Theoretische Grundlagen der Informatik II
Blatt 2

Markus Vieth, David Klopp, Christian Stricker

5. November 2015

Aufgabe 1

a)

Behauptung: EXAKT3SAT (E3S) ist in NP
Beweis:
Durch folgenden nichtdeterministischen Algorithmus:
Man rät nichtdeterministisch eine Menge $A \subset \{i \in \mathbb{N} | i < d\}$, wobei d = Anzahl der paarweise verschiedenen Literale des booleschen Ausdrucks ist. Anschließend wird jede Klausel, welche durch ein logisches AND getrennt sind, einzeln betrachtet, davon existieren $\frac{n}{3}$, wobei n = Anzahl der eingegebenen Literale. Nun wird geprüft, ob mindestens ein Literal x_i in jeder Klausel existiert, wobei i fortlaufend ist, für das gilt:

$$\text{wenn } x_i \text{ nicht negiert} : i \in A, \text{ wenn } x_i \text{ negiert} : i \notin A$$

Es gilt: $|A| \leq d \leq n \Rightarrow T(n) \in O(n^2)$
Wenn die obere Aussage wahr ist, wird JA zurückgegeben, sonst NEIN.
Da die Laufzeit der Einzelschritte polynomial ist, ist auch die Summe dieser und somit die gesamt Laufzeit polynomial.

Idee Die Menge A enthält die Indices der booleschen Variablen, welche mit 1 bzw. true belegt werden sollen, somit enthält die Menge $\overline{A} = i \in \mathbb{N} | i < d \setminus A$ die Indecs aller booleschen Variablen, welche mit 0 bzw. false belegt werden sollen. Nichtdeterministisch wird nun eine Belegung gefunden, welche den Ausdruck erfüllt, sollte eine solche existieren.

Behauptung: E3S ist NP-Hart
Beweis:
Man passt die Literale aller Klauseln an E3S an. Es sei d der größte verwendete fortlaufende Index der Literale des booleschen Ausdrucks, welcher betrachtet wird.

Klauseln mit einem Literal

$$x_i = (x_i + x_{d+1} + x_{d+2})(x_i + x_{d+1} + \overline{x_{d+2}})(x_i + \overline{x_{d+1}} + x_{d+2})(x_i + \overline{x_{d+1}} + \overline{x_{d+2}})$$

Klauseln mit zwei Literalen

$$x_i + x_j = (x_i + x_j + x_{d+1})(x_i + x_j + \overline{x_{d+1}})$$

Klauseln mit drei Literalen

$$x_i + x_j + x_k = x_i + x_j + x_k$$

Klauseln mit vier Literalen

$$x_i + x_j + x_k + x_l = (x_i + x_j + x_{d+1})(\overline{x_{d+1}} + x_k + x_l)$$

Klauseln mit fünf oder mehr Literalen

$$\sum_{\alpha=i}^{\beta} x_\alpha = (x_i + x_j + x_{d+1}) \left(\overline{x_{d+1}} + \sum_{\alpha=k}^{\beta} \right)$$

auf die rechte Summe/Batch-OR-Operation ist entweder „Klausel mit fünf oder mehr Literalen“ bzw. „Klausel mit vier Literalen“ wieder anzuwenden. Der Beweis der ersten 3 Umformungen ist trivial (siehe

Technische Informatik)
Anschließend ist der CSAT Algorithmus anzuwenden, dass Ergebnis des CSAT-Problems ist das Ergebnis des E3S Problems. Da die Laufzeit der Einzelschritte polynomial ist, ist auch die Summe dieser und somit die gesamt Laufzeit polynomial.
Somit ist das CSAT-Problem in polynomialer Laufzeit auf das E3S-Problem reduzierbar. ⇒ E3S ist NP-Hart ⇒ E3S ist NP-Voll.

q.e.d.

Aufgabe 2

a)

Behauptung: BP ist NP-Hart
Beweis:
Man berechnet:

$$b = \left\lfloor \frac{\sum_{i=1}^{n} a_i}{2} \right\rfloor$$

Nun führt man den BP-Algorithmus mit k = 2 Behältern der Größe b mit n Werten $a_i = a_i \forall i \in \{1, \ldots, n\}$ aus.
Das Ergebnis des BP-Algorithmus ist das Ergebnis des Partition-Problems.
Da die Laufzeit der Einzelschritte polynomial ist, ist auch die Summe dieser und somit die gesamt Laufzeit polynomial.
Somit ist das Partition-Problem in polynomialer Laufzeit auf das BP-Problem reduzierbar. ⇒ BP ist NP-Hart ⇒ BP ist NP-Voll.

Aufgabe 3

a)

Behauptung: Partition ist NP-Hart
Vorüberlegung:
In der Reduzierung wird der Wert $a_{n+1} = x \geq 0$ hinzugefügt, der sich wie folgt ergibt.
1. Fall $b \geq \sum_{i \notin J} a_i$

$$\sum_{i \in J} a_i = b = \sum_{i \notin J} a_i + x$$

$$\sum_{i \in J} a_i - \sum_{i \notin J} a_i = x$$

$$\sum_{i \in J} a_i - \sum_{i \notin J} a_i + \sum_{i \in J} a_i - \sum_{i \in J} a_i = x$$

$$2 \cdot \sum_{i \in J} a_i - \sum_{i \notin J} a_i - \sum_{i \in J} a_i = x$$

$$2b - \sum_{i=1}^{n} a_i = x$$

2. Fall $b < \sum_{i \notin J} a_i$

$$\sum_{i \in J} a_i = b = \sum_{i \notin J} a_i - x$$

$$\sum_{i \in J} a_i + x = b + x = \sum_{i \notin J} a_i$$

$$\sum_{i \notin J} a_i - \sum_{i \in J} a_i = x$$

$$\sum_{i \notin J} a_i - a_i \sum_{i \in J} a_i - \sum_{i \in J} a_i + \sum_{i \in J} a_i = x$$

$$\sum_{i \notin J} a_i + \sum_{i \in J} a_i - 2 \cdot \sum_{i \in J} a_i = x$$

$$\sum_{i=1}^{n} a_i - 2b = -(2b - \sum_{i=1}^{n} a_i) = x$$

$$\Rightarrow x = a_{n+1} = \left| 2b - \sum_{i=1}^{n} a_i \right|$$

Beweis:
Durch folgenden Algorithmus:
Der Wert $a_{n+1} = |2b - \sum_{i=1}^{n} a_i|$ wird berechnet.
Der Partition-Algorithmus wird für die n+1 Werte $a_1, \ldots, a_n, a_{n+1}$ angewendet.
Das Ergebnis des Partition-Algorithmus ist das Ergebnis des Subset Sum Problems.
Da die Laufzeit der Einzelschritte polynomial ist, ist auch die Summe dieser und somit die gesamt Laufzeit polynomial.
Somit ist das Subset Sum-Problem in polynomialer Laufzeit auf das Partition-Problem reduzierbar. $\Rightarrow$ Partition ist NP-Hart $\Rightarrow$ Partition ist NP-Voll.