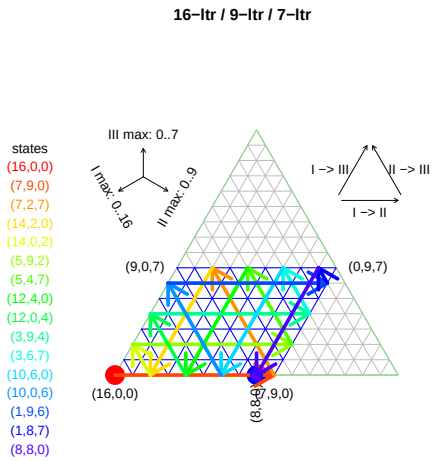
Weinaufteilung per Billard

File: dist-vino.rev
in: /home/wiwi/pwolf/lehre/aud/start

10. Februar 2011, Peter Wolf



Inhalt

1 Einleitung

Rätselecken bieten uns immer wieder Umfüllprobleme, die sich auf das gleichmäßige Aufteilen einer Weinmenge beziehen. Zur Illustration wollen wir ein paar Beispiele aus dem Internet zitieren:

- *Stirb langsam - Rätsel*

  Gestern lief mal wieder Stirb langsam (ich glaube Teil 3). In diesem Film müssen Bruce Willis und Samuel L. Jackson in der Gegend rumlaufen und einige Rätsel lösen, um damit Bombendetonationen zu verhindern. Dabei gibt es ein Rätsel, bei dem ich nie die Lösung peile, weil die Herren Willis und Jackson diese angesichts ihrer Freude über die Verhinderung der Explosion doch nur recht undeutlich herausbrüllen. Gestern habe ich zudem noch eine ganze Weile darüber nachgedacht und bin ebenfalls zu keiner Lösung gekommen. Eigentlich würde ich sogar behaupten wollen, dass noch ein entscheidender Bestandteil fehlt, um das Rätsel überhaupt lösen zu können. Andererseits gehe ich mal davon aus, dass man in einem solchen Film keinen Fehler bei so einem Rätsel einbaut und es eine Lösung gibt. Wer anderes als dieses Forum wäre nun also prädestiniert, dieses Rätsel zu lösen. Also auf geht's:

  Die beiden haben 2 Wasserkanister. Einer ist 5 Gallonen groß und der andere ist 3 Gallonen groß. Beide sitzen an einem Brunnen und haben ausreichend Wasser. Ihr Ziel ist es nun, einen der beiden Kanister mit exakt 4 Gallonen zu befüllen. Dieser muss dann auf eine Waage gestellt werden. Ist er genau so schwer wie ein Kanister mit 4 Gallonen, ist die Bombe entschärft. Ist er leichter oder schwerer wird sie explodieren. Mal unbeachtet davon, dass es da wohl immer marginale Abweichungen geben wird, wie bekommt man "exakt" 4 Gallonen in einen der beiden Kanister?

  http://www.mysnip.de/forum-archiv/thema/8007/241826/Stirb+langsam+−+Raetsel.html*:*

- *"Ein 16-Liter-Gefäß A ist ganz mit Wein gefüllt. Sein Inhalt soll ohne Verwendung einer Waage, eines Maßstabs mit Skala u. dgl. halbiert werden. Zur Verfügung stehen nur zwei leere Gefäße B und C, die 10 bzw. 6 Liter fassen."*

  Aufgaben dieser Art findet man öfter als Rätsel in Büchern über unterhaltsame Mathematik. Zu lösen sind sie dadurch, daß die Flüssigkeit aus dem großen Gefäß zum Teil in die kleineren gegossen wird und man sie anschließend in verschiedenen Portionen mehrfach hin - und hergießt.

  http://www.matheplanet.com/matheplanet/nuke/html/print.php?sid=511

- *Und nun das ehrfurchtgebietende Problem des ungarischen Barons, der sehr unzufrieden mit seinem Hofnarr war, dessen Scherze zu sehr verletzten. "Für Deine Unverschämtheiten sollst Du tausend Peitschenhiebe erhalten", sagte der schlimme Baron. "Ach geh, Baron," sagte der Narr albern, "sei fair, gib mir eine Chance." Der Baron hätte sich lieber Völkermord als Unfairness vorwerfen lassen, also stellte er dem Narr eine Aufgabe. Wenn er sie löste, würde er davonkommen.*

"Hier ist ein Fass meines besten Tokajer," sagte der Baron, "und hier sind zwei Weinkrüge - einer fasst 5 Liter und einer 3 Liter. Du darfst an den Krügen keine Zeichen anbringen, und Du musst Dir etwas ausdenken in jeden Krug genau 1 Liter zu bekommen, nicht mehr und nicht weniger. Du darfst nur das Fass und die Krüge berühren und natürlich, als besondere Vergünstigung, mit Deinen Füßen den Boden."

[Anmerkung: Wem das Krug-Rätsel jetzt schon bekannt vorkommt (gähn!), irrt! es ist eine Variation mit Trick!]

Der Hof wurde zusammengetrommelt, und der Narr musste vor dem Publikum versuchen, seinen Hals zu retten. Ich brauche nicht zu betonen, dass der Narr das Problem gelöst hat, sonst würde ich die Geschichte nicht erzählen. Mit seiner Lösungsweise bereitete er dem Baron großen Kummer, und wenn nicht während der Ausführung jedermann kräftig betrunken geworden wäre, wäre er wahrscheinlich schwer bestraft worden. Aber am anderen Morgen waren die Köpfe so benebelt, dass der ganze Vorfall vergessen war.

http://www.wer-weiss-was.de/theme113/article889728.html

Diese Rätsel eignen sich gut zum Knobeln, aber auch sehr schön, um den Gewinn von Perspektivenänderungen zu verdeutlichen. Denn besondere Repräsentationen von Zuständen und Operationen ermöglichen bei den Umfüllproblemen neue Sichten, die wie hier zu verblüffenden Lösungen führen können. Oder wissen Sie, wie sich Wein per Billard gerecht aufteilen lässt?

2 Umfüllprobleme per Billard lösen

Ja, genauso verwundert habe ich zuerst auf die Internetseite von Hans-Jürgen Caspar[1]

http://www.hjcaspar.de/hpxp/mpart/dateien/textdateien/umfuell.htm

geschaut, wie Sie, lieber Leser, sich jetzt wahrscheinlich ob der Überschrift wundern. Auf seiner Seite stellt Caspar eine interessante Idee zur Lösung bestimmter Umfüllprobleme dar und verweist seinerseits auf [Steinhaus, 1959] als Quelle. Die folgenden Ausführungen sollen die Verwunderung gegen plausible Erklärungen ersetzen. Mit ein wenig Geometrie wird die Basis für die erforderliche Sicht gelegt. Danach wird die Idee in ein lauffähiges Programm gegossen und anhand von Beispielen demonstriert.

2.1 Repräsentation per Billard

Prozentsätze als Punkte im Dreieck. Geometrie ist ein Fach von großer innerer Schönheit, auch wenn manche Schüler das in ihrer Schulzeit nicht so erkennen.

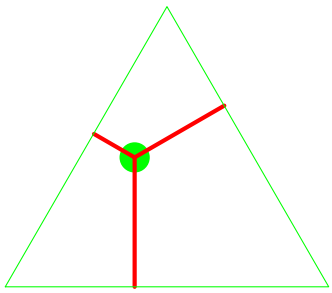
[1] siehe: http://www.hjcaspar.de/hpxp/anfang.php

Es ist unglaublich, wie unwissend die studierende Jugend auf die Universität kommt, wenn ich nur zehn Minuten rechne oder geometriere, so schläft ein Viertel derselben sanft ein.

Georg Christoph Lichtenberg

Doch oft besitzt die Geometrie aufgrund ihrer Anschaulichkeit Vorzüge, und es kann sehr hilfreich sein, ein Probleme in ein geometrisches Pendant zu übersetzen. Die Anschaulichkeit erhöht das Verständnis und hilft durch bekannte Zusammenhänge Schwierigkeitsgrade zu verringern. Hier kommt die interessante geometrische Erkenntnis zum Tragen, dass in einem gleichseitigen Dreieck die Summe der drei Abstände jedes inneren Punktes zu den drei Seiten immer gleich groß ist. Dieser Trick wird manchmal auch in der Statistik verwendet, beispielsweise zur Darstellung von drei Prozentzahlen, die sich zu 100% addieren.

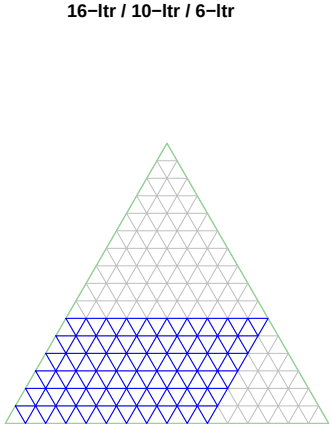
Jedem Tripel von solchen Prozentzahlen lässt sich also ein spezieller innerer Punkt eines gleichseitigen Dreiecks zuordnen. Diese Methode ist hilfreich, um verschiedene Tripel visuell zu vergleichen. Auch können wir die Entwicklung eines Tripel von drei Anteilen mit Summe 1 im Zeitablauf durch einen Linienzug in einem gleichseitigem Dreieck darstellen.



Wenn wir ausgehend von einem inneren Punkt die Prozent-Werte eines Tripels ermitteln wollen, messen wir die Abstände von den drei Seiten und schon ergeben sich die nummerischen Werte. Wollen wir den Punkt im Dreieck bestimmen, der zu einem Tripel von Prozentwerten gehört, dann müssen wir nur den Schnittpunkt der zwei Geraden finden, deren Abstände von den zugehörigen Seiten den beiden Werten entspricht. Natürlich muss die Gerade, die zu dem dritten Wert gehört, ebenfalls in dem Schnittpunkt die beiden anderen Geraden schneiden. Mit Punkten in einem gleichseitigen Dreieck können wir also die Aufteilung eines Gesamtwertes in drei Teile repräsentieren und damit Fragen zu Verhältnissen in die Welt eines gleichseitigen Dreiecks transformieren. Diese Übertragung mag sehr elegante Lösungen hervorbringen.

Die geometrischen Eigenschaften gleichseitiger Dreiecke bilden den ersten Grundstein für die Billard-Repräsentation. Denn die Verteilung der Füllmengen von drei Gefäßen ist durch drei Prozentsätze festgelegt.

Ein Parallelogramm als Tisch. Der zweite gedankliche Puzzlestein leitet sich aus der Tatsache ab, dass die Weingefäße der Rätsel durch ihr Volumen eine begrenzte Füllmenge besitzen. Damit sind nicht alle Punkte in einem gleichseitigen Dreieck mit realisierbaren Füllmengenverteilungen verbunden. Gehen wir einmal von einem großen Δ aus, also einem gleichseitigen Dreieck, dessen eine Seite horizontal verläuft und sich die gegenüberliegende Ecke dieser Seite oberhalb von der Seite befindet. In diesem Dreieck wollen wir die Menge, die sich im größten Gefäß – Gefäß `I` – befindet, durch den Abstand von der nordöstlichen Seite messen. Der Inhalt des zweitgrößten Gefäßes – Gefäß `II` – soll dem Abstand von der nordwestlichen Seite entsprechen und der Abstand von der Südseite soll die Flüssigkeitsmenge im dritten Gefäß – Gefäß `III` – anzeigen.

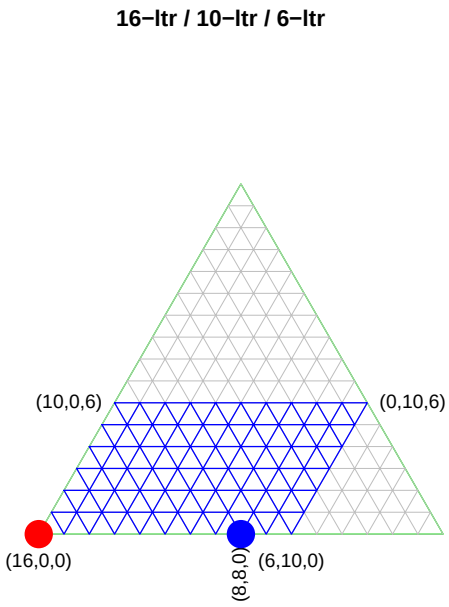


Gemäß der Begrenzung des dritten Gefäßes können wir eine waagerechte Linie durch unser Dreieck legen. Kein Punkt einer realisierbaren Verteilung kann sich oberhalb der Waagerechten befinden, und wir können das Dreieck durch diese Grenzlinie in Höhe der Füllmenge von `III` kappen. Auch das zweite Gefäß grenzt den realisierbaren Bereich ein, und in Folge wird die Ecke unten rechts vom Dreieck abschnitten. Mit diesen Überlegungen im Hinterkopf sei das letzte Bild betrachtet, das uns das Parallelogramm zulässiger Verteilungen zeigt.

Repräsentation von Litermengen. Vielleicht ist es hilfreich, nach Betrachtung des Bildes den letzten Abschnitt noch einmal zu lesen und die einzelnen Bemerkungen zu überprüfen. In dem Bild wurde von einer Problemstellung der Halbierung von 16 Litern mittels Gefäßen mit 16, 10 und 6 Litern ausgegangen. Zwar haben wir oben Prozentsätze ins Auge gefasst, jedoch ändert sich

nichts an der Konstruktion, wenn wir statt von 0% bis 100% in Kategorien von 0 bis 16 Einheiten oder Litern denken. In diesem Fall hat der Punkt unten links einen maximalen Abstand von 16 von der Nordost-Seite. Für unser Problem können nur ganzzahlige Aufteilungen der 16 Einheiten oder Liter interessant sein. Insofern lassen sich die einzelnen Einheiten durch Parallelen zu den Seiten darstellen und wir erhalten eine Zerlegung des großen Dreiecks in viele kleine Dreiecke. Mit diesen kleinen Dreiecken können wir zu einzelnen Punkten die kodierte Aufteilung bequem abzählen. Jetzt sind wir in der Lage, Zustände während des Lösungsprozesses von Weinaufteilungsproblemen zu repräsentieren.

Für das 16-10-6-Liter-Aufteilungsproblem sind einige besondere Zustände durch Füllmengen-Tripel beschriftet worden. Der rote Punkt unten links hebt den Anfangszustand (16,0,0) und der blaue den gewünschten Endzustand (8,0,8) hervor.

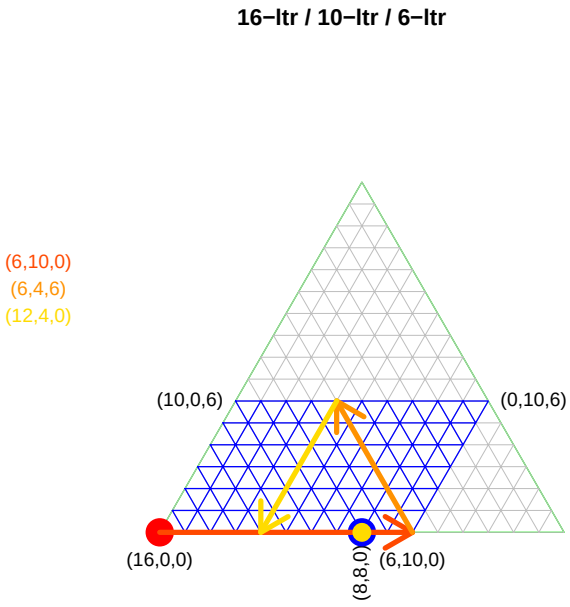


Schüttoperationen und Vektoren. Als elementare Handlung sind für die Problemlösung ausschließlich Schüttoperationen von einem Gefäß in ein anderes erlaubt. Eine solche Operation ist stets durch zwei Zustände, nämlich den Zustand vorher und dem Endzustand festgelegt. Zustandsveränderungen lassen sich also durch Vektoren im Parallelogramm charakterisieren. Da immer nur zwei Gefäße an einer solchen Schüttoperation beteiligt sind, muss die Koordinate der dritten konstant bleiben. Also bekommen wir nur Vektoren, die parallel zu einer der Seiten unseres Dreiecks sind.

Von jedem inneren Punkt gibt es sechs denkbare Schütt-*Richtungen*, von jedem inneren Punkt einer Parallelogrammseite sind dagegen nur vier Richtun-

gen möglich und von den Ecken gehen nur noch zwei Wege aus, um einen neuen Zustand zu erreichen.

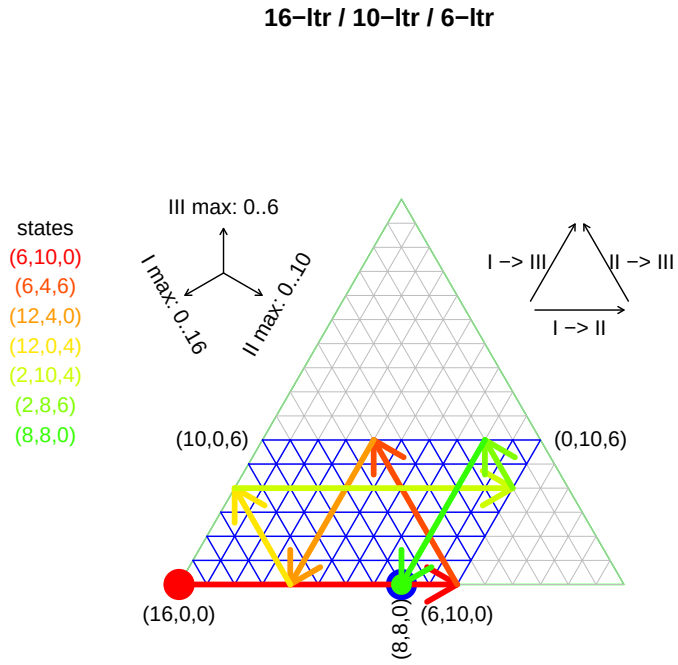
Schnell lässt sich überlegen, dass innere Punkte nicht relevant für die Umschüttprobleme sind. Denn man schüttet immer so lange, bis das Zielgefäß voll oder aber bis das Quellgefäß leer ist. Fraglich ist, ob man sich von einem Nicht-Endpunkt einer Parallelogramm-Seite zu einer Ecke bewegen sollte. Auch hier wird schnell klar, dass dadurch keine Verbesserung erreicht wird. Denn die beiden Ecken oben links und unten rechts könnte man mit einer einzige Schüttung vom Ausgangspunkt aus erreichen und die oben rechts mit zwei Operationen.@



Da man zum Erreichen eines inneren Punktes einer Seite ausgehend vom Anfangszustand mindestens zwei Operationen benötigt, gilt als allgemeine Regel, die *Ecken* zu *vermeiden*. Wir müssen uns also von den Ecken fern halten, so dass nur Pfade verbleiben, die zwischen inneren Randpunkten hin und her springen. Da ein Zurückgießen ebenfalls wenig Sinn macht, bleibt in jedem Randpunkt nur eine einzige zielführende Richtung der Zustandsänderung übrig. Fassen wir diese Überlegungen in einem Algorithmus zusammen:

1. Wir starten unten links in dem Anfangszustand
2. und wenden uns Schritt für Schritt die beschriebene Sprungstrategie an, bis wir
3. im gewünschten Endzustand landen.

Der Leser möge die Strategie anhand der folgenden Graphik überprüfen. Zur Erleichterung der Lesbarkeit enthält sie einige Legenden. Übrigens werden weiter unten bei der Umsetzung der Suche nach der nächste Schüttoperation noch einmal die jeweils neuen Richtungen argumentiert, ohne dabei geometrische Überlegungen zu zitieren. Wie zu erwarten ergibt sich eine sehr unübersichtliche Menge verstrickter Argumente.



2.2 Was hat das mit Billard zu tun?

Na ja, im wesentlichen die Erkenntnis, dass beim Billard Einfallswinkel = Ausfallswinkel gilt. Und dieses Gesetz gilt auch für unsere Strategie, wie eine nähere Inspektion der Pfeile im Bild oben zeigt. Jedoch unterscheidet sich unser Billardtisch von herkömmlichen, da er keine Rechteckform besitzt. Leider ergibt sich kein unmittelbarer Nutzen und wir müssen Caspar zustimmen, denn er schreibt:

> Niemand wird sich zur Lösung von Umfüllaufgaben extra einen Billardtisch mit 60- und 120-Grad-Winkeln bauen ...

Der Witz liegt natürlich in der *Vorstellung*, per Billard ein kleines mathematisches Rätsel zu lösen, also in der alternativen gedachten Repräsentation des Problems.

3 Wir wollen es sehen!

Die beschriebene Billard-Strategie lässt sich mit einer dynamischen R-Graphik demonstrieren. Dazu definieren wir eine Funktion, die uns diese Arbeit abnimmt. Sie plottet uns das gleichseitige Dreieck mit den angesprochenen Hilfslinien und nach und nach die Zustandveränderungen. Für den Dreiecksplot müssen wir etwas Sinus- und Cosinus-Wissen aktivieren. Das Ausschneiden des Parallelogramms ist eher eine Fingerübung. Insofern bleiben als schwierigste Aufgaben die Ermittlung der nächsten Richtung und das Auffinden des nächsten Zustands übrig. Diese beiden Aufgaben werden mit den lokalen Funktionen `new.direction()` und `find.new.state()` erledigt.

3.1 Die grobe Fassung

Grobstruktur. Mit den beiden erwähnten Hilfs-Funktionen ist folgende Struktur von `distribute.vino()` naheliegend.

1 ⟨*define distribute.vino()* 1⟩ ≡

```
distribute.vino <-
  function(II=10,III=6,dir.start=12,state.start,state.end,
           sleep=1,i.max=21,verbose=FALSE){
  ⟨check input and define some functions for distribute.vino 2⟩
  state <- state.start; dir <- dir.start
  ⟨show triangle plot 5⟩
  ⟨show path found by hand for problem II=5, III=3 14⟩
  for(i in 1:i.max){
    state.new <- find.new.state(dir,state)
    ⟨show new state 3⟩
    state <- state.new
    if(identical(state,state.end)) { cat("fertig"); break }
    dir <- new.direction(dir,state)
  }
  ⟨show result 4⟩
}
distribute.vino(5,3,dir.start=12,sleep=1)
```

Input. Als zentralen Input sind der Funktion `distribute.vino()` die Füllmengen der beiden kleineren Gefäße (`I` und `II`) mitzugeben. Die Größe des größten Gefäßes wird durch Summation der beiden kleinen Mengen berechnet. Als erste Schüttoperation wird eine Umfüllung von Gefäß `I` nach `II` vorgeschlagen und ist mit `12` kodiert. Möchte man dagegen die andere Möglichkeit probieren, so wähle man als Richtung `dir=13`. Die Bedeutung der weiteren Variablen ergeben sich aus den Argumentnamen.

Wie man sieht, werden Schüttoperationne durch eine zweistellige Zahl kodiert, wobei die zweite Ziffer das Zielgefäß angibt und die erste das Gefäß, aus dem Flüssigkeit entnommen wird. Weiterhin erkennen wir, dass in der Umsetzung ein Zustand (siehe `state`) als dreielementiger Vektor notiert wird, wobei sich Position i auf Gefäß i bezieht.

Für die Implementation sind also noch eine Reihe von Chunks zu definieren.

Fassen die in diesem Papier verwendeten Chunks in einer Liste zusammen.

Code Chunk Index

3.2 Verfeinerungsschritte

Check. Im Input-Check-Teil wird die Größe von Gefäß `I` berechnet. Falls die Richtung versehentlich als zweielementiger Vektor angegeben wurde, wird diese Festlegung in die interne Codierung durch eine zweistellige Zahl umgewandelt.

2 ⟨*check input and define some functions for distribute.vino* 2⟩ ≡ ⊂ 1

```
I <- II + III; if(II<III){ II <- I-II; III <- I-III }
if(missing(state.start)) state.start <- c(I,0,0)
if(missing(state.end  )) state.end   <- c(I/2,I/2,0)
if(length(dir)==2) dir <- 10*dir[1] + dir[2]
col <- rainbow(i.max)
```

Zwischenergebnisse. Ein paar Infos während des Lösungsprozesses sind nett, ebenfalls, wenn man in der Graphik die Veränderungen sieht.

3 ⟨*show new state* 3⟩ ≡ ⊂ 1

```
cat("number",i,"state",state,"direction",
    paste(unlist(strsplit(as.character(dir),"")),collapse="->"))
text(-0.3,0.8-(i+1)/15,
     paste(sep="","(",paste(state.new,collapse=","),")"),xpd=NA,col=col[i+1])
show.vec(state,state.new,col=col[i+1])
Sys.sleep(sleep);  if(verbose) cat("state:",state,"/ state.new:",state.new)
```

Ergebnis. Am Ende sollte das Ergebnis geeignet präsentiert werden.

4 ⟨*show result* 4⟩ ≡ ⊂ 1

```
points((I/2)/I,0,col=col[i+1],pch=16,cex=2)
cat("final state",state,"\ni =",i)
```

Bildkonstruktion. Zu Beginn wird das Grundbild aufgebaut. Hierzu sind folgende Teiltätigkeiten notwendig:

5 ⟨*show triangle plot* 5⟩ ≡ ⊂ 1
⟨*initialize plot* 6⟩
⟨*plot small triangles* 7⟩
⟨*plot region of interest* 10⟩
⟨*plot some important points* 8⟩
⟨*plot legend* 9⟩

`ymax` zeigt uns die maximale Höhe des gleichseitigen Dreiecks an.

6 ⟨*initialize plot* 6⟩ ≡ ⊂ 5
```
# initialize plot
plot(0:1,0:1,asp=1,type="n",ylim=c(-0.1,1.1),xlim=c(-.0,0.9),
     axes=FALSE,xlab="",ylab="")
# points(.4,.4,pch=16,col="green",cex=4); segments(.4,.4,.4,0,col="red",lwd=4)
# segments(.4,.4,.4+0.32*sin(pi/3),.4+0.32*cos(pi/3),col="red",lwd=4)
# segments(.4,.4,.4-0.145*sin(pi/3),.4+0.145*cos(pi/3),col="red",lwd=4)
# plot big triangle
ymax <- sin(1/6*2*pi)
dreieck <- rbind(0,1:0,c(0.5,ymax),0)
lines(dreieck,col="green")
title(paste(c(III+II,"-ltr / ",II,"-ltr / ",III,"-ltr"),collapse="" ))
```

Da sich die kleineren Dreiecke als Verkleinerungen des großen ergeben, ist es nur etwas mühsam, die Laufbereiche zu finden. Dieses hat während der Entwicklung mehrere Versuche gekostet.

7 ⟨*plot small triangles* 7⟩ ≡ ⊂ 5
```
# plot small triangles
red.dreieck <- dreieck/I
delta.x <- diff(red.dreieck[1:2]); delta.y <- ymax/I
for(j in 1:I){ # bis oben
  for(i in 1:(I-j+1)){ # bis zum Rand
    kd <- cbind(0.5*delta.x*(j-1)+red.dreieck[,1] + (i-1)*delta.x,
                red.dreieck[,2] + (j-1)*delta.y)
    lines(kd,col="gray",lwd=0.5)
  }
}
```

In dem Dreieck sollen verschiedene Punkte hervorgehoben und bezeichnet werden, damit auch ein aussagekräftiges Bild entsteht.

8 ⟨*plot some important points* 8⟩ ≡ ⊂ 5
```
text(0,-.07,
     paste("(",I,",",0,",",0,")",sep=""))
text((I-III)/I,-.07,adj=0,
     paste("(",I-II,",",II,",",0,")",sep=""))
text(delta.x*(III/2-1)-0.05,III*delta.y,adj=1,
     paste("(",I-III,",",0,",",III,")",sep=""))
text(1-delta.x*(III/2-1)+0.05,III*delta.y,adj=00,
     paste("(",I-III-II,",",II,",",III,")",sep=""))
#  points((I/2)/I,0,col="blue",pch=16,cex=2)
text((I/2)/I,-0.1,srt=90,xpd=NA,
     paste("(",I/2,",",I/2,",",0,")",sep=""))
show.state(state.start,col="red")
show.state(state.end,col="blue")
```

Die Legende soll eine Lesererleichterung darstellen. Wir zeigen, in welchen Richtungen sich die drei Koordinaten bzw. Füllmengen ablesen lassen, und außerdem, wie man sich bei den drei angedeuteten Schüttoperationen bewegen muss.

9 ⟨*plot legend* 9⟩ ≡ ⊂ 5

```
#  text(0,-.05,paste("(",I,",",II,",",III,")",sep=""))
arrows(0.1,0.7,0.1,0.8,length=.05)
text(0.1,0.85,paste("III max: 0..",III,sep=""))
arrows(0.1,0.7,0.1-0.1*sin(pi/3),0.7-0.1*cos(pi/3),length=.05)
text(-.01,0.63,srt=-60,paste("I max: 0..",II+III,sep=""))
arrows(0.1,0.7,0.1+0.1*sin(pi/3),0.7-0.1*cos(pi/3),length=.05)
text(.22,0.63,srt=60,paste("II max: 0..",II,sep=""))
# show directions
si <- 0.8-0.2*sin(pi/3); ee <- 0.01
arrows(0.9-0.2*cos(pi/3)-ee,si+ee,0.9-ee,0.8+ee,col=1,length=.05)
arrows(0.9+0.2*cos(pi/3)+ee,si+ee,0.9+ee,0.8+ee,col=1,length=.05)
arrows(0.9-0.2*cos(pi/3),si-ee,0.9+0.2*cos(pi/3),si-ee,col=1,length=.05)
text(0.9,si-.05,"I -> II")
text(0.8-4*ee,si+.1,"I -> III")
text(1+4*ee,si+.1,"II -> III")
text(-0.3,0.8-0/15,"states",xpd=NA)
text(-0.3,0.8-1/15,
     paste(sep="","(",paste(state,collapse=","),")"),xpd=NA,col=col[1])
```

Der Billardtisch ist eine Teilmenge des Dreiecks. Hier ist er.

10 ⟨*plot region of interest* 10⟩ ≡ ⊂ 5

```
rev.dreieck <- red.dreieck; rev.dreieck[,2] <- red.dreieck[c(3,3,1,3),2]
for(j in 1:III){
  for(i in 1:(I-III)){
    kd <- cbind(0.5*delta.x*(j-1)+rev.dreieck[,1] + (i-0.5)*delta.x,
                   rev.dreieck[,2] + (j-1)*delta.y)
    lines(kd,col="blue")
  }
}
```

3.3 Hilfsfunktionen

- `show.state()` soll die Frage beantworten, welche x-y-Koordinaten zu einem Zustand gehören. Außerdem soll der entsprechende Punkt in der Graphik markiert werden. `show.vec()` zeichnet uns den Vektor, der zu einer Umschüttoperation gehört. Dieser Funktion sind zwei Zustände mitzugeben.

11 ⟨*check input and define some functions for distribute.vino* 2⟩+ ≡ ⊂ 1

```
show.state <- function(state, col="blue") {
  x <- delta.x*(I-state[1]) - delta.x/2*state[3]; y <- delta.y*state[3]
  xy <- cbind(x,y); points(xy,pch=16,col=col,cex=3)
  xy
}
show.vec <- function( state, state2, col="blue" ) {
  if(identical(state,state2)) return()
  x <- delta.x*(I-state[1]) - delta.x/2*state[3]; y <- delta.y*state[3]
  xy <- c(x,y); state <- state2
  x2 <- delta.x*(I-state[1]) - delta.x/2*state[3]; y2 <- delta.y*state[3]
  arrows(x,y,x2,y2,lwd=4,col=col)
}
```

• `find.new.state()` schließlich verfolgt eine Richtung solange, bis der Rand des Billardtisches erreicht ist. Das ist dann der neue Zustand. Da das erste Gefäß genügend groß ist, um die gesamte Menge an Flüssigkeit aufzunehmen, gibt es für Schüttungen in das größte Gefäß keine Beschränkungen. Bei den anderen müssen wir schauen, ob der verbleibende Platz im Zielgefäß oder maximale Schüttmenge kleiner ist.

12 ⟨*check input and define some functions for distribute.vino* 2⟩+ ≡ ⊂ 1

```
find.new.state <- function(dir,state){
  idx <- c(floor(dir/10),dir%%10)
  switch(as.character(dir),
    "12" = { amount <- min(state[1], II-state[2]) },
    "21" = { amount <- state[idx[1]] },
    "13" = { amount <- min(state[1],III-state[3]) },
    "31" = { amount <- state[idx[1]] },
    "23" = { amount <- min(state[2],III-state[3]) },
    "32" = { amount <- min(state[3], II-state[2]) })
  state[idx] <- state[idx] + c(-1,1) * amount
  state
}
```

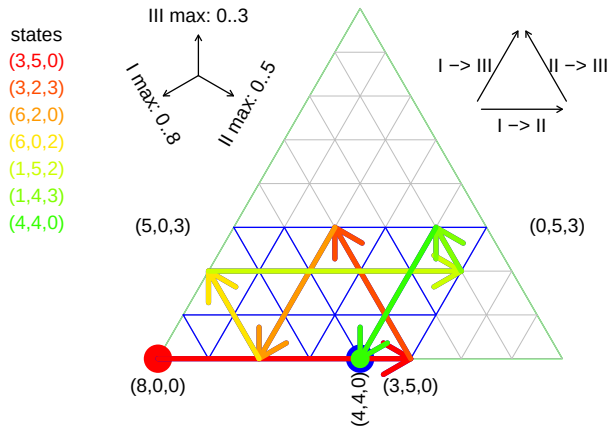
• `new.direction()` ermittelt die nächste Richtung in Abhängigkeit vom aktuellen Zustand und der alten Richtung.

$* \to 1$ Haben wir gerade ein Schüttoperation in das größte Gefäß erledigt, wird das beteiligte kleinere leer sein. Eine anschließende Umfüllung des anderen kleinen Gefäßes in das Gefäß `I` ist nicht brauchbar, da sonst der Anfangszustand erreicht wird.

$1 \to *$ Haben wir aus Gefäß `I` ein kleineres gefüllt, wird dieses dann randvoll sein. Ein anschießendes Umfüllen von `I` ins dritte macht wenig Sinn, da wir danach nur wieder ein kleineres in das größte Gefäß umfüllen könnten. Für `12` überlegen wir:

- Damit verbleiben als aussichtsreiche Umfüllungen nach `12` entweder `23` oder `31` übrig. `23` geht aber nicht, wenn Gefäß `III` bereits voll ist. Damit ergibt sich die Schüttung: $\to$ `31`.
- Wenn jedoch Gefäß `III` nicht voll ist, würde man mit `31` einen Zustand erzielen, den man bereits mit einer einzigen Schüttung ausgehend vom Startpunkt hätte erzielen können. Wir erhalten also Schüttung: $\to$ `23`.

Nach einer `13`-Operation gilt entsprechendes.



$2 \to 3$ Haben wir mit dem mittleren Gefäß das kleinste Gefäß bedient ...

3 voll ... und ist es voll geworden, macht es keinen Sinn, das zweite Gefäß durch das erste wieder aufzufüllen. Also ist `I` als Ausgangsfäß unbrauchbar. Wenn wir den Rest von Gefäß zwei ins erste gießen, haben wir einen Zustand erreicht, den wir auch mittels einer einzigen Schüttung am Anfang hätten realisieren können. Also bleibt nur: $\to$ `31` übrig.

sonst ... und ist `III` nicht voll geworden, muss nun Gefäß `II` leer sein. Damit kommen nur `I` und `III` für den Ausgang der nächsten Operation in Betracht. Schütten wir `III` in `I`, ergibt sich wieder der Ausgangszustand, so dass nur $\to$ `12` verbleibt.

$3 \to 2$ Hierzu verläuft der Gedanke wie nach der Operation `23` mit ausgetauschten Ziffern.

In ungünstigen Fällen könnte es sein, dass die Ecken unten links oder oben rechts getroffen werden. Dann wird in Richtung der Startrichtung mit passender Orientierung vorangeschritten. Entstehen dadurch Zyklen, kommt es letztlich zu keiner Lösung des Problems.

13 ⟨*check input and define some functions for distribute.vino* 2⟩+ ≡ ⊂ 1

```
new.direction <- function(dir,state){
  if(identical(state,c(I,0,0)))    return(ifelse(dir==31,12,13))
  if(identical(state,c(0,II,III))) return(ifelse(dir==13,21,31))
  switch(as.character(dir),
    "21" = dir <-  32 ,  # else "startsituation"
```

```
    "31" = dir <-  23 ,  # else "startsituation"
    "12" = if(state[3]<III) dir <-  23 else dir <- 31,
    "13" = if(state[2]<II)  dir <-  32 else dir <- 21,
    "23" = if(state[3]<III) dir <-  12 else dir <- 31,
    "32" = if(state[2]<II)  dir <-  13 else dir <- 21,
           cat("ERROR") )
  dir
}
```

Der Leser wird hoffentlich zustimmen, dass die geometrischen Überlegung anhand des Parallelogramm und der Billard-Reflexionsregel sehr viel klarer sind als die gerade aufgelisteten, verschachtelten Begründungen. Damit lohnt sich also die Transformation in der Umfüllprobleme in die geometrische Repräsentation.

3.4 Ein kleiner Blinddarm

Eine Spezialsituation. Für das Problem aus der Veranstaltung A&D wird die manuelle Lösung eingeblendet. Das hatte sich während der Entwicklung als hilfreich erwiesen.

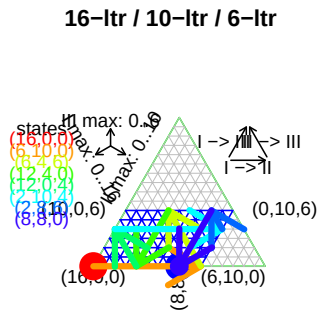
14 ⟨*show path found by hand for problem II=5, III=3* 14⟩ ≡ ⊂ 1

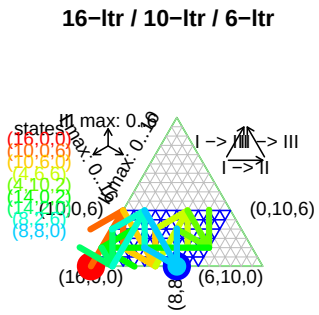
```
if(II==5 && III==3 && identical(state.start,c(8,0,0))){
  state <- state.start
  for(di in c(12,23,31,23,12,23,31)){
    state.new <- find.new.state(di,state)
    show.vec(state,state.new)
    state <- state.new
  }
  state <- state.start
}
```

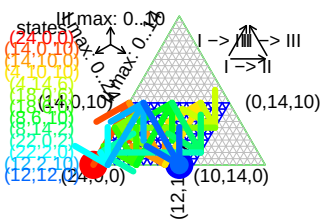
3.5 Beispiele

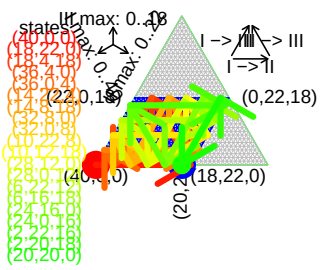
15 ⟨* 15⟩ ≡

```
no <- menu(1:4)
switch(no,
"1"=distribute.vino(II=10,III=6, dir.start=12,sleep=.1,i.max=10),
"2"=distribute.vino(II=10,III=6, dir.start=13,sleep=.1,i.max=15),
"3"=distribute.vino(II=14,III=10,dir.start=13,sleep=.1,i.max=20),
"4"=distribute.vino(II=18,III=22,dir.start=12,sleep=.1,i.max=60),
  "relax")
```









3.6 Ausblick und Schlussbemerkung

Sind damit alle Umfüllprobleme gelöst. Natürlich nicht. Denn andernfalls würden Rätselfreunde ja jetzt auch nichts mehr zu raten haben. Beispielsweise lässt sich mit der vorgestellten Funktion das Hofnarr-Problem leider nicht lösen.

Dem Leser seien folgende Denkanstöße und Fragen mitgegeben: Was ändert sich, wenn wir nur zwei Gefäße gibt, zum Nachfüllen jedoch eine beständig sprudelnde Quelle zur Verfügung steht? Wann haben Umfüllaufgaben keine Lösung? Wie können wir die aufgezeigte Strategie abändern, damit wir auch die Operation *trinke einen Liter Wein aus* integrieren können?

Abschließend möge man sich einmal

> http://www.mathematik.uni-dortmund.de/ieem/BzMU/BzMU2008/BzMU2008/BzMU2008_BENOELKEN_Ralf.pdf

anschauen. Dort werden Geschlechter spezifische Beobachtungen zum Lösungsverhalten von Umfüllaufgaben dargestellt.

Literatur

Hugo Steinhaus (1959): Kaleidoskop der Mathematik, Dt. Verl. d. Wiss.

Anhang

Object Index