Algorithmen und Datenstrukturen Prof. Dr. U. Schöning

Wintersemester 2020/21 **Übungsblatt 5** T. Büchler

Abgabe bis 11.12. **13:00 Uhr**, Besprechung: 14.12-18.12. Institut für Theoretische Informatik

---

Aufgabe 5.1 *3 Punkte.*

In der Vorlesung haben Sie einen Algorithmus kennen gelernt, der das 0/1-Rucksackproblem mittels dynamischen Programmierens löst. Finden Sie mit diesem Algorithmus die optimale Lösung für $G = 8$ und die Objekte

$$\begin{array}{lllll} g_1 = 1 & g_2 = 3 & g_3 = 3 & g_4 = 4 & g_5 = 9 \\ v_1 = 1 & v_2 = 2 & v_3 = 1 & v_4 = 3 & v_5 = 7\ . \end{array}$$

Markieren Sie die optimale Lösung in der Tabelle und geben Sie die zugehörige Auswahl von Objekten an.

Aufgabe 5.2 *1+2+1+2=6 Punkte.*

Man bezeichnet s' als Teilfolge eines Strings s, wenn s' daraus entsteht, dass null oder mehr Zeichen aus s gelöscht werden. Man nennt s' eine gemeinsame Teilfolge zweiter Strings s_1 und s_2, wenn s' sowohl eine Teilfolge von s_1 als auch von s_2 ist. Falls es keine längere Teilfolge gibt, dann heißt s' längste gemeinsame Teilfolge.

Beispiel: Gegeben seien s_1=`AXBXC`, s_2=`ABCDE`, dann ist `ABC` die längste gemeinsame Teilfolge.

a) Bestimmen Sie die Länge der längsten gemeinsamen Teilfolge von `HOORAY` und `HURRA`.

b) Schreiben Sie eine Funktion $lgT(s_1,\ s_2)$ in Pseudocode, die die Länge der längsten gemeinsamen Teilfolge berechnet. Verwenden Sie hierzu dynamische Programmierung.

c) Welche Tabellengröße verwendet Ihr Programm? Welche Laufzeit im Sinne der O-Notation hat es?

d) Sei n die Länge von s_1, m die Länge von s_2 und x die Editierdistanz. Begründen Sie, dass $x \leq m + n - 2lgT(s_1,\ s_2)$ gilt. In welchen Fällen ist es eine echte Ungleichung und wann gilt Gleichheit?

Aufgabe 5.3 *1 + 1 + 1 = 3 Punkte.*

Gegeben ist der folgende Algorithmus mit der Eingabe $n \in \mathbb{N}$:

```
Function F(n)
    if n ≤ 1 then
        return 1;
    return 3 · F(n − 1) + F(⌊n/2⌋) ;
```

a) Warum ist der Algorithmus für die Berechnung von F nicht effizient?

b) Lässt sich das Prinzip des Dynamischen Programmierens anwenden, um F zu berechnen? Begründen Sie Ihre Antwort!

c) Welche Änderung des Algorithmus ist notwendig, damit nur $\mathcal{O}(\,n\,)$ viele Additionen ausgeführt werden? Gegen Sie den abgeänderten Algorithmus an.

Aufgabe 5.4 *1+1=2 Punkte.*

In der Vorlesung wurde gezeigt, wie man das Travelling-Salesman-Problem mit dynamischem Programmieren in Laufzeit $\mathcal{O}(\, n^2 \cdot 2^n \,)$ lösen kann.

a) Wie kann man eine Menge $S \subseteq \{2, .., n\}$ als Zahl zwischen 0 und $2^{n-1} - 1$ repräsentieren? Die Abbildung $C : \mathcal{P}(\{2, \ldots, n\}) \to \{0, \ldots, 2^{n-1} - 1\}$ soll bijektiv sein.

b) Beschreiben Sie, wie sich folgende Operationen auf $C(S)$ durchführen lassen:

- Testen, ob $i \in S$.
- Zuweisung $S \leftarrow S \setminus \{i\}$.

Aufgabe 5.5 *2+1+3=6 Punkte.*

Gegeben sei eine aufsteigend sortierte Liste mit n Zahlen $a_1, ..., a_n$, d. h. $a_i < a_{i+1}$ für $1 \leq i < n$. Sei $\sigma(i) = \sum_{j=1}^{n} |a_i - a_j|$ für $1 \leq i \leq n$.

(a) Wie lässt sich $\sigma(i)$ aus $\sigma(i-1)$ in $\mathcal{O}(\,1\,)$ berechnen? Beweisen Sie diese Formel.

(b) Zeigen Sie, für welches i der Wert $\sigma(i)$ für die gegebenen a_i minimal wird.

(c) Lösen Sie die Programmier-Aufgabe "Treffpunkt".
Geben Sie Ihren Domjudge-Teamnamen bei Ihrer Abgabe an, damit Ihnen Ihre Lösung zugeordnet werden kann.