December 10, 2020

Technical issues

For this ADS1 students get 2 grades (in NEPTUN)

→ one for the practice part

→ another one for the lecture part

(they are independent)

Practice part

By the sum of the quiz results

100 – 120 → 5

80 – 99 → 4

60 – 79 → 3

40 – 59 → 2

Under 40 points you fail the practice part as well as the lecture part

Lecture part

Oral exam (without preparation time)

Exams are announced in Neptun and

you have to register there

If you fail an exam you must retake it but at most twice and you have to pay a fee for the third exam

You can also retake an exam if you are not satisfied with your grade.

Successful exam is a requirement for registering ADS2 next semester

Retake for practice part

If you have less than 40 points then you must, otherwise you can retake on quiz on December 15 (see Canvas) esp. the quiz of lower score.

This quiz will be in a separate Teams group and you have to register for the retake by a private chat message

in Teams not later than December 11.
You can only increase your score by a retake quiz.
The technical conditions are the same as before.

Offered lecture part grade

If someone has at least 100 point in total from the two midterm quizzes (retakes don't count by the department's policy) he or she doesn't have to take an oral exam he or she gets grade 5 for the lecture part as well. Everything is managed by Neptun: I offer grade 5 and you MUST accept it (or reject it). You can accept an offered grade if you don't have exam registration.

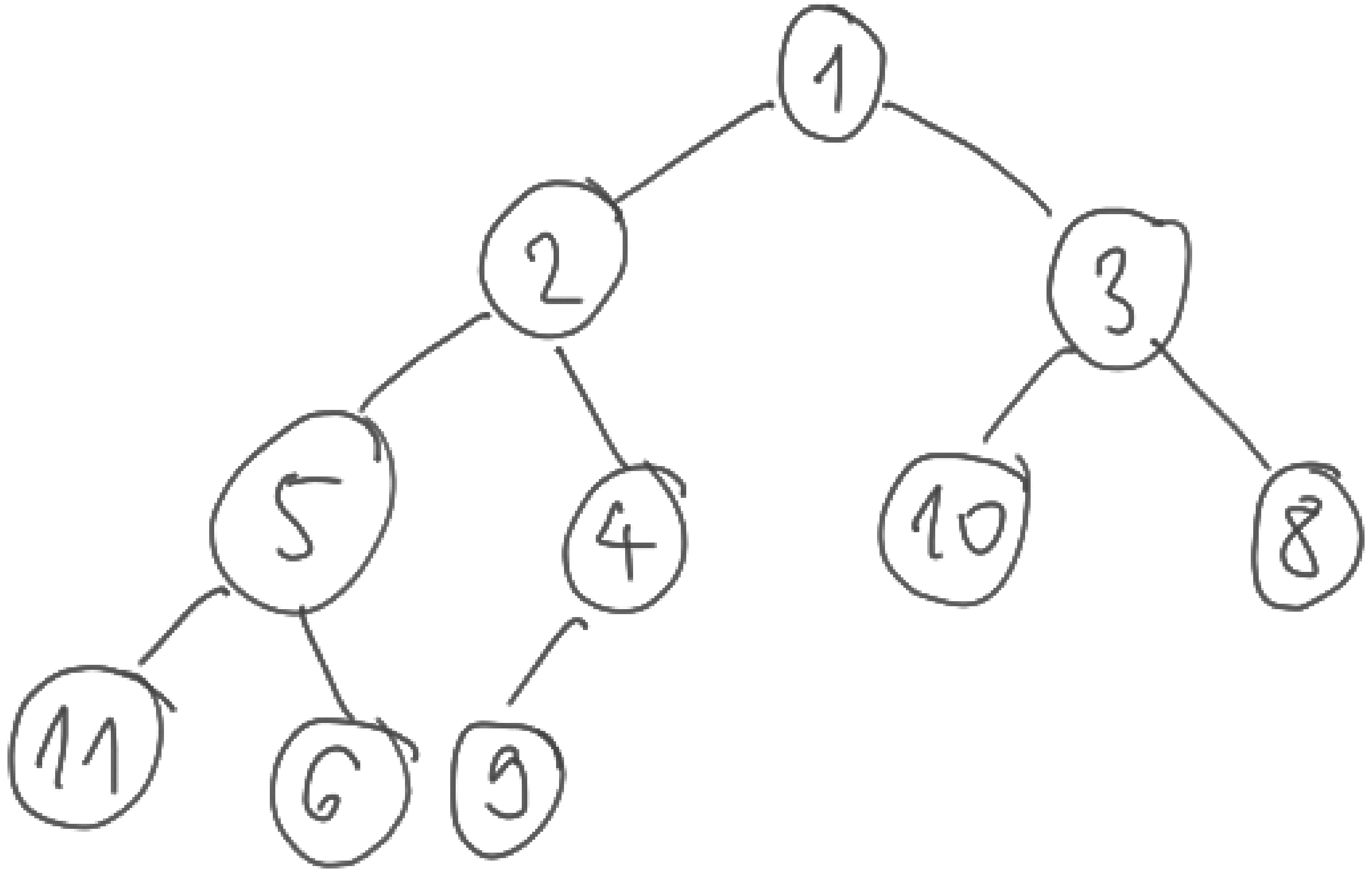
Dropping an exam

Before at least 24 hours.

If somebody misses an exam, he or she must pay a fee to retake another one

Priority queue (heap)

Visually



(nearly) complete binary tree

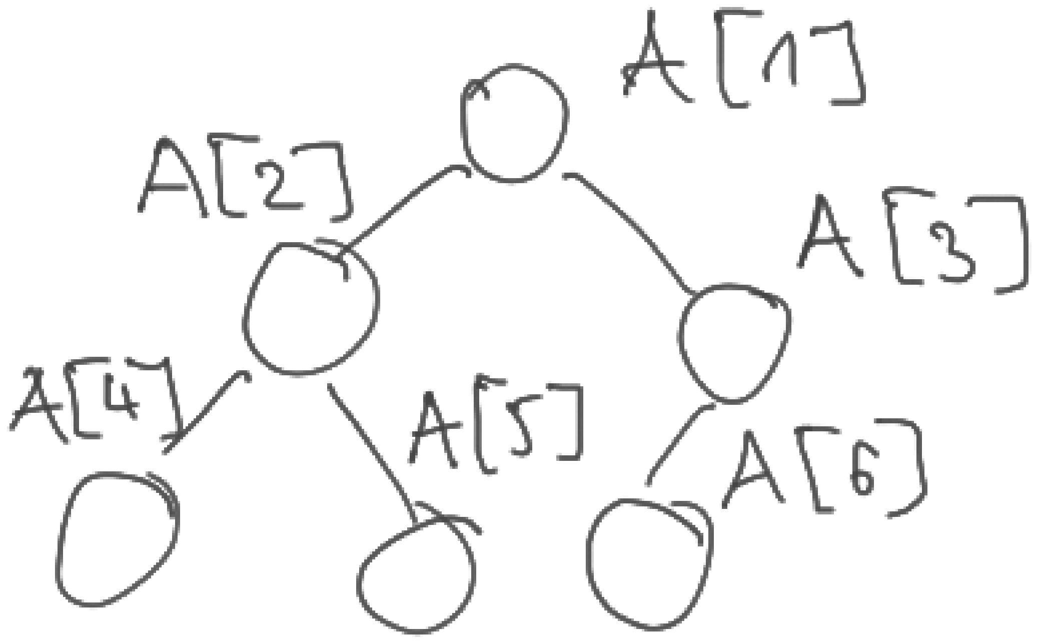
max heap

min heap

minheap property

for each vertex the key in this vertex is not greater than the keys in its children

(nearly) complete binary trees ↔ arrays

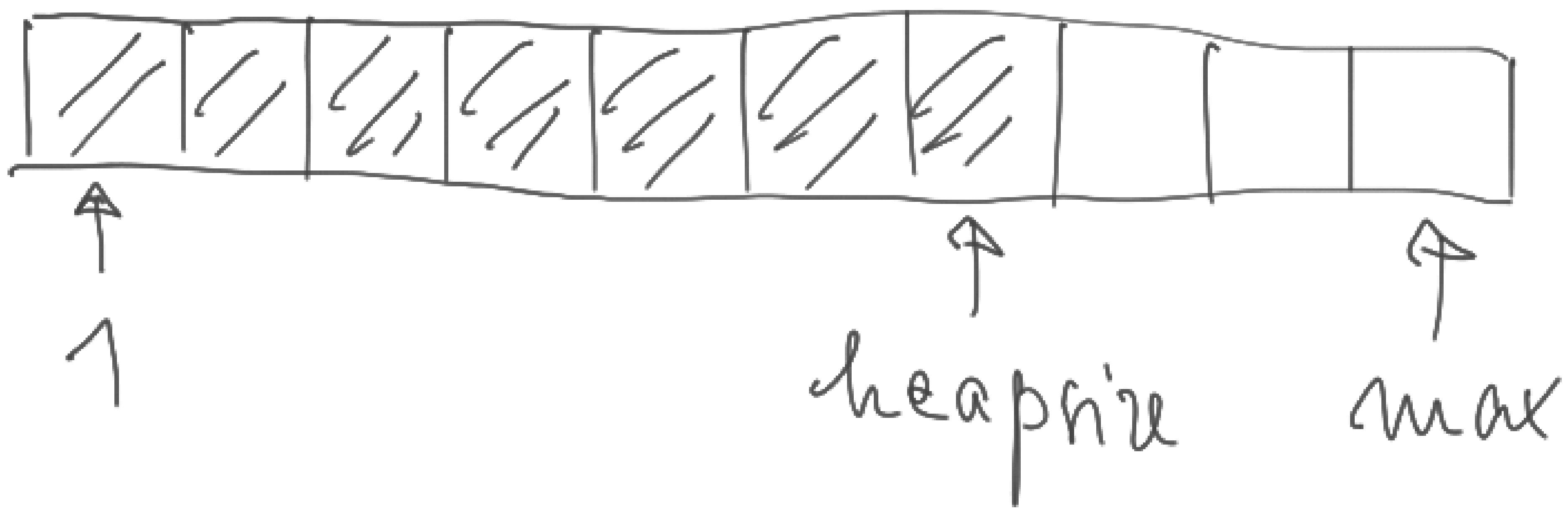


Priority queue

initialize an array A[1 : maxheapsize]

we will able to store at most maxheapsize data elements

If we store heapsize data elements, then they occupy the first part of A from index 1 to index heapsize



Two operations

① insert a new data element

② delete the smallest data element

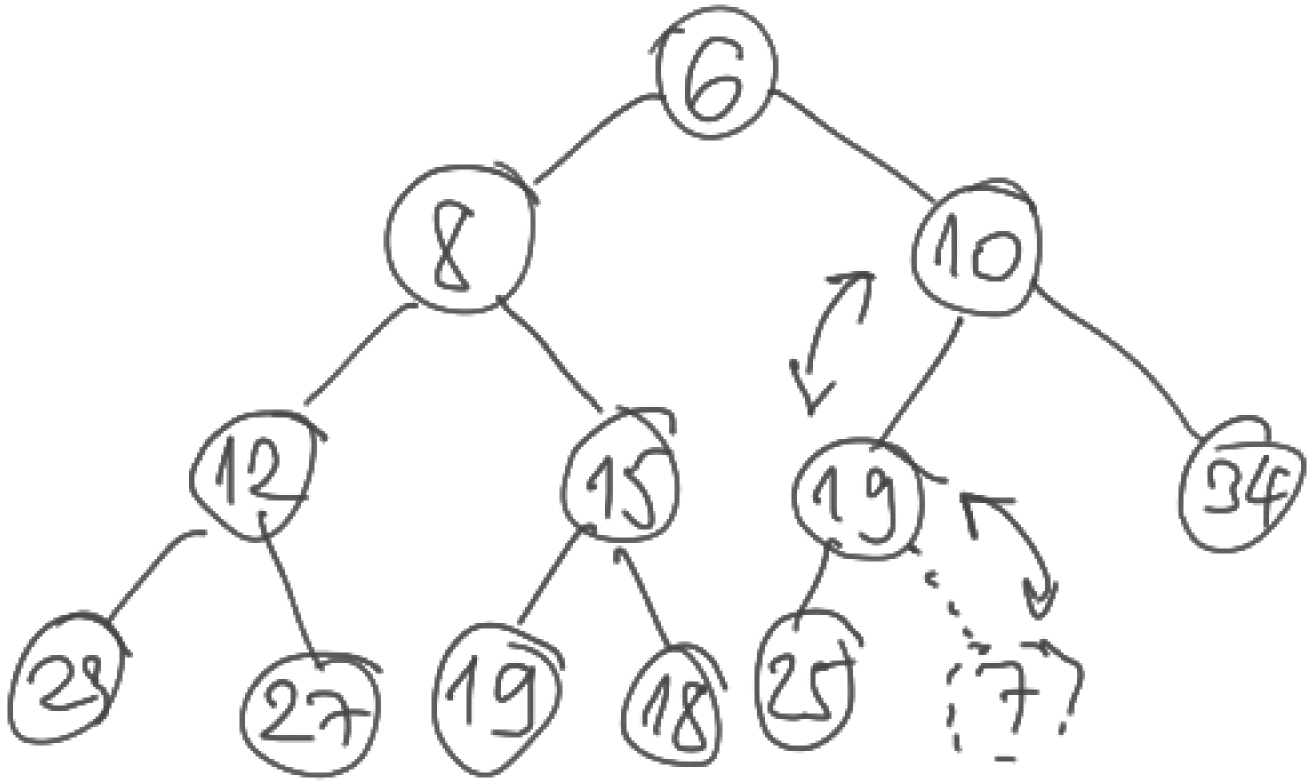
It's important to maintain the heap property

Both operations can be implemented

in $O(\log \text{heapsize})$ time

Insert

Visually



Insert $\boxed{7}$

we add first a new leaf with 7, and then we restore the heap prop.

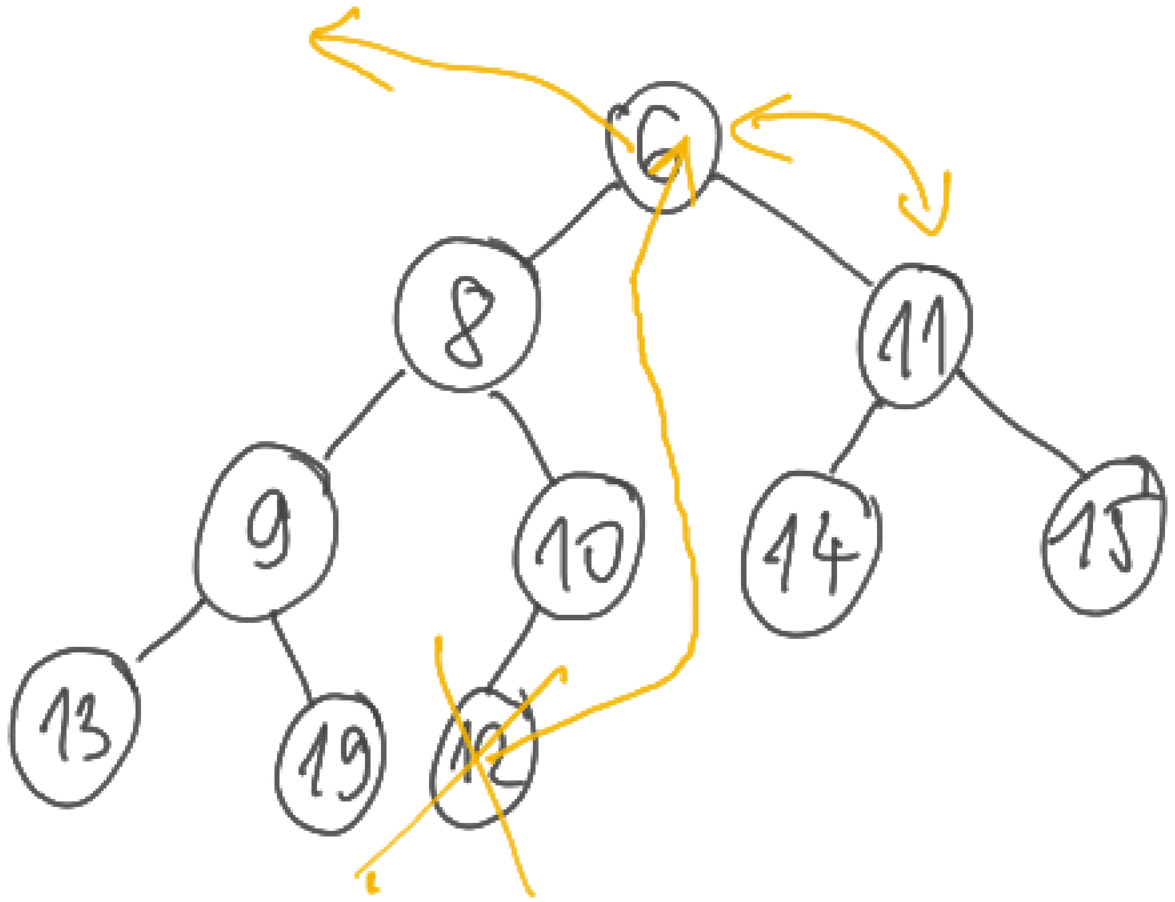
by a sequence of parent - child swaps

```
Insert (A, new)
 if heapsize[A] = maxheapsize[A] then
     error heap overflow
 heapsize[A] ++
 i := heapsize[A]
 A[i] := new
 while i > 1 AND A[int(i/2)] > A[i] do
     swap(A[i], A[int(i/2)])
     i := int(i/2)
```

Deleting the min

we know this from heapsort

Visually



we remove the "last" leaf and we replace the root's key by the key of the removed leaf, then we restore the heap prop.

by a sequence of parent-child swaps

```
MinDelete (A)
if heapsize[A] = 0 then
    error heap underflow
min := A[1]
A[1] := A[heapsize[A]]
heapsize[A] --
Heapify (A, 1)            remember in heapsort!
return min
```

For the sake of completness

```
Heapify (A, i)                    (recursive)
$l := 2i$ , $r := 2i+1$
if $l \le$ heapsize[A] AND $A[l] < A[i]$
    then smallest := $l$
    else smallest := $i$
if $r \le$ heapsize[A] AND $A[r] < A[\text{smallest}]$
    then smallest := $r$
⋮
```

```
⋮
if smallest $\neq$ i then
    swap ( A[i] , A[smallest]
    Heapify (A , smallest)
```

The cost is logarithmic in both cases because we walk along only one root - leaf path in one direction

Topic list for the exam

1. computational problems
   easy, hard, impossible
2. Comparision based sorting algorithms
   bubble sort, insertion sort
3. Divide and conquer design principle
   mergesort + merge
4. Asymptotic notations

⑤ Heapsort (special max selection sort)

⑥ Randomized algorithms

Quicksort, Quickselect

⑦ Simple data structures

stacks, queues, linked lists

⑧ Binary search trees

⑨ Heaps (priority queue)

⑩ Hashing