Batcher Sorting Networks using Recursive construction

Sorting networks are comparison networks that always sort their inputs, so it makes sense to begin our discussion with comparison networks and their characteristics. A sorting network consists of two types of items: *comparators* and *wires*. The *wires* are carrying values that traverse the network all at the same time. Each comparator connects two wires. When a pair of values, traveling through a pair of wires, encounter a comparator, the comparator swaps the values if and only if the top wire's value is greater than the bottom wire's value.
Let say C be the comparator getting x and y as data from input wires. C outputs x' and y' where $x' = min(x, y)$ and $y' = min(x, y)$.
Size. Total the number of comparators in the network.
Depth. the largest number of comparators that any input value can encounter on its way through the network. An input wire of a sorting network has depth 0. Now, if a comparator has two input wires with depths d_x and d_y, then its output wires have depth $max(d_x, d_y)$+1. There are no cycles of comparators in a sorting network, the depth of a wire is well defined.

Batchers Sorting Network

We will describe a simple recursive non-adaptive sorting procedure, named Batchers Algorithm after its discoverer.

The idea behind Batchers algorithm is the following claim. If you sort the first half of a list, and sort the second half separately, and then sort the odd-indexed entries $(1^{st}, 3^{rd}, \cdots)$ and the even indexed entries $(2^{nd}, 4^{th}, \cdots)$ separately, then you need make only one more comparison-switch per pair of keys to completely sort the list.
given any list of length n, if we sort the entries in the first half, then sort the entries in the second half, then sort the entries in odd positions, then sort the entries in even positions and lastly perform this same round of exchanges (second with third, fourth with fifth, and sixth with seventh), the list will end up sorted. Furthermore, and even more incredibly, the same fact holds for any list whose length is a multiple of 4 (as we shall see below); in that case, in the final step, we sort the $2l^{th}$ element with the $(2l+1)^{st}$ for $l = 1, 2 \cdots (n/2)$-1
Merger. Given two sorted sequences, if we reverse the order of the second sequence and th en concatenate the two sequences, the resulting sequence is bitonic (in case of binary input, If a sorting network sorts 01 inputs then it can solve for any number, called $Zero - One$ principle).

Given the sorted zero-one sequences X and Y we reverse Y to get Y^R. Concatenating X and Y^R yields a bitonic sequence. Thus, to merge the two input sequences X and Y, it suffices to perform a bitonic sort on X concatenated with Y^R.

Sorting network will be constructed from merging networks (A collection of comparison Gates arranged in certain fashion).

Lower bound on sorting network is given as;

- Each level of a sorting network can contain at most $\frac{n}{2}$ comparators
- Since the size of a sorting network is $\Omega(n \log n)$, the depth is $\Omega(\log n)$
- An elegant construction that achieves depth $O(\log_2 n)$ and size $O(n \log_2 n)$
- Much more complicated constructions have been given that achieve depth $O(\log n)$ and size $O(n \log n)$.

Procedure for Batcher's Sorting Network is given as;

Sort$(x_1, x_2, \cdots x_n)$**:**

Sort$(x_1, \cdots x_{\frac{n}{2}})$

Sort$(x_{\frac{n}{2}+1}, \cdots, x_n)$

Merge$(x_1, x_2, \cdots x_n)$

Merge$(x_1, x_2, \cdots x_n)$**:**

Merge$(x_i, \forall i$ Odd$)$

Merge$(x_i, \forall i$ Even$)$

For $i = 2$ to $n - 1$ step by 2 do

Compare(x_i, x_{i+1})

in this procedure *Compare* is the a single comparison gate in sorting network.

Analysis

Let analyse the Batchers Sorting Network.

Let $S(n)$ denote the the number of comparison needed to Sort n items, and $M(n)$ denote the number of comparisons needed to merge two sorted list of size $\frac{n}{2}$ each.

$$S(n) = 2S(n/2) + M(n),$$
$$M(n) = 2M(n/2) + n/2 - 1$$

for n = 2, a single comparator gate is suffiecent to sort and merge. Hence $M(2) = S(2) = 1$.

$$M(n) \leq \frac{n}{2} \log n = O(n \log n)$$
$$S(n) \leq \frac{n}{2} (\log n)^2 = O(n(\log n)^2)$$

Expander Graphs

The set of vertices of a bipartite graph is partitioned into two disjoint parts in such a way that each edge has one endpoint in each of the two parts; a matching is a set of pairwise disjoint edges; for each subset S of the vertex-set of a graph G, we write

$$N_G(S) = \{u : u \text{ is adjacent to at least one vertex in} S\}$$

By a bipartite (n, d, μ)-expander, we shall mean a bipartite graph G such that,

(i) G has n vertices in each part.

(ii) the edge-set that is union of d matchings,

(iii) Every nonempty set S of vertices in one part of G has S

By an strong $(2n, \varepsilon)$-halver, we shall mean a comparator network on $2n$ wires with the output wires collected in equally size blocks B_L, B_R so that, for every $k = 1, 2, \cdots n$.

(i) the network places at mostεk of its k smallest input keys into output block B_R and

(ii) the network places at most εk of its k largest input keys into output block B_L.

Theorem 23.1. *For every positive ε there is a positive integer d such that, for every positive integer n, there is a strong $(2n, \varepsilon)$-halver of depth d.*

Proof. □

Theorem 23.2. *For every choice of positive ε_B, ε_F, and δ, there is a positive integer d such that, for every choice of positive even integers a and f such that $\delta a \leq f \leq a$, there is an $(a, f, \varepsilon_B, \varepsilon_F)$-separator of depth d.*