Jun.-Prof. Dr. Bernhard Beckert
Ulrich Koch

UNIVERSITÄT KOBLENZ · LANDAU

Theoretische Informatik II
Wintersemester 2007/2008
11. Aufgabenblatt

Ausgabe: 04. 02. 2008 Besprechung: 07. 02. 2008

---

1 Komplexität 1

CHROMATIC NUMBER ist das Problem, ob ein Graph G mit k (oder weniger) Farben gefärbt werden kann.
Beweisen Sie, dass CHROMATIC NUMBER $\mathcal{NP}$-vollständig ist, indem Sie

1. zeigen, dass es in $\mathcal{NP}$ liegt, und
2. 3-CNF-SAT (das Erfüllbarkeitsproblem für aussagenlogische Formeln in CNF, wobei jede Klausel 3 Literale hat) polynomiell auf CHROMATIC NUMBER reduzieren.

Lösung:

1. $G = (V, E)$ (ungerichtet) und k sind gegeben; dabei ist $V = \{v_1, \ldots, v_m\}$. Die NTM rät eine Abbildung $color : V \to \{1, \ldots, k\}$ und prüft dann folgendermaßen, ob $color$ eine zulässige Färbung ist:

```
for i := 1...m do
   for j := 1...m do
      if (v_i, v_j) ∈ E ∧ color(v_i) = color(v_j) then
         return false
      end
   end
end;
return true
```

   Um den Zeitaufwand für diesen Algorithmus abzuschätzen, müssen wir wissen, in welcher Zeit der Test $(v_i, v_j) \in E \wedge color(v_i) = color(v_j)$ abläuft. Dafür ist die Codierung der Eingabe (G, k) des Algorithmus wichtig. (G, k) soll als $w := m'\#E'\#k'$ dargestellt werden, wobei gilt:

   - m' ist die Binärdarstellung von m ohne führende Nullen.
   - $E' \in \{0, 1\}^*$ ist eine binäre $(m \times m)$-Matrix, die an der Stelle i, j genau dann eine 1 enthält, wenn v_i und v_j benachbart sind.
   - k' ist die Binärdarstellung von k ohne führende Nullen.

Damit ergibt sich:

$$\begin{aligned}|m'| &= O(\log_2 m)\\ |E'| &= O(m^2)\\ |k'| &= O(\log_2 k)\\ |w| &= O(\log_2 m + m^2 + \log_2 k) = O(m^2 + \log k)\end{aligned}$$

Der Test $(v_i, v_j) \in E$ läuft in konstanter Zeit ab, weil E' linear codiert wird, wobei die Indizes in der Reihenfolge $i = 1, j = 1; i = 1, j = 2; \ldots; i = 1, j = m; i = 2, j = 1; \ldots; i = m, j = m$ variieren. Die NTM bewegt den Lesekopf auf E' bei jedem Durchlauf der inneren Schleife um ein Feld nach rechts.

Für den Test $color(v_i) = color(v_j)$ benutzt die NTM zwei Bänder. Jedes enthält eine Kopie von *color*. Der Kopf auf dem einen Band wird bei jedem Durchlauf der äußeren Schleife um ein Arrayelement bewegt (Index i). Der Kopf auf dem anderen Band wird bei jedem Durchlauf der inneren Schleife um ein Element bewegt (Index j); zusätzlich muss er bei jedem Durchlauf der äußeren Schleife auf das erste Element zurückgesetzt werden, also um m Elemente. Die Arrayelemente (Farben) werden binär ohne führende Nullen dargestellt, also hat jedes Element die Länge $\lfloor \log_2 k \rfloor + 1$.

Damit ist der Zeitaufwand für die innere Schleife in $O(m \log k + m \log k) = O(m \log k)$. Da die äußere Schleife aus m Ausführungen der inneren Schleife besteht, ist der Zeitaufwand für die gesamte Prüfung der geratenen Färbung in $O(m^2 \log k)$. Mit $n := |w| = O(m^2 + \log k)$ ist der Zeitaufwand in $O(n^2)$.

2. Eine Formel habe die Form $C_1 \wedge \ldots \wedge C_k$, wobei jedes C_i die Form $(L_{i,1} \vee L_{i,2} \vee L_{i,3})$ habe. Jedes $L_{i,j}$ sei entweder x_l oder $\overline{x_l}$ für eine Variable x_l; dabei sei $l \in \{1, \ldots, m\}$. Da jede Klausel nur 3 Literale hat, gilt $m \leq 3k$. Bei der Darstellung der Formel ist zu beachten, dass die Indizes l der Variablen binär (ohne führende Nullen) dargestellt werden.

   Die reduzierende Funktion f bildet Wörter, die keine syntaktisch korrekten Formeln sind, auf ε ab. Eine wohlgeformte Formel F dagegen wird auf $(G, m+1)$ abgebildet, wobei $G = (V, E)$ ein ungerichteter Graph ist. Dabei gilt:

$$\begin{aligned}V = {} & \{C_1, \ldots, C_k\} \cup \{x_1, \ldots, x_m\} \cup \{\overline{x_1}, \ldots, \overline{x_m}\} \cup \{y_1, \ldots, y_m\}\\ E = {} & \{(x_i, \overline{x_i}), (\overline{x_i}, x_i) \mid i \in \{1, \ldots, m\}\} \cup \{(y_i, y_j) \mid i \neq j\} \cup\\ & \{(y_i, x_j), (x_j, y_i) \mid i \neq j\} \cup \{(y_i, \overline{x_j}), (\overline{x_j}, y_i) \mid i \neq j\} \cup\\ & \{(C_i, x_j), (x_j, C_i) \mid x_j \text{ ist keins der Literale in } C_i\} \cup\\ & \{(C_i, \overline{x_j}), (\overline{x_j}, C_i) \mid \overline{x_j} \text{ ist keins der Literale in } C_i\}\end{aligned}$$

   Die Knoten y_i, x_i und $\overline{x_i}$ können folgendermaßen mit $m+1$ Farben gefärbt werden:

   - Jedes y_i bekommt die Farbe i.
   - Für alle $i \in \{1, \ldots, m\}$ gilt: Entweder wird x_i mit i und $\overline{x_i}$ mit $m+1$ gefärbt, oder umgekehrt.

   Da es für alle i, j mit $i \neq j$ eine Kante zwischen y_i und y_j gibt, ist G nicht mit weniger als m Farben färbbar. Weil zusätzlich jedes y_i mit jedem x_j und jedem $\overline{x_j}$ für $i \neq j$ benachbart ist und x_j mit $\overline{x_j}$ benachbart ist, braucht man auch die Farbe $m+1$. Dabei entspricht die Färbung mit $m+1$ einer Belegung des entsprechenden Literals mit $false$ und die Färbung mit j einer Belegung mit $true$.

Jeder Knoten C_i ist mit genau den Literalen benachbart, die nicht in der Klausel C_i vorkommen. Da jede Klausel nur 3 Literale enthält, folgt daraus für $m > 4$, dass es für jedes i ein j gibt, so dass C_i sowohl mit x_j als auch mit $\overline{x_j}$ benachbart ist. Also kann kein C_i mit $m+1$ gefärbt werden. Außerdem scheidet für C_i jede Farbe aus, die ein Literal trägt, das nicht in der Klausel C_i vorkommt. Damit muss C_i dieselbe Farbe bekommen wie ein Literal, das in der Klausel vorkommt und mit *true* belegt wird (vorausgesetzt, es gibt ein solches Literal). Das bedeutet, dass auch C_i die Belegung *true* hat. Damit sind alle Klauseln mit *true* belegt, und F wird wahr. Wenn dagegen für ein i alle Literale, die in C_i vorkommen, mit $false$ belegt sind, wird C_i und damit auch F falsch, und es gibt keine $(m+1)$-Färbung für G. Insgesamt ist G also genau dann mit $m+1$ Farben färbbar, wenn F erfüllbar ist.

Für den Zeitaufwand von f ergibt sich:

- Die Kanten zwischen x_i und $\overline{x_i}$ können mit einem Zeitaufwand gelegt werden, der in $O(m)$ ist.
- Der Zeitaufwand für die Kanten zwischen y_i und y_j ist in $O(m^2)$.
- Dasselbe gilt für die Kanten zwischen y_i und x_j ...
- ... und für die zwischen y_i und $\overline{x_j}$.
- Der Zeitaufwand für die Kanten zwischen C_i und x_j ist in $O(km)$.
- Dasselbe gilt für die Kanten zwischen C_i und $\overline{x_j}$.
- Der Zeitaufwand für das Legen aller Kanten ist somit in $O(m + m^2 + km) = O(m^2 + km)$, wegen $m \leq 3k$ also in $O(k^2)$.
- Da die Indizes der Variablen binär (ohne führende Nullen) dargestellt werden, ist die Länge der Eingabe F in $O(k \log_2 m)$, wegen $m \leq 3k$ also in $O(k \log k)$.
- Die Ausgabe $(G, m+1)$ wird wie in 1. dargestellt; damit ist ihre Länge in $O((k+m)^2 + \log m) = O(k^2)$.

Damit ist für $n := |F|$ der Zeitaufwand von f in $O(n^2)$.

2 Komplexität 2

In der Vorlesung wurde gesagt:

1. CNF-SAT (das Erfüllbarkeitsproblem für aussagenlogische Formeln in CNF) ist $\mathcal{NP}$-vollständig.
2. DNF-SAT (das Erfüllbarkeitsproblem für aussagenlogische Formeln in DNF) liegt in $\mathcal{P}$.

Wenn man CNF-SAT polynomiell auf DNF-SAT reduzieren könnte, wäre also bewiesen, dass $\mathcal{P} = \mathcal{NP}$.
Nun lassen sich Formeln in CNF generell durch Anwendung des Distributivgesetzes

$$A \wedge (B_1 \vee \ldots \vee B_k) \equiv (A \wedge B_1) \vee \ldots \vee (A \wedge B_k)$$

in DNF umwandeln. Warum führt dies nicht zu einer polynomiellen Reduktion?

Lösung:

Die umzuwandelnde Formel sei:

$$F_1 \equiv (L_{1,1} \vee \ldots \vee L_{1,m}) \wedge \ldots \wedge (L_{k,1} \vee \ldots \vee L_{k,m})$$

Es gilt:

$$F_1 \equiv \underbrace{(L_{1,1} \vee \ldots \vee L_{1,m}) \wedge (L_{2,1} \vee \ldots \vee L_{2,m}) \wedge \ldots \wedge (L_{k,1} \vee \ldots \vee L_{k,m})}_{km \text{ Literale}} \equiv$$

$$((L_{1,1} \vee \ldots \vee L_{1,m}) \wedge L_{2,1} \vee \ldots \vee (L_{1,1} \vee \ldots \vee L_{1,m}) \wedge L_{2,m}) \wedge$$
$$(L_{3,1} \vee \ldots \vee L_{3,m}) \wedge \ldots \wedge (L_{k,1} \vee \ldots \vee L_{k,m}) \equiv$$

$$\underbrace{(L_{1,1} \wedge L_{2,1} \vee \ldots \vee L_{1,m} \wedge L_{2,1}) \vee \ldots \vee (L_{1,1} \wedge L_{2,m} \vee \ldots \vee L_{1,m} \wedge L_{2,m})}_{2m^2 \text{ Literale}} \wedge$$
$$(L_{3,1} \vee \ldots \vee L_{3,m}) \wedge \ldots \wedge (L_{k,1} \vee \ldots \vee L_{k,m}) \equiv \cdots \equiv$$

$$\underbrace{L_{1,1} \wedge \ldots \wedge L_{k,1} \vee \ldots \vee L_{1,m} \wedge L_{2,1} \wedge \ldots \wedge L_{k,1} \vee \ldots \vee L_{1,m} \wedge L_{2,m} \wedge \ldots \wedge L_{k,m}}_{km^k \text{ Literale}} \equiv: F_2$$

Falls außerdem $k = m$ gilt, folgt daraus mit $n := km = m^2$, dass die Länge von F_2 (proportional zur Anzahl der Literale) in $O(n^{O(\sqrt{n})})$ ist. Für solche F_1 ist also schon die Länge der Ausgabe überpolynomiell in der Länge der Eingabe. Die Länge der Ausgabe ist aber eine untere Schranke für die Laufzeit des Algorithmus; folglich kann eine Reduktion von der CNF in die DNF nur mittels Anwendung des Distributivgesetzes nicht polynomiell sein.