Time Series Models: From Statistics to AI

lecturer: Daniel Durstewitz
tutors: Lukas Eisenmann, Christoph Hemmer, Alena Brändle, Florian Hess, **Elias Weber**, Florian Götz
SS2025

Exercise Sheet 10

Regulations

Please submit your solutions via Moodle in teams of **2 students**, before the exercise group on **Wednesday, July 2nd, 2025**. Each submission must include **exactly** two files:

- A `.pdf` file containing both your Jupyter notebook and solutions to analytical exercises. The Jupyter notebook can be exported to pdf by selecting **File → Download as → pdf** in JupyterLab. If this method does not work, you may print the notebook as a pdf instead. Your analytical solutions can be either scanned handwritten solutions or created using LaTeX.
- A `.ipynb` file containing your code as Jupyter notebook.

Both files must follow the naming convention:
`Lastname1-Lastname2-sheet10.pdf`
`Lastname1-Lastname2-sheet10.ipynb`

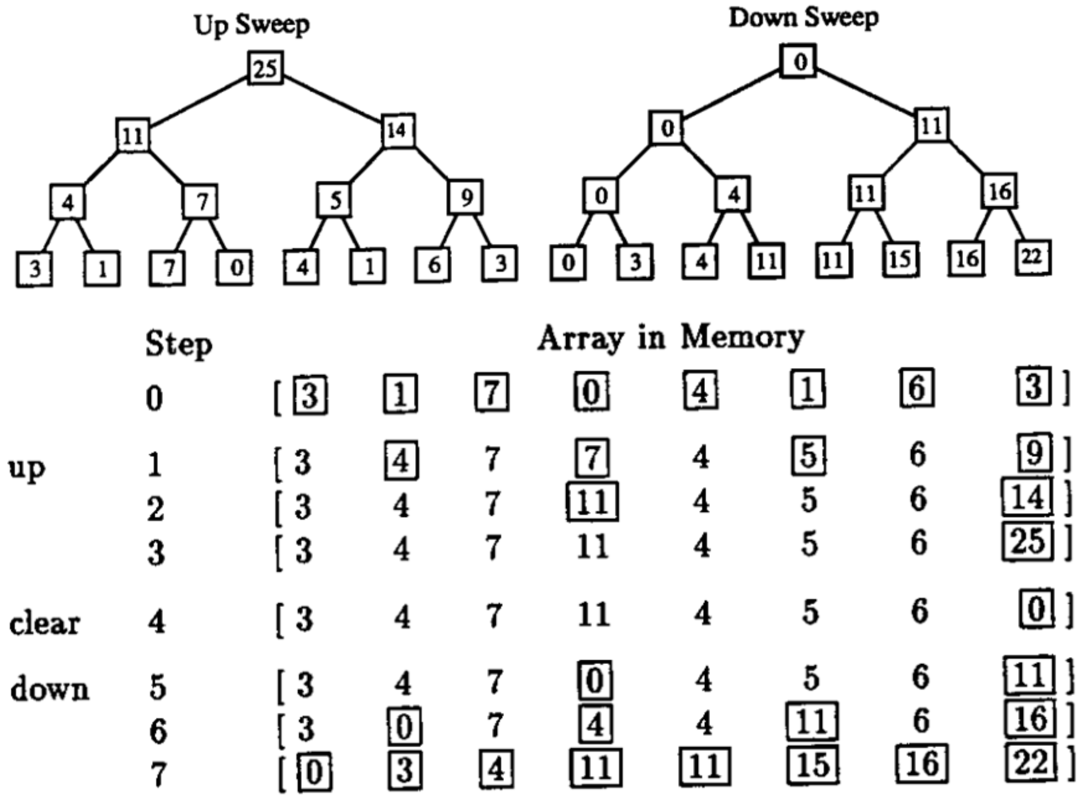
1 Parallel associative scan

In the lecture, we learned about structured state space sequence models (e.g. Gu et al. (2021); Gu et al. (2023)). These models can be parallelized over the sequence length by utilizing the parallel associative scan algorithm (Blelloch (1990); Martin & Cundy (2018)). This exercise aims to give you an understanding of that algorithm.

Consider an input sequence $a = [a_0, \dots, a_{N-1}]$ of which we want to compute the cumulative sum $\texttt{cumsum}(a) = [0, a_0, a_0 + a_1, a_0 + a_1 + a_2, \dots]$. The algorithm consists of two parts:

> The **up-sweep** constructs a binary tree bottom-up by initializing the leaves as the elements of the input sequence. Then, values are propagated upward by taking the sum of pairs of two (these sums are done in parallel), i.e. `parent` ← `left_child` + `right_child`; see Fig. 1 for a visualization. In the end, the root node is exactly the total sum of all elements in the sequence.
>
> After the tree has been constructed by the up-sweep, we first clear the root node to zero and then do a **down-sweep**, which consists of propagating values downward by setting `right_child` ← `parent` + `left_sibling` and then `left_child` ← `parent` (again, this is done in parallel). Fig. 1 also visualizes the resulting tree after the down-sweep.

If you need further explanation or are just interested, feel free to have a look at the Blelloch paper yourself.



	Step	Array in Memory							
	0	[$\boxed{3}$	$\boxed{1}$	$\boxed{7}$	$\boxed{0}$	$\boxed{4}$	$\boxed{1}$	$\boxed{6}$	$\boxed{3}$]
up	1	[3	$\boxed{4}$	7	$\boxed{7}$	4	$\boxed{5}$	6	$\boxed{9}$]
	2	[3	4	7	$\boxed{11}$	4	5	6	$\boxed{14}$]
	3	[3	4	7	11	4	5	6	$\boxed{25}$]
clear	4	[3	4	7	11	4	5	6	$\boxed{0}$]
down	5	[3	4	7	$\boxed{0}$	4	5	6	$\boxed{11}$]
	6	[3	$\boxed{0}$	7	$\boxed{4}$	4	$\boxed{11}$	6	$\boxed{16}$]
	7	[$\boxed{0}$	$\boxed{3}$	$\boxed{4}$	$\boxed{11}$	$\boxed{11}$	$\boxed{15}$	$\boxed{16}$	$\boxed{22}$]

Figure 1: Top: Tree visualization of the two sweep stages. Bottom: The changes in the input sequence after each loop iteration. Adapted from Blelloch (1990).

a) Following the example in Fig. 1, construct the trees after the up-sweep and after the down-sweep for the input sequence $a = [4, 6, 1, 0, 1, 7, 3, 2]$.

b) Computing the cumulative sum naively has time complexity $O(N)$. What is the time complexity of the parallel scan algorithm (specifically, the idealized version from here)? Explain why.
Hint: In both sweeps, the loop over the tree's nodes at a given depth is parallelized, i.e. has time complexity $O(1)$.

In the paper (see Sec. 4), Blelloch showed that his algorithm can be generalized to recurrences of the form

$$[y_i, x_i] = \begin{cases} [a_0, b_0] & i = 0 \\ [y_{i-1} \odot a_i, (x_{i-1} \otimes a_i) \oplus b_i] & i > 0 \end{cases} \tag{I}$$

given that $\oplus, \otimes, \odot$ fulfill certain properties. This is exactly what enables parallelization of structured state space sequence models. Recall from the lecture the discretized state equations (see also Smith et al. (2023)):

$$\begin{aligned} \mathbf{z}_t &= \bar{\mathbf{A}}\mathbf{z}_{t-1} + \bar{\mathbf{B}}u_t \\ y_t &= \mathbf{C}\mathbf{z}_t \end{aligned} \tag{II}$$

with diagonal matrix $\mathrm{diag}(\bar{\mathbf{A}}) \in \mathbb{R}^M$ and matrices $\bar{\mathbf{B}} \in \mathbb{R}^{M\times 1}$ and $\mathbf{C} \in \mathbb{R}^{1\times M}$.

c) Reformulate the recurrence in eq. II such that it fits eq. I.
Hint 1: Start by finding an expression for z_T by iterative insertion.
Hint 2: It is $y_i \equiv \bar{\mathbf{A}}^i$ and $x_i \equiv \mathbf{z}_i$. Then, $\otimes$ is matrix-matrix multiplication, $\odot$ is matrix-vector multiplication and $\oplus$ is vector addition.

d) This exercise requires using a GPU. If you do not have access to one, then use Google Colab. The python library `jax` implements an associative parallel scan at `jax.lax.associative_scan`. Have a look at the documentation and then complete the supplied `sheet10_template.ipynb`. That is, implement the two versions of the cumulative sum, one that is fully sequential and one that uses the parallel scan algorithm. Record their runtimes for different lengths of input sequence and visualize the results.

   Hint: `jax`'s arrays do not support in place operations. Instead of `array[i] = expression` use `array = array.at[i].set(expression)`.