Tutorien-Übungsblatt 8

Aufgabe 1

Ein Algorithmus, der eine Zahl $n \in \mathbb{N}$ als Eingabe erhält und prüft, ob n prim ist, sei gegeben durch die nachfolgende Beschreibung einer Turingmaschine $\mathcal{M}$, die die Funktion $f : \mathbb{N} \to \{0,1\}, n \mapsto \begin{cases} 1 & , \ n \text{ prim} \\ 0 & , \ \text{sonst} \end{cases}$ realisiert:

1. Initialisiere Zähler z mit 2 und Ausgabe a mit 1
2. Berechne $n \ \% \ z$
3. Prüfe, ob $n \ \% \ z = 0$ gilt:
- Falls ja: Setze a auf 0 und gehe zu Schritt 4.
- Falls nein: Gehe zu Schritt 4.
4. Prüfe, ob $z < n$ gilt:
- Falls ja: Erhöhe z um 1 und gehe zu Schritt 2.
- Falls nein: Lösche das Band, schreibe a auf das Band und stoppe

Geben Sie die Komplexität des oben angegebenen Algortihmus bezüglich der Eingabe n und auch bezüglich der Länge der Binärdarstellung von n an!

Aufgabe 2

Die Komplexitätsklasse **co-P** sei definiert als die Menge der Sprachen $\mathcal{L}$, deren Komplementsprache $\mathcal{L}^C$ in der Komplexitätsklasse **P** liegt.
Erinnerung: Zu einer Sprache $\mathcal{L}$ über einem Alphabet Σ ist die Komplementsprache $\mathcal{L}^C = \Sigma^* \setminus \mathcal{L}$.

Beweisen Sie: **co-P** = **P**

Aufgabe 3 - Graph-2-Färbbarkeit

Gegeben seien ein ungerichteter Graph $G = (V, E)$ mit Knoten $v \in V$ und Kanten $e = (v_1, v_2) \in E$ mit $v_1, v_2 \in V$ und zwei Farben A und B.
Beweisen Sie: Die Sprache 2-COLOR $=$
$\{G \mid G = (V, E)$ ungerichteter Graph mit $\exists$ totale Funktion $g : V \to \{A, B\} : \forall \ (v_1, v_2) \in E : g(v_1) \neq g(v_2)\}$
liegt in der Komplexitätsklasse **P**.

Aufgabe 4

Das Problem 2-SAT ist folgendermaßen definiert:

2-SAT
Gegeben eine in ihrer Größe polynomiell beschränkte aussagenlogische Formel F in konjuktiver Normalform, wobei jede Klausel genau 2 Literale enthält. F hat also die Form

$$F = \Lambda_{i=1}^{n}(L_i \vee N_i),$$

wobei L_i und N_i Literale sind, also von der Form X oder $\neg X$ für eine Variable X sind.
Gibt es eine erfüllende Belegung für F?

Geben Sie eine polynomielle Reduktion von 2-COLOR auf 2-SAT an!

Lösung zu Aufgabe 1

Die Komplexität bezüglich der Eingabe n ist $O(n)$. Sei die Länge der Binärdarstellung von n bezeichnet mit $|n|$, dann gilt: $|n| = \log_2(n)$. Daraus folgt, dass $n = 2^{|n|}$ ist. Somit ist die Komplexität in $O(2^{|n|})$.

Lösung zu Aufgabe 2

P $\subseteq$ co-P:
Sei $\mathcal{L} \in \mathbf{P}$. Dann gibt es eine polynomiale deterministische Turingmaschine T mit der Zustandsmenge Q und der Endzustandsmenge $F \subseteq Q$, die $\mathcal{L}$ akzeptiert, die also für alle Eingaben $w \in \Sigma^*$ nach polynomialer Zeit hält und nur genau dann in einem Endzustand $q \in F$ hält, wenn $w \in \mathcal{L}$ gilt. Konstruiere daraus die polynomiale deterministische Turingmaschine T', die ähnlich aussieht wie T und nur folgende Änderungen aufweist:

1. Alle bisherigen Endzustände $q \in F$ sind in T' keine Endzustände mehr.
2. Die Zustandsmenge Q wird um zwei neue Zustände q^+, q^- erweitert, wobei T' in diesen beiden Zuständen für jedes Zeichen $a \in \Gamma$ immer sofort hält und q^+ der einzige Endzustand von T' ist.
3. Statt in einem ehemaligen Endzustand $q \in F$ zu halten, geht T' von diesem Zustand für jedes Zeichen $a \in \Gamma$ in den neuen Zustand q^- über.
4. Statt für ein Zeichen $a \in \Gamma$ in einem Zustand $q \in Q \setminus F$ zu halten, geht T' von diesem Zustand für das Zeichen a in den neuen Zustand q^+ über.

T' akzeptiert genau die Komplementärsprache $\mathcal{L}^C$ und es gilt daher $\mathcal{L}^C \in \mathbf{P}$. Damit gilt also auch $\mathcal{L} \in$ **co-P**.

co-P $\subseteq$ P:
Sei $\mathcal{L} \in$ **co-P**. Dann ist die Komplementsprache $\mathcal{L}^C \in \mathbf{P}$. Damit gibt es eine polynomiale deterministische Turingmaschine T^C, die $\mathcal{L}^C$ in polynomialer Zeit entscheidet. Analog zu oben kann man daraus eine polynomiale deterministische Turingmaschine T konstruieren, die $\mathcal{L}$ akzeptiert. Damit ist $\mathcal{L} \in \mathbf{P}$.

Wir haben nun gezeigt, dass $\mathbf{P} \subseteq$ **co-P** und **co-P** $\subseteq \mathbf{P}$, damit erhalten wir insgesamt $\mathbf{P} =$ **co-P**.

Lösung zu Aufgabe 3

Vorgehen:
Wir zeigen zuerst, dass ein Graph genau dann 2-färbbar ist, wenn er bipartit (siehe unten) ist. Es reicht dann zum Beweisen von 2-COLOR $\in \mathbf{P}$, einen polynomialen Algorithmus anzugeben, der prüft, ob ein Graph bipartit ist.

Definition:
Ein Graph $G = (V, E)$ heißt *bipartit*, wenn sich seine Knoten in zwei disjunkte Mengen M und N einteilen lassen, so dass keine Knoten innerhalb der Teilmengen miteinander durch eine Kante verbunden sind. D.h. für jede Kante $(v_1, v_2) \in E$ gilt $(v_1 \in M) \wedge (v_2 \in N)$ oder $(v_1 \in N) \wedge (v_2 \in M)$.

Jeder bipartite Graph ist 2-färbbar:
Sei $G = (V, E)$ ein bipartiter Graph. Damit gibt es zwei disjunkte Teilmengen M und N wie in der obigen Definition von bipartit beschrieben. Färbe nun alle Knoten in der Menge M mit der Farbe A und alle Knoten in der Menge N mit der Farbe B. Zwei Knoten haben genau dann die gleiche Farbe, wenn sie in der gleichen Menge (M oder N) liegen. Da Knoten innerhalb der jeweiligen Mengen nicht durch Kanten miteinander verbunden sind, und damit nur Kanten zwischen einem Knoten aus M und einem Knoten aus N existieren, haben zwei durch Kanten verbundene Knoten nie die gleiche Farbe. Damit ist G 2-färbbar, d.h. $G \in$ 2-COLOR.

Jeder 2-färbbare Graph ist bipartit:
Sei $G \in$ 2-COLOR ein 2-färbbarer Graph. Dann gibt es eine 2-Färbung $g : V \to \{A, B\}$ für G. Definiere die Mengen M und N so, dass in M alle Knoten v mit der Farbe $g(v) = A$ und in N alle Knoten v mit der Farbe $g(v) = B$ liegen. Knoten innerhalb von M oder innerhalb von N können nicht durch Kanten miteinander verbunden sein, denn sonst wäre ja g keine korrekte 2-Färbung. Damit haben wir die Knoten von G in zwei disjunkte Mengen unterteilt, so dass keine Kanten zwischen Knoten innerhalb einer Menge existieren. Damit ist G bipartit.

Insgesamt haben wir nun gezeigt, dass ein Graph G genau dann in 2-COLOR ist, wenn er bipartit ist. Um zu testen, ob ein Graph G in 2-COLOR ist, können wir also einen Algorithmus verwenden, der testet, ob G bipartit ist.

Ein Algorithmus, der in polynomialer Zeit prüft, ob ein Graph $G = (V, E)$ bipartit ist, ist z.B. der folgende:

1. Setze zu Beginn die zwei Mengen M und N auf die leere Menge: $M = N = \varnothing$.
2. Falls der Graph nicht zusammenhängend ist, führe den Rest des Algorithmus für jeden der nicht miteinander verbundenen Teilgraphen aus.
3. Beginne mit einem beliebigen Knoten $v_0 \in V$. Füge v_0 zu M hinzu und füge außerdem alle Knoten, die mit v_0 durch eine Kante verbunden sind, der Menge N hinzu.
4. Für alle Knoten $v \in M$: Füge alle Knoten, die mit v durch eine Kante verbunden und noch nicht in der Menge M oder N sind, der Menge N hinzu.
5. Für alle Knoten $v \in N$: Füge alle Knoten, die mit v durch eine Kante verbunden und noch nicht in der Menge M oder N sind, der Menge M hinzu.
6. Wiederhole 4. und 5., bis alle Knoten von G in einer der beiden Mengen M oder N sind.
7. Gib als Ergebnis aus: Falls Knoten innerhalb von M oder innerhalb von N durch eine Kante miteinander verbunden sind, ist der Graph nicht bipartit. Ansonsten ist der Graph bipartit.

Aufwand:
Dieser Algorithmus hat polynomialen Aufwand in der Anzahl n der Knoten: Das Sortieren in die Mengen M und N hat einen Aufwand von $O(n^2)$ (Für jeden Knoten alle Kanten durchgehen: $O(n^2)$, für jeden verbundenen Knoten testen, ob er schon in M oder N liegt: $O(n)$). Das Testen, ob eine Kante zwischen zwei Knoten aus M oder zwei Knoten aus N existiert, hat auch einen Aufwand von $O(n^2)$. Damit hat der Algorithmus insgesamt polynomiale Laufzeit in der Länge der Darstellung des Graphen (die etwa der Anzahl Knoten entspricht).

Korrektheit:
Der Algorithmus funktioniert deshalb, weil in einem bipartiten Graphen $G = (V, E)$ und der Aufteilung in die zwei Mengen M und N (nach der Definition) für jeden Knoten $v \in M$ gilt, dass alle mit v verbundenen Knoten in N liegen, und umgekehrt. Der Algorithmus geht also erstmal davon aus, der Graph sei bipartit und stellt dann eine Partition in die Mengen M und N her. Falls G wirklich bipartit ist, ist das Ergebnis eine korrekte Aufteilung in zwei disjunkte Mengen, innerhalb derer keine Kanten existieren. Falls nicht, so kann keine solche korrekte Aufteilung herauskommen, und damit müssen Knoten innerhalb von M oder innerhalb von N miteinander verbunden sein.

Lösung zu Aufgabe 4

Sei $G = (V, E)$ ein ungerichteter Graph, für den wir feststellen wollen, ob $G \in$ 2-COLOR gilt. Dazu seien wieder die beiden Farben A und B gegeben. Wir bilden für die Reduktion zunächst die Menge der zur Verfügung stehenden Variablen $\chi := \{X_v \mid v \in V\}$. Falls der gegebene Graph G 2-färbbar ist, so gibt es eine 2-Färbung $g : V \to \{A, B\}$, andernfalls gibt es zumindest eine Abbildung $g : V \to \{A, B\}$. Diese verwenden wir zur Definition einer Variablenbelegung:

$$f : \chi \to \{\text{TRUE}, \text{FALSE}\}, X_v \mapsto \begin{cases} \text{TRUE} & , \quad \text{falls } g(v) = A \\ \text{FALSE} & , \quad \text{falls } g(v) = B \end{cases}$$

Nun bilden wir jede Kante $(u, v) \in E$ auf die passende aussagenlogische Formel $(X_u \oplus X_v)$ ab. Die obige Variablenbelegung f ergibt genau dann für alle diese aussagenlogischen Formeln eine gemeinsame erfüllende Variablenbelegung, wenn die zugrunde liegende Abbildung g bereits eine korrekte 2-Färbung und der Graph G damit 2-färbbar ist. Nun müssen die Formeln noch in die richtige Form gebracht werden:

$$(X_u \oplus X_v) \cong ((X_u \wedge \neg X_v) \vee (\neg X_u \wedge X_v)) \cong ((X_u \vee X_v) \wedge (\neg X_u \vee \neg X_v))$$

Wir erhalten somit für jede Kante $(u, v) \in E$ die beiden Klauseln $X_u \vee X_v, \neg X_u \vee \neg X_v$ und für den gesamten Graphen G die 2-SAT-Instanz $K := \bigcup_{(u,v)\in E} \{X_u \vee X_v, \neg X_u \vee \neg X_v\}$, die also genau dann eine erfüllende Belegung besitzt, wenn G 2-färbbar ist. Wir haben damit also eine Reduktion gefunden und müssen nur noch zeigen, dass diese mit polynomialem Aufwand möglich ist. Dies ist aber recht einfach einzusehen, da zur Bildung der Klauselmenge K nach deren Definition nur einmal alle Kanten durchlaufen werden müssen, der Aufwand liegt also in $O(|E|) = O(|V|^2)$.