October 8, 2020

Heapsort

→ an efficient max selection sort

It consists of two parts

1. Preprocessing
   → Rearrange the array to satisfy the heap property

2. Swap the max with the last element, then restore the heap property and repeat

(1) Making a heap

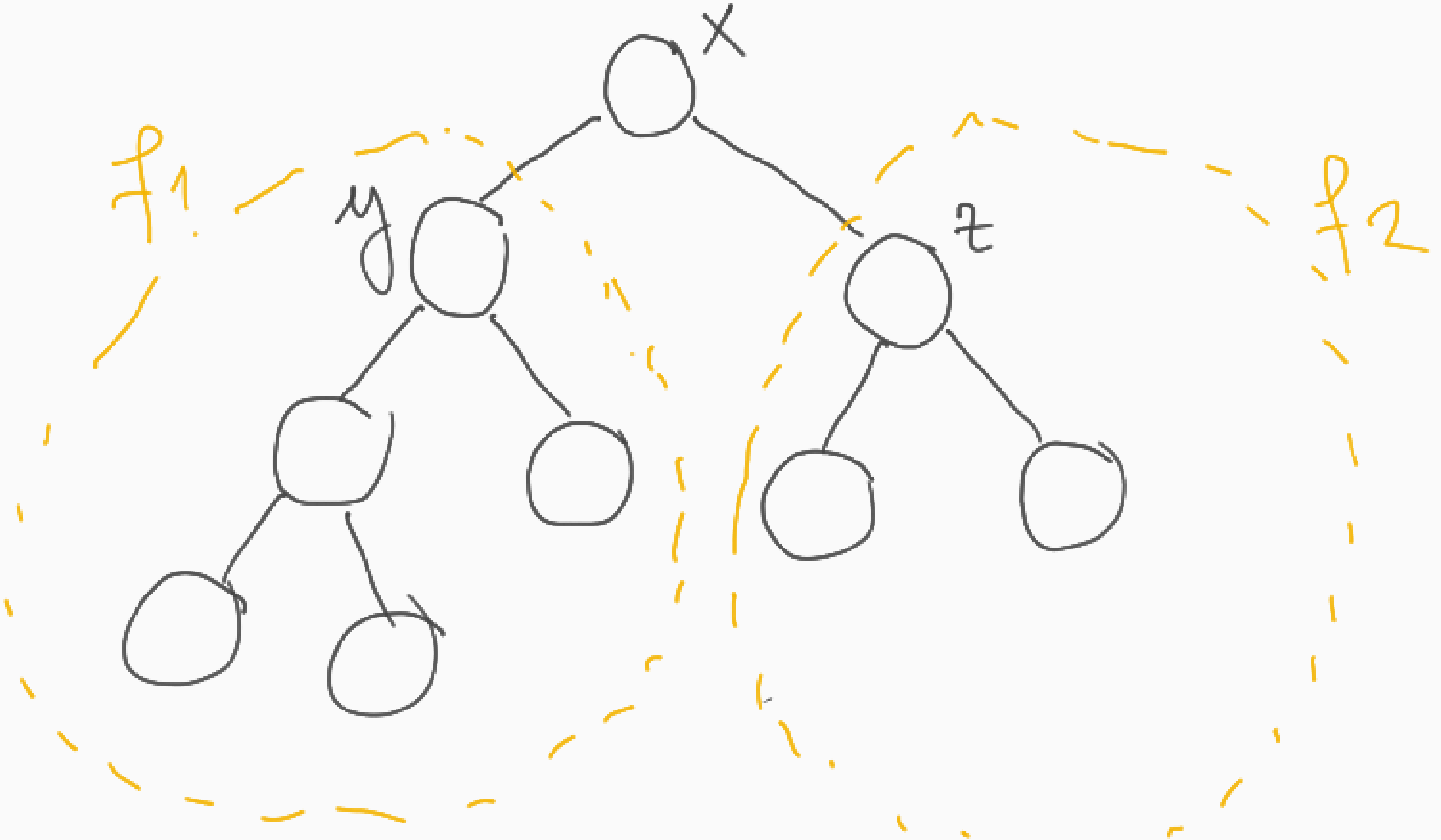
We have $A[1:n]$, rearrange this such that $A[1] \geq A[2], A[3]$

$$A[2] \geq A[4], A[5]$$

$$\vdots$$

We will use the tree representation to visualize the algorithm

Consider a subtree f of the tree representation. Let f_1 and f_2 be the left and right subtree, resp., of the root of f



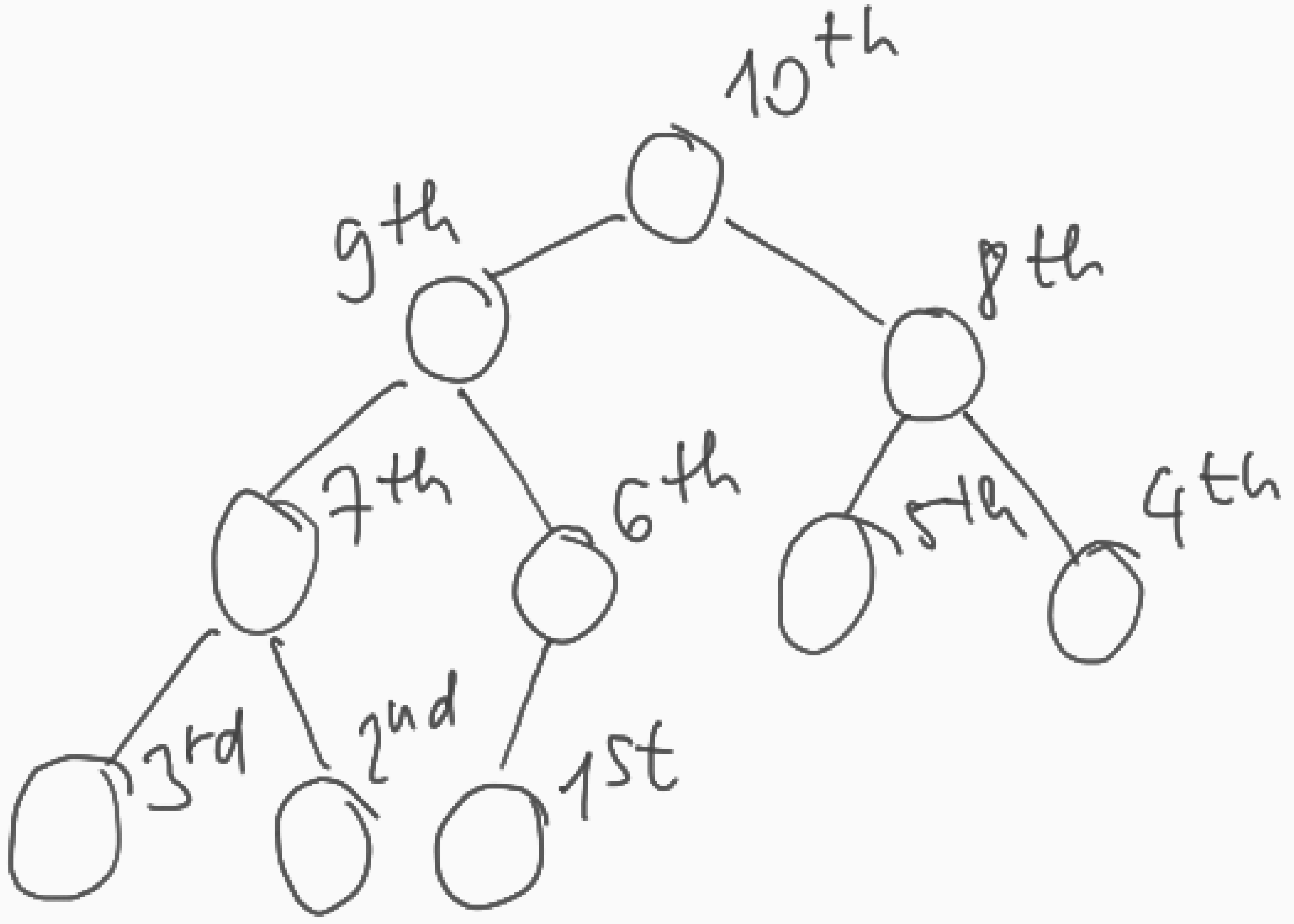
Suppose that f_1 and f_2 are heaps, but f is not a heap.

Then x is not the biggest element among x, y, z; i.e., the biggest element is y or z, say y.

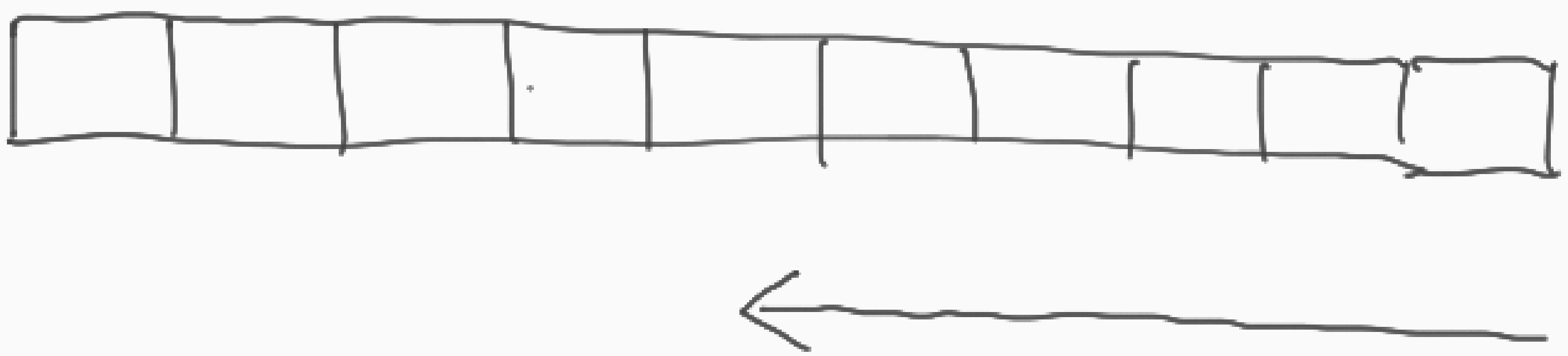
Now swap x and y and repeat recursively the same for f_1 instead of f

After this sequence of swaps the heap property is satisfied for f as well.

After this the algorithm is easy: execute this procedure for each vertex from bottom to top and from right to left in a certain level.



leaves



array

Heapify (A, i) $\qquad$ $A[1:n]$

$l = 2i$

$r = 2i + 1$

if $l \le n$ AND $A[l] > A[i]$

then maxind $= l$

else maxind $= i$

if $r \le n$ AND $A[r] > A[\text{maxind}]$

then maxind $= r$

if maxind $\neq i$ then

swap$(A[i], A[\text{maxind}])$

Heapify (A, maxind)

HeapBuilding (A) $\qquad A[1:n]$

for $i = n$ downto 1 do

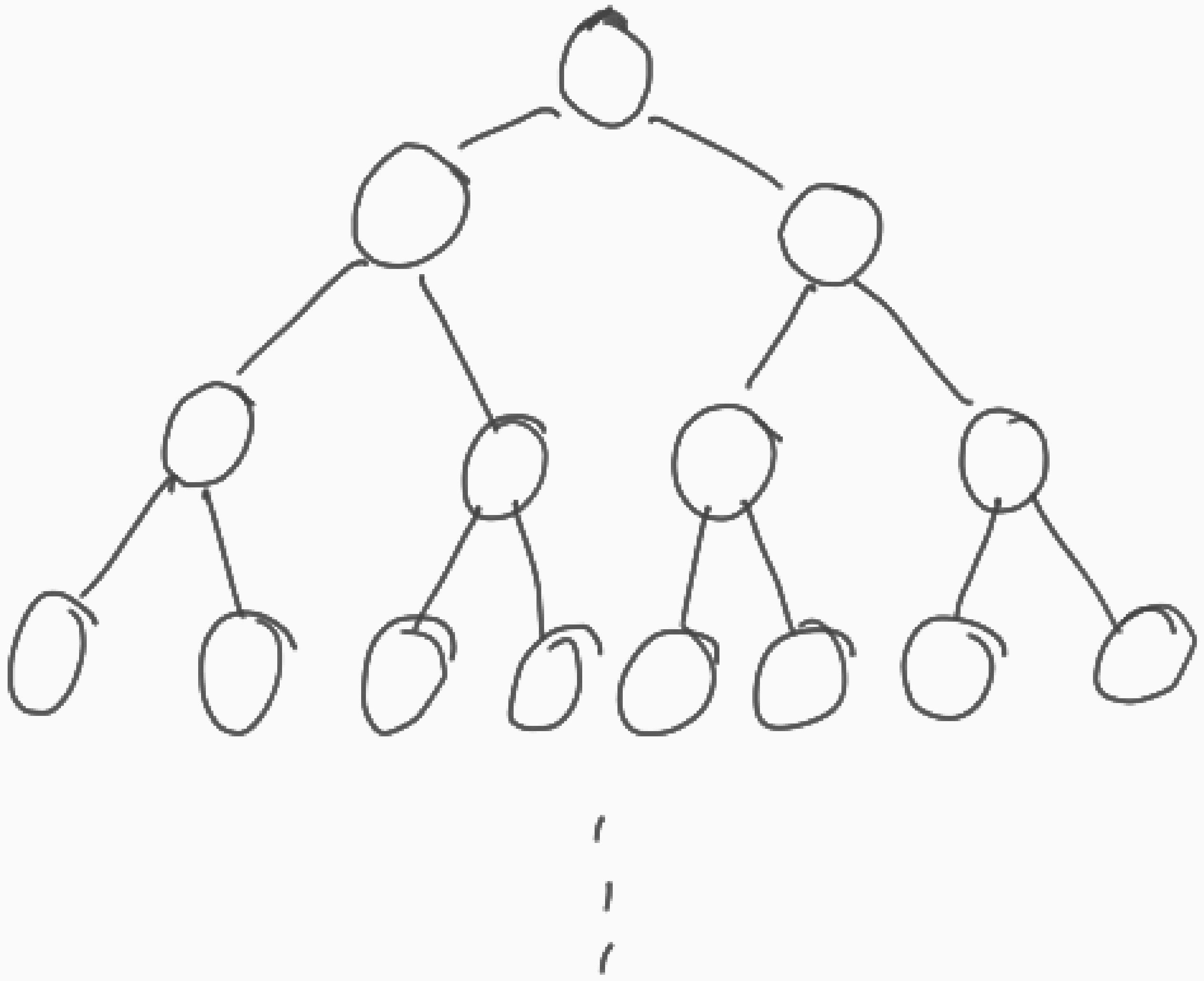
$\quad$ Heapify (A, i)

In fact we can write $n/2$ instead of 1 in the for statement.

Time complexity : $O(n)$

(It's clear, that $O(n \log n)$)

Complete binary tree with n vertices



$1 \leftarrow 1^{st}$ level

$2 \leftarrow 2^{nd}$ level

$4 \leftarrow 3^{rd}$ level

$8 \leftarrow 4^{th}$ level

$\vdots$

$2^{k-1} \leftarrow k^{th}$ level

$$1 + 2 + 2^2 + \dots + 2^{k-1} = \underbrace{2^k - 1}_{n}$$

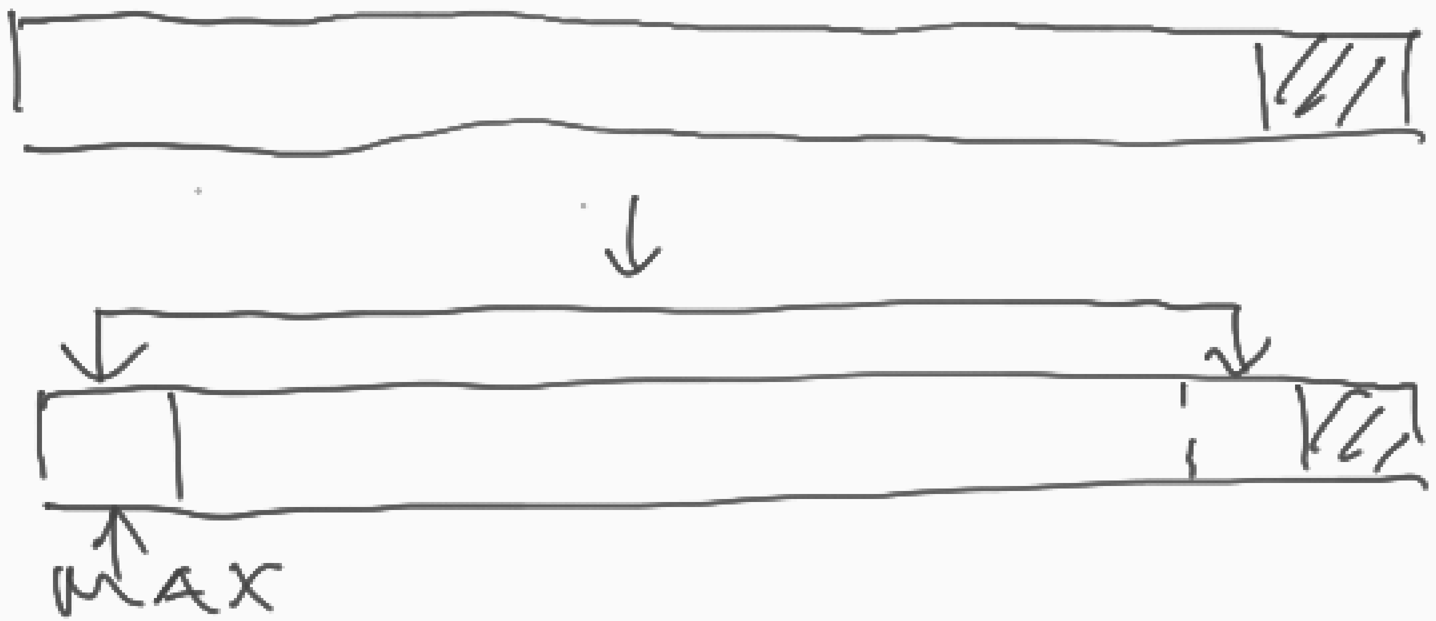
$$k \sim \log_2(2^k - 1) = \log_2 n$$

We turn to ②

this is the clever max selection sequence

How does it work

$A[1:n]$ heap



Heapify for $A[1:n-1]$
from $A[1]$

⋮

Heapsort (A) $\qquad$ $A[1:n]$

HeapBuilding(A)

for $i = n$ downto 2 do

$\quad$ Swap $(A[1], A[i])$

$\quad$ Heapify $(\underbrace{A[1:i-1]}, 1)$

indicates that we
take smaller and smaller
prefix

Time complexity: $O(n \log n)$

(Time complexity for Heapify: $O(\log n)$)

<u>Summary</u>

Heapsort is a

- deterministic
- in-place
- $O(n \log n)$ worst case complexity

Theorem

Any comparision based sorting algorithm's time complexity is $\Omega(n \log n)$ where n is the number of elements

Therefore heapsort is an asymptotically optimal sorting algorithm.

Quiz 1

October 15 16:00 - 17:30 Lecture time

17:30 - 17:45 for taking photos, to produce a pdf, to upload to Teams

You have to be online with turned on camera during the whole quiz

You have to sit in front of the camera in such a way that not only your face but your hands, desk, papers are visible

No electronic devices are allowed.

Closed book, closed notes quiz.

The video conference (the quiz) will be in the Lecture Group

Uploading will be in the Practice Group in Assignments