Algorithmen und Datenstrukturen, Blatt 6

03.06 2024

Yorick Schiffer – ys151
Paul Tröster – pt144

Ege Tekin – et130

Aufgabe 1

(a)

```
function findClosestValueInBstRecursive(tree, target, closest)
    if tree is None then
        return closest
    endif

    if abs(target - closest) > abs(target - tree.value) then
        closest = tree.value
    endif

    if target < tree.value then
        return findClosestValueInBstRecursive(tree.left, target, closest)
    elseif target > tree.value then
        return findClosestValueInBstRecursive(tree.right, target, closest)
    else
        return closest
    endif
endfunction
```

Die Zeitkomplexität beträgt O(N^2), da der Algorithmus zwei verschachtelte Schleifen verwendet, die jeweils fast N Iterationen durchlaufen, um die Abstände aller möglichen Punktpaare zu berechnen und zu vergleichen.

(b)

```
function findClosestValueInBstIterative(tree, target, closest)
    currentNode = tree

    while currentNode is not None do
        if abs(target - closest) > abs(target - currentNode.value) then
            closest = currentNode.value
        endif

        if target < currentNode.value then
            currentNode = currentNode.left
        elseif target > currentNode.value then
            currentNode = currentNode.right
        else
            break
        endif
    endwhile

    return closest
endfunction
```

Die iterative Lösung hat eine Zeitkomplexität von O(N), weil im ungünstigsten Fall, bei einem unausgeglichenen Baum (wie einer Liste), jeder Knoten einmal besucht wird, bevor der Zielwert erreicht wird.

Um den 'schwersten Pfad' bzw die größte Summe in einem Binären Suchbaum zu finden,
kann man in Post Order durch den Baum traversieren, und immer von den Blättern beginnend die größte Summe
speichern, und mir der nächsthöheren vergleichen.

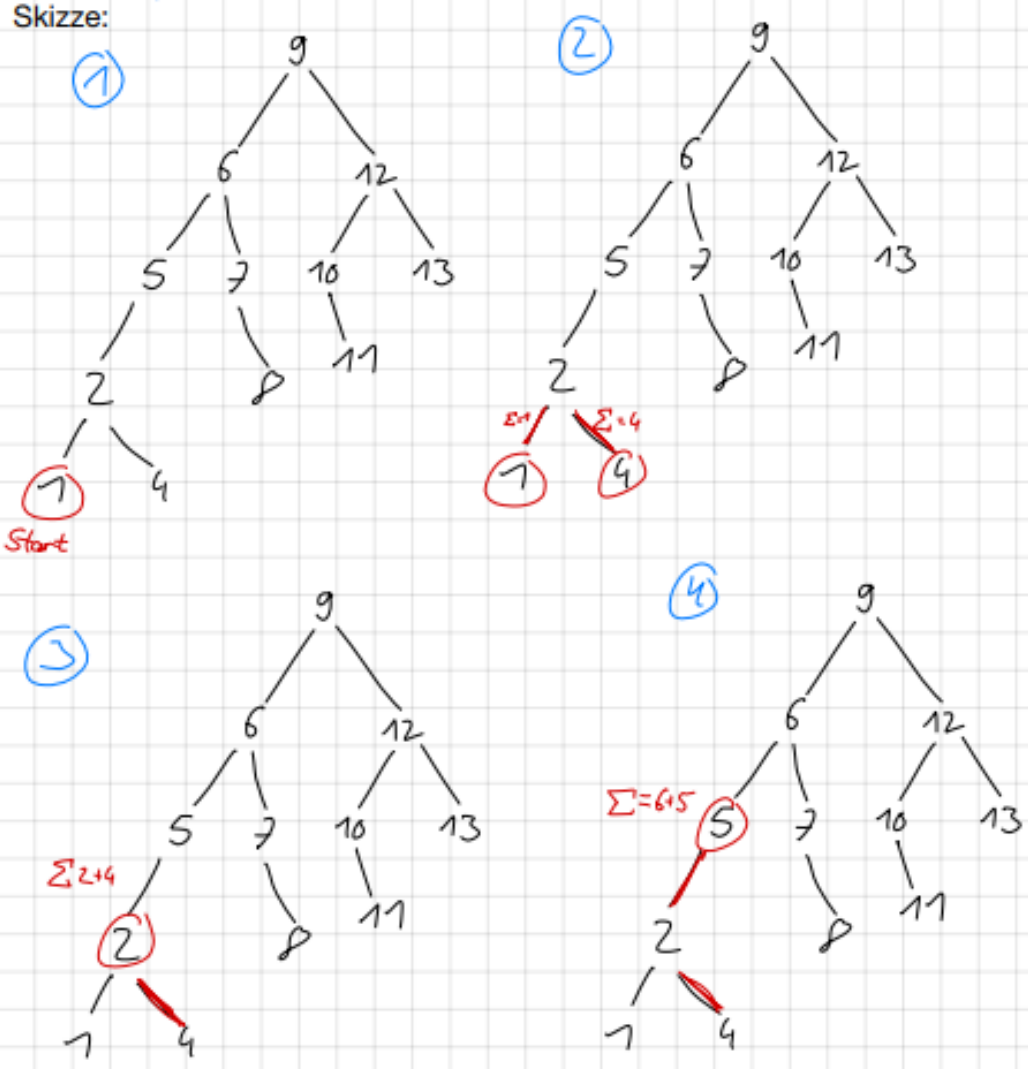
Man geht also zuerst zum 'untersten, linken' Blatt des Baums und begint damit, bei dem der nächst höheren Knoten zu vergleichen, ob der andere Pfad einen höheren wert hat. Die höhere Summe wird dann als höchste Summe des höheren knotens gespeichert und dann wiederum mit dem Knoten links oder rechts danneben verglichen.

So ruft man jeden Knoten/ Blatt des Baumes einmal auf und vergleicht die Summen der linken und rechten Unterbäume.

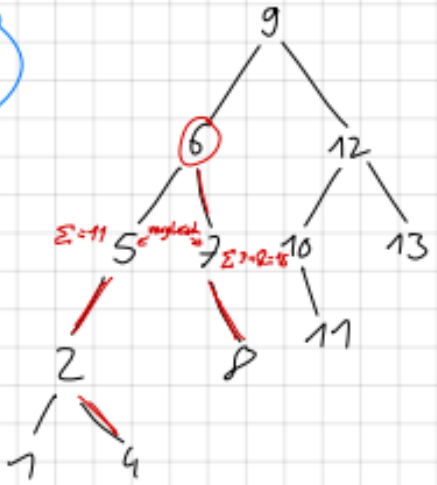
Das funktioniert sogar bei unsortierten Bäumen, da einfach nur die Summen weiter aufaddiert werden und egal ist ob die linken knoten kleiner sind als die rechten.

(ein paar schritte wurden übersprungen)

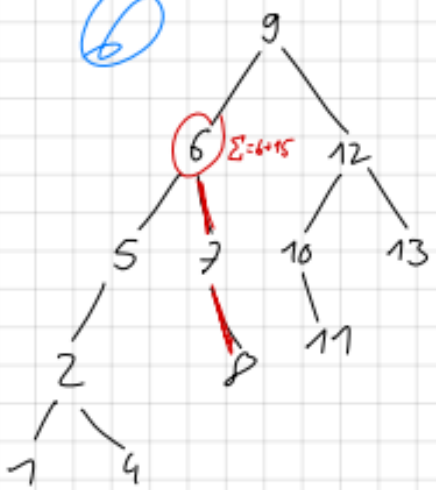
Skizze:



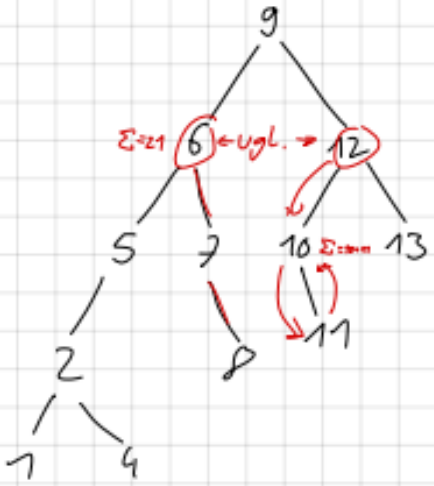
5


9
6
12
Σ=11
5
7
Σ=7+2=9
10
13
11
2
8
1
4

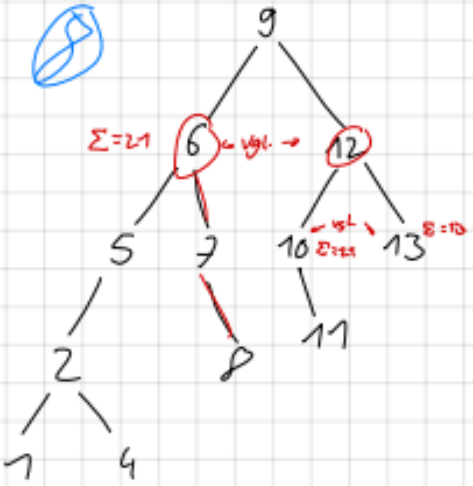
6


9
6
Σ=6+15
12
5
7
10
13
11
2
8
1
4

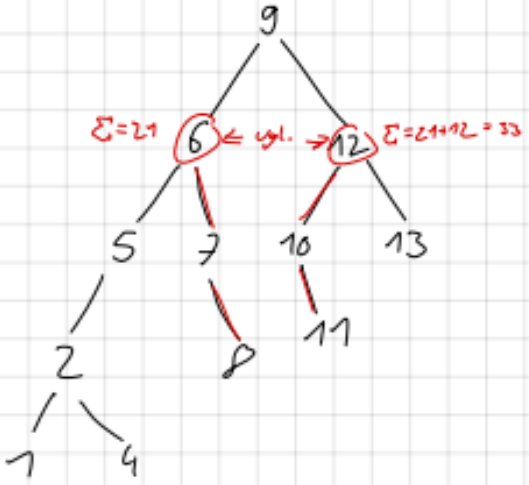
7


9
Σ=21
6
vgl.
12
5
7
10
Σ=...
13
11
2
8
1
4

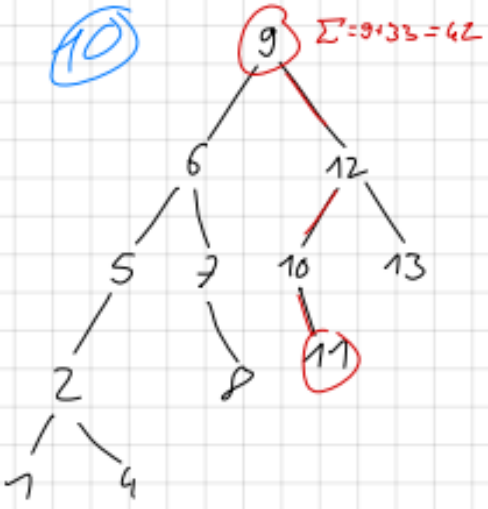
8


9
Σ=21
6
vgl.
12
5
7
10
Σ=21
vgl.
13
Σ=13
11
2
8
1
4

9


9
Σ=21
6
vgl.
12
Σ=21+12=33
5
7
10
13
11
2
8
1
4

10


9
Σ=9+33=42
6
12
5
7
10
13
11
2
8
1
4