Algorithmen & Datenstrukturen

Übungsstunde 6

Michelle Honegger - CAB G59 - 27.10.2024

Miniquiz 5

- Passwort: **datastructure4**
- 10 Fragen
- 12 Minuten
- 0.1 Punkte pro richtige Antwort

Programm

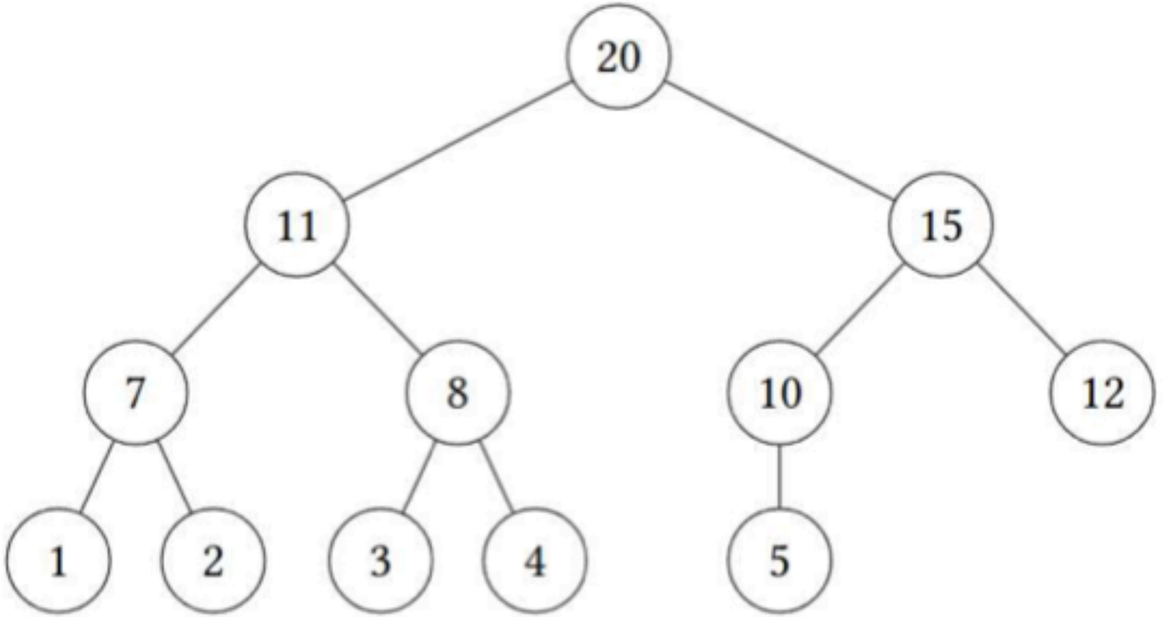
- Miniquiz
- Rückmeldung Serie 4
- Theory Recap
- Prüfungsaufgaben
- Besprechung Aufgaben

Rückmeldung Serie 4

- Vorsichtig mit indexing
- Elemente und Positionen unterscheiden
- Bei ‘exact number of calls’ braucht es auf-/abrunden von log,√,/2 u.ä.
- Rot ist meine Farbe

Besprechung Miniquiz

Consider the following heap.



Suppose we extract the number 20. What number will occupy the position currently held by 15, after the insertion?

Answer:

In a max-heap, every key at depth i is larger than every key at depth $i + 1$.

- ○ True
- ○ False

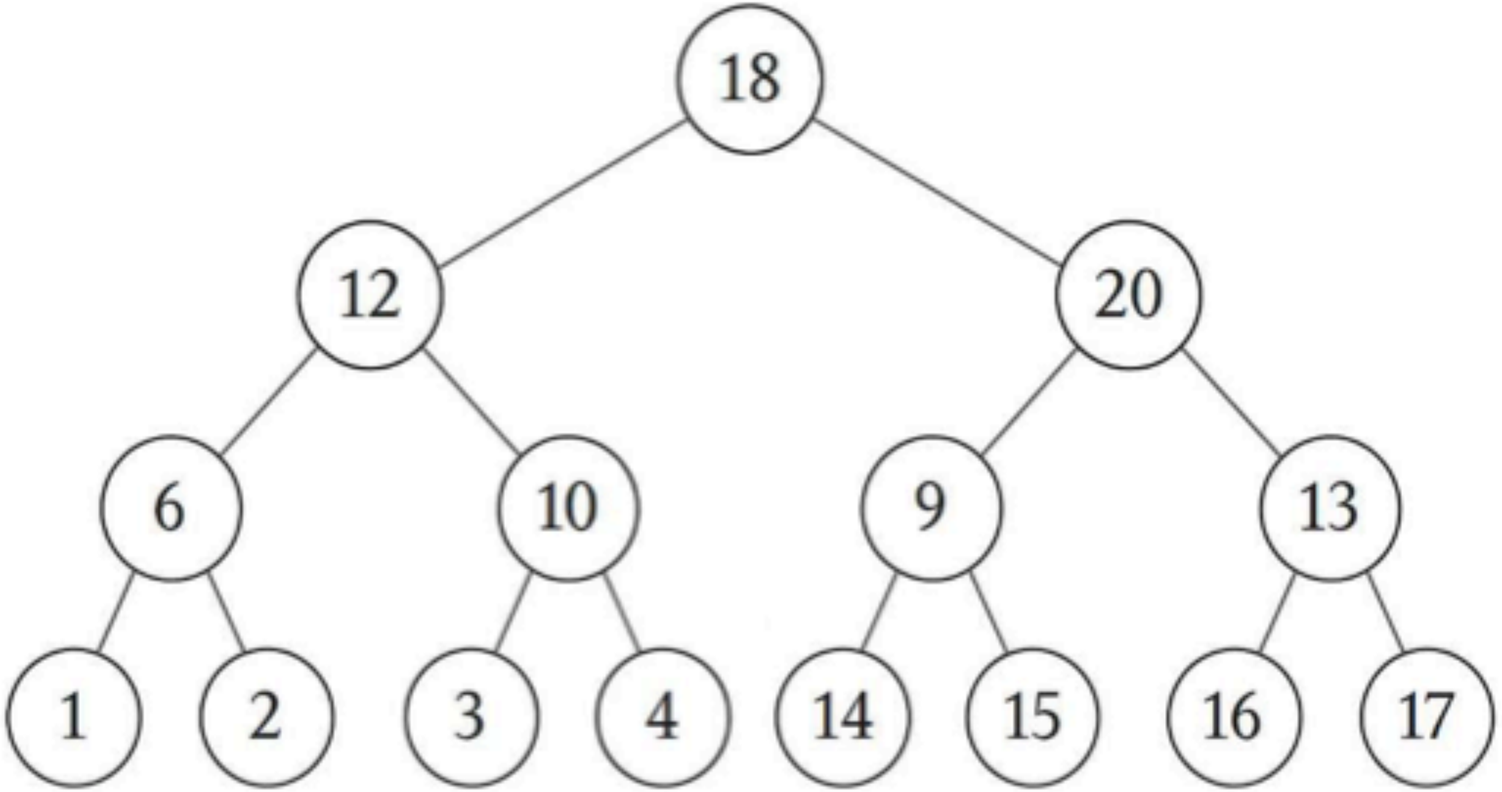
Binary Search in a doubly linked list takes time $O(\log n)$.

- ○ True
- ○ False

Consider the operation get(i), which returns the i-th item in a list. A 2-3-tree is a more efficient data structure for this operation than an array.

- ○ True
- ○ False

Consider the following tree.



How many vertices violate the max-heap property?

Answer:

A 2-3 tree with height h has $\Theta(2^h)$ leaves.

- ○ True
- ○ False

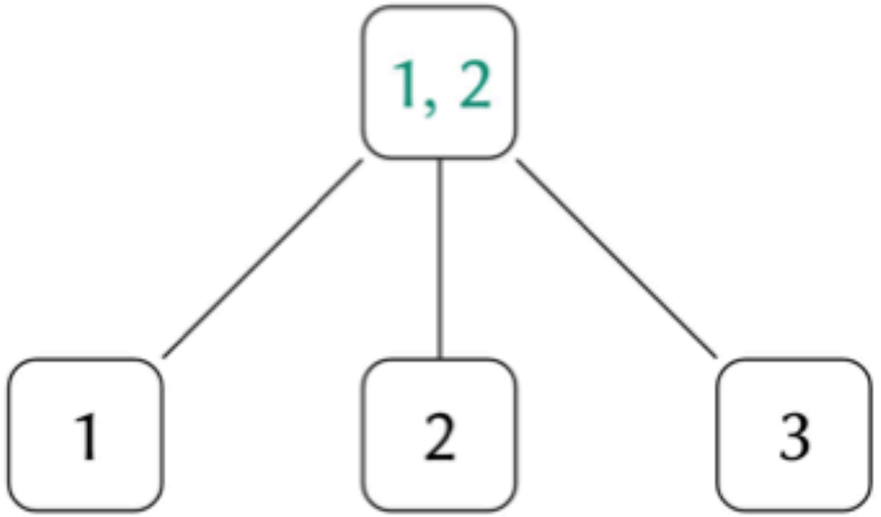
There exists an algorithm that can compute the edit distance between two strings $A[1 \ldots n]$ and $B[1 \ldots m]$ in time $O(mn)$ using extra space $O(n)$.

- ○ True
- ○ False

For two strings $A[1 \ldots n]$ and $B[1 \ldots n]$, let L be a longest common subsequence. Then an optimal sequence of edit operations to transform A into B (i.e., of minimal length) will not change any element of L.

- ○ True
- ○ False

Consider the following 2-3 tree:



What is the element in the root of the tree after we insert the number 4?

- a. 1,2
- b. 1, 2, 3
- c. 1, 3
- d. 2
- e. 1
- f. 4
- g. 3, 4

Consider the pseudocode below:

```
1: function Foo(n)
2:     if n ≤ 1 then
3:         return n
4:     else
5:         return Foo(n-1) + Foo(n-2)
6:     end if
7: end function
```

What is the asymptotic runtime of this function?

- ○ a. $\Theta(\log n)$
- ○ b. $\Theta(n)$
- ○ c. $\Theta(n^2)$
- ○ d. $\Theta(C^n)$, for some constant $C > 0$.

Theory Recap

Inhalt

- Abstrakte Datentypen 2
- Dynamische Programmierung 1

Binärbaum

- Für Knoten x sind Schlüssel in linkem Teilbaum < x und Knoten im rechten Teilbaum > x
- search, insert und detete in O(h)
  - Problem je nach Struktur (Baum unique für gegebene Reihenfolge -> h = n-1 möglich)
  - Lösung: Flexibilität durch 2-3 Bäume

2-3 Baum

- Jeder Parent Knoten hat 2-3 Kinder
- Bei 3 Kindern brauchen wir 2 Werte beim Parent Node
  - Linkes Kind - kleiner gleich linker Wert
  - Mittleres Kind - grösser linkem und kleiner gleich rechtem Wert
  - Rechtes Kind - grösser als rechter Wert

2-3 Baum Cont’d

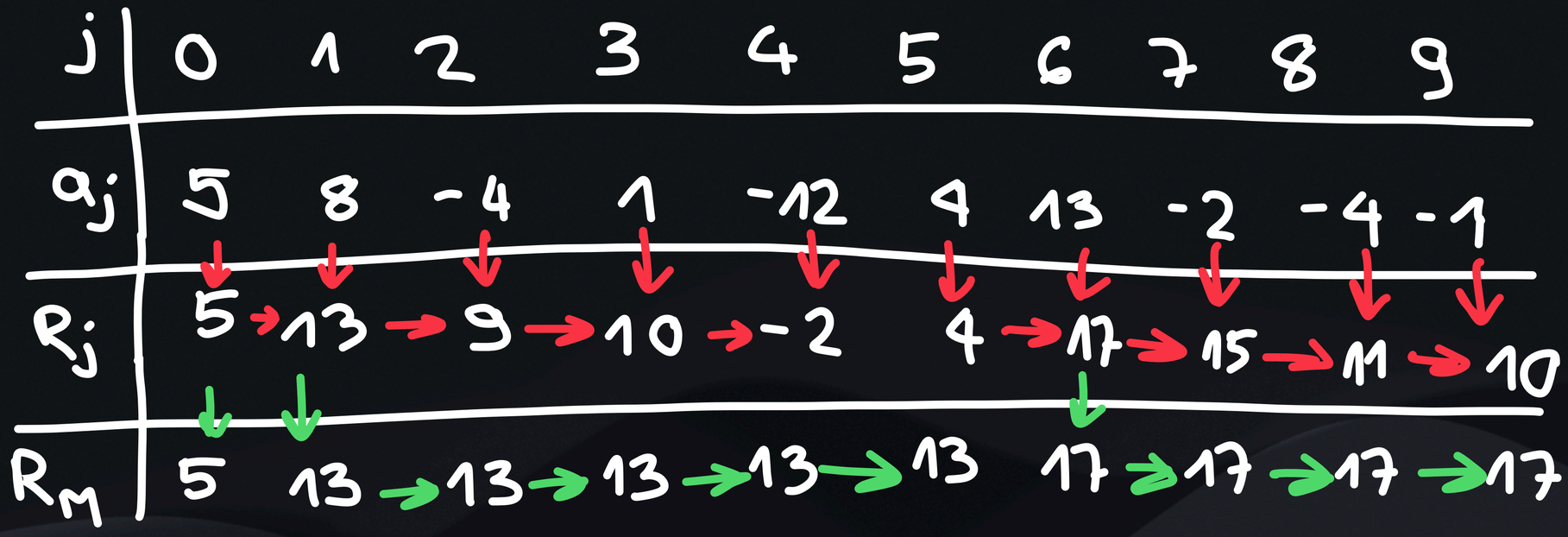
- find(x): Gleich wie in Binärbaum
- insert(x): Füge Blatt und Separator an richtiger Stelle ein, falls 3 Separatoren und 4 Kinder dadurch, immer Mittleren Separator hochziehen
- delete(x): Lösche Knoten und Separator bei Elternknoten, falls nur noch 1 Kind danach, verschmelze mit Geschwisterknoten

Fibonacci

- Top-down schlecht
  - Laufzeit ≥ Θ(1.5^n)
  - Selbes Resultat wird mehrfach berechnet
- Memoization/Bottom-up
  - Laufzeit ≤ O(n)
  - speichern bereits berechnete Resultate in Array

Maximum Subarray Sum

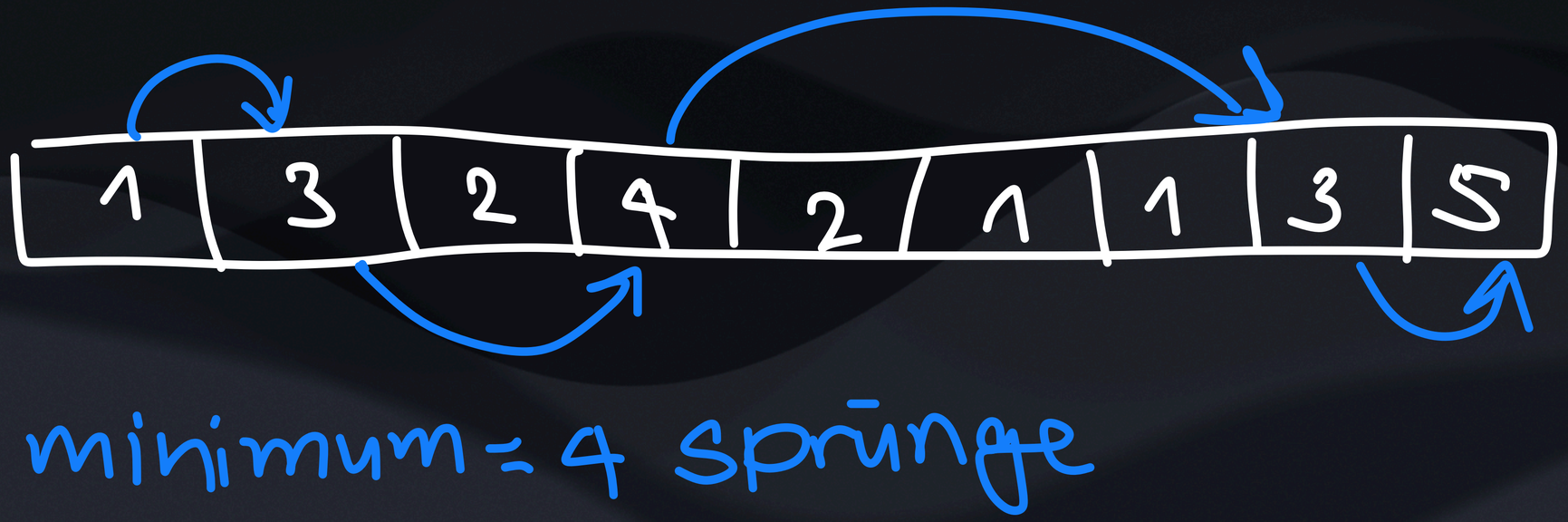
Beste Lösung



$$R_j = \begin{cases} R_{j-1} + a_j & \text{wenn } R_{j-1} \geq 0 \\ a_j & \text{sonst} \end{cases}$$

Jump Game

- Input: Array der Länge n mit positiven ganzen Zahlen
- Restriktion: von Feld i dürfen wir höchstens A[i] nach vorne
- Gesucht: minimale Zahl an Sprüngen um n zu erreichen



Jump Game

- Teilproblem: dp[k]:= maximaler index, der mit k Spüngen erreichbar ist
- Rekursion: dp[k] = $max\{i + A[i] \mid dp[k-2] < i \leq dp[k-1]\}$
- Laufzeit: O(n)

Longest Common Subsequence

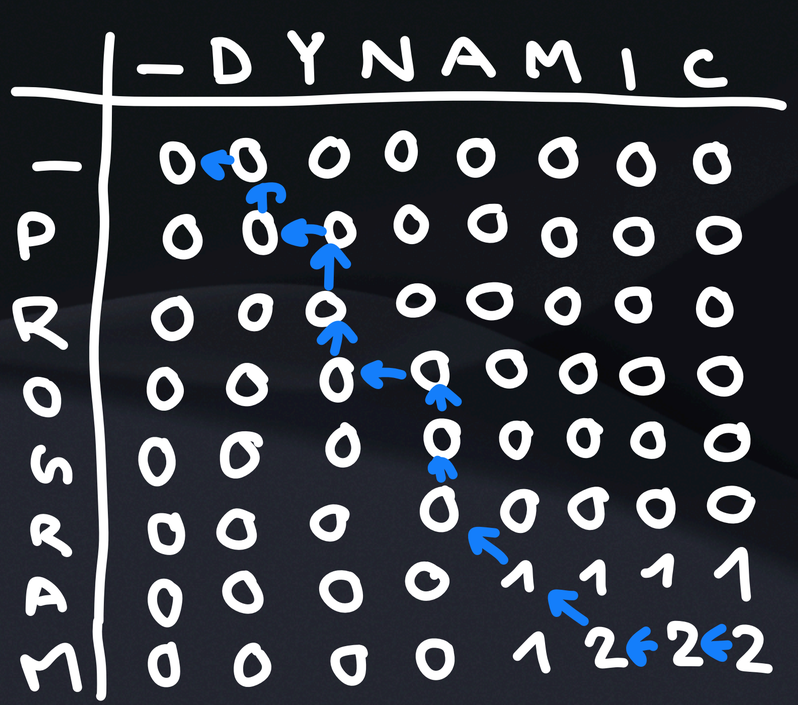
- Input: 2 Arrays mit Wörtern
- Restriktion: Teilfolge muss Reihenfolge beibehalten kann aber Stellen skippen
- Gesucht: Längste gemeinsame Teilfolge der Arrays

Longest Common Subsequence

- Teilproblem: dp[i,j] := Länge LGT von A[1..i], B[1..j]
- Rekursion:

$$dp[i][j] = \begin{cases} 0 & \text{, if } i = 0 \text{ or } j = 0 \\ 1 + dp[i-1, j-1] & \text{, if } A[i] = B[j] \quad \leftarrow \text{benutze } A[i] \text{ \& } B[j] \\ \max\{dp[i-1][j], dp[i][j-1]\} & \text{, sonst} \end{cases}$$

Laufzeit $O(nm)$

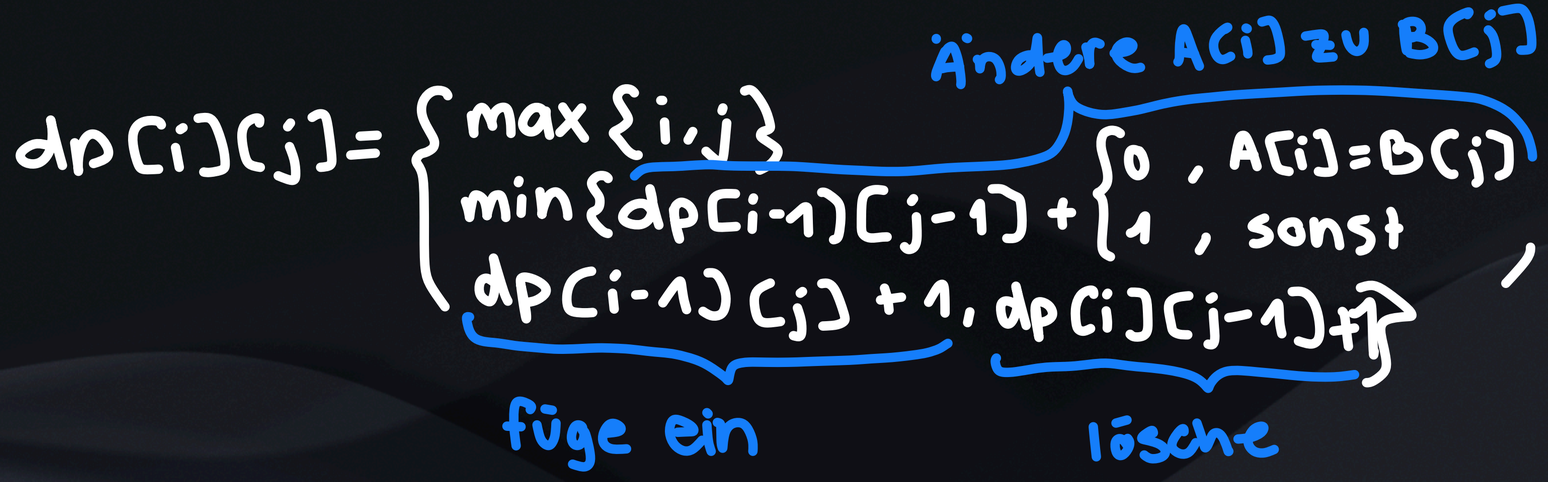


Edit Distance

- input: 2 Arrays mit Wörtern
- Restriktion: A in B umwandeln mit einfügen, löschen, ändern
- Gesucht: Editierdistanz

Edit Distance

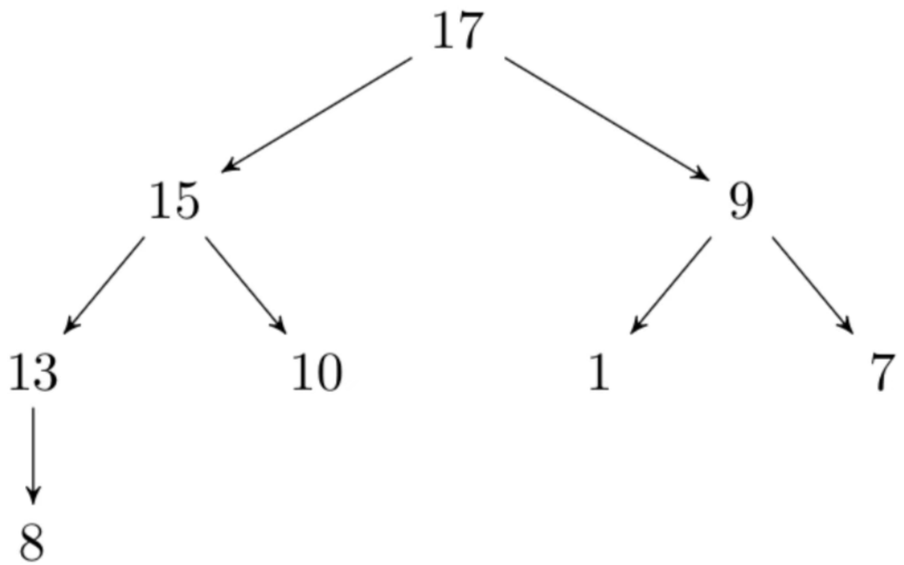
- Teilproblem: dp[i,j] := Editierdistanz von A[1..i] zu B[1..j]
- Rekursion: Was passiert mit A[i] in optimaler Lösung?



		_	T	I	R	E	D
		0	1	2	3	4	5
E		1	1	2	3	3	4
T		2	1	2	3	4	4
H		3	2	2	3	4	5

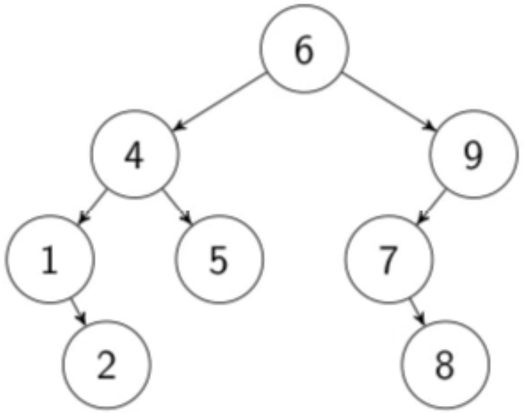
Prüfungsaufgaben

ii) Draw the Max-Heap obtained from the following Max-Heap by performing the operation DELETE−MAX once.



g) *Binary search trees*: Draw the binary search tree that is obtained when inserting into an empty tree the keys `7`, `2`, `1`, `5`, `4`, `8`, `9` in this order.

h) *Binary search trees*: Draw the resulting binary search tree obtained by deleting the keys `6` and `9` in this order from the following binary search tree.



Exercise Recommendations

Aufgabenblatt 5

- 6.1 Bonus
- 6.2 Da 2-3 Bäume neu sind -> Alle Aufgaben dazu machen!
- 6.3 Bonus
- 6.4 Bonus
- 6.5 Sehr gute Aufgabe zur Prüfungsvorbereitung (genau solche Aufgaben kommen häufig in Theorie und Programmierprüfung) falls jetzt keine Zeit, in Lernphase lösen

Leetcode of the Week

- 55. Jump Game
- https://leetcode.com/problems/jump-game

- 45. Jump Game II
- https://leetcode.com/problems/jump-game-ii

Peergrading

- Aufgabe 5.4
- Grading Scheme in Lösungen beachten