Bitonic Sorting

1 Parallel Merge Sort

1.1 Serial Merge Sort

The runtime for serial merge sort using divide and conquer is $O(n \log n)$, which has been proved here.

1.2 Parallel Merge Sort

First explore some special cases:

1. Divide into 2 parts, when merge need to traverse n:

$$T(n,p) = T(\frac{n}{2}, \frac{p}{2}) + O(n) \tag{1}$$

2. Divide into 4 parts, the first merge needs to traverse $\frac{n}{2}$, the second merge needs to traverse n:

$$T(n,p) = T(\frac{n}{4}, \frac{p}{4}) + O(\frac{n}{2}) + O(n) \tag{2}$$

Now for p processors, divide into p parts:

$$T(n,p) = T(\frac{n}{p}, 1) + O(n + \frac{n}{2} + \frac{n}{4} + ... + \frac{n}{p/2}) \tag{3}$$

Recall the serial merge sort runtime:

$$T(n) = T(n,1) = O(n \log n) \tag{4}$$

Therefore we have:

$$T(n,p) = O(\frac{n}{p} \log \frac{n}{p} + n) \tag{5}$$

To use resources optimally, the number of processors should match the level of parallelism that the algorithm can effectively exploit. For parallel merge sort, this means having a number of processors that aligns with the depth of the recursion tree, which is $\log n$. Using more processors than this would not make the algorithm run faster because the bottleneck becomes the merging process, which has to wait for all sub-problems at a given level to be completed before it can proceed to the next level of merging.

2 Bitonic Sort

2.1 Overview

Let l_1 and l_2 be non-decreasing sorted lists, including:

$$l_1 : x_0, x_1, ..., x_{n-1} \tag{6}$$

$$l_2 : y_0, y_1, ..., y_{n-1} \tag{7}$$

Then we can generate a minimum array and a maximum array:

$$l_{min} : \min(x_0, y_{n-1}), \min(x_1, y_{n-2}), ..., \min(x_{n-1}, y_0) \tag{8}$$

$$l_{max} : \max(x_0, y_{n-1}), \max(x_1, y_{n-2}), ..., \max(x_{n-1}, y_0) \tag{9}$$

An example is shown below:

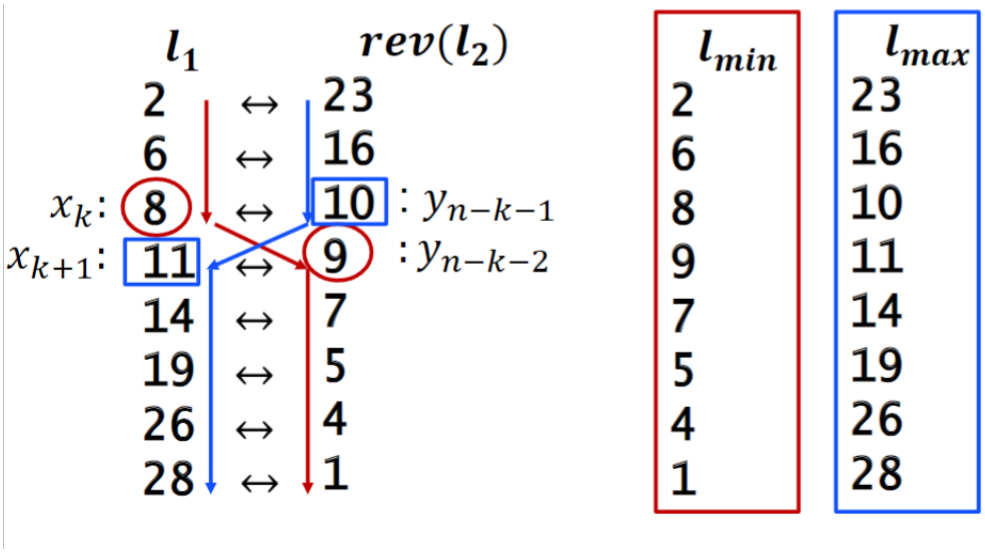

Figure 1: Bitonic Sort Example

Based on the observation, we can see that:

$$\max(l_{min}) \leq \min(l_{max}) \tag{10}$$

This could be proven. Assume the transition index as k in l_1, then:

$$\max(l_{min}) = \max(x_k, y_{n-k-2}) \tag{11}$$

And:

$$\min(l_{max}) = \min(x_{k+1}, y_{n-k-1}) \tag{12}$$

The proof steps include:

1. Because l_1 is sorted, so $x_k \leq x_{k+1}$.

2. Because the transition is after k, so $x_k \leq y_{n-k-1}$
3. Also because the transition is after k, so $y_{n-k-2} \leq x_{k+1}$
4. Because l_2 is sorted, so $y_{n-k-2} \leq y_{n-k-1}$

Therefore, we can easily see that $\max(l_{min}) \leq \min(l_{max})$.

2.2 Runtime

The runtime for bitonic sort is $O(n \log \frac{n}{p})$, and the proof is shown below:

- $T(n,p) = T\left(\frac{n}{2},\frac{p}{2}\right) + O\left(\frac{n}{p}\right) + T\left(\frac{n}{2},\frac{p}{2}\right)$
- $T(n,p) = 2T\left(\frac{n}{2},\frac{p}{2}\right) + O\left(\frac{n}{p}\right)$
- $T(n,p) = 2\left(2T\left(\frac{n}{4},\frac{p}{4}\right) + \left(\frac{n/2}{p/2}\right)\right) + O\left(\frac{n}{p}\right)$
- $T(n,p) = 2^2T\left(\frac{n}{4},\frac{p}{4}\right) + O\left(2\frac{n}{p} + \frac{n}{p}\right)$
- $T(n,p) = 2^3T\left(\frac{n}{8},\frac{p}{8}\right) + O\left(2^2\frac{n}{p} + 2\frac{n}{p} + \frac{n}{p}\right)$
- $T(n,p) = pT\left(\frac{n}{p},1\right) + O\left(\frac{n}{p}\left(1 + 2 + 2^2 + \cdots \frac{p}{2}\right)\right)$ $(p-1)$
- Each processor sorts locally in $O\left(\frac{n}{p}\log\frac{n}{p}\right)$ time.
- $T(n,p) = O\left(\frac{n}{p}\log\frac{n}{p} + n\right)$ $= O\left(n\log\frac{n}{p}\right)$

Figure 2: Bitonic Sort Runtime Proof

2.3 Bitonic Sequence

We call $x_0, x_1, ..., x_{n-1}$ as bitonic sequence if there is an index k such that (could be one of the three):

1. $x_0, x_1, ..., x_k$ is non-decreasing and $x_{k+1}, x_{k+2}, ..., x_{n-1}$ is non-increasing.
2. $x_0, x_1, ..., x_k$ is non-increasing and $x_{k+1}, x_{k+2}, ..., x_{n-1}$ is non-decreasing.
3. There is a cyclical shift of the sequence that makes the first two situations true.

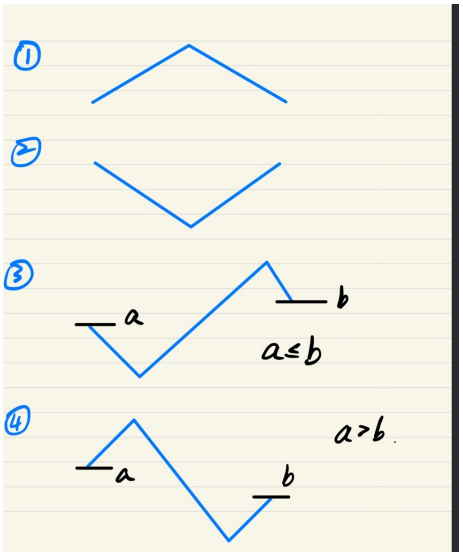

Figure 3: Bitonic Sequence

2.4 Bitonic Split

2.4.1 Definition

Bitonic split of a **bitonic sequence** $l = x_0, x_1, ..., x_{n-1}$ is defined as decomposition of l into:

$$l_{\min} = \min\left(x_0, x_{\frac{n}{2}}\right), \min\left(x_1, x_{\frac{n}{2}+1}\right), \ldots, \min\left(x_{\frac{n}{2}-1}, x_{n-1}\right) \tag{13}$$

$$l_{\max} = \max\left(x_0, x_{\frac{n}{2}}\right), \max\left(x_1, x_{\frac{n}{2}+1}\right), \ldots, \max\left(x_{\frac{n}{2}-1}, x_{n-1}\right) \tag{14}$$

2.4.2 Lemma

Let l be a bitonic sequence, and l_{min} and l_{max} result from the bitonic split, then we have:

1. l_{min} and l_{max} are bitonic.
2. $max(l_{min}) \leq min(l_{max})$

The proof is shown below. if l is the type T1 bitonic sequence, then:

1. When $x_0 > x_{\frac{n}{2}}$:

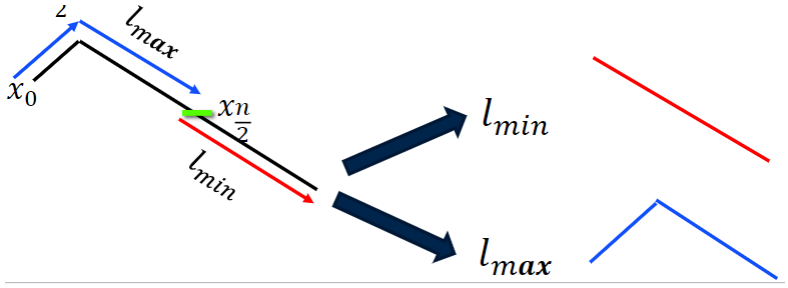

Figure 4: Case 2

2. When $x_0 > x_{\frac{n}{2}}$:

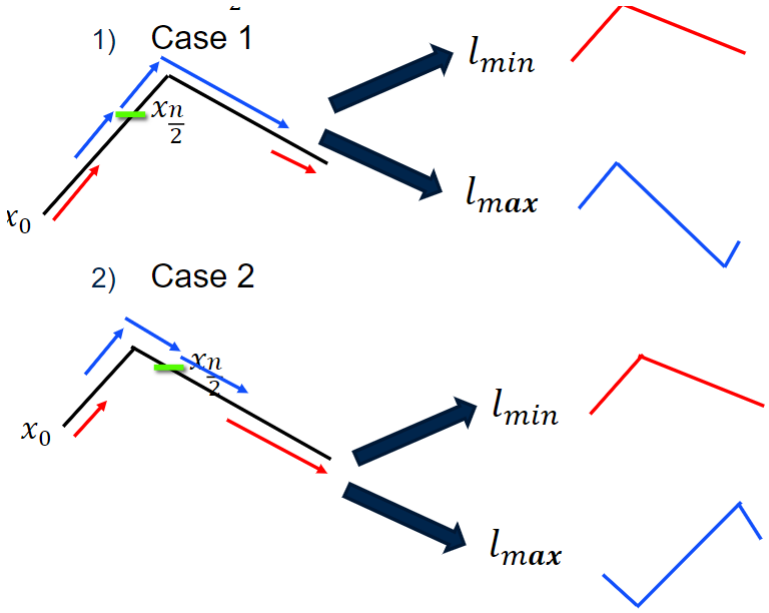

Figure 5: Case 2

2.4.3 Sub-Lemma

Let l be a bitonic sequence and l' be a cyclical shift. Assume bitonic split l into l_{min} and l_{max} and bitonic split l' into l'_{min} and l'_{max}, then we have:

1. l'_{min} is a cyclical shift of l_{min}
2. l'_{max} is a cyclical shift of l_{max}

So the cyclical shift does not change the bitonic nature of the sequence nor the min(or max) element in the sequence.

2.5 Bitonic Merge

Bitonic merge means turning a bitonic sequence into a sorted sequence using repeated bitonic split operations. The runtime could be expressed as:

$$BM(p, p) = O(\log p) \tag{15}$$

2.6 Bitonic Sort Example

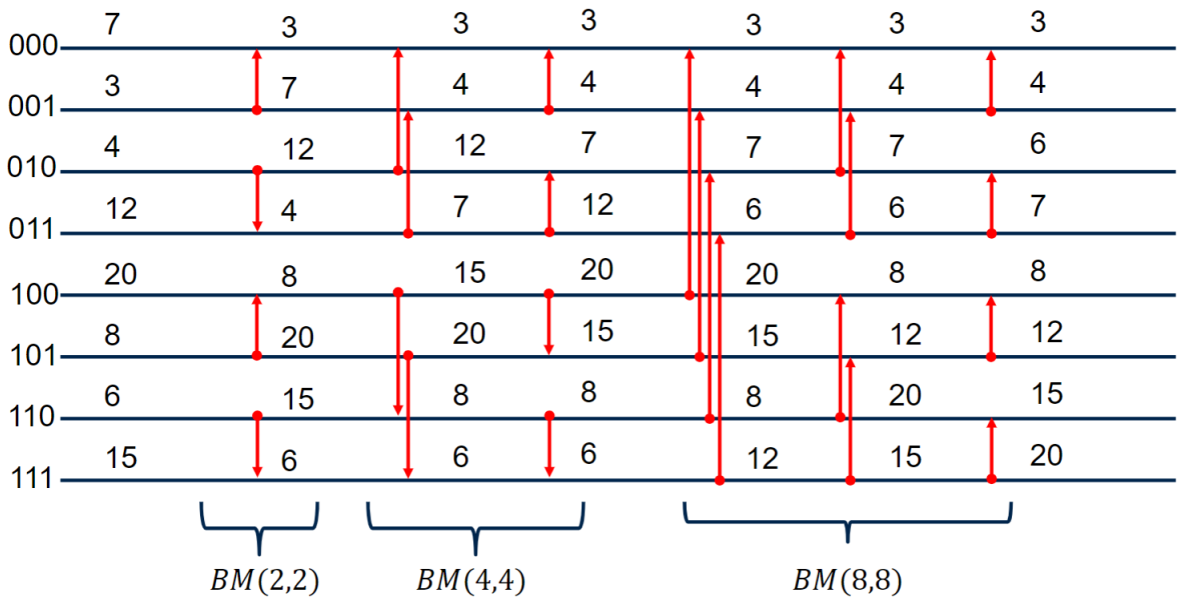

Figure 6: Bitonic Sort Example

Assume we have processors $p_1, p_2, ..., p_8$ and the corresponding value $7, 3, 4, 12, 20, 8, 6, 15$. Then we first divide these values into small sets with only 2 values. Then, we perform **bitonic merge** in each of the set. This includes fisrt **bitonic split** the set and then combine them in sorted way. This is done by putting the split result into a specific processor order. Notice that here **in the first set of 2, we choose ascending order, which means $p_1 < p_2$, and in the second set of 2, we choose descending order, which means $p_3 > p_4$.** This is for the next level's merging. Because now we have only two values, bitonic split the set and put the results into specific order will be the sorting:

$$[p_1(7), p_2(3)] \rightarrow [l_{min}, l_{max}] \rightarrow [p_1(3), p_1(7)] \tag{16}$$

$$[p_3(4), p_4(12)] \rightarrow [l_{max}, l_{min}] \rightarrow [p_3(12), p_4(4)] \tag{17}$$

Because in this level we have two values in each set, so $BM(2, 2) = \log_2 2 = 1$. Now we move to the next level, with each set containing 4 values. Similar with before, **in the first set of 4, we choose ascending order, which means $p_1 < p_2 < p_3 < p_4$, and in the second set of 4, we choose descending order, which means $p_5 > p_6 > p_7 > p_8$.** Now we have:

$$[p_1(3), p_2(7), p_3(12), p_4(4)] \rightarrow [l_{min}, l_{max}] \rightarrow [p_1(3), p_2(4), p_3(12), p_4(7)] \tag{18}$$

$$[p_5(8), p_6(20), p_7(15), p_8(6)] \rightarrow [l_{max}, l_{min}] \rightarrow [p_5(15), p_6(20), p_3(8), p_4(4)] \tag{19}$$

But this is not enough for sorting the first set of 4. We still need to divide it into set of 2, then perform bitonic merge for it, for example:

$$[p_3(12), p_4(7)] \rightarrow [l_{min}, l_{max}] \rightarrow [p_3(7), p_4(12)] \tag{20}$$

$$[p_5(15), p_6(20)] \rightarrow [l_{max}, l_{min}] \rightarrow [p_5(20), p_4(15)] \tag{21}$$

This will take $\log_2 4 = 2$ steps. Similar with $BM(8, 8)$, which will take 3 steps. Therefore, the bitonic sort runtime (computation) could be expressed as:

$$BS(p, p) = BM(2, 2) + BM(4, 4) + ...BM(p, p) \tag{22}$$

$$BS(p, p) = 1 + 2 + 3 + ... + \log p = O(\log^2 p) \tag{23}$$

Because each step in this case need communications between processors, so the communication time is $O((\tau + \mu) \log^2 p)$.

If we have $n > p$, then we need to **sort n/p local elements before doing the bitonic sort, which will take** $O(n \log n)$, so:

$$\text{Computation T} = O(\frac{n}{p} \log \frac{n}{p} + \frac{n}{p} \log^2 p) \tag{24}$$

For the communication time, recall that μ is the time taken to send each unit of data. Here, each processor contains n/p data, and τ is the time for interaction between each processor, so:

$$\text{Communication T} = O(\tau \log^2 p + \mu \frac{n}{p} \log^2 p) \tag{25}$$