Max Wisniewski , Alexander Steen

Tutor: Lena Schlipf

Aufgabe 1 Varianten der Vorlesungsbeispiele

(a) Betrachten Sie folgende Variation des Einkaufsproblems:
Zu einer Ware i existiert nun abgesehen vom Preis p_i und einem Wert w_i nun auch noch eine Häufigkeit h_i. Weiterhin haben wir ein Budget B und wollen als Ziel nun eine *Multmenge* von Artikeln finden, das bei B die Summe der Werte maximiert.
Zeigen Sie, dass diese Variante auch mit Laufzeit $O(nB)$ läuft.

Lösung:

Der Algorithmus ist in seiner Rekursionsgleichung davon ausgegangen, dass wir nach dem Kauf eines Artikels diesen nicht nocheinmal betrachten wollen und ist Aufgrund dieser Annahmen in jedem Schritt eine Zeile in der Tabelle nach oben gegangen (pro Zeilen hatten wir einen Artikel). Streichen wir das unbedingte nach oben gehen, erhalten wir die folgende Rekursionsvorschrift (Die Bezeichner sind, wie in der Vorlesung gewählt):

$$\begin{array}{lcrcl} \forall b \leq B & : & E[0,b] & = & 0 \\ \forall 0 \leq m \leq n & : & E[m,0] & = & 0 \\ \forall b < p[m] & : & E[m,b] & = & E[m-1,b] \\ \forall b \geq p[m] & : & E[m,b] & = & \max\{E[m-1,b], E[m,b-p[m]+w[m]\} \end{array}$$

Der Unterschied hier besteht nur in der letzten Zeile. Wenn wir einen Artikel genommen haben, so können wir uns dafür entscheiden ihn erneut zu nehmen.

Was man an dieser Stelle nun gut sehen kann, ist dass wir nur auf ein anderes Feld im Array zugreifen ($E[m, b-p[m]]$ statt $E[m-1, b-p[m]]$). Der Algorithmus wird sich auch nur in diesem Zugriff unterscheiden.

Wir haben also wider nur $\Theta(n \cdot B)$ Zellen (Platzverbrauch) und pro Zelle müssen wir 2 Zahlen vergleichen und einen Wert addieren.
Dies macht eine Laufzeit von $\Theta(n \cdot B)$

(b) In der Vorlesung haben Sie gesehen, wie das Rundreiseproblem mit Hilfe von dynamischen Programieren gelöst werden kann. Arbeiten Sie die Details des Algorithmus aus und geben Sie Pseudecode an, um eine optimale Tour zu berechnen.

Lösung:

Aus der Vorlesung haben wir eine Menge $S \subseteq \{1, .., n-1\}$ Teilmenge von $m \in \{0, ..., n-1\} \setminus S$. $T[S, m]$ sind nun die kosten des optimalsten Weges, der in 0

anfängt, alle Städte aus S besucht und in m endet. Die Rekursion, mit der wir das Ergebnis finden wollen ist:

$$\begin{array}{rcccl} \forall m & : & T[\emptyset, m] & = & d(0, m) \\ & & T[S, m] & = & \min\limits_{a \in S} T[S \setminus \{a\}, a] + d(a, m) \end{array}$$

Unsere erste Überlegung um den Algorithmus aufzuschreiben ist, dass wir alle Teilmenge von $\{1, ..., n-1\}$ aufstellen müssen. Bei jeder dieser Teilfolgen interessiert uns nur, ob das Element drin ist oder nicht. Wir verwenden also eine Bitmap, die sich am günstigesten über die Zahl 2^{n-1} darstellen. (Die 0 ist nicht Teil unserer Menge, dies hat zur Folge, dass wir im Algorithmus die Elemente immer um eins versetzt angucken müssen).

```
for m := 0 to n − 1 do
  T[0,m] := 0;
for m := 0 to n − 1 do
  for s := 1 to 2^{n-1} do
    minValue := ∞;
    for a:= 0 to n − 1 do
      if s & 1 << (a-1) then
        minValue := min (minValue, T[s & ~(1 << (a-1)), a] +
            d(a,m));
    T[s, m] = minValue;

return T[2^{n-1}, 0];
```

Platzbedarf: Wir haben ein Feld der Dimension $n \times 2^{n-1}$, dies kostet uns also $\Theta(n \cdot 2^n)$ Platz.
Laufzeit: Pro Feld, müssen wir alle unsere Städte durchgehen und gucken, ob das Bit in unserer Zahl gesetzt ist. Das testen kostet uns konstante Laufzeit, das minimum bestimmen auch. Sollte die Bestimung der Distanz auch konstant sein, so haben wir pro Feld eine Laufzeit von $\Theta(n)$.
Insgesammt haben wir (was man auch an den 3 for Schleifen sehen könnte) eine Laufzeit von $\Theta(n^2 \cdot 2^n)$

Aufgabe 2 Münzwechseln

(a) Entwerfen Sie einen Algorithmus, der berechnet, auf weiviele Arten ein Euro (und allgemeiner n *Cent*) mit beliebig viel Münzen ≤ 1 Euromünze gewechselt werden kann. Die Lösung soll dynamische Programmierung verwenden und für beliebige Wärungen funktionieren.

Lösung:

(b) Analysieren Sie die Laufzeit Ihres Algorithmus.

Lösung:

(c) Implementieren Sie Ihren Algorithmus in *Java* und wenden sie ihn auf das Problem in *a)* an, sowie 1 $ in 1-, 5-, 10- und 25- Centstücke an.

Lösung:

Aufgabe 3 Versteckte Markov-Modelle

(a) Betrachten Sie das folgende Markov-Modell.

- Zustände: $Q = \{q, r, s\}$
- Alphabet: $\Sigma = \{a, b\}$
- Anfangsverteilung: $\{q : 0.1, r : 0.4, s : 0.5\}$
- Ausgabeverteilung für q $\{a : 0.2, b : 0.8\}$
- Ausgabeverteilung für r $\{a : 0.7, b : 0.3\}$
- Ausgabeverteilung für s $\{a : 0.5, b : 0.5\}$
- Übergangsverteilung für q: $\{q : 0.8, r : 0.1, s : 0.1\}$
- Übergangsverteilung für r: $\{q : 0.3, r : 0.3, s : 0.4\}$
- Übergangsverteilung für s: $\{q : 0.2, r : 0.4, s : 0.4\}$
-

Benutzen Sie den Viterbi-Algorithmus, um die wahrscheinlichste Erklärung für die Ausgabefolge *abba* zu ermitteln.

Lösung:

(b) Bei einer Implementierung des Viterbi-Algorithmus rechnet man oft mit den Werten $\log p_i$ statt den Wahrscheinlichkeiten p_i. Erklären Sie, warum das eine gute Idee ist.

Lösung:

(c) Eine Anwenung von versteckten Markov-Modellen ist die Fehlerkorrektur. Beschreiben Sie, wie man mit einem *versteckten Markov-Modell* eine Übertragung eines Satzes deutscher Sprache über eine Leitung die zu 10

Lösung: