Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Ferien-Übungsblatt 8

Vorlesung Algorithmentechnik im WS 09/10

Ausgabe 4. Februar 2010
Abgabe keine

Problem 1: Probabilistische Komplexitätsklassen [vgl. Kapitel 8 im Skript] **

Studieren Sie die Komplexitätsklassen aus Kapitel 8 im Skript.

(a) Zeigen Sie: Für jedes Polynom $q(n) \geq 1$ kann man in der Definition von $\mathcal{RP}$ anstelle von $1/2$ auch $1 - 2^{-q(n)}$ einsetzen, also Fehlerwahrscheinlichkeit $2^{-q(n)}$ statt $1/2$ fordern, ohne an der Klasse was zu ändern.

Die Klasse co-$\mathcal{RP}$ enthält die Entscheidungsprobleme Π für die es einen polynomiellen, randomisierten Algorithmus $\mathcal{A}$ gibt, so dass für alle Instanzen $\mathcal{I}$ von Π gilt:

$$\begin{cases} \mathcal{I} \in Y_\Pi \longrightarrow \Pr[\mathcal{A}(\mathcal{I}) \text{ ist "Nein"}] = 0 \\ \mathcal{I} \notin Y_\Pi \longrightarrow \Pr[\mathcal{A}(\mathcal{I}) \text{ ist "Nein"}] \geq 1/2 \end{cases}$$

Die Klasse $\mathcal{ZPP}$ (zero-error probabilistic polyn.-time) ist definiert durch $\mathcal{ZPP} := \mathcal{RP} \cap co - \mathcal{RP}$.

(b) Welche Art von Algorithmen sind in $\mathcal{ZPP}$ enthalten? Begründen Sie Ihre Antwort.

(c) Zu einem Problem Π sei $\mathcal{A}$ ein $\mathcal{RP}$-Algorithmus und $\overline{\mathcal{A}}$ ein co-$\mathcal{RP}$-Algorithmus wobei $\mathcal{A}$ und $\overline{\mathcal{A}}$ jeweils Laufzeit $p(n)$ haben. Geben Sie für Π einen $\mathcal{ZPP}$-Algorithmus $\mathcal{A}'$ an, der $\mathcal{A}$ und $\overline{\mathcal{A}}$ verwendet. Geben Sie weiterhin die erwartete Laufzeit von $\mathcal{A}'$ an.

Hinweis: Es gilt $2 - \sum_{i=0}^{k} i2^{-i} = (k+2)2^{-k}$.

Problem 2: Vergleich von Wörtern [vgl. Kapitel 8 im Skript] **

In dieser Aufgabe widmen wir uns folgender Problemstellung. Gegeben seien zwei Bitfolgen $a_1 \cdots a_n$ und $b_1 \cdots b_n$, von denen wir uns vorstellen, dass sie an verschiedenen Orten vorliegen. Möchte man nun überprüfen, ob die Bitfolgen identisch sind, so lässt sich das natürlich bewerkstelligen, indem man alle n Bits von einem Ort zum anderen überträgt. Der Monte-Carlo-Algorithmus 1 benötigt dafür jedoch bloß $\Theta(\log n)$ Bits. Eine Bitfolge $a_1 \cdots a_n$ kann auf natürliche Weise als eine Zahl $a := \sum_{i=1}^{n} a_i 2^{i-1}$ interpretiert werden.

(a) Beschreiben Sie, wie Sie Algorithmus 1 dazu benutzen können um über die Distanz zu überprüfen ob $a_1 \cdots a_n = b_1 \cdots b_n$ ist. Was müssen Sie dazu übertragen? Zeigen Sie, dass die Anzahl übertragener Bits in $\mathcal{O}(\log n)$ liegt.

(b) Zeigen Sie, dass die Wahrscheinlichkeit, dass Algorithmus 1 ein falsches Ergebnis liefert in $\mathcal{O}(1/n)$ liegt. Benutzen Sie dafür die folgenden beiden Resultate:

Algorithmus 1 : Überprüfe ob Bitfolgen gleich sind

Eingabe : Bitfolgen $a_1 \cdots a_n$ und $b_1 \cdots b_n$
Ausgabe : Entscheidung ob Bitfolgen gleich sind

$p \leftarrow \langle$Primzahl, zufällig gleichverteilt aus denen kleiner oder gleich $n^2 \log n^2$ gewählt$\rangle$
$a \leftarrow \sum_{i=1}^{n} a_i 2^{i-1}$
$b \leftarrow \sum_{i=1}^{n} b_i 2^{i-1}$
Wenn $a \mod p \equiv b \mod p$
  return ja
sonst
  return nein

Satz 2.1 (Chebyshew)**.** *Für die Anzahl* $\mathfrak{p}(n)$ *der Primzahlen kleiner oder gleich* n *gilt*

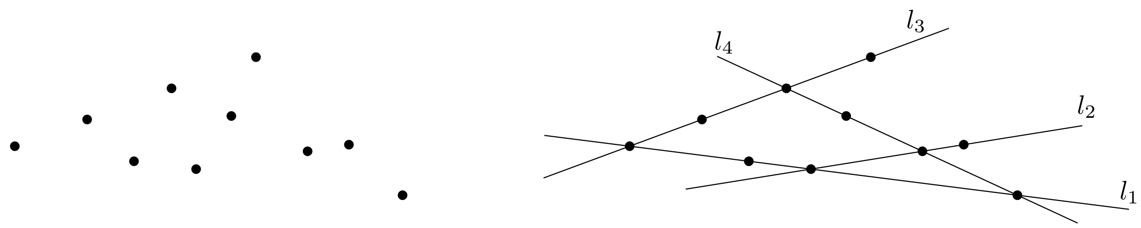
$$\frac{7}{8}\frac{n}{\ln n} \leq \mathfrak{p}(n) \leq \frac{9}{8}\frac{n}{\ln n}$$

Es ist also $\mathfrak{p}(n) \in \Theta(n/\ln n)$.

Satz 2.2. *Die Anzahl* k *verschiedener Primteiler einer Zahl* $m \leq 2^n$ *ist höchstens* n.

Problem 3: Line-Shooting Problem [vgl. Kapitel 10 im Skript] ***

Gegeben sind n Punkte $P \subset \mathbb{R}^2$ ($|P| < \infty$) in der Ebene mit $p_i := (x_i, y_i)$ für alle $p_i \in P$ sowie eine Zahl $k \in \mathbb{N}$. Gefragt ist nun ob es eine Menge L von k Geraden gibt so, dass die Geraden $l_j \in L$ alle Punkte $p_i \in P$ überdecken – das heißt für jeden Punkt $p_i \in P$ eine Gerade $l_j \in L$ existiert so, dass $p_i \in l_j$. Eine Instanz des Line-Shooting Problems ist dann durch $\mathcal{I} := (P, k)$ gegeben.



(a) Ja-Instanz $\mathcal{I} = (P, 4)$.
(b) Lösung von $\mathcal{I}$ mit 4 Geraden.

Abbildung 1: Beispiel für eine Instanz $\mathcal{I} = (P, k)$ des Line-Shooting Problems mit gegebener Punktemenge P und $k = 4$. Eine zulässige Lösung ist in Abbildung (b) dargestellt.

(a) Beweisen Sie, dass das Line-Shooting Problem FPT bezüglich k ist. Geben Sie dazu einen Algorithmus mit beschränkter Suchbaumtiefe an, der eine Instanz (P, k) exakt löst. Dabei soll die Suchbaumtiefe und der Verzweigungsgrad polynomiell in k, und der Aufwand pro Knoten polynomiell in $|P| =: n$ sein. Analysieren Sie die Laufzeit Ihres Algorithmus im $\mathcal{O}$-Kalkül.

Hinweis: Verwalten Sie in jedem Baumknoten k Bins, die maximal zwei Punkte aufnehmen können (und damit ggf. eine Gerade definieren). Ein Suchbaum der Höhe $\mathcal{O}(k)$ genügt.

(b) Argumentieren Sie, warum bezüglich der Punktemenge P aus Abbildung 1a die Instanz $\mathcal{I} = (P, 3)$ eine Nein-Instanz des Problems ist.

(c) Gegeben sei eine Instanz $\mathcal{I} = (P, k)$ des Line-Shooting Problems mit $n = |P|$. Geben Sie ein Algorithmus zur Kernbildung von $\mathcal{I}$ an, der die Instanz $\mathcal{I}$ auf eine Instanz $\mathcal{I}'$ reduziert, so dass $|\mathcal{I}'|$ polynomiell in k ist. Beweisen Sie die Korrektheit der Transformation und geben Sie die Gesamtlaufzeit zur Lösung von $\mathcal{I}$ mit Ihrem Verfahren an. Ist Ihr Verfahren effizienter, als das aus Aufgabe (a)?