilde{z}$.

Monotonicity Lemma (proof)

Induction step: show $\hat{z} = \tilde{z}$.

- For any $k \notin \{i, j\}$,

$$\begin{aligned}\hat{z}_k &= (\text{cas}_{i,j}(\mathcal{S}_0(f(x))))_k, && \text{definition of } \hat{z} \\ &= (\mathcal{S}_0(f(x)))_k, && \text{definition of } \text{cas}_{i,j} \\ &= (f(\mathcal{S}_0(x)))_k && \text{induction hypothesis} \\ &= \tilde{y}_k, && \text{definition of } \tilde{y} \\ &= \tilde{z}_k, && \text{definition of } \text{cas}_{i,j}\end{aligned}$$

- The i^{th} output:

$$\begin{aligned}\hat{z}_i &= (\text{cas}_{i,j}(\mathcal{S}_0(f(x))))_k, && \text{definition of } \hat{z} \\ &= \min((\mathcal{S}_0(f(x)))_i, (\mathcal{S}_0(f(x)))_j), && \text{definition of } \text{cas}_{i,j} \\ &= \min((f(\mathcal{S}_0(x)))_i, (f(\mathcal{S}_0(x)))_j), && \text{induction hypothesis} \\ &= f(\min((\mathcal{S}_0(x))_i, (\mathcal{S}_0(x))_j), && f \text{ is monotonic} \\ &= f(\text{cas}_{i,j}((\mathcal{S}_0(x))_i, (\mathcal{S}_0(x))_j)), && \text{definition of } \text{cas}_{i,j} \\ &= \tilde{z}_i, && \text{definition of } \tilde{z}\end{aligned}$$

- The j^{th} output: equivalent to the argument for the i^{th} output.

The 0-1 Principle

If a sorting network correctly sorts all inputs consisting only of 0's and 1's, then it correctly sorts inputs of any values.

I'll prove the contrapositive.

- If a sorting network does not correctly sort inputs of any values, then it does not correctly sort all inputs consisting only of 0's and 1's.
- Let S be a sorting network, let x be an input vector, and let $y = S(x)$, such that there exist i and j with $i < j$ such that $y_i > y_j$.

- Let $f(x) = \begin{cases} 0, & \text{if } x < y_i \\ 1, & \text{if } x \geq y_i \end{cases}$

$$\tilde{y} = S(f(x))$$

- By the definition of f , $f(x)$ is an input consisting only of 0's and 1's.
- By the monotonicity lemma, $\tilde{y} = f(y)$. Thus,

$$\tilde{y}_i = f(y_i) = 1 > 0 = f(y_j) = \tilde{y}_j$$

- Therefore, S does not correctly sort an input consisting only of 0's and 1's.
- \square

Announcements and reminders

- Nov. 22: Review Lin & Snyder, Chapter 5, *Scalable Parallelism* (the Bitonic Sort example).
Start reading Lin & Snyder Chapter 6: *Programming with Threads*
- Nov. 24: Finish reading Lin & Snyder Chapter 6.

Review

- Why don't traditional, sequential sorting algorithms parallelize well?
- Try to parallelize another sequential sorting algorithm such as heap sort? What issues do you encounter?
- We proved that 0-1 principle for sorting networks. Show that the 0-1 principle does **not** apply to arbitrary programs. In particular, show a simple program (sequential is fine) that sorts all inputs of 0's and 1's correctly, but does not sort arbitrary inputs correctly.