CSE 202–Day 7
Sorting Networks

Neil Rhodes
UC San Diego
Comparators

• Two input wires containing values
• Two output wires with lesser value on top wire and greater value on bottom wire
• Compares two elements and exchanges if out of order
Network of Comparators

- Operate in unison (as soon as inputs are available)
- Interested in *depth* (time) of the network
  - Depth of wire: 0 for input wires
  - $\text{Max}(dx, dy) + 1$ for comparator with input wires $x$ and $y$ with depths $dx$ and $dy$
- Interested in *size* (number of comparators)
- No cycles
Network of Comparators Representation

- Comparator is vertical line comparing wires
- n inputs on the left
- n outputs on the right
Sorting Network

• Similar to Insertion Sort
• Depth: $O(\log n)$
• Size: $O(n^2)$
0-1 principle

- A sorting network sorts any sequence of arbitrary input iff it sorts any sequence of 0’s and 1’s
- Assume some network N sorts all 0 and 1 sequences but not some specific sequence of arbitrary input
- Let X be a sequence N doesn’t sort. Let X[I] be a value that appears to early in the output. Let Y be a sequence of 0,1 based on X such that. If X[j] < X[I], Y[j] = 0, otherwise Y[j] = 1
- Claim: N doesn’t sort Y correctly
0-1 principle

- Claim: If $x[j] < x[I]$ appears on a wire when $N$ operates on $X$, then 0 appears on the wire when $N$ operates on $y$ (same for $\geq$, 1)

- Proof by induction on depth
  - Base case: apparent
  - Inductive step: Assume true for depths < $d$, show true for depth $d$ wire.
    - If input wires are both < $x[I]$, outputs on $Y$ are both 0
    - If one input wire, $z$, < $x[I]$, comparator will output $z$ on top. By inductive hyp. Inputs on $Y$ will be 0 and 1 and comparator will output 0 on top.
0-1 principle

- To prove a sorting network sorts, need only prove it sorts all sequences of 0’s and 1’s (\(2^n\) possibilities)
Bitonic sorting network

• Bitonic sequence
  – Sequence of monotonically increasing and then monotonically decreasing values, or sequence that can be circularly shifted into such
  – Degenerate case: any sequence of \( \leq 2 \) values is bitonic
  – Examples
    • (1,4,8,6,3), (3,1,4,8,6), (8,6,3,1,4)
  – Binary sequences 0^i1^j0^k or 1^i0^j1^k are bitonic
Half-cleaner

- \(n/2\) comparators
- Input is a bitonic sequence
- Line \(i\) is compared with line \(i+n/2\)
- Depth 1
- On two inputs, a half-cleaner sorts
- Claim: Output is two bitonic sequences half the size
- Claim: One of the sequences is *clean* (top is all 0’s or bottom is all 1’s)
Half-cleaner

- Proof of bitonic claims (input is 00..11..10..0 case; other case similar)
  - If 1’s are only in bottom half, output is input
  - If 1’s are only in top half, they’re moved to the bottom; top is clean, bottom is bitonic
  - If 1’s are in both, but aren’t compared, 1’s go to bottom; top is clean, bottom is bitonic
  - If 1’s are in both and are compared, 0’s go to top; top is bitonic; bottom is clean
Bitonic Sorter

• Recursively combine half-sorters
  – Sorts bitonic sequences
  – Depth: \( \lg n \)
  – Size: \( O(n \log n) \)
Merging network

• Want to merge two sorted sequences.
• Key insight:
  – Given X and Y sorted sequences, X followed by Reverse(Y) is a bitonic sequence
  – Modification to inputs of bitonic sorter will create a merger (don’t actually reverse Y, modify first level of comparators of bitonic sorter to create a merger).
Sorting network

- Use Merge network to create sorting network (bottom-up version of Mergesort)
- $D(n) = D(n/2) + \log n$
- $D(n) = \text{__________}$
- Size: \text{__________}
AKS Sorting network

- Depth: __________
- Size: __________
- Huge constants make it impractical for most inputs
AKS Sorting Network

- Imagine a perfect halver network that takes \( n \) inputs and puts the smallest \( n/2 \) inputs into one side, and the largest \( n/2 \) inputs into another. These could be used to generate a recursive sorting network.
  
  - Problem:
    - \( \log n \) depth in recursion
    - \( \log n \) depth in halver
    - \( \log^2 n \) total depth
AKS Sorting network

• Don’t require perfection; settle for pretty good:
  – A constant (small) fraction of smaller $n/2$ inputs go into wrong half (and vice-versa)
  – Depth of pretty-good halver is constant time
  – Build pretty-good halver using expander graph
    • Bipartite graph $(U, V, E)$ is an alpha-beta expander if all subsets of size $\leq \alpha|U|$ of $U$ have at least $\beta$ times as many neighbors in $V$ (and vice-versa)