\[ T(n) = 2 T\left(\frac{n}{2}\right) + \Theta(n) = \Theta(n \log n) \]

\[ x_0, \ldots, x_m \quad y_0, \ldots, y_n \]

\[ \min(x, y) \]

\text{MergeSort}

- split in \( \frac{n}{2} \) & recursively
- sort each half then merge two sorted lists
Problem:

Merge is sequential

⇒ parallel time $\Theta(n)$
Bitonic Sort

\[ A = a_0 \leq a_1 \leq \ldots \leq a_m \geq a_{m+1} \geq \ldots \]

Bitonic Split \( A_0 = a_0, \ldots, a_{n/2-1} \)

\[ \text{max} = \max(a_0, a_{n/2}) \quad A_1 = a_{n/2} \ldots a_{n-1} \]

\[ \text{max} = \min(a_0, a_{n/2}) \quad A_1 = a_{n/2} \ldots a_{n-1} \]

Bitonic split parallel
Parallel Alg.

\[ P_0 \, P_1 \, \ldots \, P_{p-1} \]

Distributed array, each process owns \( n/p \) elements.
1) Each process sorts its local array in parallel.

2) In loop stage, perform parallel bitonic split, create bitonic sequence, increase and decrease shared among processes.
\( P_0 \) \( P_1 \) \( P_2 \) \( P_3 \) \( P_4 \) \( P_5 \) \( P_6 \) \( P_7 \)
Each step perform partially binary split until sorted
- requires sort on local array
Can avoid local
sorts if maintain
sorted increasing/decreasing
lists using a parallel
merge at each step at
bitwise split