Prof. Dr. Christian Sohler
Nicole Funk, Dr. Matthijs Ebbens, Sebastian Zaun

Sommersemester 2023

6. Übungsblatt

zur Vorlesung

Grundzüge der Informatik I

Abgabe über Ilias bis zum 17.4. 14:00 Uhr.
Besprechung in Kalenderwoche 21.

Aufgabe 1 *(4 Punkte)*

Wir betrachten den Algorithmus $SubsetSum(A, U, n)$ der Vorlesung. Gegeben sei die folgende Menge

$$A = \{19, 5, 7, 4, 12, 9, 2, 6, 4\} \tag{1}$$

und der Wert

$$u = 15 \tag{2}$$

Wenden Sie den Algorithmus der Vorlesung an, um zu entscheiden, ob es eine Teilmenge $L \subseteq A$ gibt, sodass $\sum_{x \in L} x = u$ gilt. Geben Sie dabei das vollständige Array Ind, sowie das Ergebnis des Algorithmus an.

Aufgabe 2 *(4 Punkte)*

Gegeben sei ein Array $A[1..n]$ mit $1 \leq A[i] \leq 3$ für alle $1 \leq i \leq n$. Eine Spielfigur startet auf der ersten Stelle des Arrays und muss die n-te Stelle erreichen. Befindet sich die Figur auf der Stelle i für $1 \leq i \leq n$, so darf sie mit einem Sprung bis zu $A[i]$ Stellen nach vorne ziehen. Im unten gezeigten Beispiel darf die Figur also von der zweiten Stelle aus bis zu $A[2] = 3$ Stellen weiterspringen, also jede der Stellen $3, 4$ und 5 mit einem Sprung erreichen. Gesucht ist die minimale Anzahl von Sprüngen, um beginnend auf der ersten Stelle des Arrays die n-te Stelle zu erreichen.

Für das folgende Beispiel mit $n = 8$ beträgt die minimale Anzahl an Sprüngen 3 und ergibt sich durch die Sprungfolge $1 \curvearrowright 2 \curvearrowright 5 \curvearrowright 8$.

$$A = \begin{pmatrix} \mathbf{2} & \mathbf{3} & 1 & 2 & \mathbf{3} & 1 & 2 & \mathbf{1} \end{pmatrix}$$

Sei $M[i]$ die minimal benötigte Anzahl von Sprüngen, um ausgehend von der i-ten Stelle die n-te Stelle zu erreichen. Geben Sie eine rekursive Formulierung für $M[i]$ an. Erklären Sie die Funktionsweise dieser. Gehen Sie dabei auf jede Fallunterschiedung ein.

1a)

U	I	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
19	**1**	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	**2**	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
7	**3**	1	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0
4	**4**	1	0	0	0	1	1	0	1	0	1	0	1	1	0	0	0
12	**5**	1	0	0	0	1	1	0	1	0	1	0	1	1	0	0	0
9	**6**	1	0	0	0	1	1	0	1	0	1	0	1	1	1	1	0
2	**7**	1	0	1	0	1	1	1	1	0	1	0	1	1	1	1	1
6	**8**	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	**9**	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1

1: true
0: false

Ausgabe: <u>true</u>

2.)

$$M(i) = \begin{cases} \infty & \text{falls } i > n \\ 0 & \text{falls } i = n \\ 1 + M(i+1) & \text{falls } A[i]=1 \\ 1 + \min\{M(i+1), M(i+2)\} & \text{falls } A[i]=2 \\ 1 + \min\{M(i+1), M(i+2), M(i+3)\} & \text{falls } A[i]=3 \end{cases}$$

$1 \le i \le n+2$

alternative Lösung

$$M(i) = \begin{cases} 0 & \text{falls } i=n \\ 1 + M(i-1) & \text{falls } A[i]=1 \text{ oder } i=n-1 \\ 1 + \min\{M(i+1), M(i+2)\} & \text{falls } A[i]=2 \text{ oder } (i=n-2 \text{ und } A[i] \ge 2) \\ 1 + \min\{M(i+1), M(i+2), M(i+3)\} & \text{falls } A[i]=3 \end{cases}$$

$1 \le i \le n$

alternative Lösung anders rum

$S(i)$ beschreibt die minimale Anzahl an Sprüngen, um von $A[1]$ nach $A[i]$ zu gelangen.

$$S(i) = \begin{cases} 0 & \text{falls } i = 1 \\ 1 + S(1) & \text{falls } i = 2 \\ 1 + S(2) & \text{falls } i = 3 \text{ und } A[1] < 2 \\ 1 + \min\{S(i-1), S(i-2)\} & \text{falls } (i = 3 \text{ und } A[1] \geq 2) \\ & \text{oder } (i \geq 4 \text{ und } A[i-2] \geq 2 \text{ und } A[i-3] < 3) \\ 1 + \min\{S(i-1), S(i-2), S(i-3)\} & \text{falls } (i \geq 4 \wedge A[i-2] \geq 2 \text{ und } A[i-3] = 3) \\ 1 + \min\{S(i-1), S(i-3)\} & \text{falls } (i \geq 4 \wedge A[i-2] < 2 \text{ und } A[i-3] = 3) \\ 1 + S(i-1) & \text{falls } (i \geq 4 \wedge A[i-2] < 2 \text{ und } A[i-3] < 3) \end{cases}$$

mit $1 \leq i \leq n$

kürzere Version

$$S(i) = \begin{cases} \infty & \text{falls } i < 1 \\ 0 & \text{falls } i = 1 \\ 1 + S(i-1) & \text{falls } i \geq 4 \text{ und } A[i-2] < 2 \text{ und } A[i-3] < 3 \\ 1 + \min\{S(i-1), S(i-2)\} & \text{falls } i \geq 4 \text{ und } A[i-3] < 3 \\ 1 + \min\{S(i-1), S(i-3)\} & \text{falls } i \geq 4 \text{ und } A[i-2] < 2 \\ 1 + \min\{S(i-1), S(i-2), S(i-3)\} & \text{sonst} \end{cases}$$

mit $-2 \leq i \leq n$

Aufgabe 3 *(4 + 2 + 4 + 2 Punkte)*

Gegeben sei eine Menge A mit n Zahlen. Die Anzahl der Partitionen von A kann mit der sogenannten Stirling Zahl zweiter Art berechnet werden.

Diese Zahl $S(n,k)$ ist rekursiv definiert als:

$$S(n,k) = \begin{cases} 0 & \text{falls } k = 0, n > 0 \\ 1 & \text{falls } k = n \\ S_{n-1,k-1} + k \cdot S_{n-1,k} & \text{sonst.} \end{cases} \tag{3}$$

und beschreibt wie viele Partitionen einer n elementigen Menge in k disjunkte Teilmengen es gibt.

a) Geben Sie einen rekursiven Algorithmus in Pseudocode an, welcher bei Eingabe einer Zahl n unter Verwendung von $S(n,k)$ die Anzahl aller Partitionen einer Menge mit n Elementen berechnet.

b) Analysieren Sie die asymptotische Worst-Case-Laufzeit Ihres Algorithmus aus Teilaufgabe a).

c) Geben Sie einen Algorithmus in Pseudocode an, der auf dem Prinzip der dynamischen Programmierung beruht, und bei Eingabe einer Zahl n unter Verwendung von $S(n,k)$ die Anzahl aller Partitionen einer Menge mit n Elementen berechnet.

d) Analysieren Sie die asymptotische Worst-Case-Laufzeit Ihres Algorithmus aus Teilaufgabe c).

3.)

a)

Teilmengen Rekursiv (n)

$a = 0$

for $k = 1$ to n do

$a = a + \text{Stirling}(n,k)$

return a

b)

1

$n+1$

$\sum_{k=1}^{n} T(n,k)$

1

$\sum_{i=1}^{n} T(n,k) + n + 3 \in O(n \cdot 2^n)$

Stirling (n,k) — $\underline{T(n,k)}$

if $k=0$ and $n>0$ — $1 \quad 1 \quad 1$

return 0 — 1

else if $k=n$ — $1 \quad 1$

return 1 — 1

else

return $\text{Stirling}(n-1,k-1) + k \cdot S(n-1,k)$ — $T(n-1,k-1) + T(n-1,k)$

$$\hookrightarrow T(n,k) = \begin{cases} 2 \\ 3 \\ T(n-1,k-1) + T(n-1,k) + 2 \end{cases}$$

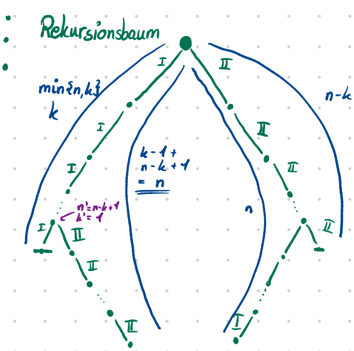
$$T(n,k) = T(n-1,k-1) + T(n-1,k) + 2$$

$$= (T(n-2,k-2) + T(n-2,k-1) + 2) + (T(n-2,k-1) + T(n-2,k) + 2) + 2$$

$\vdots$

I: $n-1, k-1$

II $n-1, k$



$\leftarrow$ nicht vollständiger Baum

maximale Tiefe : $h = n$

Jeder Knoten hat konstante Laufzeit

Es gibt weniger innere Knoten als Blätter, also dominieren die Blätter

Blätter in vollständigem Baum : 2^h

$\hookrightarrow T(n) \in O(2^n)$

c)

```
TeilmengenDynamisch(n)
    k = n
    S = new Array[0...n][0...k]
    for i=1 to n do
        S[i][0] = 0
    for j=0 to k do
        S[j][j] = 1
    for i=2 to n do
        for j=2 to i-1 do
            S[i][j] = S[i-1][j-1] + j · S[i-1][j]
    a = 0
    for j=1 to n do
        a = a + S[n][j]
    return a
```

d)

1

$n \cdot n = n^2$

$n+1$

n

$n+2$

$n+1$

n

$\sum_{i=2}^{n} i-1 = \frac{1}{2}(n^2-n)$

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

1

$n+1$

n

1

$n^2+\frac{1}{2}n^2+\frac{1}{2}n^2+7n+\frac{1}{2}n-\frac{1}{2}n$
$+8-\frac{1}{2}\cdot 2$

$= 2n^2+7n+7$

$\in O(n^2)$