# Weekplan: Massively Parallel Computation

Philip Bille    Inge Li Gørtz    Eva Rotenberg

## References and Reading

[1] Massively Parallel Algorithms, M. Ghaffari, 2019.

[2] Parallel Algorithms, Chapter 6, M. Ghaffari, 2019.

[3] Scribe notes from "Algorithms for Massive Data Science", Ben Mosely, 2019.

We recommend reading [1] for an overview of the area. We recommend reading [2] sections 6.7-6.9 and the scribe notes [3] in detail.

## Exercises

**1 Summing with other Parameters** Consider the sum problem from the lectures and the parameters $P$ and $S$. Solve the following exercises.

**1.1** Give an efficient algorithm when $S = \Theta(N^{3/4})$ and $P = \Theta(N^{1/4})$.

**1.2** Give an efficient algorithm when $S = \Theta(N^{1/4})$ and $P = \Theta(N^{3/4})$ .

**1.3** Give an efficient algorithm when $S = \Theta(N^{\varepsilon})$ and $P = \Theta(N^{1-\varepsilon})$.

**2 Sorting Properties and Output** Solve the following exercises.

**2.1** $[w]$ The single round algorithm for sorting in the MPC model is often called "quicksort" or "generalized quicksort". Why? What is the connection to quicksort?

**2.2** The output of the MPC sorting algorithm is not in the required format. Show how to convert it to the required format in $O(1)$ additional rounds.

**3 Prefix Sum, Distinct Elements, and Word Count** Solve the following exercises. Assume $S = P = \widetilde{\Theta}(\sqrt{N})$

**3.1** Let $A$ be an array of $N$ integers distributed among processors. Each entry in $A$ is stored as $(i, A[i])$. Show how to compute the prefix sum of $A$ (the array $P[i] = \sum_{j \leq i} A[j]$) efficiently in the MPC model. The array $P$ should be represented similar to $A$.

**3.2** Let $L$ be a list of $N$ integers, show how to compute the number of distinct elements in $L$ in the MPC model.

**3.3** Let $W$ be a list of $N$ strings each of constant length. The *word count* of $W$ is the list of pairs of distinct words and their frequency in $W$. Computing the word count is a classic "hello world"-exercise for the MapReduce framework. Show how to implement it efficiently in the MPC model.

**4 Sorting Analysis** Solve the following exercises.

**4.1** Carefully verify that each step of the sorting algorithm satisfies the bounds on $S$ and $P$.

**4.2** Show that $|X| \leq 4P \ln N$ whp. *Hint:* First compute the expected size of $X$ and then apply a Chernoff bound.

**5 Minimum Spanning Tree Correctness** Show that the MPC minimum spanning tree algorithm correctly outputs a minimum spanning tree. *Hint:* show that the discarded edges cannot be part of an MST using standard properties of MSTs.

**6 Dynamic Programming** Let $S$ and $T$ be strings of length $N$ and consider the classic $O(N^2)$ time solution for computing the longest common subsequence of $S$ and $T$. Show how to implement the algorithm efficiently on the MPC model. Assume $S = P = \widetilde{\Theta}(\sqrt{N})$.

**7** [$*$] **String Matching** Let $P$ and $T$ be strings of lengths $M$ and $N$, respectively. The *string matching problem* is to determine if $P$ occurs as a substring of $T$. Show how to solve the string matching problem efficiently on the MPC model. Assume $S = P = \widetilde{\Theta}(\sqrt{N})$. Further, assume that $T$ and $P$ are partitioned into equal sized substrings of size $O(S)$ and stored distributed among the processors. Solve the following exercises.

**7.1** Solve the string matching problem with the assumption that $M < S$.

**7.2** Solve the string matching problem for any $M \leq N$. *Hint:* use *Karp-Rabin fingerprints* to efficiently hash substrings.