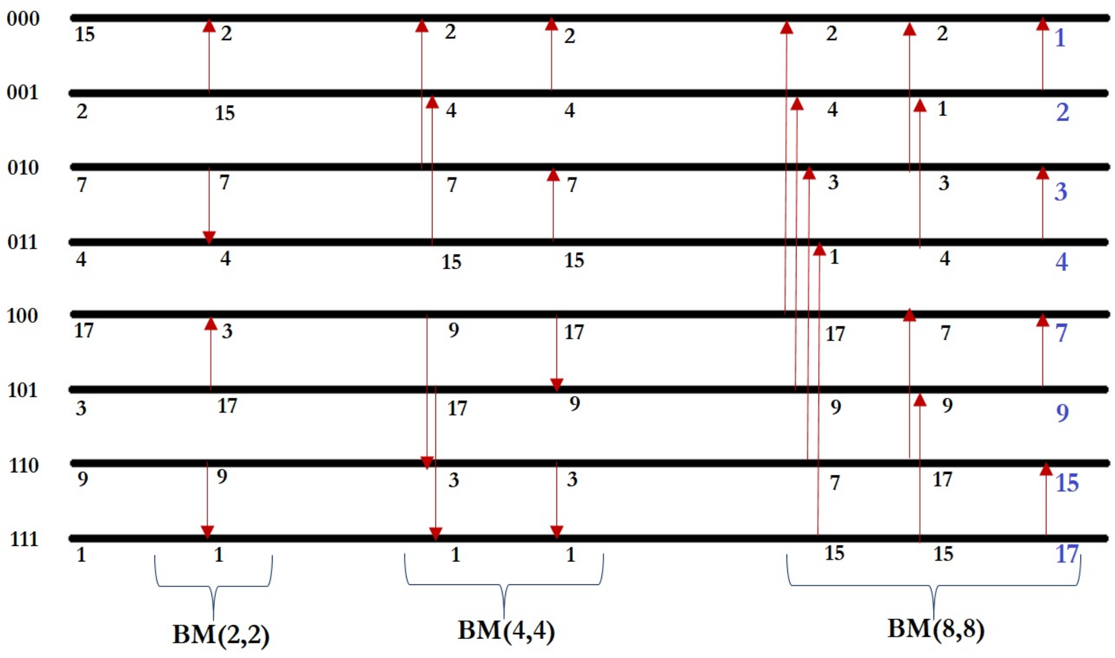
1. CSE 6220: HW3

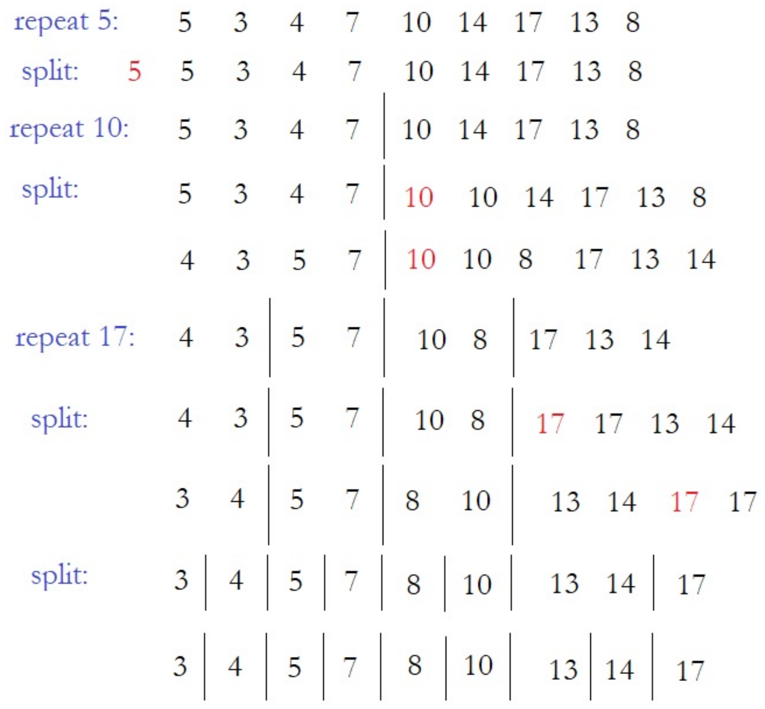
Note: This assignment is submitted by: **Somdut Roy (GTID: sroy86)**

1. Figure below describes the 8-element bitonic element sorting and its functioning with the given example. Numbers in blue font (at the extreme right) denote the final result.



2. For sequences with odd length, we can repeat an element in the sequence. That way, parity (even or odd) can be changed but the sequence remains bitonic. The extra element can be removed after the split is performed. In the figure below, the example is sorted and important steps (for the sequence that follows)

are mentioned (in blue) at the left. We have duplicated the first element and added it to the beginning in each step that demanded it.


repeat 5: 5 3 4 7 10 14 17 13 8
split: 5 5 3 4 7 10 14 17 13 8
repeat 10: 5 3 4 7 | 10 14 17 13 8
split: 5 3 4 7 | 10 10 14 17 13 8
4 3 5 7 | 10 10 8 17 13 14
repeat 17: 4 3 | 5 7 | 10 8 | 17 13 14
split: 4 3 | 5 7 | 10 8 | 17 17 13 14
3 4 | 5 7 | 8 10 | 13 14 17 17
split: 3 | 4 | 5 | 7 | 8 | 10 | 13 14 | 17
3 | 4 | 5 | 7 | 8 | 10 | 13 | 14 | 17

3. (a) We reverse the n numbers in second sequence. As an upper bound, a processor will send it elements to two processors (two permutations). With $\frac{m+n}{p}$ communications per processor, time taken will be bounded by $O(\tau + \mu(\frac{m+n}{p}))$.
   (b) With m ascending numbers followed by n descending numbers, a bitonic sequence is already formed. This treated with bitonic merge, will create the desired result.
   Computation time: $O((\frac{m+n}{p})log(p))$.
   Communication time: $O(\tau log(p) + \mu(\frac{m+n}{p})log(p))$.

4. k processors have m messages while other $(p-k)$ processors can be assumed to have messages for zero size. If $k-to-k$ broadcast is taken into account, it takes $O(\tau log(k) + \mu mk)$ time. Now km messages are sent to the other $(p-k$ processors and the messages size would not exceed km at any time. It will be done in remaining $log(p) - log(k)$ steps. The time taken in that will be $O((log(p) - log(k))(\tau + \mu km))$. When added together, the total communication time will be $O(\tau log(p) + \mu mk(log(\frac{p}{k}) + 1))$.

5. First term of algorithm 1: $\tau log(p)$ is less than the first term of algorithm 2: τp and second term of algorithm 2: μmp is less than the second term of algorithm 1: $\mu mplog(p)$.
   So algorithm 1 will be faster if $\mu mplog(p) < \tau p$.
   Therefore, algorithm 1 is faster for $m < O(\frac{\tau}{\mu log(p)})$, otherwise algorithm 2 is faster.