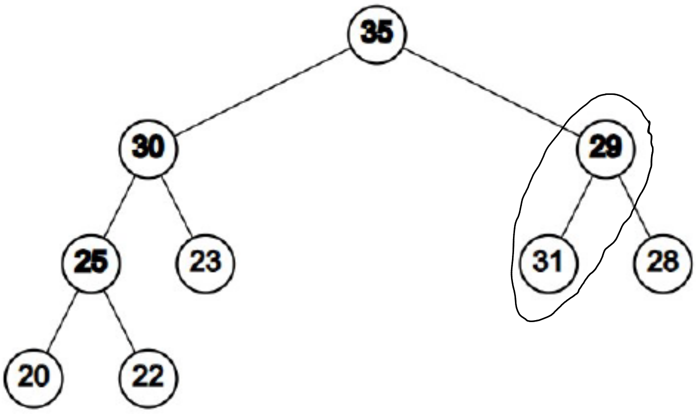
QUIZ-NACHBESPRECHUNG



True or false: *The binary tree above satisfies the max heap condition (for every node).*

False

Let T be a (max) heap. Let v be a node at depth 2 in T, and let w be a node at depth 4 in T.

Which of the following statements about the keys of v and w is true?

a. The key of v **must be larger than or equal to** the key of w.

b. The key of w **must be larger than or equal to** the key of v.

c. None of the above.

In the lecture you saw the $\mathrm{ExtractMax}$ procedure to remove the root of a (max) heap, and subsequently restore the heap condition.

Let ℓ be the rightmost leaf in the lowest level of a max heap consisting of at least 2 nodes, and assume all keys in the heap are unique.

True or false: *The node ℓ is guaranteed to be a leaf again after running the* $\mathrm{ExtractMax}$ *procedure.*

(A leaf is a node without children)

False

True or false: *the runtime of the quick sort algorithm depends on the way in which the pivot is chosen.*

True

Let B be the binary decision tree of a comparison-based sorting algorithm for arrays of length n.

Which of the following statements are true for B?

a. B contains at most 2^n nodes.

b. \(B \) contains at least \(n!\) nodes.

c. \(B \) has depth at least \(\Omega(n \log n) \).

d. \(B \) has depth at most \(O(n \log n) \).

SERIE 3 - COMMON MISTAKES

$$= \lim_{n\to\infty} \frac{3}{\ln(n)}$$

$$\lim_{n\to\infty} n^{\frac{3}{\ln(n)}} = e^{\ln(n)\cdot\frac{3}{\ln(n)}} = e^3 \neq n^0 = 1$$

NACHBESPRECHUNG SERIE 4

claim	true	false
$\frac{n}{\log n} \leq O(\sqrt{n})$	☐	☐
$\log(n!) \geq \Omega(n^2)$	☐	☐
$n^k \geq \Omega(k^n)$, if $1 < k \leq O(1)$	☐	☒
$\log_3 n^4 = \Theta(\log_7 n^8)$	☒	☐

claim	true	false
$\frac{n}{\log n} \geq \Omega(n^{1/2})$	☐	☐
$\log_7(n^8) = \Theta(\log_3(n^{\sqrt{n}}))$	☐	☐
$3n^4 + n^2 + n \geq \Omega(n^2)$	☐	☐
(*) $n! \leq O(n^{n/2})$	☐	☒

$\log_3(n^4) = \underline{\frac{4}{\ln(3)}} \cdot \ln(n)$

$\log_7(n^8) = \underline{\frac{8}{\ln(7)}} \cdot \ln(n)$

$$n! = \prod_{i=1}^{n} i \geq \prod_{i=0{,}1n}^{n} i \geq \prod_{i=0{,}1n}^{n} 0{,}1n = \left(\frac{n}{10}\right)^{0{,}9n}$$

$$\lim_{n\to\infty} \frac{n^{0{,}5n}}{\left(\frac{n}{10}\right)^{0{,}9n}} = \lim_{n\to\infty} 10^{0{,}9n} \cdot n^{-0{,}4n} = \underbrace{\left(\frac{10^{\frac{9}{4}}}{n}\right)}^{0{,}4n} = 0$$

Algorithm 4

(a)
$i \leftarrow 1$
while $i \leq n$ **do**
  $j \leftarrow 1$
  while $\sqrt[i]{j} \leq n$ **do**
    $f()$
    $j \leftarrow j + 1$
  $i \leftarrow i + 1$

Hint: *You may use the formula for a finite geometric series without proof*

$$\sum_{i=0}^{n} ar^i = \frac{a(r^{n+1} - 1)}{r - 1} \text{ for } r \neq 1.$$

Algorithm 5

(b) **function** A(n)
  $i \leftarrow 1$
  while $i \leq n$ **do**
    $j \leftarrow i$
    while $j \leq n$ **do**
      $f()$
      $f()$
      $j \leftarrow j + 1$
    $i \leftarrow i + 1$
  $k \leftarrow \lfloor \frac{n}{2} \rfloor$
  for $\ell = 1 \ldots 3$ **do**
    if $k > 0$ **then**
      $A(k)$

$$T(n) = \sum_{i=1}^{n}\sum_{j=i}^{n} 2 + 3T\left(\frac{n}{2}\right) = \sum_{i=1}^{n} 2(n-i+1) + 3T\left(\frac{n}{2}\right)$$

$$= 2n^2 - 2\sum_{i=1}^{n} i - 1 + 3T\left(\frac{n}{2}\right)$$

$$= 2n^2 - 2\sum_{i=0}^{n-1} i + 3T\left(\frac{n}{2}\right)$$

$$= 2n^2 - 2 \cdot \frac{(n-1)n}{2} + 3T\left(\frac{n}{2}\right)$$

$$= n^2 + n + 3T\left(\frac{n}{2}\right)$$

$n^2 + 3T(\frac{n}{2}) \leq T(n) \leq 2n^2 + 3T(\frac{n}{2})$

$a = 3$
$b = 2$
$c = 1 \vee 2$

$\log_2(a) = \log_2 3$
$< \log_2 4 = 2 = b$

$\Rightarrow \Theta(n^2)$

$T(n) = n^2 + n + 3T(\frac{n}{2})$

(c)* Prove that the function $T : \mathbb{N} \to \mathbb{R}^+$ from the code snippet in part (b) is indeed increasing.

Hint: *You can show the following statement by mathematical induction: "For all $n' \in \mathbb{N}$ with $n' \leq n$ we have $T(n'+1) \geq T(n')$".*

B.C.: $T(2) = 2^2 + 2 + 3\underbrace{T(1)}_{=2} = 12 \geq 2 = T(1)$

I.H.: siehe Hint

I.S.: $T(k+1) = (k+1)^2 + (k+1) + 3T(\lfloor \frac{k+1}{2} \rfloor)$

$\geq k^2 + k + 3T(\lfloor \frac{k+1}{2} \rfloor) \underset{\substack{\uparrow \\ \text{I.H.}}}{\geq} k^2 + k + 3T(\lfloor \frac{k}{2} \rfloor) = T(k)$

Case Distinction:

① k gerade

$\lfloor \frac{k+1}{2} \rfloor = \lfloor \frac{k}{2} \rfloor$

② k ungerade

$\lfloor \frac{k+1}{2} \rfloor = \lfloor \frac{k}{2} \rfloor + 1$

FINDING CLOSED FORM EXPRESSIONS FOR RECURSIVE FORMULAS

Beispiel

$T(1) = c$ und $T(n) = T(\frac{n}{2}) + d$ $\quad n = 2^k$ für $k \in \mathbb{N}$

1. Möglichkeit: Kleine Werte berechnen und "raten"

$T(1) = c \quad T(2) = c + d \quad T(4) = c + 2d$

$T(n) = d \cdot \log_2(n) + c$

2. Möglichkeit: Teleskopieren

$$T(n) = T(\tfrac{n}{2}) + d = T(\tfrac{n}{4}) + d + d = T(\tfrac{n}{8}) + 3d = \ldots$$

$$= \log_2(n) \cdot d + c$$

am Ende Gültigkeit formal mit Induktion beweisen!

MASTER THEOREM

Prüfungsaufgabe (HS20)

Theorem 1 (Master theorem) *Let $a, C > 0$ and $b \geq 0$ be constants and $T : \mathbb{N} \to \mathbb{R}^+$ a non-decreasing function such that for all $k \in \mathbb{N}$ and $n = 2^k$,*

$$T(n) \leq aT(n/2) + Cn^b.$$

Then

- *If $b > \log_2 a$, $T(n) \leq \mathcal{O}(n^b)$.*
- *If $b = \log_2 a$, $T(n) \leq \mathcal{O}(n^{\log_2 a} \cdot \log n)$.*
- *If $b < \log_2 a$, $T(n) \leq \mathcal{O}(n^{\log_2 a})$.*

Consider the following recursive function that takes as an input a positive integer m that is a power of two (that is, $m = 2^k$ for some integer $k \geq 0$).

Algorithm 1 $g(m)$

```
if m > 1 then
    g(m/2)
    g(m/2)
    for i = 1, ..., 6⌊√m⌋ do
        f()
else
    f()
```

Let $T(m)$ be the number of calls of the function f in $g(m)$.

i) Give a recursive formula for $T(m)$. Don't forget to provide the base case as well.

$$T(m) = \begin{cases} 2 \cdot T\left(\frac{m}{2}\right) + \sum_{i=1}^{6\sqrt{m}} 1 = 2T\left(\frac{m}{2}\right) + 6\lfloor\sqrt{m}\rfloor & \text{if } m > 1 \\ 1 & \text{else} \end{cases}$$

ii) Determine $T(m)$ in $\mathcal{O}$-notation. Your answer should be as tight as possible.

$a = 2 \qquad b = \frac{1}{2} \qquad C = 6$

$b = \frac{1}{2} < \log_2 a = \log_2 2 = 1$

$\Rightarrow T(m) = \Theta(m^{\log_2 a}) = \Theta(m)$

SORTIEREN-RECAP LETZTE WOCHE

	Vergleiche	Vertauschungen	Speicher
Bubblesort	$O(n^2)$	$O(n^2)$	$O(1)$
Selectionsort	$O(n^2)$	$O(n)$	$O(1)$
Insertionsort	$O(n \log n)$	$O(n^2)$	$O(1)$
Mergesort	$O(n \log n)$	$O(n \log n)$	$O(n)$

QUICKSORT

- rekursives Aufteilen des Arrays (ähnlich wie bei Mergesort)
- Wahl eines Pivotelements p
  - $\rightarrow$ wenn $A[i] < p$: linkes Teilarray
  - $\rightarrow$ sonst rechtes Teilarray

QUICKSORT$(A[1..n], l, r)$

---

if $l < r$ **then**
    $k \leftarrow$ Aufteilen(A, l, r) $\triangleright$ *Teile $A[l..r]$ in zwei Gruppen auf*
    Quicksort$(A, l, k-1)$ $\triangleright$ *Sortiere linke Gruppe*
    Quicksort$(A, k+1, r)$ $\triangleright$ *Sortiere rechte Gruppe*

AUFTEILEN$(A[1..n], l, r)$

$p \leftarrow A[r]$ $\triangleright$ *Pivotelement*
$k \leftarrow$ Zahl der Elemente $\leq p$ in $A[l..r]$
$B \leftarrow$ neues Array mit $r - l + 1$ Zellen $\triangleright$ *so gross wie $A[l, \ldots, r]$*
$B[k] \leftarrow p$ $\triangleright$ *Pivot muss an k-te Stelle*
$i \leftarrow l$ $\triangleright$ *Anfang des linken Teils von B*
$j \leftarrow k+1$ $\triangleright$ *Anfang des rechten Teils von B*
for $s \leftarrow l, l+1, \ldots, r$
    if $A[s] \leq p$ **then**
        $B[i] \leftarrow A[s]$ $\triangleright$ *Schreibe $A[s]$ in linke Hälfte*
        $i \leftarrow i + 1$
    else
        $B[j] \leftarrow A[s]$ $\triangleright$ *Schreibe $A[s]$ in rechte Hälfte*
        $j \leftarrow j + 1$
kopiere B nach $A[l..r]$

worst case: man wählt als Pivot ein Element am Rand des sortierten Bereichs

Laufzeit: $O(n^2)$, mit randomisierter Pivotwahl: $O(n \log n)$

HEAPSORT

Max-Heap: binäre Baum der folgende Bedingungen erfüllt:

1) Vollständigkeit
2) Heap-Bedingung: Der Schlüssel jedes Knotens ist grösser oder gleich den Schlüsseln seiner Kinder

---

HEAPSORT($A[1..n]$) $O(n \log n)$

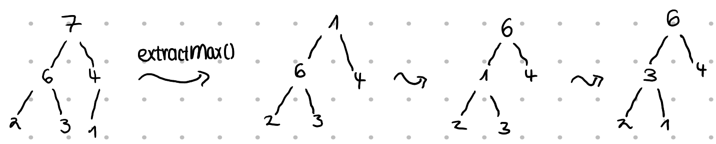
$H \leftarrow$ Heapify(A) ▷ *Wandle Array in Heap um.*

for $i \leftarrow n, n-1, \dots, 1$ **do** ▷ *Entferne Elemente aus Heap*

  $A[i] \leftarrow$ ExtractMax(H) $\leftarrow O(\log n)$

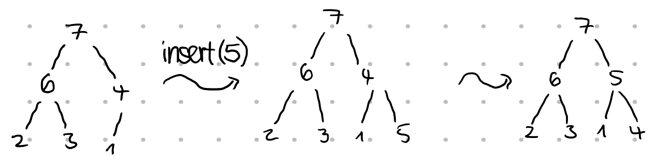
ExtractMax(H) $O(\log n)$

einfaches Beispiel:



- Wurzel entfernen
- letztes Blatt an Wurzelposition verschieben
  → evtl. verletzte Heapbedingung wiederherstellen

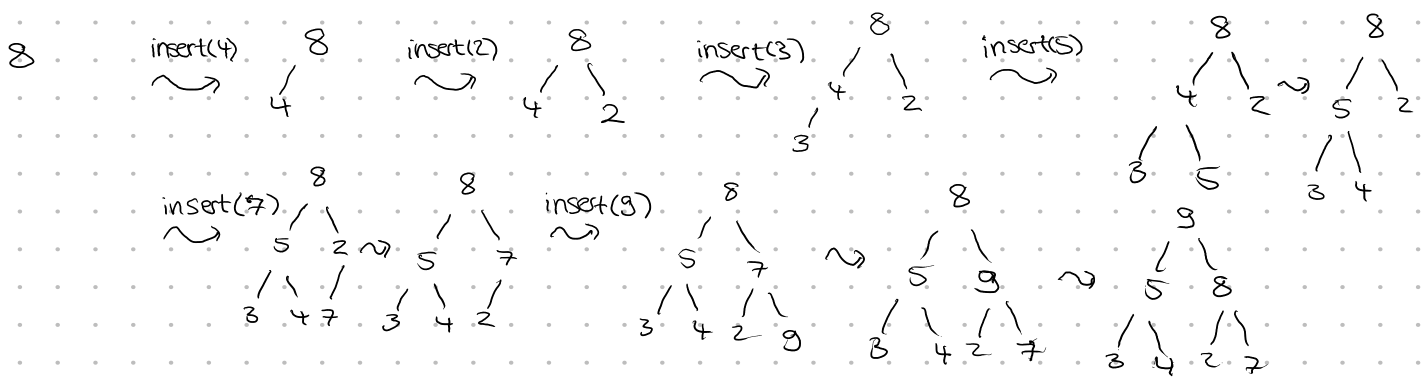
Insert(H,p) $O(\log n)$



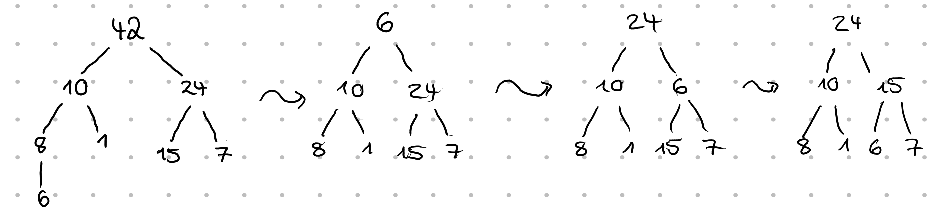
- neuen Knoten v mit Schlüssel p an nächster freier Stelle einfügen
- evtl. verletzte Heapbedingung durch Tausch mit Elternknoten wiederherstellen

Prüfungsaufgabe HS20 (2P.)

a) Füge die Schlüssel 8,4,2,3,5,7,9 nacheinander in einen leeren Heap ein



b) Zeichne den Max-Heap nach ExtractMax() (mit Zwischenschritten)



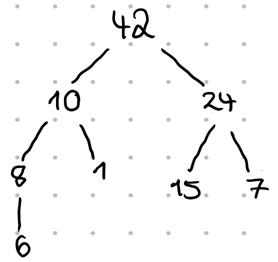
Darstellung eines Binärbaums im Speicher

→ Array

→ Kinder des Knotens k sind an Stelle 2k+1 und 2k+2

Aufgabe:

Stelle folgenden Binärbaum als Array im Speicher dar

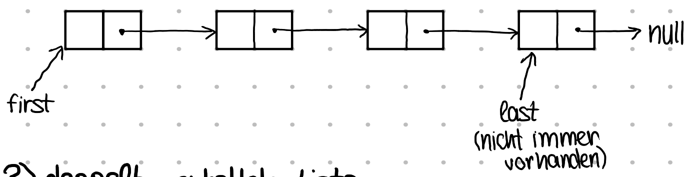


0	1	2	3	4	5	6	7
42	10	24	8	1	15	7	6

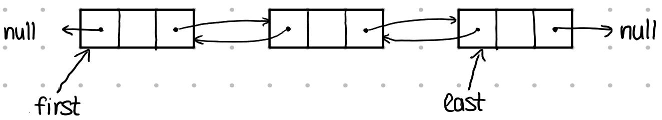
ADT LISTE

1) Arrays (schon bekannt aus EProg)

2) einfach verkettete Liste



3) doppelt verkettete Liste



Laufzeitenüberblick

	Array	einf. verkettete Liste	dopp. verkettete Liste
insert(k,L)	$O(1)$	$O(1)$	$O(1)$
get(i,L)	$O(1)$	$O(n)$	$O(n)$
insertAfter(k,k',L)	$O(n)$	$O(1)$	$O(1)$
delete(k,L)	$O(n)$	$O(n)$	$O(1)$