Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CHAI
Humanities-Centered AI

Transformer-based Algorithmics

Malte Luttermann – Institute for Humanities-Centered Artificial Intelligence (CHAI)

May 6, 2026

Version from May 5, 2026

Topics for Today

- Selection
- Priority Queues with Binary Heaps

Acknowledgement

- Some of the upcoming content is taken from the following lectures and corresponding exercises and has been translated and partially modified:
  - Prof. Dr. Ralf Möller: »Algorithmen und Datenstrukturen – Vorlesung«
  - Dr. Magnus Bender: »Algorithmen und Datenstrukturen – Übung«

Exercise 1 – Quickselect

Given the sequence $A = 12, 47, 13, 99, 7, 3, 72$.
Use *Quickselect* to determine the $k = 2$-smallest element.

Idea. Like Quicksort, but recurse only into the side that contains the k-smallest element:

- Pick a pivot, swap it to the end, partition $A[l..r]$ around the pivot.
- Let i be the pivot's final position. Then:
  - if $k = i$: **found** – return $A[k]$,
  - if $k < i$: recurse on $A[l..i-1]$,
  - if $k > i$: recurse on $A[i+1..r]$.

Exercise 1 – Solution: 1st Call `Quickselect`$(A, l{=}1, r{=}7, k{=}2)$

- Pick pivot index randomly: $z = 2$, i.e., **pivot** $= 47$
- Swap pivot to end: $A[r{=}7] = 47$, $A[z{=}2] = 72$

Partitioning (`while`):

	i	j	A
	1	7	12, 72, 13, 99, 7, 3, **47**
Swap $A[i]{=}72$, $A[j]{=}3$	2	6	12, 3, 13, 99, 7, 72, **47**
Swap $A[i]{=}99$, $A[j]{=}7$	4	5	12, 3, 13, 7, 99, 72, **47**

- Place pivot at position: $A[r{=}7] = 99$, $A[i{=}5] = 47$
- Resulting array:

- Since $k = 2 < i = 5$: search continues in $A[1..4]$.
- $\Rightarrow$ Recurse: `Quickselect`$(A, l{=}1, r{=}4, k{=}2)$

Exercise 1 – Solution: 2nd Call `Quickselect`$(A, l{=}1, r{=}4, k{=}2)$

- Current array: $A = 12, 3, 13, 7, 47, 72, 99$ (active range $A[1..4]$)
- Pick pivot index randomly: $z = 3$, i.e., **pivot** $= 13$
- Swap pivot to end of range: $A[r{=}4] = 13$, $A[z{=}3] = 7$

Partitioning (`while`):

i	j	A
1	4	12, 3, 7, **13** 47, 72, 99

- Place pivot at position: $A[r{=}4] = 13$, $A[i{=}4] = 13$ (no change)
- Resulting array:

- Since $k = 2 < i = 4$: search continues in $A[1..3]$.
- $\Rightarrow$ Recurse: `Quickselect`$(A, l{=}1, r{=}3, k{=}2)$

Exercise 1 – Solution: 3rd Call `Quickselect`$(A, l{=}1, r{=}3, k{=}2)$

- Current array: $A = 12, 3, 7, 13, 47, 72, 99$ (active range $A[1..3]$)
- Pick pivot index randomly: $z = 3$, i.e., **pivot** $= 7$
- Swap pivot to end of range: $A[r{=}3] = 7$, $A[z{=}3] = 7$ (no change)

Partitioning (`while`):

	i	j	A
	1	3	12, 3, **7** 13, 47, 72, 99
Swap $A[i]{=}12$, $A[j]{=}3$	1	2	3, 12, **7** 13, 47, 72, 99

- Place pivot at position: $A[r{=}3] = 12$, $A[i{=}2] = 7$
- Resulting array:

3	**7**	12	13	47	72	99
1	2	3	4	5	6	7

- Since $k = 2 = i$: **target found!**
- $\Rightarrow$ Result: $A[k{=}2] = \mathbf{7}$

Exercise 2 – Min-Heap Construction

Given the following array:

A:	42	23	68	99	16	6	15	45	8	66
	1	2	3	4	5	6	7	8	9	10

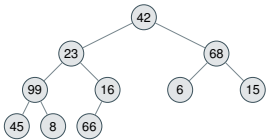
Convert the array into a **Min-Heap**: Insert the elements into a binary tree (level by level, left-to-right) and then convert the tree into a Min-Heap using `siftDown`. Draw the tree after inserting all elements as well as after each `siftDown`.

Min-Heap Property

For every node v: $A[v] \leq A[\text{left\_child}(v)]$ and $A[v] \leq A[\text{right\_child}(v)]$
$\Rightarrow$ The **smallest element** is always at the root.

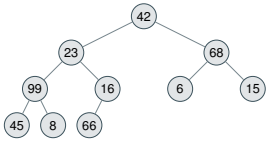
Exercise 2 – Solution: Build Min-Heap (1/2)

Initial tree:

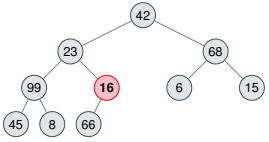


Exercise 2 – Solution: Build Min-Heap (1/2)

Initial tree:

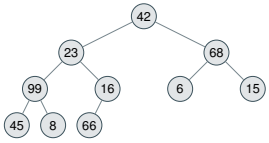


`siftDown`(16): no swap

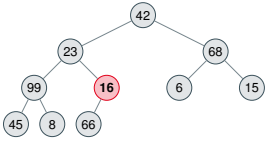


Exercise 2 – Solution: Build Min-Heap (1/2)

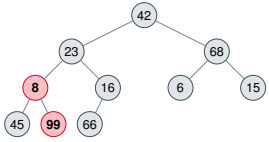
Initial tree:



`siftDown`(16): no swap

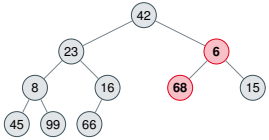


`siftDown`(99): swap 99 $\leftrightarrow$ 8



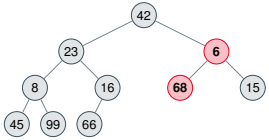
Exercise 2 – Solution: Build Min-Heap (2/2)

`siftDown`(68): swap 68 $\leftrightarrow$ 6

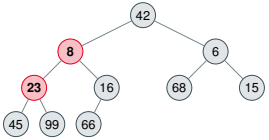


Exercise 2 – Solution: Build Min-Heap (2/2)

`siftDown`(68): swap 68 ↔ 6

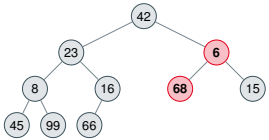


`siftDown`(23): swap 23 ↔ 8

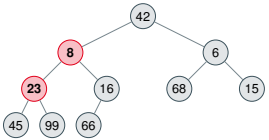


Exercise 2 – Solution: Build Min-Heap (2/2)

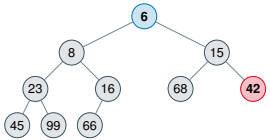
`siftDown`(68): swap 68 $\leftrightarrow$ 6



`siftDown`(23): swap 23 $\leftrightarrow$ 8



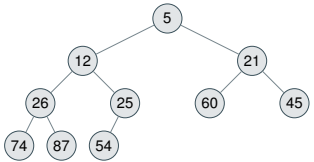
`siftDown`(42): 42 $\leftrightarrow$ 6, then 42 $\leftrightarrow$ 15



$\Rightarrow$ Min-Heap

Exercise 3 – Min-Heap: Insert and Delete

Given the following Min-Heap:

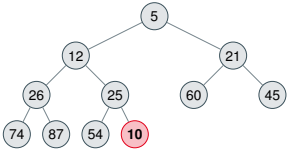


(a) Insert the values **10** and **4** (in this order) into the Min-Heap above and restore the Min-Heap property using `siftUp` after each insertion. Draw the tree directly after the insertion as well as after `siftUp`.

(b) Delete the root of the Min-Heap above (*not* the result of (a)!) **twice** in succession and restore the Min-Heap property using `siftDown`. Draw the tree directly after the deletion as well as after `siftDown`.

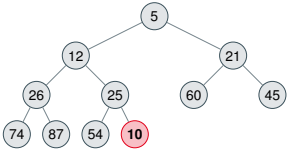
Exercise 3 (a) – Solution: Insert 10

Insert 10 (append at next free position):

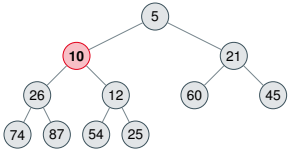


Exercise 3 (a) – Solution: Insert 10

Insert 10 (append at next free position):

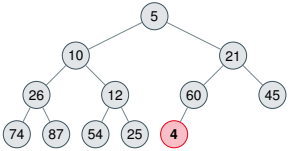


After `siftUp`: $10 \leftrightarrow 25$, then $10 \leftrightarrow 12$:



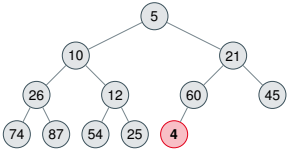
Exercise 3 (a) – Solution: Insert 4

Insert 4 (append at next free position):

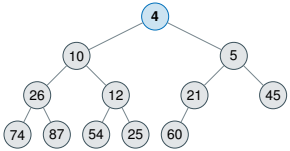


Exercise 3 (a) – Solution: Insert 4

Insert 4 (append at next free position):

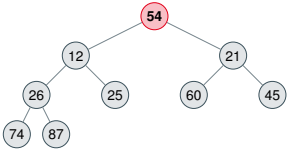


After `siftUp`: $4 \leftrightarrow 60$, $4 \leftrightarrow 21$, $4 \leftrightarrow 5$:



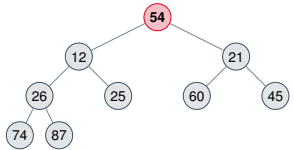
Exercise 3 (b) – Solution: First `deleteMin`

Delete root 5 (move last leaf 54 to root):

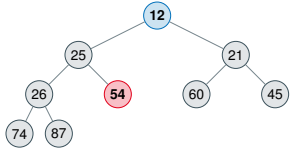


Exercise 3 (b) – Solution: First `deleteMin`

Delete root 5 (move last leaf 54 to root):

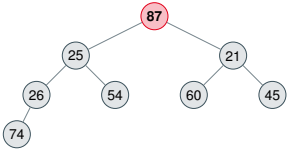


After `siftDown`: $54 \leftrightarrow 12$, then $54 \leftrightarrow 25$:



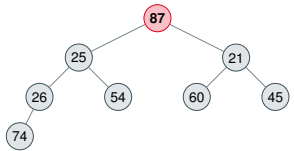
Exercise 3 (b) – Solution: Second `deleteMin`

Delete root 12 (move last leaf 87 to root):



Exercise 3 (b) – Solution: Second `deleteMin`

Delete root 12 (move last leaf 87 to root):



After `siftDown`: $87 \leftrightarrow 21$, then $87 \leftrightarrow 45$: