Institut für Informatik
Prof. Dr. Heiko Röglin
Tobias Brunsch
Clemens Rösner

Algorithmen und Berechnungskomplexität I
Wintersemester 2013/14
Abgabe: 19.11.2013, 12:30 Uhr

Übungsblatt 5

Aufgabe 5.1

2+4 Punkte

(a) Wir betrachten die folgende Instanz des Rucksackproblems mit 4 Objekten.
Es seien $p_1 = 3$, $p_2 = 1$, $p_3 = 2$, $p_4 = 5$, $w_1 = 2$, $w_2 = 4$, $w_3 = 3$ und $w_4 = 6$. Stellen Sie die Tabelle mit den Werten $P(i, w)$, die durch den Algorithmus DynKP berechnet werden, auf und geben Sie eine optimale Lösung für $t = 10$ und $t = 7$ an.

(b) Wir haben gesehen, wie wir mit DynKP optimale Lösungen für das Rucksackproblems mit einer Laufzeit von $O(n^2 \cdot W)$ berechnen können, wobei n die Anzahl der Objekte und $W = \max_{i \in \{1,\dots,n\}} w_i$ das maximale Gewicht eines Objektes bezeichnet. Zeigen Sie, wie eine optimale Lösung des Rucksackproblems mit einer Laufzeit von $O(n^2 \cdot P)$ berechnet werden kann, wobei $P = \max_{i \in \{1,\dots,n\}} p_i$ der größte Nutzen eines Objektes ist.

Aufgabe 5.2

6 Punkte

Wir betrachten folgendes Problem, das uns auch in der Vorlesung schon einmal begegnet ist. Ein Kassierer möchte Wechselgeld mit möglichst wenigen Münzen zurückzahlen. Die zur Verfügung stehenden Münzsorten seien gegeben durch $C = (c_1, \dots, c_m)$ mit $1 = c_1 < c_2 < \dots < c_m$. Wir nehmen an, dass von jeder Münzsorte hinreichend viele Münzen vorhanden sind. Somit können wir das Problem für einen gegebenen Wechselgeldbetrag $W \in \mathbb{N}$ formulieren als

$$\begin{aligned} \textbf{Minimiere } & \alpha_1 + \dots + \alpha_m \\ \textbf{Nebenbedingungen: } & \alpha_1 c_1 + \dots + \alpha_m c_m = W, \\ & \alpha_1, \dots, \alpha_m \in \mathbb{N}_0 = \{0, 1, 2, \dots\}. \end{aligned}$$

Für dieses Problem haben wir in der Vorlesung bereits gesehen, dass je nachdem wie C und W aussehen der Greedyalgorithmus nicht zwangsweise die optimale Lösung liefert.

Geben Sie einen Algorithmus an, der ein Münzsystem $C = (c_1, \dots, c_m)$ und einen Wechselgeldbetrag W als Eingabe erhält und in Zeit $O(m \cdot W)$ eine optimale Lösung berechnet.

Aufgabe 5.3

6 Punkte

Wir betrachten das Partitionsproblem. Beim Partitionsproblem geht es darum, für eine gegebene Multimenge $A = \{a_1, a_2, \dots, a_n\}$ von natürlichen Zahlen zu entscheiden, ob eine Partition $A_1 \cup A_2 = A$ existiert, sodass die Summe der Elemente aus A_1 gleich der Summe der Elemente aus A_2 ist, und gegebenenfalls eine solche Partition $A_1 \cup A_2$ zu finden. Da aus einer Bipartition $A_1 \cup A_2$ von A folgt, dass wir A_2 auch als $A \setminus A_1$ schreiben können, reicht es auch zu bestimmen, ob eine Teilmenge A_1 von A existiert, für die $\sum_{a_i \in A_1} a_i = \frac{S}{2}$ gilt, wobei wir S durch $\sum_{i=1}^{n} a_i$ definieren.

Benutzen sie dynamische Programmierung um das Partitionsproblem mit einer Laufzeit von $O(n \cdot S)$ zu lösen.

Aufgabe 5.4

6 Punkte

Wir wollen die beiden Algorithmen DynKP und ApproxKP auf einer Menge mit n Objekten vergleichen. In der Vorlesung haben wir bereits gesehen, dass DynKP eine optimale Lösung liefert und ApproxKP eine Lösung liefert, die mindestens halb so gut ist, wie die optimale Lösung. Wir wollen nun untersuchen, wie gut die von ApproxKP gelieferten Lösungen auf zufälligen Instanzen sind.

Erzeugen Sie für $n = 2, \ldots, 200$ jeweils 100-mal hintereinander je ein Feld p mit n zufälligen Einträgen für die Nutzenwerte und ein anderes Feld w mit n zufälligen Einträgen für die Gewichte. Dabei sei jeder Eintrag eine zufällige natürliche Zahl kleiner gleich 1000. Führen Sie für eine zufällig gewählte Kapazität $t \in \{\max_{i\in\{1,\ldots,n\}} w_i, \ldots, \sum_{i\in\{1,\ldots,n\}} w_i\}$ sowohl DynKP als auch ApproxKP auf diesen Feldern aus. Bestimmen Sie für jedes n die minimale, die maximale und die durchschnittliche Approximationsgüte der Lösung von ApproxKP und stellen Sie diese als Graphen in Abhängigkeit von n dar. Die Approximationsgüte ist der Quotient aus dem optimalen Nutzen und dem Nutzen der Lösung, die ApproxKP berechnet.

Hinweis: Zur Darstellung eignet sich zum Beispiel *gnuplot* (siehe *Hinweise zu gnuplot* auf der Homepage).

Aufgabe 5.5

6 Zusatzpunkte

Wir betrachten das folgende Problem. Gegeben ist ein Bild B bestehend aus einer Menge von Zeilen. Jede Zeile ist dabei eine Zeichenfolge, die aus den Buchstaben R und G besteht. Jedes Zeichen entspricht einem Feld in dem Bild, das entweder grün oder rot gefärbt werden soll, je nachdem, welcher Buchstabe an der entsprechenden Stelle steht. Mit der Farbrolle können wir nur horizontale Streifen mit Höhe 1 färben. Zusätzlich kann zu jeder Zeit immer nur eine Farbe benutzt werden und ein einmal gefärbtes Feld darf nicht mit einer anderen Farbe überstrichen werden.

Das folgende Bild B kann zum Beispiel mit 6 Streifen korrekt gefärbt werden:

$B = \{$“*RRRRRRRRRRRRRR*”,
“*GGGGGGGGGGGGGG*”,
“*GGGGGGGGGGGGGG*”,
“*GGGGGRRRRRGGGGG*”$\}$

Wir benötigen je einen Streifen für die ersten 3 Zeilen und 3 Streifen für die letzte Zeile.

Zusätzlich zu dem Bild B enthält eine Instanz noch eine natürliche Zahl n, die angibt, wie viele verschiedene Streifen wir maximal färben dürfen. Ist n zu klein, so können wir nicht bei jedem Bild alle Felder richtig färben. In diesem Fall wollen wir möglichst viele Felder richtig färben. Ob wir ein Feld nicht färben oder falsch färben, macht für uns dabei keinen Unterschied.

Sind in dem obigen Beispiel beispielsweise nur $n = 5$ Streifen erlaubt, so ist es optimal die ersten drei Zeilen mit je einem Streifen komplett korrekt zu färben. In der letzten Zeile können dann mit den beiden noch erlaubten Streifen nur 10 der 15 Felder korrekt gefärbt werden. Sind nur $n = 3$ Streifen erlaubt, so ist es optimal, nur die ersten drei Zeilen komplett korrekt zu färben.

Geben Sie einen möglichst effizienten Algorithmus an, der mithilfe von dynamischer Programmierung für eine beliebige Instanz berechnet, welche Streifen wir in welcher Farbe färben sollten um möglichst wenige Felder falsch zu färben.