Karlsruher Institut für Technologie

Tutorium 12

Algorithmen I SS 14

Institut für Theoretische Informatik

Greedy Algorithmen

- Global optimale Lösungen bestehen oft aus lokal optimalen Lösungen
- Wähle für die *aktuelle Situation* die optimale Entscheidung.
- Keine Reversion von vergangenen Entscheidungen
- $\Rightarrow$ Einfach und schnell

Beispiele

Greedy Algorithmen die optimale Ergebnisse liefern:

- Dijkstra für kürzeste Wege
- Beide MST Algorithmen
  - Kruskal
  - Jarnik-Prim
- Selection Sort

Dynamische Programmierung

Dynamische Programmierung

Karlsruher Institut für Technologie

Definition

Dynamische Programmierung ist ein algorithmisches Muster, das vor allem zur Lösung von Optimierungproblemen eingesetzt wird.

Idee:

- Problem in Teilprobleme zerlegen
- Lösungen der Teilprobleme speichern
- Endlösung aus Lösungen der Teilproblemen zusammensetzten

Dynamische Programmierung: Zwischenspeichern der Ergebnisse

- in einem ein- oder mehrdimensionalen Array
- Basisfälle vorinitialisieren
- Rest der Tabelle wird mit Werten für 'nicht berechnet' zB. -1 initialisiert
- für jeden Funktionsaufruf Table-Lookup:
  - steht ein Wert in der Tabelle: zurückgeben
  - sonst: Wert berechnen, in der Tabelle speichern und zurückgeben

Top-down

- 'Vom Problem zur Lösung'
- rekursiver Funktionsaufruf
- overhead durch viele Funktionsaufrufe
- es kann bei extrem großen Problemen zu stackoverflow kommen

Besser: Bottom-Up

- die Tabelle wird sukzessiv mit Lösungen gefüllt
- das Endergebniss wird danach einfach aus der Tabelle abgelesen

Levenshtein-Distance

Die sog. *Levenshtein Distanz* gibt eine Editierdistanz, also eine Art Abstand, zwischen zwei Wörtern an. Sie ist definiert als die minimale Anzahl an Einfüge-, Lösch- und Ersetzungsoperationen, um ein Wort in das andere umzuwandeln. Lösung mit dynamischer Programmierung!

$$lev_{a,b}(i,j) = min \begin{cases} lev_{a,b}(i-1,j)+1 & \textit{(a einfügen/b löschen)} \\ lev_{a,b}(i,j-1)+1 & \textit{(a löschen/b einfügen)} \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} & \textit{(ersetzen)} \end{cases}$$

Palindrome

Palindrome sind Wörter, die rückwärts gleich wie vorwärts sind (z.B. „anna“, „kajak“, „reittier“).
Aufgabe 1: Entwickle einen Algorithmus, der für eine Zeichenkette der Länge n in $\mathcal{O}(n^2)$ Zeit die Länge der längsten palindromischen Teilsequenz.
Beispiel: „FA**K**UL**T**Ä**TSF**E**ST** INFORMA**TIK** 2014“
Aufgabe 2: Erweitere den Algorithmus, sodass er zusätzlich diese Sequenz ausgibt.

Miners

Zwei Kohleminen müssen mit Essen versorgt werden. Es gibt drei verschiedene Arten von Essen: Fleisch, Fisch und Brot. Die Arbeiter mögen Abwechslung und sind produktiver, wenn sie verschiedene Arten von Essen bekommen. Für die nächste Produktion entscheidend sind immer die letzten drei Sendungen.

- Wenn alle Sendungen gleich sind $\Rightarrow$ 1 Kohleeinheit
- Wenn es zwei verschiedene Sendungen gab $\Rightarrow$ 2 Kohleeinheiten
- Wenn alle verschieden waren $\Rightarrow$ 3 Kohleeinheiten

Für eine gegebene Reihenfolge an Essenssendungen kann jeweils entschieden werden, welche Mine sie erhält.
Aufgabe: Berechne die maximale Anzahl an produzierten Kohleeinheiten.