Spaghetti sort

- Input: $S = \{s_1, \ldots, s_n\} \subseteq [1, 2]$.
- Have much Spaghetti (this are longish and very narrow tubes of pasta).
- cut $i$th piece to be of length $s_i$, for $i = 1, \ldots, n$.
- take all these pieces of pasta in your hand.
- make them stand up vertically, with their bottom end lying on a horizontal surface.
- lower your handle till it hit the first (i.e., tallest) piece of pasta.
- Take it out, measure it height, write down its number
- and continue in this fashion till done.
- Linear time sorting algorithm.
- ...but sorting takes $\Omega(n \log n)$ time.
What is going on?

- Faster algorithm achieved by changing the computation model.
- Allowed new "strange" operations (cutting a piece of pasta into a certain length, picking the longest one in constant time, and measuring the length of a pasta piece in constant time).
- Using these operations we can sort in linear time.
- So, are there other useful computation models?

Circuits are fast...

- Computing the following circuit naively takes 8 units of time.
- Use parallelism!

Sorting with a circuit – a naive solution

- **Comparator** gate:
  - \( x' = \min(x, y) \)
  - \( y' = \max(x, y) \)
- Draw it as:
  - \( x \)
  - \( x' = \min(x, y) \)
  - \( y \)
  - \( y' = \max(x, y) \)
Our circuits would be depicted by horizontal lines, with vertical segments (i.e., gates) connecting between them. For example, see complete sorting network depicted on the right. The inputs come on the wires on the left, and are output on the wires on the right. The largest number is output on the bottom line. Somewhat surprisingly, one can generate circuits from known sorting algorithms.

**Definitions**

**Definition**
A *comparison network* is a DAG, with $n$ inputs and $n$ outputs, where each gate has two inputs and two outputs.

**Definition**

*depth* of a wire is 0 at input. For gate with two inputs of depth $d_1$ and $d_2$, the depth on the output wire is $1 + \max(d_1, d_2)$.

*depth* of comparison network is maximum depth of an output wire.

**Definition**
A *sorting network* is a comparison network such that for any input, the output is monotonically sorted. The *size* of a sorting network is the number of gates in the sorting network. The *running time* of a sorting network is just its depth.

**Sorting network based on insertion sort**

1. Inner loop of insertion sort is:

2. Insertion sort as a network:

**Lemma**

The sorting network based on insertion sort has $O(n^2)$ gates, and requires $2n - 1$ time units to sort $n$ numbers.
**The Zero-One Principle**

**Definition**

The **zero-one principle** states that if a comparison network sort correctly all binary inputs (for all inputs is 0 or 1) then it sorts correctly all inputs.

Need to prove the zero-one principle.

**Lemma**

A comparison network transforms input sequence  
\[ a = \langle a_1, a_2, \ldots, a_n \rangle \implies b = \langle b_1, b_2, \ldots, b_n \rangle \]

Then for any monotonically increasing function \( f \), the network transforms  
\[ f(a) = \langle f(a_1), \ldots, f(a_n) \rangle \implies f(b) = \langle f(b_1), \ldots, f(b_n) \rangle \]

**Proof continued**

1. **Claim:** if a wire carry a value \( a_i \), when the sorting network get input \( a_1, \ldots, a_n \), then for input \( f(a_1), \ldots, f(a_n) \) this wire would carry the value \( f(a_i) \).
2. **Proof by induction on the depth on the wire at each point.**
3. If point has depth 0, then its input and claim trivially hold.
4. Assume holds for all points in circuit of depth \( \leq q_i \), and consider a point \( p \) on a wire of depth \( i + 1 \).
5. \( G \): gate which this wire is an output of.
6. By induction, claim holds for inputs of \( G \). Now, the claim holds for the gate \( G \) itself. Apply above single gate proof for \( G \).
7. \( \implies \) claim holds at \( p \).

**0/1 sorting implies real sorting**

**Theorem**

If a comparison network with \( n \) inputs sorts all \( 2^n \) binary strings of length \( n \) correctly, then it sorts all sequences correctly.
Proof: 0/1 sorting implies real sorting

- Assume for contradiction that fails for input $a_1, \ldots, a_n$. Let $b_1, \ldots, b_n$ be the output sequence for this input.
- Let $a_i < a_k$ be the two numbers that are output in incorrect order (i.e. $a_k$ appears before $a_i$ in output).
- $f(x) = \begin{cases} 
0 & x \leq a_i \\
1 & x > a_i.
\end{cases}$
- By lemma for input $\langle f(a_1), \ldots, f(a_n) \rangle$, circuit would output $\langle f(b_1), \ldots, f(b_n) \rangle$.
- This sequence looks like: 000..0????f(a_k)????f(a_i)??1111
- but $f(a_i) = 0$ and $f(a_j) = 1$. Namely, the output is a sequence of the form ????1????0????, which is not sorted.
- bin. input $\langle f(b_1), \ldots, f(b_n) \rangle$ sorting net? fails. A contradiction.

Part I

A bitonic sorting network

Definition

A bitonic sequence is a sequence which is first increasing and then decreasing, or can be circularly shifted to become so.

Example

The sequences $(1, 2, 3, \pi, 4, 5, 4, 3, 2, 1)$ and $(4, 5, 4, 3, 2, 1, 1, 2, 3)$ are bitonic, while the sequence $(1, 2, 1, 2)$ is not bitonic.

Observation

A binary bitonic sequence (i.e., bitonic sequence made out only of zeroes and ones) is either of the form $0^i1^j0^k$ or of the form $1^i0^j1^k$, where $0^i$ (resp. $1^j$) denote a sequence of $i$ zeros (resp., ones).
Half cleaner...

**Definition**

*half-cleaner*: a comparison network, connecting line \( i \) with line \( i + n/2 \).

Half-Cleaner\([n]\) denote half-cleaner with \( n \) inputs. Note, that the depth of a Half-Cleaner\([n]\) is one.

Half cleaner half sorts a bitonic sequence...

**Lemma**

If the input to a half-cleaner (of size \( n \)) is a binary bitonic sequence then for the output sequence we have that

(i) the elements in the top half are smaller than the elements in bottom half, and

(ii) one of the halves is clean, and the other is bitonic.

Proof

**Proof.**

If the sequence is of the form \(0^i1^j0^k\) and the block of ones is completely on the left side (i.e., its part of the first \( n/2 \) bits) or the right side, the claim trivially holds. So, assume that the block of ones starts at position \( n/2 - \beta \) and ends at \( n/2 + \alpha \).

If \( n/2 - \alpha \geq \beta \) then this is exactly the case depicted above and claim holds. If \( n/2 - \alpha < \beta \) then the second half is going to be all ones, as depicted on the right. Implying the claim for this case. A similar analysis holds if the sequence is of the form \(1^i0^j1^k\).
Bitonic sorter - sorts bitonic sequences...

(i) recursive construction of $\text{BitonicSorter}[n]$.  
(ii) opening up the recursive construction, and  
(iii) the resulting comparison network.

Merging sequence

- Merging question: Given two sorted sequences of length $n/2$, how do we merge them into a single sorted sequence?
  - Concatenate the two sequences...
  - ... second sequence is being flipped (i.e., reversed).
  - Easy to verify that the resulting sequence is bitonic, and as such we can sort it using the $\text{BitonicSorter}[n]$.
- Given two sorted sequences $a_1 \leq a_2 \leq \ldots \leq a_n$ and $b_1 \leq b_2 \leq \ldots \leq b_n$, observe that the sequence $a_1, a_2, \ldots, a_n, b_n, b_{n-1}, b_{n-2}, \ldots, b_2, b_1$ is bitonic.

Bitonic sorter... the result

Lemma

$\text{BitonicSorter}[n]$ sorts bitonic sequences of length $n = 2^k$, it uses $(n/2)k = (n/2)\lg n$ gates, and it is of depth $k = \lg n$.

Merger: Using a bitonic sorter

Merging two sorted sequences into a sorted sequence

(i) $\text{Merger}$ via flipping the lines of bitonic sorter.  
(ii) $\text{BitonicSorter}$.  
(iii) $\text{Merger}$ after we “physically” flip the lines.  
(iv) Equivalent drawing of the resulting $\text{Merger}$.
**Lemma**

The circuit $\text{Merger}[n]$ gets as input two sorted sequences of length $n/2 = 2^{k-1}$, it uses $(n/2)k = (n/2)\log n$ gates, and it is of depth $k = \log n$, and it outputs a sorted sequence.

**Proof**

The number of gates is

$$G(n) = 2G(n/2) + \text{Gates}(\text{Merger}[n]).$$

Which is $G(n) = 2G(n/2) + O(n\log n) = O(n\log^2 n)$.

As for the depth, we have that $D(n) = D(n/2) + \text{Depth}(\text{Merger}[n]) = D(n/2) + O(\log(n))$, and thus $D(n) = O(\log^2 n)$, as claimed. 

---

(i) $\text{FlipCleaner}[n]$, and

(ii) $\text{Merger}[n]$ described using $\text{FlipCleaner}$. 

---

**Lemma**

The circuit $\text{Sorter}[n]$ is a sorting network (i.e., it sorts any $n$ numbers) using $G(n) = O(n\log^2 n)$ gates. It has depth $O(\log^2 n)$. Namely, $\text{Sorter}[n]$ sorts $n$ numbers in $O(\log^2 n)$ time.
Faster sorting networks

One can build a sorting network of logarithmic depth (see Ajtai et al. [1983]). The construction however is very complicated. A simpler parallel algorithm would be discussed sometime in the next lectures. BTW, the AKS construction Ajtai et al. [1983] is better than bitonic sort for $n$ larger than $2^{8046}$.