HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN · FH MÜNCHEN

Algorithmen und Datenstrukturen (IB) Praktikum
Aufgabenblatt 7 – Rekursion

Ausgabe: 3. Mai 2018
Abgabe: 15. Mai 2018, 18:00 Uhr

Allgemeines

Sie finden im Moodle ein Projekt, in dem für die folgenden Aufgaben jeweils Methodenrümpfe und Tests vorgegeben sind.

1 Rekursion und Iteration (Pflicht)

Wortpalindrome sind Wörter, die sowohl vorwärts als auch rückwärts gelesen aus derselben Zeichenfolge bestehen. Beispiele sind Regallager, Ada oder Retsinakanister[1]. Implementieren Sie in der Klasse `hm.edu.cs.algdat.rekursion.Rekursion` die Methoden `isPalindromeIterative` und `isPalindromeRekursiv` jeweils so, dass Sie `true` genau dann zurückliefern, wenn das übergebene Wort ein Palindrom ist. Dabei soll `isPalindromeIterative` ohne Rekursion auskommen (iterative Implementierung mit einer `for`-Schleife), und `isPalindromeRekursiv` rekursiv implementiert werden, d.h. ohne Schleifen auskommen.

Hinweise:

- Verwenden Sie die Methode `String.toUpperCase()`, um das Palindrome ohne Berücksichtigung von Groß- und Kleinschreibung zu erkennen,
- greifen Sie über `String.charAt(..)` auf einzelne Zeichen des Strings zu und
- (optional, geht auch anders) nutzen Sie `String.substring()`, um Teilstrings zu extrahieren.

2 Dynamische Programmierung (Pflicht)

Eine Neuregelung der Öffnungszeiten stellt es Gemeinden frei, an beliebig vielen Sonntagen die Geschäfte zu öffnen, unter der Voraussetzung, dass zwischen zwei verkaufsoffenen Sonntagen immer mindestens 3 nicht verkaufsoffene Sonntage liegen. Die Gemeinde Entenhausen will die neue Regelung bestmöglich nutzen und ermittelt für alle 52 Sonntage des Jahres die erwarteten Umsätze (in 10.000 Talern):

20,18,26,16,7,6,18,21,17,14,13,21,17,18,5,25,28,20,9,22,28,29,15,28,10,8,6,21,5,11,28,16,
5,22,21,5,12,25,27,22,20,11,16,15,11,24,22,18,29,13,15,23

Nun ist man ratlos, wie man auf dieser Basis freie Sonntage auswählt. Helfen Sie den Entenhausenern!

[1] Satzpalindrome wie O, Genie, der Herr ehre dein Ego! betrachten wir nicht.

2.1 Rekursive Formulierung

Schreiben Sie zunächst in der Klasse `DynamicProgramming` in den Methoden `maxSumWithDistanceRecursive(..)` eine rekursive Lösung für das Problem, aus Zahlen in einem Array die größte Summe zu bilden, in der nur Zahlen mit einem Mindestabstand vorkommen! Die Hilfsmethode mit dem zusätzlichen Parameter `index` sollte dabei die jeweils optimale Lösung bis zur Stelle `index` zurückgeben.

Zum Beispiel sollte für `array={2,4,1,1,5,6,2}` und `distance=3` die 4 und die 6 ausgewählt werden.

Hinweis: Verwenden Sie in Ihrer Fallunterscheidung die Fälle index $=0$, $0<$ index $<$ distance und distance $\leq$ index.

Welche Umsätze können die Entenhausener erreichen? (Das Array mit den Werten ist als Konstante in der Vorgabe enthalten?)

2.2 Dynamische Programmierung

Schreiben Sie die Rekursion um in eine dynamische Programmierung. Nutzen Sie dafür die Methode `maxSumWithDistanceDP(..)` Verwenden Sie ein Array `sum`, um die Werte zu speichern!

3 Rekurrenz (optional)

Sie haben für die Laufzeit eines rekursiven Algorithmus die folgende Rekurrenzbeziehung ermittelt:

$$T(n)=\begin{cases}O(1) & \text{für} n=1\\ 4T(n/2)+O(n^2) & \text{sonst}\end{cases}$$

Ermitteln Sie mithilfe der Master-Methode eine Lösung und nutzen Sie die Substitutionsmethode, um die Lösung zu validieren. Lassen Sie bei der Substitutionsmethode den Induktionsanker weg oder verwenden Sie $n_0=2$!