Fuelle zwei / viele Container

Lösung zum Problem
des Füllens zweier oder mehrerer 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.py*:

```
def fuelle(stuecke, container_1, container_2):
    ...
    # Zunächst mit dem Stück im ersten versuchen - wenn es geht
    if not(wird_zu_voll(stuecke[0], container_1)):
        ergebnis=fuelle(stuecke[1:],
                        fuelle_ein(stuecke[0],container_1),
                        container_2)
        if ergebnis: return ergebnis
    # Dann mit dem Stück im zweiten versuchen - wenn es geht
    ...
```

Fuelle zwei Container

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

```
def fuelle(stuecke, container_1, container_2):
  ...
  # Dann mit dem Stück im zweiten versuchen - wenn es geht
  if not(wird_zu_voll(stuecke[0], container_2)):
    ergebnis=fuelle(stuecke[1:],
                    container_1,
                    fuelle_ein(stuecke[0],container_2))
    if ergebnis: return ergebnis
  # In jedem Fall auch ohne das Stück versuchen:
 ...
```

Fuelle zwei Container

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

```
def fuelle(stuecke, container_1, container_2):
    ...

    # In jedem Fall auch ohne das Stück versuchen:
    ergebnis=fuelle(stuecke[1:], container_1, container_2)
    if ergebnis: return ergebnis
    return False
```

Fuelle zwei Container

- Am Beginn des Verzweigungsteils muss der Erfolgsfall (*beide gefüllt*) getestet und behandelt werden und im Misserfolgsfall der Wert **`False`** zurückgegeben werden:

```
if ist_voll(container_1) & ist_voll(container_2):
   return container_1, container_2
if len(stuecke)==0:
   return False
```

Fuelle viele Container

Und wie erweitern wir
auf mehr als zwei Container ?

Fuelle viele Container

- Bei mehr Containern muss mit einer Containerliste gearbeitet werden.
- Der Erfolgsfall wird in einer Hilfsfunktion getestet.

```
def fuelle(stuecke, alleContainer):
  if alle_voll(alleContainer):
     return alleContainer
  if len(stuecke)==0:
     return False
```

Fuelle viele Container

- Rekursive Variante: Die Bearbeitung wird einer Funktion übergeben, welche die Containerliste rekursiv durchtestet.
- 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
- Ist dieser Schritt in die Tiefe nicht erfolgreich wird der unbearbeitete Container für die weitere Bearbeitung (*des nächsten Containers*) übernommen.

Fuelle viele Container

- Eine iterative Behandlung dafür:

```
# Zunächst mit dem Stück in allen versuchen - wenn es geht
for i in range(len(alleContainer)):
   if not(wird_zu_voll(stuecke[0], alleContainer[i])):
      ergebnis=fuelle(stuecke[1:],
                  alleContainer[:i]+ # alle davor
                  [fuelle_ein(stuecke[0],alleContainer[i])]+
                  alleContainer[i+1:]) # alle danach
      if ergebnis: return ergebnis
# In jedem Fall auch ohne das Stück versuchen:
ergebnis=fuelle(stuecke[1:],alleContainer)
if ergebnis: return ergebnis
return False
```