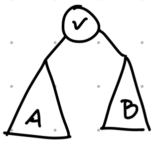
Exercise Sheet 6

(a)



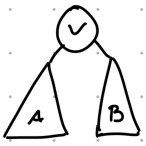
$\Rightarrow O(\log n)$

(c)

$s_l(v)$ $s_r(v)$

$\Downarrow$

Anz. Knoten im linken Teilbaum von v

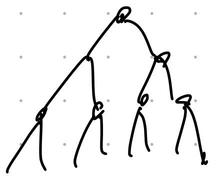


Fälle

1. $s_l(v) = i_0 - 1$ $\Rightarrow$ return v

2. $s_l(v) \geq i_0$ $\Rightarrow$ Wir müssen im linken Teilbaum suchen mit rekursivem Aufruf

3. $s_l(v) < i_0 - 1$ $\Rightarrow$ Rekursiver Aufruf im rechten Teilbaum mit $i_1 = i_0 - (s_l(v) + 1)$



Exercise 6.3

$A_1 = 1$, $A_2 = 2$, $A_3 = 3$, $A_4 = 4$

$A_n = A_{n-1} + A_{n-3} + 2A_{n-4}$ für $n \geq 5$.

(a) A (n)

```
if n ≤ 4
   return n
else
   return A(n-1) + A(n-3) + 2·A(n-4)
```

(b) $\Omega(c^n)$

$T(1) = T(2) = T(3) = T(4) = 1$

$T(n) = T(n-1) + T(n-3) + T(n-4) + d$

$\geq T(n-4) + T(n-4) + T(n-4)$

$= 3 \cdot T(n-4)$

$T(n) \geq 3T(n-4) \geq 3 \cdot 3T(n-8) \geq \ldots \geq 3^k \cdot T(n-4 \cdot k)$ $\qquad k < \frac{n}{4}$

$T(n) \geq 3^{n/4}$ passt noch nicht für $n = 1, 2, 3, 4$

$\Rightarrow T(n) \geq \frac{1}{3} \cdot 3^{n/4}$ $\qquad T(4) \geq 3 \cdot \frac{1}{3}$

$\Omega\left(\left(3^{1/4}\right)^n\right)$

(c)

mem $\leftarrow$ n-dimensional, filled with -1 's

function A_Mem(n)

```
if mem[n] ≠ -1
    return mem[n]
if n ≤ 4
    mem[n] ← n
    return n
else
    mem[n] ← A-Mem(n-1) + A-mem(n-3) + 2· A-mem(n-4)
    return mem[n]
```

$\Rightarrow \Theta(n)$

(d)

(1) Def. of DP-Table

$1 \times n$

$DP[i] \mathrel{\hat{=}} A_i$

Calculation of an entry

Base cases:

$DP[i] \leftarrow i$ für $1 \leq i \leq 4$

$i \geq 5$ $\qquad DP[i] \leftarrow DP[i-1] + DP[i-3] + 2 \cdot DP[i-4]$

Order

i aufsteigend

Extracting the solution

$DP[n]$

Runtime

$\Theta(n)$

Exercise 6.4

A -

(a)

$R_k = \{A[i]+i \mid 1 \leq i \leq M[k-1]\}$

$[2, 4, 2, 2, 1, 1, 1, 1, 5, 2]$

k	$M[k]$	R_k
0	1	-
1	3	$[1+2]$
2	6	$[1+2, 2+4, 3+2]$
3	7	$[1+2, 2+4, 3+2, 4+2, 5+1, 6+1]$
4	8	
5	9	
6	14	

$$\sum_{k=1}^{K} |R_k| = 34.$$

(b)

$R'_k = \{A[i]+i \mid M'[k-2] < i \leq M'[k-1]\}$

$$\sum_{k=1}^{K} |R'_k| = 8$$

$[2, 4, 2, 2, 1, 1, 1, 1, 5, 2]$

k	$M'[k]$	R'_k
0	1	-
1	3	
2	6	$[2+4, 3+2]$
3	7	$[4+2, 5+1, 6+1]$
4	8	$[7+1]$
5	9	$[8+1]$

6 14 [9+5]

(c)

$M[0] = M'[0] \qquad M[1] = M'[1]$

I.H. : $M[k-2] = M'[k-2] \wedge M[k-1] = M'[k-1] \qquad \Rightarrow \underline{\max R_k = \max R'_k}$

$M[k] \overset{\text{def}}{=} \max R_k$

$M'[k] \overset{\text{def}}{=} \max R'_k$

$M[0] = M'[0]$

$R_1 = \{1 + A[1]\} \Rightarrow M[1] = M'[1]$

I.S. z.z.: $\max R_k = \max R'_k$ für $k \geq 2$

$\underline{\max R_k \geq \max R'_k}$ weil $R'_k \subseteq R_k$

zu zeigen: $\max R_k \not> \max R'_k$

$\Rightarrow \max R_k \leq \max R'_k$

(I.H.) $M'[k-2] = M[k-2] \wedge M'[k-1] = M[k-1]$

$R_k \setminus R'_k = \{i + A[i] \mid 1 \leq i \leq M[k-2]\} = R_{k-1}$

$\Rightarrow \max (R_k \setminus R'_k) = M[k-1] < \max R_k$

$\Rightarrow \max R'_k >= \max R_k$

Exercise 6.2

$S = ((\ldots)[\ldots])$ ✓

$S =$ [abc) X

([ )]

function isWellFormed(s)

```
S ← emptyStack()

for i ∈ {0, ..., |s|-1} do

    if s[i] = "(" then
        S.push("(")
    else if s[i] = "[" then
        S.push("[")

    else if s[i] = ")" then
        if S.pop() ≠ "("
            return false
    else if s(i) = "]" then
        if S.pop() ≠ "["
            return false

return S.isEmpty()
```

Vorlesung Recap

Subset Sum

$A = [1, \dots, n]$ $b \in \mathbb{N}$

$I \subseteq A$, sodass $\sum_{i \in I} = b$

$T(i,s) :=$ " Ist s eine Teilsumme von $A[1, \dots, i]$? "

$i-1 \to i$

Fall 1: s ist bereits eine Teilsumme von $A[1, \dots, i-1]$

z.B. $A = [3, 4, 2]$, $b = 7$

In $A = [3, 4]$ haben bereits eine 7-Summe!
ist in $A = [3, 4, 2]$ sowie schon eine 7-Summe.

$\Rightarrow T(i,s) = T(i-1, s)$

Fall 2: $s - A[i]$ ist bereits Teilsumme von $A[1, \dots, i-1]$

Sei $A[i] = k$

Es könnte in $A[1, \dots, i-1]$ bereits $s-k$ Summe geben.

Addieren wir k, erhalten wir s-Summe!

z.B. $A = [1, 4, 3, 5]$, $b = 10$

in $A = [1, 4, 3]$ bereits 5-Summe

also gibt es in $A = [1, 4, 3, 5]$ $5+5 = 10$ - Summe!

$\Rightarrow T(i,s) = T(i-1, s-A[i])$

$$\Rightarrow T(i,s) = T(i-1, s) \vee T(i-1, s-A[i])$$

B.C. $T(i, 0) = true$

$T(0, s) = false$

Rucksack-Problem

Rucksack max. W Gewicht tragen kann , n , w_i , v_i

w_i: Gewicht, v_i: Wert

$$I \subseteq \{1, \dots, n\} \quad \sum_{i \in I} w_i \leq W$$

$$\sum_{i \in I} v_i \text{ maximal}$$

$MW(i,w) :=$ max. Wert, den man aus $1, \dots, i$ mit Gewichtslimit w erreichen kann.

Fall 1: Wir haben bereits optimale Lösung, verwende i nicht

$$A = \begin{array}{c} \\ \text{value} \\ \text{weight} \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 100 & 20 & 15 \\ 2 & 3 & 1 \end{bmatrix} \end{array}$$

$W = 5$

$MW(2,w) = 120$

$MW(3,w) = 120$

$$\Rightarrow MW(i,w) = MW(i-1,w)$$

Fall 2: Nehme i-tes element in Rucksack, der Platz hat für w_i

$$\begin{array}{c} \\ \text{value} \\ \text{weight} \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 50 & 5 & 10 & 20 \\ 3 & 1 & 1 & 2 \end{bmatrix} \end{array}$$

$W = 5$

$MW(3,5) = 50 + 5 + 10 = 65$

$MW(3,3) = 50$

$$MW(4,5) = MW(3,3) + V_4 = 50 + 20 = 70$$

$$\Rightarrow MW(i,w) = V_i + MW(i-1, w-w_i)$$

$$\Rightarrow MW(i,w) = \max \{ MW(i-1,w), v_i + MW(i-1, w-w_i) \}$$

Längste aufsteigende Teilfolge

Array $A(1, \dots, n)$

$DP[i] \hat{=}$ das letzte Element der Teilfolge i

$DP(0) = -\infty$

$DP[i] = +\infty \quad i > 0$

```
for (k=1, k<n, k++)
    int l = Search(DP, A[k])      ▷ gibt grösstes Element zurück, A[l] < A[k]
    // DP[l] < A(k) ≤ DP[l+1]
    DP[l+1] = A[k]
```

Exercise 7.1 *k-sums* **(1 point).**

We say that an integer $n \in \mathbb{N}$ is a *k-sum* if it can be written as a sum $n = a_1^k + \cdots + a_p^k$ where $a_1, \ldots, a_p$ are distinct natural numbers, for some arbitrary $p \in \mathbb{N}$.

For example, 36 is a 3-*sum*, since it can be written as $36 = 1^3 + 2^3 + 3^3$.

Describe a DP algorithm that, given two integers n and k, returns True if and only if n is a k-sum. Your algorithm should have asymptotic runtime complexity at most $O(n^{1+\frac{2}{k}})$.

Hint: *The intended solution has complexity* $O(n^{1+\frac{1}{k}})$. $= O(n \cdot n^{\frac{1}{k}}) = O(n \cdot \sqrt[k]{n})$

In your solution, address the following aspects:

1. *Dimensions of the DP table*: What are the dimensions of the DP table?
2. *Definition of the DP table*: What is the meaning of each entry?
3. *Computation of an entry*: How can an entry be computed from the values of other entries? Specify the base cases, i.e., the entries that do not depend on others.
4. *Calculation order*: In which order can entries be computed so that values needed for each entry have been determined in previous steps?
5. *Extracting the solution*: How can the solution be extracted once the table has been filled?
6. *Running time*: What is the running time of your solution?

1. Dimensions:

$(1+n) \times (\lfloor n^{\frac{1}{k}} \rfloor + 1)$ $\quad m = \lfloor n^{\frac{1}{k}} \rfloor$ ist die grösste Ganzzahl, sodass $m^k \leq n$.

2. Definition:

$DP[i][j] \;\hat{=}\;$ ist true $\Leftrightarrow$ i kann geschrieben werden als:

$$i = a_1^k + a_2^k + \ldots + a_p^k$$

$$\{a_1, \ldots, a_p\} \subseteq \{1, \ldots, j\}$$

3. Computation:

B.C.: $\quad DP[i][0] = \text{False} \qquad 1 \leq i \leq n$

$DP[0][j] = \text{True} \qquad 0 \leq j \leq m = \lfloor n^{\frac{1}{k}} \rfloor$

$DP[i][j] = \underbrace{DP[i-j^k][j-1]}_{\text{wir nehmen } j^k \text{ dazu}} \vee DP[i][j-1] \qquad j^k \leq i \leq n$

$DP[i][j] = DP[i][j-1] \qquad$ sonst

4. Calculation order

1. Base cases
2. Austeigende j, aufsteigende i

5. Extracting the solution

return $DP[n][\lfloor n^{1/k} \rfloor]$

6. Runtime

Jeder Eintrag benötigt $O(1)$, wir haben $n \cdot \lfloor n^{\frac{1}{k}} \rfloor$ Einträge $\Rightarrow O(n^{1+\frac{1}{k}})$

Peer Grading Ex. 6.1