# 14. Aufgabenblatt

(Besprechung in den Tutorien 06.02.2023–10.02.2023)

## Aufgabe 1. Subset Sum

Betrachten Sie das folgende Problem und den dazugehörigen Algorithmus.

SUBSET SUM
**Eingabe**: Eine Multi-Menge $U := \{u_1, u_2, \ldots, u_m\}$ von natürlichen Zahlen und eine Zahl $B \in \mathbb{N}$.
**Frage:** Existiert eine Teilmenge $U' \subseteq U$, die sich zu $B$ aufsummiert?

**Algorithm 1:** Algorithmus für SUBSET SUM

**Input:** Eine Multi-Menge $U := \{u_1, u_2, \ldots, u_m\}$ von natürlichen Zahlen und eine natürliche Zahl $B$.
**Output:** `true` genau dann, wenn es eine Teilmenge $U' \subseteq U$ mit $\sum_{u \in U'} u = B$ gibt.

```
1  ▷ Sei T[i, j] eine Boolesche Tabelle mit 0 ≤ i ≤ m und 0 ≤ j ≤ B, die angibt, ob es
     in {u_1, u_2, ..., u_i} eine Teilmenge gibt, die sich zu j aufsummiert. Initial sind alle
     Einträge false.
2  foreach 0 ≤ i ≤ m do
3  |  T[i, 0] ← true
4  end
5  foreach 1 ≤ i ≤ m (aufsteigend) do
6  |  foreach 1 ≤ j ≤ B (aufsteigend) do
7  |  |  if j ≥ u_i then
8  |  |  |  T[i, j] ← T[i − 1, j] ∨ T[i − 1, j − u_i]
9  |  |  else
10 |  |  |  T[i, j] ← T[i − 1, j]
11 |  |  end
12 |  end
13 end
14 return T[m, B]
```

1. Analysieren Sie die Laufzeit des Algorithmus.

   ——————Lösungsskizze——————

   Zunächst müssen wir alle $(m+1) \cdot (B+1)$ Einträge in der Tabelle auf false setzen, das kostet $O(mB)$ Zeit.

   Die for-Schleife in Zeile 2 wird $m$-mal durchlaufen, wobei jeder Durchlauf konstante Zeit benötigt. Somit benötigt die for-Schleife in Zeile 2 $O(m)$ Zeit.

   Zeile 7–11 kann in konstanter Zeit durchgeführt werden. Somit kann die innere for-Schleife in Zeile 6 in Zeit $O(B)$ durchgeführt werden. Daher benötigt die äußere for-Schleife in Zeile 5 insgesamt $O(mB)$ Zeit.

   Zeile 14 benötigt konstante Zeit.

   Somit läuft der Algorithmus in $O(mB + m + mB + 1) = O(mB)$ Zeit.

   ———————————————

2. Ist dadurch gezeigt, dass SUBSET SUM in P liegt?

   ————————Lösungsskizze————————

   Nein. Da Zahlen mit logarithmisch vielen Bits dargestellt werden können, ist die Eingabegröße in $O(\sum_{i=1}^{m} \log u_i + \log B)$. Somit ist die Laufzeit $O(mB) = O(m2^{\log B})$ exponentiell in der Eingabegröße.

   ————————————————————

*Hinweis*: Eine Multi-Menge ist eine Menge, in der Elemente mehrfach vorkommen können, d.h. $\{1, 1\} \neq \{1\}$.

**Aufgabe 2. NP, PSPACE, und deterministische Exponentialzeit**

Diskutieren Sie, warum NP $\subseteq$ PSPACE $\subseteq \bigcup_{k\geq 1}$ DTIME$(2^{n^k})$ gilt.

————————Lösungsskizze————————

Sei $L \in$ NP. Dann existiert eine NTM $N$, sowie ein Polynom $p$ mit $\text{time}_N(n) \leq p(n)$ und $T(N) = L$. Wir erstellen nun eine DTM $D$, welche auf Eingabe $w$ alle möglichen (nichtdeterministischen) Übergänge von $N$ nacheinander ausprobiert (Tiefensuche im Berechnungsbaum bis zur Tiefe $p(|w|)$). Wenn einer der Versuche in $p(|w|)$ Schritten akzeptiert, so akzeptiert auch $D$. Damit ist $T(D) = L$. Ein jeder dieser Versuche von $D$ kann aber maximal $p(|w|)$ viele Bandzellen beschreiben (höchstens eine pro Schritt). Also ist $L \in$ PSPACE.

Sei nun $L \in$ PSPACE. Dann existieren eine DTM $M = (Z, \Sigma, \Gamma, \delta, q_0, F, \square)$ und ein Polynom $p$, sodass $M$ bei Eingabe $w$ höchstens $p(|w|)$ viele Bandzellen manipuliert und $T(M) = L$ gilt. Wir konstruieren nun eine Maschine $M'$ wie folgt:

- Input: $w$ mit $n := |w|$
- Simuliere $M$ auf $w$ für $p(n) \cdot |Z| \cdot |\Gamma|^{p(n)} + 1$ viele Schritte.
- Falls Simulation akzeptiert hat, so akzeptiere, andernfalls verwerfe.

Dabei ist $p(n) \cdot |Z| \cdot |\Gamma|^{p(n)}$ die Anzahl der möglichen Konfigurationen von $M$ auf $w$ und setzt sich wie folgt zusammen:

- $p(n)$ mögliche Kopfpositionen
- $|Z|$ mögliche Zustände
- Auf jeder der $p(n)$ Zellen kann jedes Zeichen aus $\Gamma$ stehen, also $|\Gamma|^{p(n)}$ Möglichkeiten.

Wenn $M$ nach dieser Zeit nicht hielt, muss eine Konfiguration doppelt vorgekommen sein. Damit geriet $M$ jedoch in eine Endlosschleife, also akzeptieren wir nicht. Somit gilt $T(M') = L$.

Das Simulieren von $M$ und das Zählen der Schritte benötigt $O(p(n))$ Zeit (man beachte, dass $|\Gamma|$ und $|Z|$ konstant groß sind). Damit gilt

$$\text{time}_{M'}(n) \in O((p(n))^2 \cdot |\Gamma|^{p(n)}) \subseteq O(2^{(1+\log|\Gamma|)p(n)}) \subseteq O(2^{(p(n))^2}).$$

————————————————————

**Aufgabe 3. Generalized Geography**

In der Vorlesung wurde das generalisierte Geographiespiel eingeführt:

Eingabe: Ein gerichteter Graph $G$ mit Startknoten $v$.
Spielregeln: Die Spielerinnen wählen abwechselnd einen "nächsten Knoten" unter den noch nicht gewählten Nachfolgern des aktuellen Knotens. Wer keinen Nachfolger mehr auswählen kann, verliert das Spiel.

Wir betrachten das dazugehörige Entscheidungsproblem.

**Generalized Geography (GG)**
**Eingabe:** Ein gerichteter Graph $G = (V, E)$ und $v \in V$.
**Frage:** Hat Spielerin 1 eine Gewinnstrategie, die mit einem Nachbarn von $v$ startet?

1. Zeigen Sie, dass entweder Spielerin 1 oder Spielerin 2 eine Gewinnstrategie hat.

2. Sei $\phi = \exists x_1 \forall x_2 \exists x_3 \dots \exists x_n F$ eine quantifizierte aussagenlogische Formel, wobei $F$ in konjunktiver Normalform mit freien Variablen $x_1, \dots, x_n$ ist.

   Geben Sie eine polynomzeitberechenbare Instanz $(G, v)$ an, sodass $\phi$ genau dann wahr ist, wenn $(G, v) \in \mathrm{GG}$.

——————Lösungsskizze——————

1. Da $G$ endlich ist, kann es keine unendliche Folge von Zügen geben. Außerdem gewinnt Spielerin 1 per Definition genau dann wenn Spielerin 2 verliert.

   Es gilt also: Spielerin 1 hat eine Gewinnstrategie $\iff$ Spielerin 1 kann einen Nachbarknoten $u$ von $v$ auswählen kann, sodass Spielerin 2 von dort aus immer verliert $\iff$ Spielerin 2 hat keine Gewinnstrategie.

2. Seien $C_1, \dots, C_m$ die Klauseln von $F$. Der Graph $G$ hat folgende Knoten $V := \{a_i, b_i, x_i, \overline{x}_i \mid i = 1, \dots, n\} \cup \{c, c_1, \dots, c_m\}$ und folgende Kanten:

   - $(a_i, x_i), (a_i, \overline{x}_i), (x_i, b_i), (\overline{x}_i, b_i)$ für alle $i = 1, \dots, n$
   - $(b_1, a_2), (b_2, a_3), \dots, (b_{n-1}, a_n), (b_n, c)$
   - $(c, c_j)$ für alle $j = 1, \dots, m$
   - $(c_j, \overline{\ell})$ für jede Klausel $C_j$ und jedes Literal $\ell \in C_j$.

   Der Startknoten ist $v := a_1$.

   Falls $\phi$ wahr ist, hat Spielerin 1 eine Gewinnstrategie für $(G, v)$, denn sie kann immer einen Zugfolge (entsprechend der Variablenbelegung) finden, die einen Pfad zu einer erfüllten Klausel ergibt (Spielerin 2 wählt immer irgendeinen Knoten $c_j$). Da diese Klausel erfüllt ist, gibt es darin ein Literal $\ell$, das wahr ist. Also wurde der Knoten $\overline{\ell}$ zuvor nicht gewählt und Spielerin 1 kann diesen wählen, wodurch sie gewinnt.

   Falls $\phi$ falsch ist, kann Spielerin 2 immer einen Pfad zu einer nicht erfüllten Klausel finden, womit dann Spielerin 1 keinen möglichen Zug hat und verliert. Somit hat Spielerin 1 keine Gewinnstrategie.