17 Cellular Automata and Sorting

17.1 SIMD Computers

A cellular automaton is composed of cells. Each cell is in one of a fixed set of states. Every cell is updated simultaneously. Thus we refer to the current overall state as the generation. The update rule is local: the next state of a cell is determined by its state and the states of its neighbors. One famous cellular automaton is the Game Of Life. But first some other examples...

Cellular automata have been used as models of parallel computers; in particular, for SIMD computation (Single Instruction Multiple Data).

17.2 Odd-Even Transposition Sort

Consider an array of $n$ cells arranged in a row. Assume each cell has a value. We need to sort say increasing from left to right. (We will assume that all values are distinct; this is not really necessary but it simplifies the discussion slightly.)

One algorithm is called odd-even transposition sort. (If you’ve studied sorting before, this is equivalent to Bubble Sort on an ordinary computer.)

At odd ticks, cells $i$ and $i + 1$ compare for all odd $i$, and swap values if out of order.

At even ticks, cells $i$ and $i + 1$ compare for all even $i$, and swap values if out of order.

For example:

\begin{align*}
4 & 3 & 6 & 1 & 2 & 5 \\
3 & 4 & 1 & 6 & 2 & 5 \\
3 & 1 & 4 & 2 & 6 & 5 \\
1 & 3 & 2 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6
\end{align*}
Clearly, the process will continue if the list is not sorted. Why does it converge? Well, focus on the smallest value. That value will repeatedly move left (except possibly at the first tick) until it is in the 1st cell and then stay there forever. Then focus on the 2nd smallest value: once the 1st cell is dead, the 2nd smallest value will migrate to the 2nd cell, and so on. This proves convergence, and provides an $O(n^2)$ algorithm.

But actually, the algorithm sorts in at most $n$ steps. The key to proving this is to show a generalization of the argument for the minimum value:

**Lemma 3** For all $j \in \{1, 2, \ldots, n\}$, within $n$ steps the first $j$ cells are occupied by the first $j$ values.

**Proof.** Fix your favorite value of $j$. Call the first $j$ values small, and the remaining $n - j$ values large. Let $S_i$ be the number of the cell containing the $i^{th}$ small value (reading left to right). Assume $S_1 > 1$. Then while $S_1$ might not change the first tick, thereafter every tick it decreases, since it has only large values to its left. Similarly, if $S_2 > 2$, then after the first two ticks it will definitely start decreasing. And so on. QED

### 17.3 Snake Sort or Shearsort

Consider now a grid of cells. That is, $n$ cells arranged in an $\sqrt{n}$ by $\sqrt{n}$ array. Each cell starts with a single (distinct) value as before. The algorithm is in phases.

- **At odd phases**, sort each row by the above algorithm.
- **At even phases**, sort each column by the above algorithm.

But: while columns are always sorted top-to-bottom, odd rows are sorting increasing left-to-right and even rows increasing right-to-left.

Do an example! It’s magic…

**References**