Fuelle zwei / viele Container

Lösung zum Problem
des Füllens zweier oder mehr Container
mit Tiefensuche

Fuelle zwei / viele Container

- Das Ziel bei der Behandlung des Füllens mehrerer Container ist die Betrachtung, wie sich der Suchaufwand dabei entwickelt. (→ Berechnungsaufwand)
- Hierzu sollte eine Möglichkeit eingebaut werden, die Laufzeit jeweils messen zu können (→ *laufzeit.scm*)

Fuelle zwei Container

- Ob man sich vor der Behandlung des Füllens mehrerer Container speziell mit dem Füllen von zwei Containern beschäftigen sollte, ist fraglich.
- Der Vorteil kann sein, dass die Schülerinnen und Schüler vermutlich leichter
  - den Schritt von einem Container zu zwei Containern hin verstehen
  - und dann vermutlich bei der Frage nach dem Vorgehen bei mehr Containern eine Lösung mit Containerlisten vorschlagen.

Fuelle zwei Container

- Die wesentlichen Ergänzungen bei zwei Containern gegenüber *fuelle-ts.scm*:

```
(define (fuelle stuecke container-1 container-2)
...
  ((fuelle          ; ersten versuchen
    (rest stuecke)
    (fuelle-ein (first stuecke) container-1)
    container-2))
  ((fuelle          ; alternativ zweiten versuchen
    (rest stuecke)
    container-1
    (fuelle-ein (first stuecke) container-2)))
  (else
```

Fuelle zwei Container

- Da in den jeweiligen Fällen entweder der gefüllte Container oder im Misserfolgsfall der Wert #f zurückgegeben wird, muss dieser Fall gesondert behandelt werden:

```
(cond
 ((not container-1) #f)
  ; Rueckgabe #f von fuelle-ein verarbeiten
 ((not container-2) #f)
  ; Rueckgabe #f von fuelle-ein verarbeiten
```

Fuelle viele Container

- Bei mehr Containern muss mit einer Containerliste gearbeitet werden.
- Da für die jeweils bearbeiteten Container ein Akku benötigt wird, bietet sich eine innere Funktion an:

```
(define
  (fuelle stuecke container-liste)
  (define
    (tiefensuche stuecke c-liste-1 c-liste-2)
  …
  (tiefensuche stuecke container-liste '())
 )
```

Fuelle viele Container

- In die zweite Liste werden die erfolglos versuchten Container geschoben.

```
((zu-voll? (first stuecke) (first c-liste-1))
(tiefensuche
 stuecke
 (rest c-liste-1)
 (cons (first c-liste-1) c-liste-2)))
```

Fuelle viele Container

- Der Schritt in die Tiefe wird mit dem aktuellen Stück im aktuellen Container versucht, zusammen mit den anderen unbearbeiteten und den bearbeiteten als neue Ausgangsliste

```
((tiefensuche
  (rest stuecke)
  (append
  (cons
   (fuelle-ein (first stuecke) (first c-liste-1))
   (rest c-liste-1))
  c-liste-2)
  '()))
```

Fuelle viele Container

- Ist dieser Schritt in die Tiefe nicht erfolgreich wird (*else*) der unbearbeitete Container für die weitere Bearbeitung (*des nächsten Containers*) in die zweite Liste übernommen.

```
(else       ; versuchen
(tiefensuche
 stuecke
 (rest c-liste-1)
 (cons (first c-liste-1) c-liste-2)))
```

Fuelle viele Container

- Abbruchfälle sind
  - Misserfolgsfall → #f
  - und der Erfolgsfall, dass alle Container gefüllt sind.