Antworten zum Fragenkatalog

Thema Komplexität WS 07/08

von Marius Kischel und Marco Vreydal

Keine Garantie auf Richtigkeit der Antworten

1) Die Worst-Case Laufzeit eines Algorithmus ist die maximale Laufzeit auf einer Eingabe der Länge n bezüglich des logarithmischen Kostenmaßes auf der RAM
2) P = { P' | P' ist Problem für das ein Polynominalzeitalgorithmus existiert}
3) Eine NTM ist eine TM mit einer Zustandsübergangsrelation anstatt einer Zustandsübergangfunktion. Das heißt, es gibt für eine Konfiguration mehrere Folgekonfiguration. Die NTM rät immer den optimalen Übergang.
   Die NTM akzeptiert ein Wort x, genau dann wenn es mindestens einen Rechenweg gibt, der in eine akzeptierende Endkonfiguration führt.
   Die Laufzeit für den Fall $x \in L$ ist immer der kürzeste Rechenweg. Für $x \notin L$ ist die Laufzeit definiert als 0.
4) Die Komplexitätsklasse NP enthält diejenigen Probleme, die durch eine NTM in polynomieller Zeit lösbar sind.
   Alternative Definition: Ein Problem liegt in der Komplexitätsklasse NP, wenn es einen Verifizierer V gibt, der für ein Zertifikat y mit $|y| \leq p(n)$ für die Eingabe y#x in polynomieller Zeit berechnet, ob y eine Lösung für die Eingabe x ist.
   Beweis: Sei V ein Verifizierer der den oben genannten Anforderungen genügt. M sei eine NTM die L entscheidet. M erzeugt Lösung y in polynomieller Zeit, startet dann V auf y#x und übernimmt die Ausgabe. Wenn $x \in L$ so akzeptiert V =>
   $\exists y \in \{0,1\}^* mit |y| \leq p(n) => x \in L$
   Andere Richtung: Sei M eine NTM die L erkennt. y weise auf, welche Übergänge M gewählt hat. V simuliere und wähle die Übergange an der Stelle i so wie y_i dieses bezeichne. Es gilt $x \in L <=> M\ akzeptiert <=> \exists y \in \{0,1\}^* <=> V\ akzeptiert$ Hier wird davon ausgegangen, dass es immer nur zwei mögliche Übergänge gibt. Wenn es mehr gibt werden diese kodiert. Dies ist jedoch eine maximal konstante Anzahl.
5) Clique:
   Eingabe: Ein Graph G(V,E) und eine Konstante c
   Entscheidungsvariante: Ja, falls es eine Menge K die aus mindestens c Knoten besteht gibt, in welcher jeder mit jedem Knoten durch eine Kante verbunden ist. Sonst nein.
   Optimierungsvariante: Finde eine Menge K mit maximal vielen und mindestens c Knoten.

   KP:
   Eingabe: Eine Menge N, welche n Objekte mit den Gewichtungen (w1, ... , wn), den Profiten (p1, ... , pn), eine Konstante p (nur EV) und ein Gewicht k.
   Entscheidungsvariante: Ja, falls es eine Teilmenge K aus N gibt welche das Gewicht k nicht überschreitet und mindestens einen Profit im Wert von p erzielt. Sonst nein.
   Optimierungsvariante: Finde eine Teilmenge K mit Maximalgewicht k und maximalem Profit.

BPP:
Eingabe: Eine Menge N aus n Objekten mit den Gewichten (w1, ... ,wn) und eine Menge M aus m Objekten mit jeweils einer Kapazität von k. Eine Konstante c (nur EV).
Entscheidungsvariante: Ja, falls N auf c Objekte der Menge M aufgeteilt werden kann. Sonst nein.
Optimierungsvariante: Teile N auf eine minimale Anzahl von Objekten aus M auf.
TSP:
Eingabe: Ein Graph G(V,E) und eine Konstante c (nur EV)
Entscheidungsvariante: Ja, falls es einen Weg durch G gibt, bei dem jeder Knoten genau einmal besucht wird und welcher (unter Berücksichtigung der Kantengewichte) höchstens die Länge c hat. Sonst nein.
Optimierungsvariante: Suche den kürzesten Weg (unter Berücksichtigung der Kantengewichte) durch G, welcher alle Knoten genau einmal besucht.

6) Richtung pol. EV -> pol. OV

Clique:
Führe Binärsuche im Bereich (1, ... ,|V|) durch um das maximale c zu ermitteln. Lösche sukzessive alle Knoten und frage nach, ob noch immer eine c-Clique existiert. Falls ja, fahre mit dem nächsten Knoten fort. Falls nein, füge den Knoten wieder hinzu und fahre mit dem nächsten Knoten fort.
O(|V| * p(n)) + O(log(n) * p(n))

KP:
Führe Binärsuche durch um das Maximum an Profit zu ermitteln. Entferne sukzessive jedes Element und frage nach, ob die Optimale Lösung noch existiert. Falls ja, so verwirf das Element. Falls nein, so packe es in den Sack.
O(n* p(n)) + O(log(n) * p(n))

BPP:
Führe Binärsuche durch um das Minimum an Transportmitteln zu finden. Verschmelze paarweise jedes mit jedem Element und überprüfe ob das Minimum noch existiert. Falls ja so fahre fort. Falls nein so löse die Elemente wieder voneinander.
O(log(n) * p(n)) + O(n^3)

TSP:
Führe Binärsuche durch um den kürzesten Weg durch den Graphen zu ermitteln. Setze sukzessive alle Kanten auf unendlich und überprüfe ob der kürzeste Weg noch vorhanden ist. Falls ja, so gehe zur nächsten Kante über. Falls nein, so setze das Gewicht der Kante wieder zurück.
O(log(n) * p(n)) + O(|V| * p(n))

Rückrichtung trivial. Bestimmt optimales Ergebnis und vergleiche mit Eingabe für Entscheidungsvariante.

7) Sei die polynomielle Laufzeit der gegebenen NTM p(n), so geht die TM, welche die NTM simuliert alle Möglichkeiten der Zustandsübergänge durch, da sie nicht die korrekte Lösung erraten kann. Man geht also einen Weg durch und ruft einen Polynomialzeitverifizierer V auf, der Laufzeit q(n) hat. Wenn V akzeptiert, akzeptiert die TM, sonst wird die nächstmögliche Zustandsübergangsfunktionskombinationsfolge simuliert.
$|\Sigma \cup \Gamma x Q x \{L, N, R\}| = c$ ist die maximale Anzahl an Verzweigungen pro Berechnungsschritt. Im Worst-Case müssen diese alle simuliert werden. Damit ergibt sich als Worstcase Laufzeit O($c^{p(n)} q(n)$)

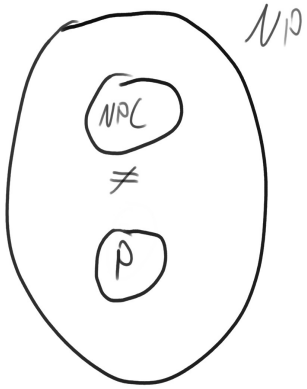
8) $\leq und \leq_p$ unterscheiden sich dadurch, dass $\leq_p$ eine Reduktion ist, deren Funktion in polynomieller Zeit berechenbar ist, während die Laufzeit der Funktion einer normalen Reduktion keine Rolle spielt.

9) Ein Problem P ist NP-Hart, wenn sich jedes Problem aus NP auf P reduzieren lässt. P wird als NP-Vollständig bezeichnet, falls zusätzlich $P \in NP$ gilt.

10) Um zu zeigen dass ein Problem NP-vollständig ist, müssen wir zeigen, dass es in NP liegt und dass sich alle Probleme in NP auf dieses Problem reduzieren lassen. Der erste Schritt kann dadurch erreicht werden, dass ein Zertifikat für SAT eine gültige Belegung der aussagenlogischen Formel ist.
Der zweite Teil besteht darin, eine Funktion zu finden, die eine NTM (für ein beliebiges anderes Problem in NP) in eine ausssagenlogische Formel umformt, die nur dann erfüllt werden kann, wenn die NTM in einen akzeptieren Entzustand gelangen kann. Dies geschieht dadurch, dass die gesamte NTM in einer aussagenlogischen Formel kodiert wird, in der Zustand, Kopfposition und Zeichen an jeder Kopfposition zum Zeitpunkt t kodiert wird. Diese werden dann so zueinander in Verbindung gebracht, dass die Formel nur erfüllt sein kann, wenn die üblichen Anforderungen an eine NTM erfüllt werden (nur ein Zustand aktiv zu jedem Zeitpunkt, nur eine Kopfposition, an jeder Bandposition steht nur ein Zeichen). Dann werden gültige Konfigurationen in eine aussagenlogische Form gebracht. Dann wird die Zustandsübergangsrelation kodiert, so dass nur die erreichbaren Konfigurationen erfüllbar sind. Dies wird dann alles zusammengeführt mit einer Überprüfung, dass Startzustand und Startkopfposition stimmen und die Turingmaschine am Ende (zum Zeitpunkt p(n)) in einem akzeptierenden Zustand ist. (Es wird vereinfachend davon ausgegangen, dass die NTM nach dem akzeptieren in eine Endlosschleife fällt). Es reicht aus p(n) Zeitpunkte zu betrachten, da eine polynomiell beschränkte NTM nur polynomiell viele Rechenschritte ausführen kann.

11) ---

12) NP ist wie ein Ei mit zwei Eidottern. Dabei ist der eine Dotter P und der andere NPC. Anders gesagt NP umfasst NPC und P. Alle Probleme in NP lassen sich auf NPC Probleme reduzieren, d.h. intuitiv, dass die NPC Probleme die schwierigsten Probleme in NP sind.



13) Wird die Eingabe eines NP-harten Problems P in unärem Code eingegeben und das Problem ist anschließend in polyinomieller Zeit lösbar, so spricht man von einem schwach NP-harten Problem. Existiert allerdings kein polynimieller Laufzeitalgorithmus so ist das Problem stark NP-hart.

14) Die Laufzeit hängt nicht mehr von der Kodierungslänge ab, sondern von der Größe der Eingabezahl.

15) Ein Approximationsschema heißt FPTAS, falls seine Laufzeit polynomiell sowohl von n als auch von $\frac{1}{\epsilon}$ abhängt. Dabei ist n die Kodierungslänge. 1 - ϵ bezeichnet den Optimierungsfaktor bei Maximierungsproblemen und 1 + ϵ bezeichnet diesen bei Minimierungsproblemen.
Falls die Laufzeit nur von n abhängt (für jedes konstante $\epsilon > 0$) heißt das Approximationsschema PTAS.

16) Das Problem wird durch eine Rekursion gelöst, welche das Problem in $O(n^2 * P)$ lösen kann. Dabei ist P der maximale Einzelnutzen. Der FPTAS skaliert dabei die Nutzenwerte so, dass sie alle aus natürlichen Zahlen bestehen. Der Skalierungsfaktor α ergibt sich dabei durch die Gleichung $\alpha = \frac{N}{\epsilon * P}$. Durch die skalierten Nutzenwerte ist P nun durch $\alpha * P = \frac{N}{\epsilon}$ beschränkt.

17) Die Approximationsgüte ist ein Wert der angibt wie nah die approximierte Lösung an der optimalen Lösung liegt.