Theoretische Grundlagen der Informatik
Blatt 1

Markus Vieth

23. Oktober 2015

1	2	3	Σ
4,5	11	6	21,5 / 25

1 Aufgabe 1

a) Zu zeigen: $n^2 + 2n + 1 \in O(n^2)$ ✓
Beweis:

$$\lim_{n\to\infty}\left(\frac{n^2+2n+1}{n^2}\right) = \lim_{n\to\infty}\left(1+\frac{2}{n}+\frac{1}{n^2}\right) = 1$$

$$\Rightarrow \lim_{n\to\infty}\left(\frac{n^2+2n+1}{n^2}\right) = \limsup_{n\to\infty}\left(\frac{n^2+2n+1}{n^2}\right) = 1 < \infty$$

$$\Rightarrow n^2 + 2n + 1 \in O(n^2)$$

q.e.d.

b) Zu zeigen: $n^{10} + 9n^9 + 20n^8 + 145n^7 \in O(n^{10})$ ✓
Beweis:

$$\lim_{n\to\infty}\left(\frac{n^{10}+9n^9+20n^8+145n^7}{n^{10}}\right) = \lim_{n\to\infty}\left(1+\frac{9}{n}+\frac{20}{n^2}+\frac{145}{n^3}\right) = 1$$

$$\Rightarrow \lim_{n\to\infty}\left(\frac{n^{10}+9n^9+20n^8+145n^7}{n^{10}}\right) = \limsup_{n\to\infty}\left(\frac{n^{10}+9n^9+20n^8+145n^7}{n^{10}}\right) = 1 < \infty$$

$$\Rightarrow n^{10} + 9n^9 + 20n^8 + 145n^7 \in O(n^{10})$$

q.e.d.

c) Zu zeigen: $(n+1)^4 \in O(n^4)$ ✓
Beweis:

$$\lim_{n\to\infty}\left(\frac{(n+1)^4}{n^4}\right) = \lim_{n\to\infty}\left(1+\frac{4}{n}+\frac{6}{n^2}+\frac{4}{n^3}+\frac{1}{n^4}\right) = 1$$

$$\Rightarrow \lim_{n\to\infty}\left(\frac{(n+1)^4}{n^4}\right) = \limsup_{n\to\infty}\left(\frac{(n+1)^4}{n^4}\right) = 1 < \infty$$

$$\Rightarrow (n+1)^4 \in O(n^4)$$

q.e.d.

d) Zu zeigen: $(n^2+n)^2 \in O(n^4)$ ✓
Beweis:

$$\lim_{n\to\infty}\left(\frac{(n^2+n)^2}{n^4}\right) = \lim_{n\to\infty}\left(1+\frac{2}{n}+\frac{1}{n^2}\right) = 1$$

$$\Rightarrow \lim_{n\to\infty}\left(\frac{(n^2+n)^2}{n^4}\right) = \limsup_{n\to\infty}\left(\frac{(n^2+n)^2}{n^4}\right) = 1 < \infty$$

$$\Rightarrow (n^2+n)^2 \in O(n^4)$$

q.e.d.

e) Zu zeigen: $n + 0,001n^3 \in O(n^3)$ ✓
Beweis:

$$\lim_{n\to\infty}\left(\frac{n+0,001n^3}{n^3}\right) = \lim_{n\to\infty}\left(0,001+\frac{1}{n^2}\right) = 0,001$$

$$\Rightarrow \lim_{n\to\infty}\left(\frac{n+0,001n^3}{n^3}\right) = \limsup_{n\to\infty}\left(\frac{n+0,001n^3}{n^3}\right) = 0,001 < \infty$$

$$\Rightarrow n + 0,001n^3 \in O(n^3)$$

q.e.d.

f) Zu zeigen: $n^3 - 1000n^2 + 10n^9 \in O(n^9)$
Beweis:

$$\lim_{n\to\infty}\left(\frac{n^3-1000n^2+10n^9}{n^9}\right) = \lim_{n\to\infty}\left(1+\frac{1}{n^6}-\frac{1000}{n^7}\right) = 1$$

$$\Rightarrow \lim_{n\to\infty}\left(\frac{n^3-1000n^2+10n^9}{n^9}\right) = \limsup_{n\to\infty}\left(\frac{n^3-1000n^2+10n^9}{n^9}\right) = 1 < \infty$$

$$\Rightarrow n^3 - 1000n^2 + 10n^9 \in O(n^9)$$

q.e.d.

g) Zu zeigen: $n + \log n \in O(n)$ ✓
Beweis:

$$\lim_{n\to\infty}\left(\frac{n+\log n}{n}\right) = \lim_{n\to\infty}\left(1+\frac{\log n}{n}\right) = 1$$

$$\Rightarrow \lim_{n\to\infty}\left(\frac{n+\log n}{n}\right) = \limsup_{n\to\infty}\left(\frac{n+\log n}{n}\right) = 1 < \infty$$

$$\Rightarrow n + \log n \in O(n)$$

q.e.d.

h) Zu zeigen: $2^n + n^2 \in O(2^n)$ ✓
Beweis:

$$\lim_{n\to\infty}\left(\frac{2^n+n^2}{2^n}\right) = \lim_{n\to\infty}\left(1+\frac{n^2}{2^n}\right) = 1$$

$$\Rightarrow \lim_{n\to\infty}\left(\frac{2^n+n^2}{2^n}\right) = \limsup_{n\to\infty}\left(\frac{2^n+n^2}{2^n}\right) = 1 < \infty$$

$$\Rightarrow 2^n + n^2 \in O(2^n)$$

q.e.d.

i) Zu zeigen: $\frac{n^3+2n}{n^2+0,75} \in O(n)$ ✓
Beweis:

$$\lim_{n\to\infty}\left(\frac{n^3+2n}{(n^2+0,75)n}\right) = \lim_{n\to\infty}\left(\frac{1+\frac{2}{n^2}}{1+\frac{0,75}{n^2}}\right) = 1$$

$$\Rightarrow \lim_{n\to\infty}\left(\frac{n^3+2n}{(n^2+0,75)n}\right) = \limsup_{n\to\infty}\left(\frac{n^3+2n}{(n^2+0,75)n}\right) = 1 < \infty$$

$$\Rightarrow \frac{n^3+2n}{n^2+0,75} \in O(n)$$

q.e.d.

4,5/5

2 Aufgabe 2

a) Ein Problem ist in P, wenn es sich um ein Entscheidungsproblem handelt, für das ein Algorithmus zur Lösung existiert, dessen worst-case Laufzeit polynomiell ist. ✓

b) Ein Problem ist in NP, wenn es sich um ein Entscheidungsproblem handelt, für das ein Algorithmus zur Überprüfung existiert, dessen worst-case Laufzeit polynomiell ist. ✓

c) Ein Problem ist NP-schwierig, wenn für jedes Problem aus NP ein Algorithmus existiert, welcher es in polynomieller Zeit auf das NP-schwere Problem reduzieren kann. Insbesondere muss das NP-schwere Problem selbst nicht in NP liegen. ✓

d) Ein Problem ist NP-vollständig, wenn es NP-schwer ist und in NP liegt. ✓

e) Man muss zeigen, dass sich das Problem in NP befindet, in dem man einen Algorithmus findet, welcher das Entscheidungsproblem in polynomieller Laufzeit überprüft. Anschließend zeigt man, dass man jedes Problem aus NP auf dieses Problem reduzieren kann. Da meist mindestens ein NP-vollständiges Problem bekannt ist, reicht es dieses mithilfe eines Algorithmus mit polynomieller Laufzeit auf das zu untersuchende Problem zu reduzieren, da die Reduktion transitiv ist. ✓

f) Ein NP-vollständiges Problem kann von einem deterministischen Rechner in polynomieller Laufzeit überprüft werden. Des Weiteren kann jedes Problem aus NP in polynomieller Laufzeit auf ein NP-vollständiges Problem reduziert werden. (✓)

½

11/12

3 Aufgabe 3

(Bei der Lösung der Aufgabe wird von einem nicht-deterministischen Rechner ausgegangen. Alternativ kann der nicht-deterministische Teil am Anfang des Algorithmus weggelassen werden und die Eingabe muss um eine zu prüfende Lösung ergänzt werden.)
Im Folgenden Werden Mengen im Pseudo-Code wie Felder behandelt, in denen jedes Element nur einmal Auftreten kann.

a) Es sei V die Menge alle Knoten des Graphen und E die Menge der Kanten.

Algorithmus:

```
rate nicht-deterministisch ein Tupel mit |V| Knoten aus V und Speichere dieses in V'

für alle V'[i]
    für alle V'[j]
        wenn i != j
            wenn V'[i] == V'[j]
                return nein

für alle V'[i]
    für alle E[j]
        wenn {V'[i],V'[i+1 mod |V'|]} == E[j]
            v++
            break

wenn v == |V'|
    return ja
sonst
    return nein
```

Erklärung: Zuerst wird ein Tupel mit $|V|$ Knoten nicht-deterministisch geraten. In der 1. Schleife wird in $|V| \cdot |V|$ (relevanten) Vergleichen geprüft, ob jeder Knoten genau einmal vorkommt. In der 2. Schleife wird in $|V| \cdot |E|$ Vergleichen geprüft, ob es zwischen einem Knoten im Tupel und dem folgendem eine Kante im Graphen gibt. Sollte dies für alle Knoten zutreffen, insbesondere zwischen dem letzten und dem ersten, so existiert ein Hamiltonkreis im Graphen.
Da beide Schleifen unabhängig in polynomieller Zeit durchlaufen, hat der ganze Algorithmus polynomielle Laufzeit. (✓)

Eingabe: Als Eingabe werden benötigt, die Menge der Knoten und die Menge der Kanten des zu untersuchenden Graphen. ✓

Viel zu viel 😊

2/2

b) Es sei V die Menge alle Knoten des Graphen, E die Menge der Kanten, d die zu unterschreitenden Kosten und K die Tabelle der Kosten der Kanten, wobei $K[i]$ den Kosten der Kante $E[i]$ entspricht.

Algorithmus:

```
rate nicht-deterministisch ein Tupel mit |V| Knoten aus V und speichere diese in V'

für alle V'[i]
    für alle V'[j]
        wenn i != j
            wenn V'[i] == V'[j]
                return nein

für alle V'[i]
    für alle E[j]
        wenn {V'[i],V'[i+1 mod |V'|]} == E[j]
            v++
            k = k + K[j]
            break

wenn v == |V'| && k <= d
    return ja
sonst
    return nein
```

Erklärung: Zuerst wird ein Tupel mit |V| Knoten nicht-deterministisch geraten. In der 1. Schleife wird in $|V| \cdot |V|$ (relevanten) Vergleichen geprüft, ob jeder Knoten genau einmal vorkommt. In der 2. Schleife wird in $|V| \cdot |E|$ Vergleichen geprüft, ob es zwischen einem Knoten im Tupel und dem folgendem eine Kante im Graphen gibt, zusätzlich werden die Kosten der entsprechenden Kanten aufsummiert. Sollte dies für alle Knoten zutreffen, insbesondere zwischen dem letzten und dem ersten, und sollten die Kosten kleiner oder gleich d sein so existiert ein Lösung des Traveling-Salesman im Graphen.
Da beide Schleifen unabhängig in polynomieller Zeit durchlaufen, hat der ganze Algorithmus polynomielle Laufzeit.

Eingabe: Als Eingabe werden benötigt, die Menge der Knoten, die Menge der Kanten des zu untersuchenden Graphen, die zu unterschreitenden Kosten und die Kosten der Kanten.

1/2

c) Es sei V die Menge alle Knoten des Graphen und E die Menge der Kanten.

Algorithmus:

```
rate nicht-deterministisch eine Menge mit n Knoten aus V und speichere diese in V'

kopiere E in E'

für alle V'[i]
    für alle V'[j]
        wenn i != j
            wenn V'[i] == V'[j]
                return nein

für alle V'[i]
    für alle E'[j]
        für alle E'[j][k]
            wenn V'[i] == E'[j][k]
                lösche E'[j]

wenn |E'| == 0
    return ja
sonst
    return nein
```

Erklärung: Zuerst wird nicht-deterministisch eine Menge mit n Knoten aus V geraten. Anschließend wird die Menge der Kanten E zur Bearbeitung in E' kopiert, Laufzeit $|E|$. In der ersten Schleife wird geprüft, ob V' auch wirklich eine Menge ist, Laufzeit $n \cdot n$. Die zweite Schleife löscht nun jede Kante in E', welche an einem Knoten aus V' endet, Laufzeit $n \cdot |E|$. Ist am Ende die Menge E' leer, so endet jede Kante aus E in einem Knoten aus V'. Somit wäre ein Vertex-Cover im gegebenen Graphen mit n Knoten möglich.
Da beide Schleifen unabhängig in polynomieller Zeit durchlaufen, hat der ganze Algorithmus polynomielle Laufzeit.

Eingabe: Als Eingabe werden benötigt, die Menge der Knoten und die Menge der Kanten des zu untersuchenden Graphen, sowie die Anzahl der zu verwendenden Knoten.

d) Es sei V die Menge alle Knoten des Graphen und E die Menge der Kanten.

Algorithmus:

```
rate nicht-deterministisch eine Menge E' mit |V|-1 Kanten aus E

kopieren V in V'

für alle E'[i]
    für alle E'[j]
        wenn i != j
            wenn E'[i] == E'[j] oder (E'[i][1] == E'[j][2] und E'[i][2] == E'[j][1])
                return nein

für alle E'[i]
    für alle E'[i][j]
        für alle V'[k]
            wenn E'[i][j] == V'[k]
                lösche V'[k]

wenn |V'| == 0
    return ja
sonst
    return nein
```

Erklärung: Zuerst wird nicht-deterministisch eine Menge E' mit $|V|-1$ Kanten aus E geraten (Erklärung siehe unten). Anschließend wird V in V' kopiert, damit V' bearbeitet werden kann, Laufzeit $|V|$. In der ersten Schleife wird geprüft, die ungerichteten Kanten in E' paarweise verschieden sind, Laufzeit $c \cdot (|V|-1)^2$, mit $c \in \mathbb{N}$. Die zweite Schleife löscht jeden Knoten in V', der in einer Kante aus E' vorkommt, Laufzeit $2 \cdot (|V|-1) \cdot |V|$. Ist V' am Ende leer, so existiert ein Spannbaum im Graphen, sonst nicht.
Da alle Schleifen unabhängig in polynomieller Zeit durchlaufen, hat der ganze Algorithmus polynomielle Laufzeit.

Eingabe: Als Eingabe werden benötigt, die Menge der Knoten und die Menge der Kanten des zu untersuchenden Graphen.

Erklärung: Der Algorithmus basiert auf der Überlegung, dass ein Spannbaum einen Knoten besitzt, welcher als Wurzel fungiert. Jeder weitere Knoten ist in einem solchen Spannbaum über genau einen Weg mit der Wurzel verbunden. Per Induktion könnte man nun zeigen, dass jeder Knoten außer der Wurzel genau eine zusätzliche Kante benötigt, um einen Baum zu bilden. Des Weiteren ist es unmöglich alle Knoten mit $|V|-1$ Kanten abzudecken und einen Zyklus zu schaffen.

Es geht deterministisch

1/2

6/8