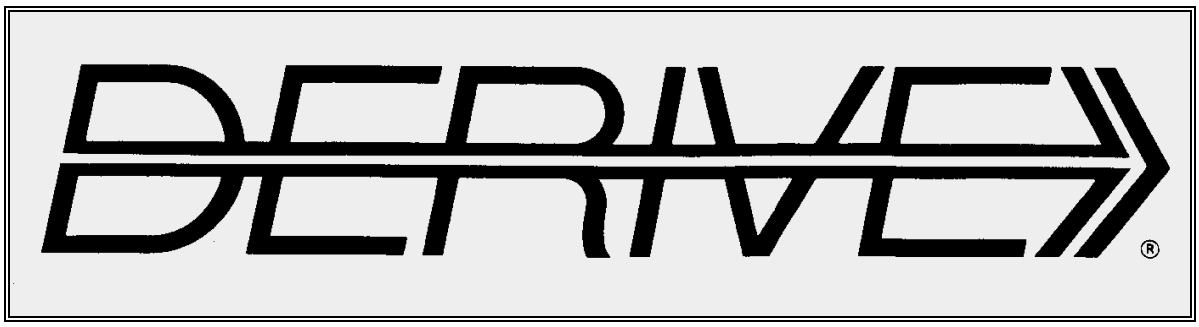
THE DERIVE - NEWSLETTER #87

ISSN 1990-7079

THE BULLETIN OF THE



USER GROUP

+ CAS-TI

Contents:

September 2012

Unser DUG-Mitglied Johannes Zerbs ersucht um Bekanntgabe seiner Mathe Chilled Website. Dieser Bitte komme ich gerne nach. Was besonders bemerkenswert ist: Es handelt sich hier um kein kommerzielles Angebot! Ja, so was gibt es auch noch, Josef.

Mathematik Online Buch

Mathe Chilled- ein Online Mathe Buch:
Ein Rätselblock nach den neuesten Standards und Kompetenzen

www.mathe-chilled.at

Gestern Mathe. Heute Mathe Chilled. Morgen in Mathe chillen....

As TIME Goes by …

Past TIME and future TIME

The “Kissing Students” are one of the most famous landmarks of Tartu, hosting town of TIME 2012 (left).

View of the old town of Krems (right). Krems is the entrance to the Wachau which is surely the most romantic part of the Danube valley between the Blackwood Forest and the Black Sea. Krems will host TIME 2014. Here is the place where it all began in 1992 with the first DERIVE Conference.

Liebe DUG-Mitglieder,

leider bin ich wieder einmal nicht rechtzeitig mit unserem Newsletter fertig geworden.

Einerseits war ich sehr mit anderen Aufgaben beschäftigt und andererseits war dieser DNL sehr zeitaufwändig. Die Übersetzung des Statistikartikels ging rasch von statten – ermöglicht durch die gute Kommunikation mit Guido Herweyers. Dagegen brauchte mein eigener Branch & Bound-Beitrag viel Zeit bis er meinen Wünschen entsprach. Aber das war ein schöner Ausflug in meine Lehrervergangenheit.

Piotr Trebisz hat den letzten Teil seiner Schneckenhausserie geschickt. Ein spannender Austausch von E-Mails ermöglichte interessante Ergänzungen zur Darstellung seiner schönen konischen Spiralflächen. Aus Platzgründen muss ich das für den nächsten DNL zurückstellen.

Auch Dietmar Oertel wartet schon lange auf die Veröffentlichung seiner Arbeiten an den Primzahlen und Taylorreihen. Ich ersuche auch ihn um Geduld. Sie sind „under construction".

Noch kurz zur TIME 2012: Herzlichen Dank an die Organisatoren vor Ort, Marina und Eno samt ihrem Team an der Universität Tartu, Estland. Großen Dank auch an alle Teilnehmer, die hervorragende Hauptvorträge und hochklassige Lektionen angeboten haben.

Eine Reihe von Vortragenden hatte nicht die Absicht, ihre Beiträge zur Veröffentlichung in Journalen einzureichen. Sie haben ihre Papiere dem DNL zur Verfügung gestellt. Vielen Dank dafür.

Schließlich möchte ich noch besonders auf die Themen im User Forum aufmerksam machen. Sie finden zwei kurze Hinweise auf TI-Nspire-Funktionen und zwei Probleme, die auf unsere DUG-Mitglieder warten.

Ganz aktuell: gemeinsam mit der Donau Universität Krems wird ACDCA die TIME 2014 vom 1. – 5. Juli 2014 in Krems an der Donau veranstalten.

Dear DUG Members,

unfortunately I am too late with this newsletter once more.

On the one hand I was busy with other duties and on the other hand production of this DNL was very time consuming. Translation of the statistics contribution was – thanks good communication with Guido Herweyers – quickly done. My own article on Branch & Bound took much time until it met my expectations – and during writing new ideas wanted to be realized … This was a nice trip back to my past as teacher.

Piotr Trebisz sent the last part of his snail shell series. An exciting email exchange brought interesting additions to the representation of his beautiful conic spiral surfaces. Because lack of space I must leave this for the next issue.

Dietmar Oertel has also been waiting a long time for publication of his prime number and Taylor series investigations. Please be patient. They are "under construction".

Concerning TIME 2012: many thanks to the local organizers, Marina, Eno and their team at Tartu University, Estonia. Many thanks also to all participants and presenters. We had excellent keynotes and great lectures.

Some presenters didn't intend to submit their full papers for publication in special journals. They provided their presentations for the DNL. Thanks for this offer.

Finally I'd like to draw your attention to the User Forum. You will find two short notes on Nspire functionality and two problems waiting to be tackled by DUG members.

Latest – great – news: The Danube University Krems and ACDCA will organize TIME 2014 from 1 to 5 July 2014 in Krems, Lower Austria.

Viele Grüße, kindest regards

The *DERIVE-NEWSLETTER* is the Bulletin of the *DERIVE* & CAS-*TI User Group*. It is published at least four times a year with a content of 40 pages minimum. The goals of the *DNL* are to enable the exchange of experiences made with *DERIVE*, *TI*-CAS and other CAS as well to create a group to discuss the possibilities of new methodical and didactical manners in teaching mathematics.

Editor: Mag. Josef Böhm
D´Lust 1, A-3042 Würmla, Austria
Phone: ++43-(0)660 3136365
e-mail: nojo.boehm@pgv.at

Contributions:
Please send all contributions to the Editor. Non-English speakers are encouraged to write their contributions in English to reinforce the international touch of the *DNL*. It must be said, though, that non-English articles will be warmly welcomed nonetheless. Your contributions will be edited but not assessed. By submitting articles the author gives his consent for reprinting it in the *DNL*. The more contributions you will send, the more lively and richer in contents the *DERIVE* & CAS-*TI Newsletter* will be.

Next issue: December 2012

Preview: Contributions waiting to be published

Some simulations of Random Experiments, J. Böhm, AUT, Lorenz Kopp, GER
Wonderful World of Pedal Curves, J. Böhm, AUT
Tools for 3D-Problems, P. Lüke-Rosendahl, GER
Hill-Encryption, J. Böhm, AUT
Simulating a Graphing Calculator in *DERIVE*, J. Böhm, AUT
Do you know this? Cabri & CAS on PC and Handheld, W. Wegscheider, AUT
An Interesting Problem with a Triangle, Steiner Point, P. Lüke-Rosendahl, GER
Graphics World, Currency Change, P. Charland, CAN
Cubics, Quartics – Interesting features, T. Koller & J. Böhm, AUT
Logos of Companies as an Inspiration for Math Teaching
Exciting Surfaces in the FAZ / Pierre Charland´s Graphics Gallery
BooleanPlots.mth, P. Schofield, UK
Old traditional examples for a CAS – what's new? J. Böhm, AUT
Truth Tables on the TI, M. R. Phillips, USA
Where oh Where is It? (GPS with CAS), C. & P. Leinbach, USA
Embroidery Patterns, H. Ludwig, GER
Mandelbrot and Newton with *DERIVE*, Roman Hašek, CZK
Tutorials for the NSpireCAS, G. Herweyers, BEL
Some Projects with Students, R. Schröder, GER
Structures in the Set of Prime Numbers, D. Oertel, GER
Dirac Algebra, Clifford Algebra, D. R. Lunsford, USA
Laplace Transforms, ODEs and CAS, G. Picard & Ch. Trottier, CAN
A New Approach to Taylor Series, D. Oertel, GER
Cesar Multiplication, G. Schödl, AUT
Henon & Co; Find your very own Strange Attractor, J. Böhm, AUT
Rational Hooks, J. Lechner, AUT
Mathematical Model for Snail Shells (4), P. Trebisz, GER
Simulation of Dynamic Systems with various Tools, J. Böhm, AUT
An APL-like SHAPE function in DERIVE 6, D. R. Lunsford, USA
Brussels Gate in Dendermonde, E. van Lantschoot, GER
and others

Impressum:
Medieninhaber: *DERIVE* User Group, A-3042 Würmla, D´Lust 1, AUSTRIA
Richtung: Fachzeitschrift
Herausgeber: Mag. Josef Böhm

Statistics with TI-Nspire 3.1/3.2 (Part 3)

Visualising and Simulating Dynamically with TI-Nspire

Guido Herweyers, KHBO Campus Oostende
guido.herweyers@khbo.be

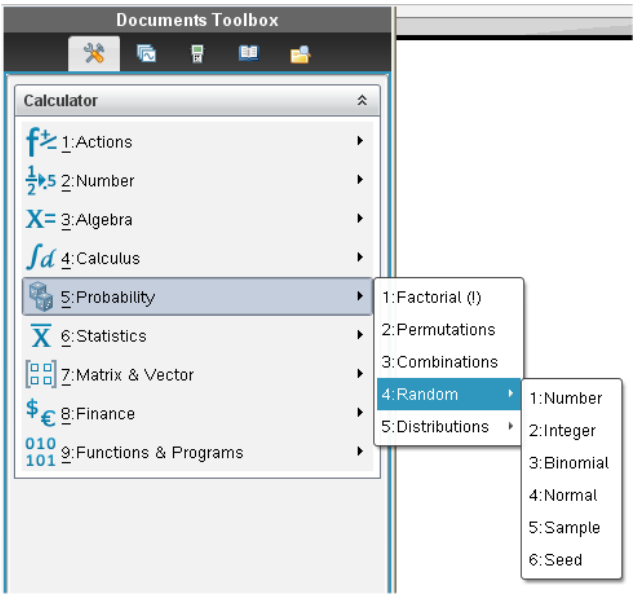
Part 2: Simulating with TI-Nspire

(1) Generating random numbers

Open a calculator page. In the menu **5: Probability > 4: Random** can you find the commands for generating random numbers.

Click on **Seed** or type **randseed** followed by space. Then type any natural number in order to initiate a new series of random numbers.

Examples for generating random numbers (you may compare with the TI-84 Plus commands) are:



→ rand(): a real random number between 0 and 1.
So gives 3 + 5 · rand() a random number within the interval (3,8).

→ rand(10): a list of 10 random numbers between 0 and 1.

→ randint(1,6): a natural number from 1 to 6 (throwing a die).

→ rand(1,6,20): a list of 20 integers within [1,6] (throwing a die 20 times).

→ randbin(10,0.5): the number of heads when throwing a coin 10 times.

→ randbin(10,0.5,20): a list of the results repeating "= TIMES the coin experiment from above.

→ randnorm(175,10): a normal distributed random number with mean = 175 and standard deviation = 10.

→ randnorm(175,10,50): a sample (list) of 50 random numbers from a normal distribution with mean = 175 and standard deviation = 10.

RandSeed 834	*Done*
randInt(1,6,10)	{5,5,6,3,5,5,5,5,1,5}
randInt(1,6,10)	{2,3,1,3,4,2,6,6,1,3}
randBin(10,0.5,20)	{3,2,7,6,6,3,4,7,4,6,6,6,6,5,6,3,7,6,4,6}
3+5· rand(4)	{4.84643,6.80426,7.00736,4.05879}

(2) Samples with and without replacement

A sample from a list of numerical or categorical data is taken using the command **randsamp**. Its syntax is:

with replacement: **randsamp(list name, sample size, 1)**
without replacement: **randsamp(list name, sample size)**

Drawing a random sample without replacement:
randsamp(list name, sample size,1)
Example: Lottery drawing (6 out of 45)
RandSeed 100 ▸ *Done*
lottery:=seq(*k*,*k*,1,45)
randSamp(**lottery**,6,1) ▸ {42,29,7,23,26,45}

Drawing a random sample with replacement:
randsamp(list name, sample size)
Example: 10 turns of a roulette wheel
roulette:=seq(*k*,*k*,0,36)
▸ {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36}
randSamp(**roulette**,10) ▸ {27,32,11,18,14,5,17,24,30,21}

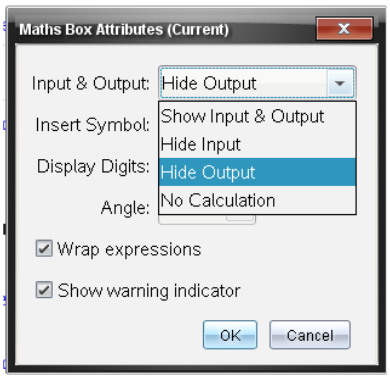
(You can find the roulette rules together with the win chances in http://www.casino-gids.be/artikels/spelregels/beginners/roulette.php.)

The screen shown above was created in a **Notes Application**.

The output of a **Maths Box** used in a **Notes Application** can be shown or hidden.

Perform a right mouse click in the Maths Box and choose the respective attributes.

The **lottery** list {1,2, …, 44, 45} is hidden and the **roulette** list is shown.



The Notes Application is very useful. It is an elementary text processor combined with a dynamic calculator in an environment which enables easy quick repetitions of simulations.

- You are able to combine text and mathematical expressions (like in a calculator page). A mathematical expression is written in a **Maths Box**, which can be activated by pressing **Ctrl + M**. As shown above you can hide the result of the evaluation: right click on the Maths Box and then choose the **Maths Box Attributes**.
- The Notes Application is dynamic (in contrary to the static Calculator Application): changes of a definition result in an immediate recalculation of all expressions in the Notes.
- A simulation – which is an expression containing **rand…** in any form – is automatically reevaluated by clicking on the expression followed by ENTER (or by pressing Ctrl+ENTER in order to perform quick repetitions of the simulation). Unfortunately this does not work any longer in version 3.2. Here you need a respective column in the spreadsheet where you can repeat the simulation by pressing **Ctrl+R**. Read more about this on page 6.

Hint: Having defined a column in a spreadsheet application using a simulation command then the simulation is immediately reevaluated after clicking the spreadsheet and then pressing **Ctrl+R**.

	A times	B rolls
◆		=randint(1,6,a1)
1	10	6
2		6
3		1
4		4
5		3
6		5
7		1
8		3
9		6
10		2

Ctrl+R

	A times	B rolls
◆		=randint(1,6,a1)
1	10	4
2		3
3		6
4		6
5		4
6		5
7		4
8		5
9		2
10		2

Ctrl+R

	A times	B rolls
◆		=randint(1,6,a1)
1	10	4
2		5
3		2
4		1
5		3
6		6
7		3
8		4
9		1
10		4

(3) Rolling two Dice

We split the screen into three applications: Lists & Spreadsheet, Notes, and Data & Statistics. Sample size **n** and simulation **sumd** (sum of the two dice) are defined within the Notes. Spreadsheet and bar chart as well will change dynamically when performing a new simulation by changing the sample size **n** and by repeating the simulation or by clicking in the **sumd** Maths Box and then pressing ENTER. Investigate the influence of the sample size on the variability of the sample and the "convergence" to the theoretic probabilities of the sum of two dice.

	A sumd	B resu…	C freq	D relfreq	E probabilities	F
◆						
5	9	6	12	0.12	0.1389	
6	7	7	20	0.2	0.1667	
7	4	8	13	0.13	0.1389	
8	5	9	12	0.12	0.1111	
9	9	10	6	0.06	0.0833	
10	4	11	4	0.04	0.0556	
11	7	12	4	0.04	0.0278	
12	9					
13	11					
14	7					
15	4					
16	10					
17	10					
18	7					
19	8					
20	6					
21	6					

E7 =round(5/36,4)

Simulation of rolling n times two dice and observing the sum.

Rolls **n**:=100

The sums of the dice is put in column A.

sumd:=randint(1,6,n)+randint(1,6,n)

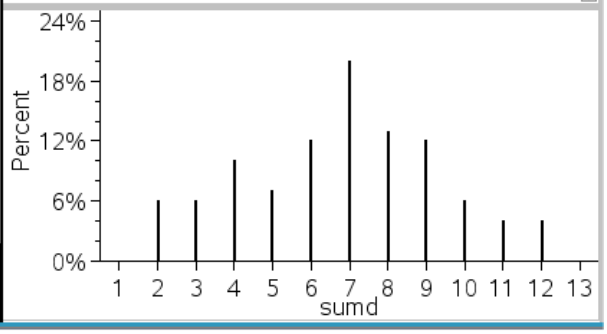
The frequencies of the sums is put in column C.

freq:=seq(countif(sumd,?=k),k,2,12)

The relative frequencies are given by

relfreq:=approx(freq/n).

Compare these values with the resepctive probabilities!



It is really a pity that the full text in this paper is only static and not dynamical …

(4) Probability distribution of the sample mean

In the following we will discuss the principle how to find an approximation for the probability model of the sample mean.

- Take a big number of samples from a population with a discrete or continuous probability distribution.
- Note for each sample its mean and append its value to a list via automatic **capturing** in a column of the spreadsheet application.
- A dot plot of the list shows whether the probability model of the sample means should be a discrete or a continuous one.
- Observe graphically the convergence of the percental distribution of the bar chart for a discrete model or the convergence of the density histogram (with small enough class widths) for a continuous model to a stable situation.
- The "equilibrium situation" gives an approximation of the probability distribution of the sample means.

For didactical reasons it is recommended to take the sample from a concrete given population of numbers like the results of main part 4 of the life guards which is a population of 367 data showing a negatively (= left) skewed distribution (long lower tail).

This is much more realistic for the students than taking a sample from an artificial population which is generated by applying a probability model (e.g. the uniform distribution for rolling a die).

Introduce the data as shown below in the **Notes**. Sometimes it is necessary to execute a command with Ctrl+ENTER (instead of only ENTER) in order to get the result in decimal form (instead of a fraction). Examples for this are **samplemean** and **mean(h4)**.

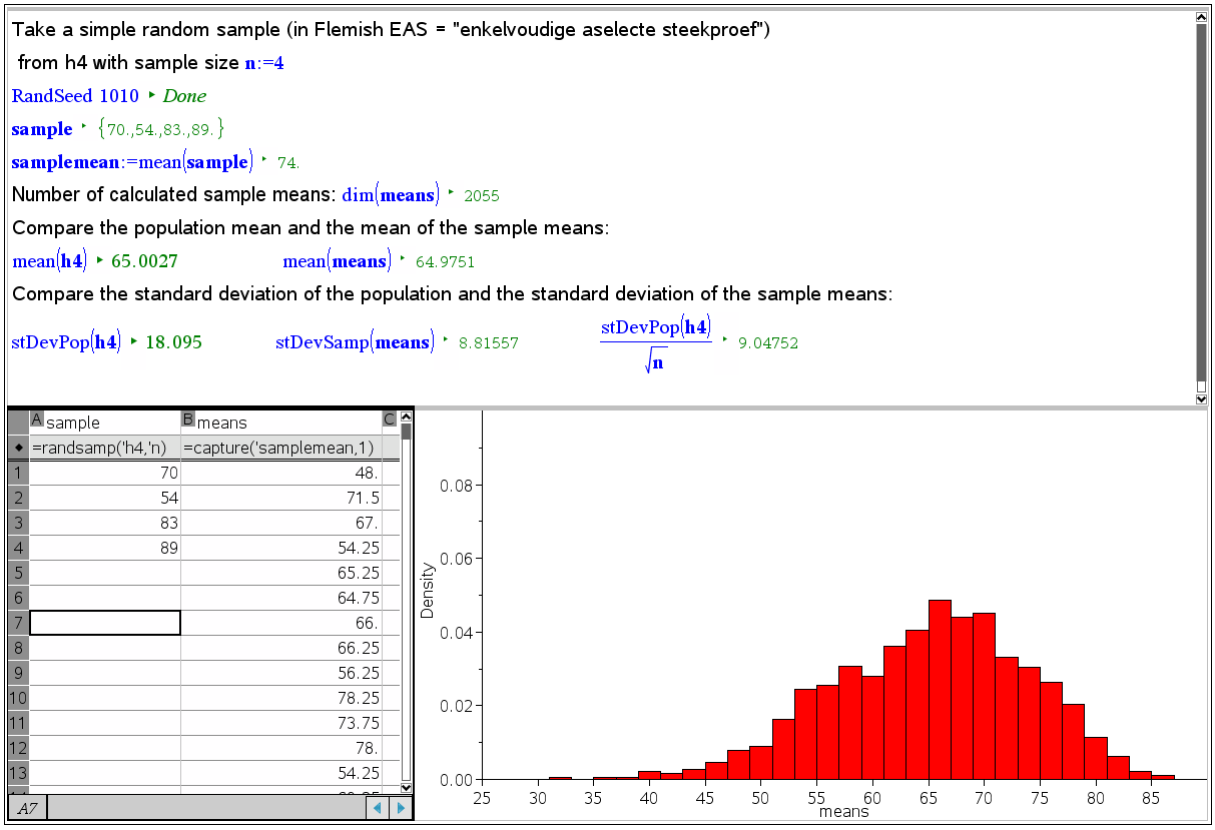
The mean of each sample will be automatically added to the list means, which has been defined in the spreadsheet application: click into the formula cell and call **Data > Data Capture > Automatic** in the Spreadsheet Toolbox, then change **capture(var,1)** to **capture(samplemean,1)**.

For taking random samples provide a Maths Box (Ctrl+M) and type **sample:=randsamp(h4,n)**. Another Math Box is necessary to calculate the samplemeans: **samplemean:=mean(sample)**. These results are collected in the spreadsheet column means. To get a really good impression you need many samples, so keep pressing Ctrl+ENTER in the **sample**-Maths Box for repeating the process many times. Accomplish the Notes according to the screenshot given below … **but** …

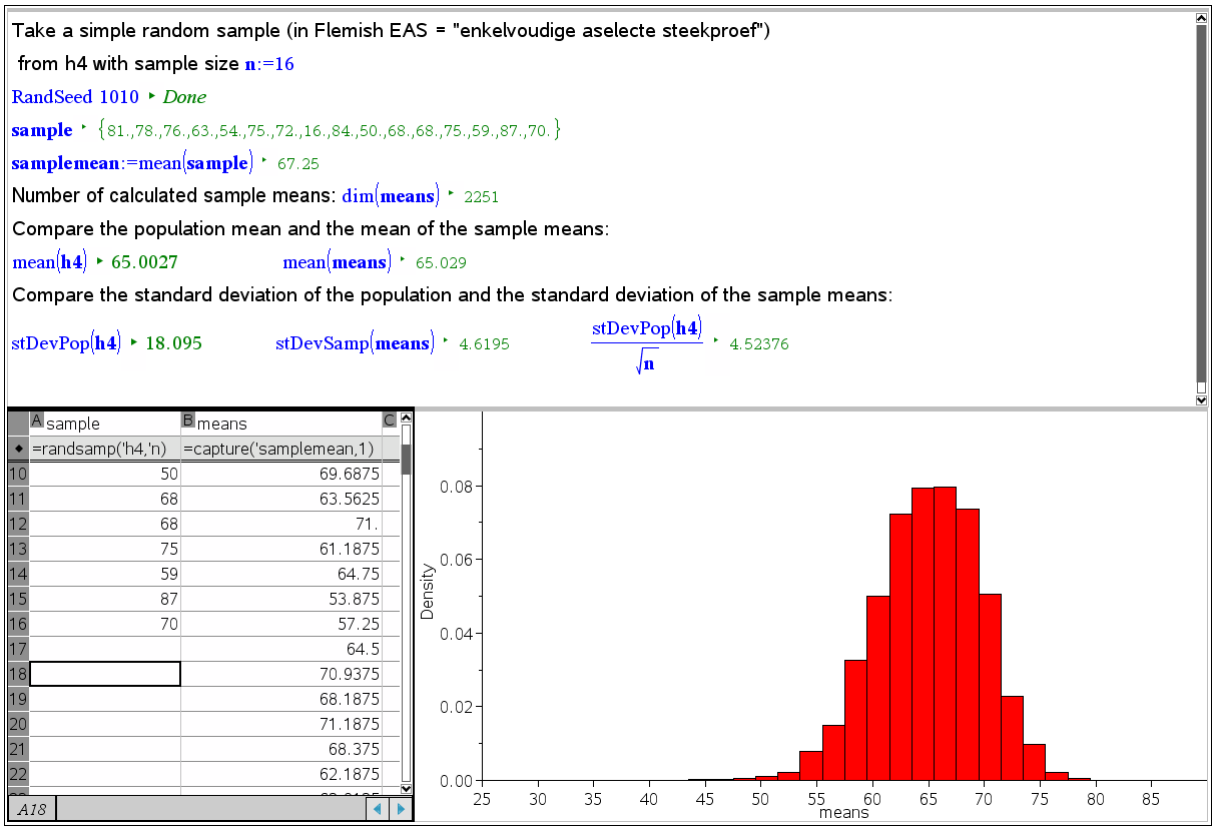
… inspecting this screenshot you will find another column sample in the spreadsheet. Why this? Unfortunately TI-Nspire changed its behaviour from Version 3.1 to 3.2 as mentioned on page 4 in such a way that pressing Ctrl+ENTER in the Maths Box does not repeat the simulation. You have to define the **sample** in a column. Then click into any cell of the spreadsheet and press **Ctrl+R**.

Perform a sufficient large number of samples.

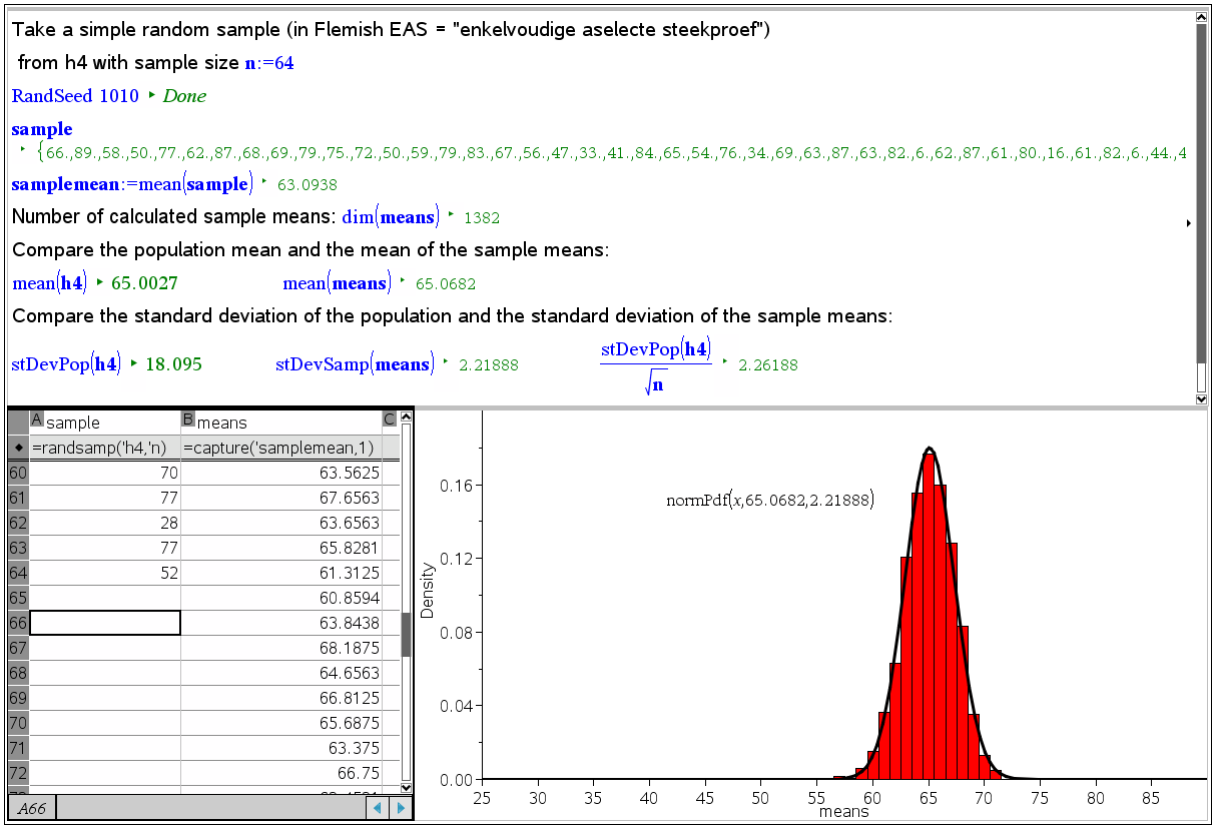
Don't forget to change the **RandSeed**s or ask the students to take different RandSeeds otherwise they could get identical results – and this is will not give a "random like" impression.



Working with sample size **n** = 4 we find the probability distribution of the mean less skewed than the populations's one. You can see a simulation of 2055 samples. The average of all means is 64.98 which is very close to the mean of the population 65.00. The deviation is almost halved.



Sample size **n** = 16 and a series of simulation of 2251 samples. Deviation is halved again.



Sample size **n** = 64, 1382 samples. Sample is … Illustration of the Central Limit Theorem.

The reader is invited to experiment with samples taken from other (discrete or continuous) populations. This shall lead to the conjecture that the probability model of the sample mean population – on condition that the sample size is sufficient large – can be approximated by a bell-shaped model – the normal distribution.

The bell-shaped probability model of a normal distributed population is completely defined by its population mean μ and its standard deviation σ. The influence of these parameters on the graph of the density function will be investigated by introducing sliders.

How to achieve the screen given below? This is the answer:

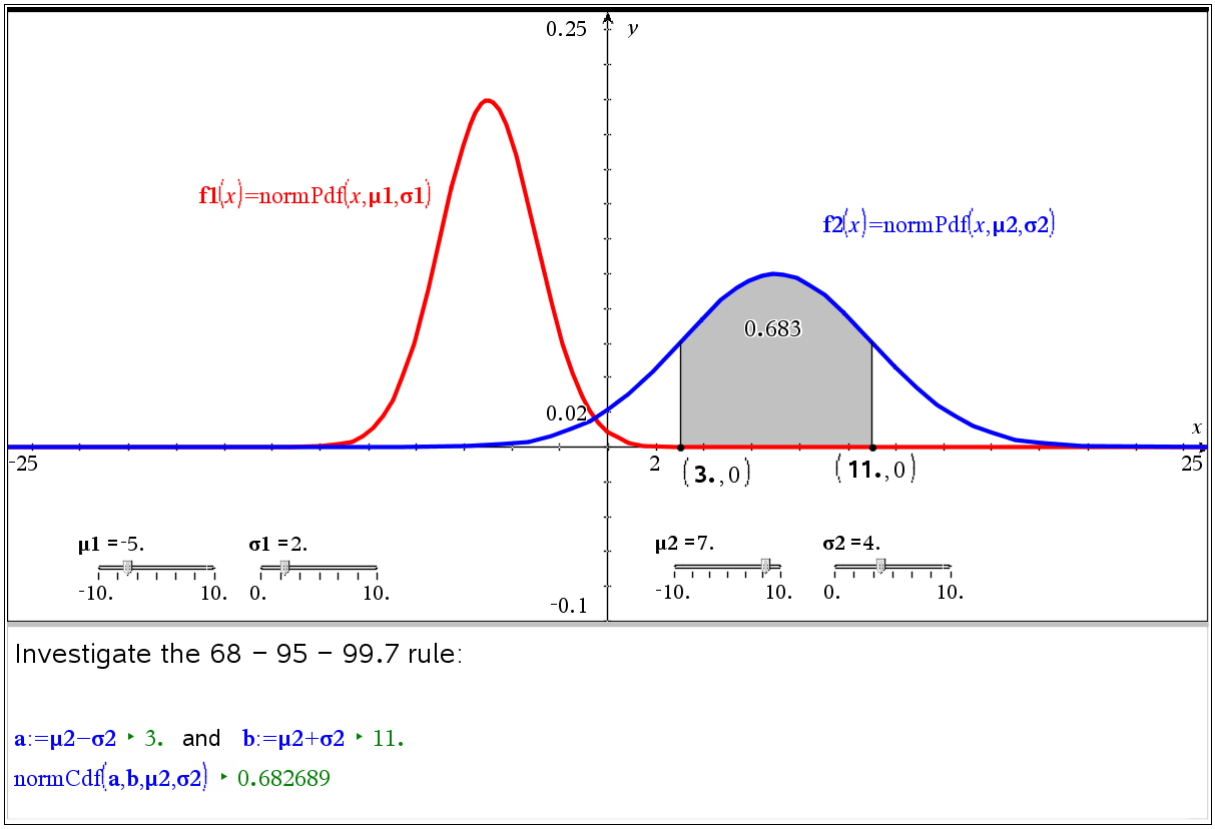
- Split the page horizontally into a Graphs-application and a Notes-Application.
- Activate the Graphs and introduce four sliders via the Graphs & Geometry Toolbox: **1:Actions > B:Insert Slider**. The Greek letters can be found among the **Utilities** in the Documents Toolbox.

(Take care that you take the right μ.

The μ-character shown in the first row is a system variable which cannot be used.)

Insert sliders for two means μ_1, μ_2 and two standard deviation σ_1 and σ_2.

- Define the functions f1(x) and f2(x) via the entry line which appears after clicking on the >> symbol bottom left (version 3.1). In version 3.2 make a right mouse click on the graph page. **2:Hide/Show > 5: Show Entry Line**.
- Define two points on the *x*-axis using **Points & Lines > Point On** in Graphs & Geometry Toolbox > **8:Geometry**. Then press ESC for leaving this menu.
- Show the coordinates of these points via **1:Actions > 8:Coordinates and Equations**.
- Switch to the Notes and define the variables **a** and **b** as shown below.
- Return to the Graphs-application, move the cursor to the *x*-coordinate of the first point (the word "text" appears). Make a right mouse click on it and choose in the appearing pop-up menu **6:Variables > 2:Link to > a**. Now this point is linked with the variable **a**. Repeat this procedure for the second point linking its *x*-coordinate with variable **b**.
- Calculate the area under the graph of f2(x) between a and b using the **Analyse Graph** menu > **7:Integral**. Click on the graph and then fix the two points as boundaries.



Hint: In the Calculator Application and in the Spreadsheet Application you have access to the Maths Operators. Go to **Statistics > Distributions** and there you will find a couple of probability distributions offered with the normal distribution among them.

If $X \sim N(\mu,\sigma)$, then

→ normPdf(x,μ,σ) = f(x) with f being the normal density function

→ normCdf(a,b,μ,σ) = P($a \le x \le b$) and

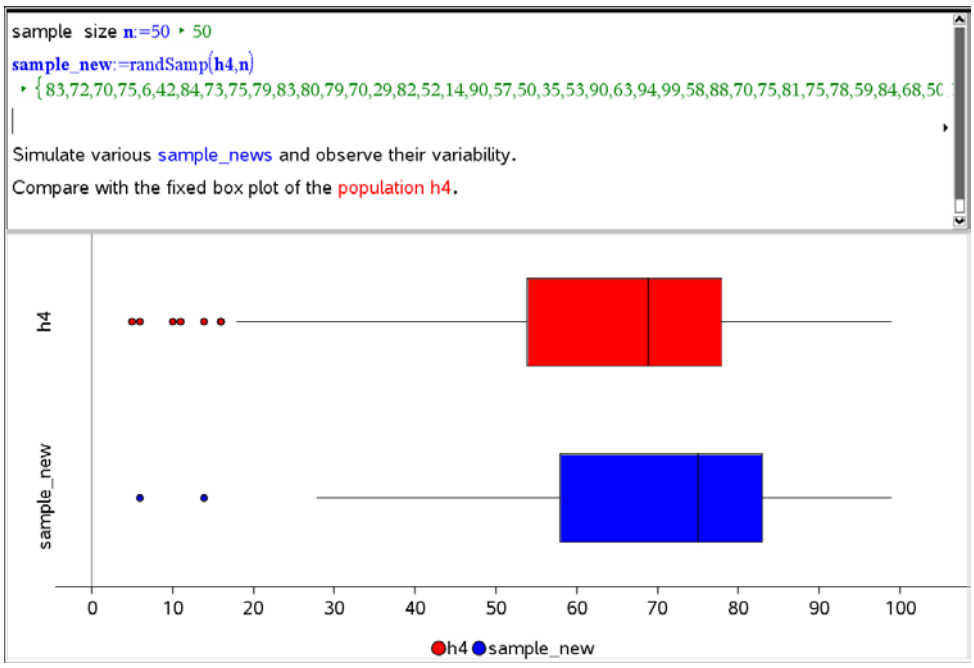
→ invnorm(a,μ,σ) = x with $a = P(X \le x)$

(5) Variability of samples in box plots

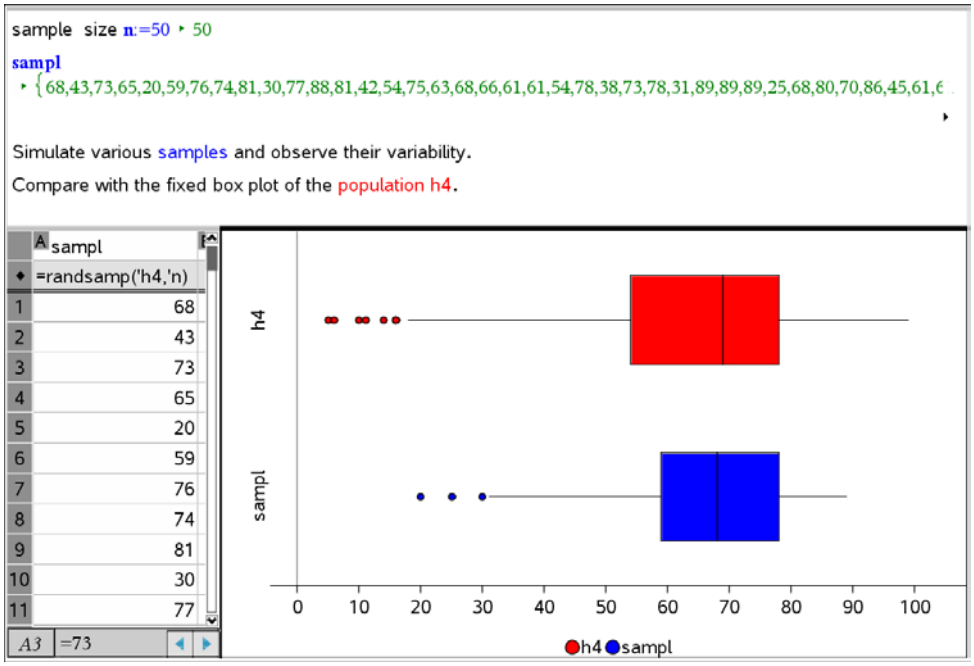
It is worth while to observe the variability of the samples by generating box plots for samples following in a row.

When taking small samples then the box plots should vary very much and should also differ from the population box plot significantly. Increasing the sample size the box plots of samples following each other are differing less and less. The random generated box plots shall also become more and more similar to the population box plot.

Take again h4 (367 life guards data) as population and take random samples of sample size **n**.



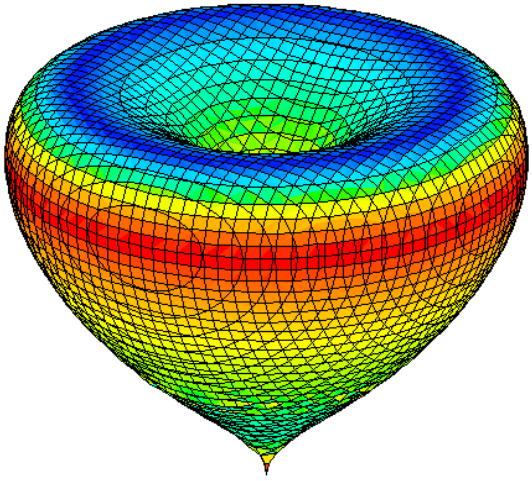
Random generated box plots with version 3.1 (above) and version 3.2 (below).

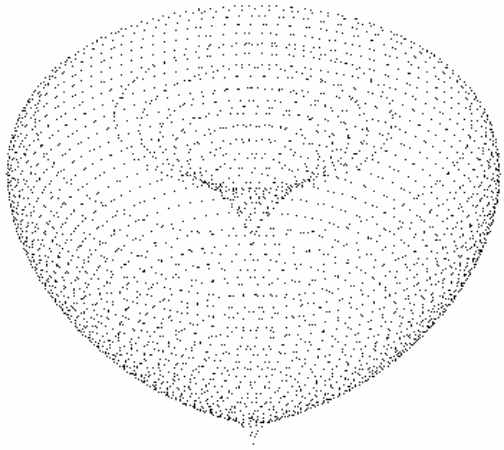


Next DNL: Part 4: Discovering Probability Distributions

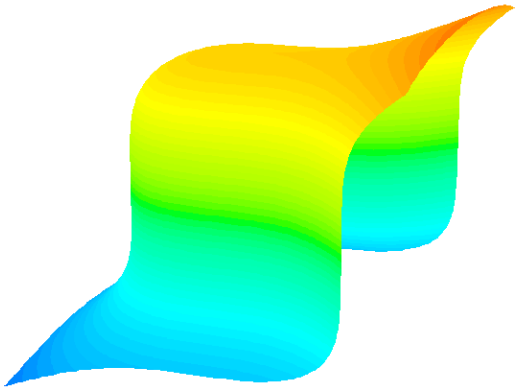
It was in DNL#71/72 when I presented some surfaces whose equations I found in the FAZ (Frankfurter Allgemeine Zeitung) some time ago. One empty page in this DNL is reason enough to have another three solids with DPGraph (left) and DERIVE (right), Josef

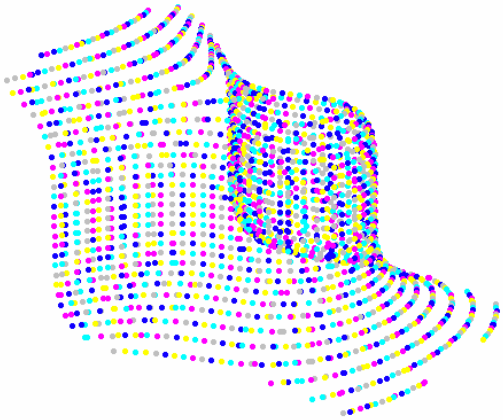
$$\text{VECTOR}\left(\text{ContourDots\_XY}\left(\frac{(x^2+y^2+z^2-1)^3}{10} = (x^2+y^2)\cdot z^3,\ 1,\ -2,\ 2,\ -2,\ 2,\ 0.1,\ 0.1\right),\ 1,\ -1,\ 3,\ 0.1\right)$$



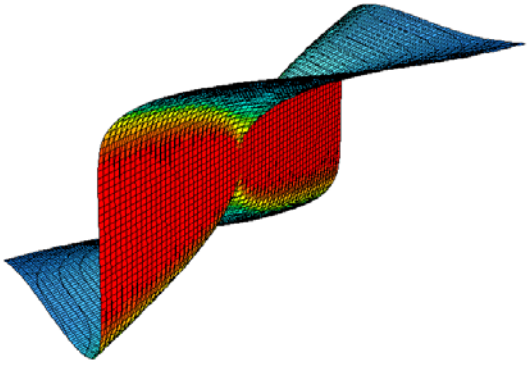


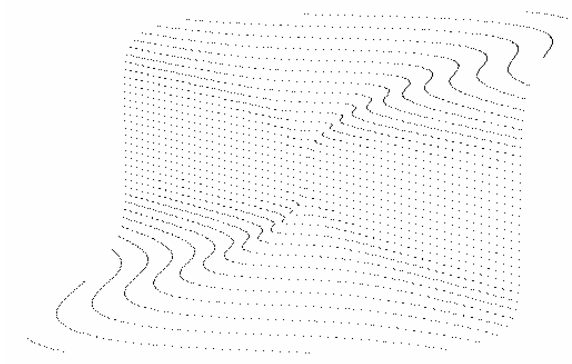
$$\text{VECTOR}(\text{ContourPts\_XY}(x^2+y^3+z^5 = 1,\ 1,\ -2,\ 2,\ -2,\ 2,\ 0.1,\ 0.1),\ 1,\ -2,\ 2,\ 0.1)$$





$$\text{VECTOR}(\text{ContourDots\_XY}(x^3+y^3-z^7,\ 1,\ -3,\ 3,\ -3,\ 3,\ 0.1,\ 0.1),\ 1,\ -3,\ 3,\ 0.1)$$





Overcoming Branch & Bound by Simulation

Josef Böhm

When I was a teacher at the St.Pölten College for Business Administration (Handelsakademie) I had to teach several years "*Planungsmathematik*" which is a possible German word for *Operations Research*. This was in the early 70ties. The curriculum covered topics as Linear Programming, Critical Path Methods, and heuristic methods as Branch & Bound. Using Branch & Bound we solved Travelling Salesman problems, Knapsack problems and Assignment problems. Very small problems can be solved by complete enumeration, i.e. trying all possible combinations and checking them for validity and for leading to a maximum or minimum of the given goal function.

All these problems are NP hard (they have no polynomial algorithm for solving). Bigger problems become very time consuming.

> Branch and Bound (B&B) is by far the most widely used tool for solving large scale *NP*-hard combinatorial optimization problems. B&B is, however, an algorithm paradigm, which has to be filled out for each specific problem type, and numerous choices for each of the components exist. Even then, principles for the design of efficient B&B algorithms have emerged over the years[*].

Guido Herweyers explains simulation performed with TI-Nspire. So I believe that it might be a good idea to demonstrate applying simulation for solving the problems mentioned above by simulation.

1 The Travelling Salesman Problem (*Rundreiseproblem*)

> Given a set of locations, and known distances between each pair of locations, the Travelling Salesman Problem is the problem of finding a tour that visits each location exactly once and that minimises the total distance travelled.

Let's start with 5 towns. The distances are given in the table below. The matrix needs not to be symmetric.

	A	B	C	D	E
A	–	5	10	6	7
B	5	–	8	8	5
C	10	8	–	8	5
D	6	8	8	–	7
E	7	5	5	7	–

It needs 5 miles from A to B, 10 miles from A to C, etc. We will find the shortest trip starting in A visiting all other places and then returning to A.

Route A-B-C-D-E-A would be 35 miles.

[1] http://www.diku.dk/OLD/undervisning/2003e/datV-optimer/JensClausenNoter.pdf
http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/c/Clausen:Jens.html
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.5.7475

We could do it in the enumerative way and check all possible $\frac{(n-1)!}{2} = 12$ roundtrips in our case. This is not so bad, but thinking on 20 cities to be visited gives 60822550204416000 possible tours! So enumeration is not the best choice to find a solution.

Question for students: Why are there $\frac{(n-1)!}{2}$ different routes possible?

You can find a lot of materials about algorithms to tackle this classic problem. Among the heuristic methods you will meet the Branch & Bound Method. I will present this method later.

Now my idea is as follows: I generate a series of random tours (= random permutations of the locations), calculate the respective mileage (cost, …) and then select the tour with the minimum value. Finally I will apply statistic means for estimating the quality of the "solution" which is in fact only an approximation.

My DERIVE approach:

```
#1:  [Precision := Approximate, Notation := Decimal, NotationDigits := 6]
```

$$\#2: \quad distances := \begin{bmatrix} 0 & 5 & 10 & 6 & 7 \\ 5 & 0 & 8 & 8 & 5 \\ 10 & 8 & 0 & 8 & 5 \\ 6 & 8 & 8 & 0 & 7 \\ 7 & 5 & 5 & 7 & 0 \end{bmatrix}$$

```
#3:  towns := [A, B, C, D, E]
```

I produce a random permutation of the "towns":

```
#4:  p := SORT(VECTOR([RANDOM(1), k], k, 5))
```

$$\#5: \quad p := \begin{bmatrix} 0.0270721 & 1 \\ 0.113457 & 3 \\ 0.368256 & 4 \\ 0.577532 & 5 \\ 0.995023 & 2 \end{bmatrix}$$

```
#6:  VECTOR(towns , k, p↓↓2) = [A, C, D, E, B]
              k
```

As I am missing the useful randsamp-command which is available for TI-Nspire I took the chance to define a `randperm()`-function for DERIVE. (See also page 31.)

```
#7:  randperm(l) := VECTOR(l   , k_, (SORT(VECTOR([RANDOM(1), k], k, DIM(l))))↓↓2)
                          k_
#8:  randperm(towns) = [C, B, A, D, E]
```

$$\#9: \quad VECTOR(randperm(towns), w, 5) = \begin{bmatrix} C & E & B & D & A \\ E & C & D & B & A \\ B & C & E & D & A \\ E & C & B & A & D \\ C & D & B & A & E \end{bmatrix}$$

But for now I will proceed on my travel route. I append the starting point because I have to finish the roundtrip and calculate the mileage which is again 35. Is this the shortest roundtrip?

```
#10:  [A, C, D, E, B]

#11:  [A, C, D, E, B, A]

      distances    + distances    + distances    + distances    + distances    = 35
#12:          1,3            3,4            4,5            5,2            2,1
```

It would be very boring to do this one route after the other. The solution is collecting all steps in one program:

```
      travel(dist, locs, numb, p, cost, way, dummy) :=
        Prog
          [dummy := RANDOM(0), simul := []]
          Loop
            p := (SORT(VECTOR([RANDOM(1), k], k, DIM(locs))))↓↓2
            p := APPEND(p, [p↓1])
#13:        way := VECTOR(locs↓k, k, p)
            cost := Σ(dist↓(p↓j)↓(p↓(j + 1)), j, 1, DIM(locs))
            way := APPEND(way, [cost])
            simul := APPEND(simul, [way])
            numb :- 1
            If numb = 0
              RETURN simul
```

I didn't include *simul* in the parameter list because it is a global variable which is necessary for my intention. *p*, *cost*, *way*, and *dummy* are local variables.

This is my first run of the program performing 5 simulations. Of course, It can happen that tours are appearing twice or even more often.

```
#14:  testrun := travel(distances, towns, 5)

                 [ A  B  E  C  D  A  29 ]
                 [                      ]
                 [ D  B  A  C  E  D  35 ]
                 [                      ]
#15:  testrun := [ B  C  E  A  D  B  34 ]
                 [                      ]
                 [ E  B  C  D  A  E  34 ]
                 [                      ]
                 [ A  C  D  E  B  A  35 ]
```

The first route is the shortest one with length 29.

We can assume that the lengths L of the routes follow a normal distribution[1]. First we calculate mean and standard deviation of the sample and then the probability $p(L \leq 29)$.

```
#16:  [AVERAGE(testrun↓↓7), STDEV(testrun↓↓7)] = [33.4, 2.50998]
```

$$\#17:\quad \text{NORMAL}\left(\frac{29 - 33.4}{2.50998}\right) = 0.0398008$$

There is a probability of only 4% to find a better solution, i.e. a shorter route combining all towns.

A sample of size 5 is too small. I increase the number of simulations 1000 random routes.

```
#18:  run1 := travel(distances, towns, 1000)

#19:  minimum(run) := MIN(run↓↓DIM(run↓1))

#20:  minimum(run1) = 29

#21:  [AVERAGE(simul↓↓7), STDEV(simul↓↓7)] = [34.376, 2.75503]
```

$$\#22:\quad \text{NORMAL}\left(\frac{29 - 34.38}{2.755}\right) = 0.0254208$$

When evaluating expression #19, the resulting matrix (1000 rows) is stored as global variable `simul`. I need `simul` for determining mean and standard deviation of the route lengths. If we would not take `simul` in #21 but `run1` instead then a new simulation would be performed. If you try `dim(simul)=` you will get immediately 1000 but when simplifying `dim(run1)=` it will need some time to first perform internally the next 1000 simulations and then show the number of rows = 1000.

The probability to find a better solution has decreased to 2.5% which is pretty good.

$$\#23:\quad \text{DIM}(\text{SELECT}(v_7 = 29, v, simul)) = 87$$

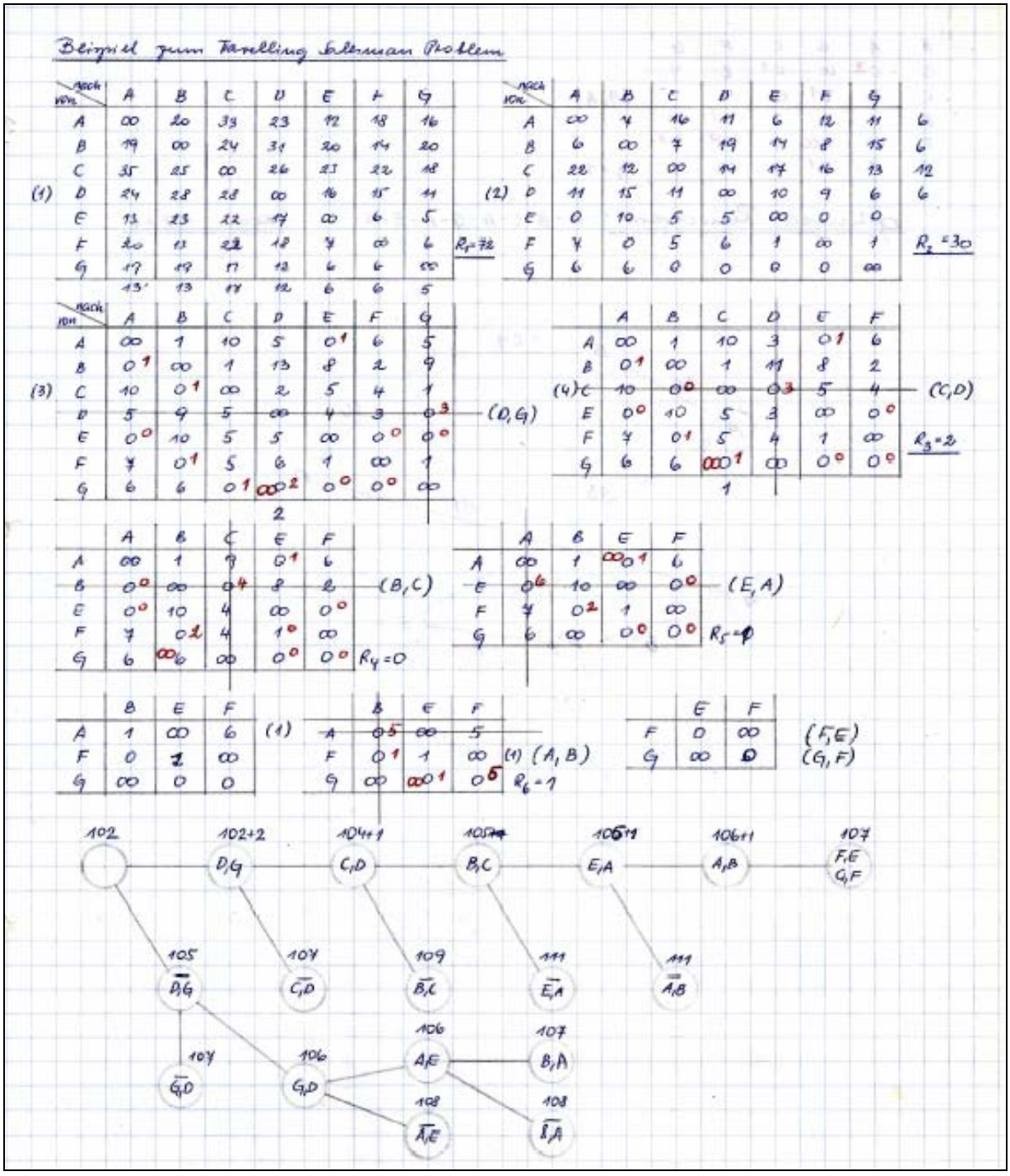
$$\#24:\quad (\text{SELECT}(v_7 = 29, v, simul))_{[1,\,2,\,3]} = \begin{bmatrix} A & D & C & E & B & A & 29 \\ C & E & B & A & D & C & 29 \\ C & E & B & A & D & C & 29 \end{bmatrix}$$

There are 87 routes with minimal length 87 which need not be all the same. I'd like to see the first three of them and am finding out that they all are the same. Maybe that there is at least one another one among the remaining 84 solutions. But we cannot be absolutely sure that 29 is the shortest possible route. It remains an approximation.

Selecting the minimum together with returning the optimal route could also be included into the program `travel`. I did without in the first application but I will do this for the other problems.

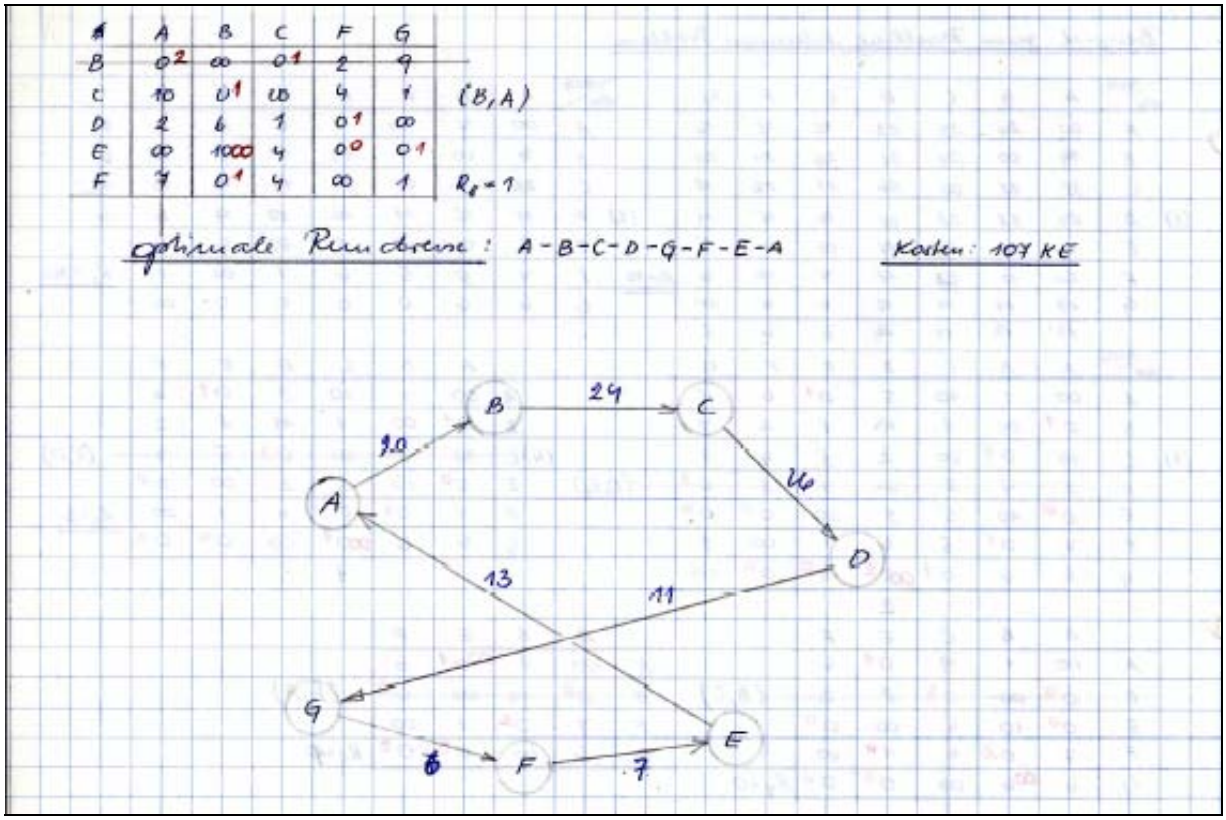
You will see that my programming style will improve stepwise. This could also be done with students. I remember that at similar occasions the students had ideas what should – and often really could – be improved

Let's have a problem with more locations. This is a problem I used as teacher demonstrating the Branch & Bound Method. As I said above this was in the early 70ties. I will show my original "preparation" provided for the classroom, doing all calculations by hand. The distance matrix – which is now defined as a cost matrix – is not symmetric and there are 7 locations.



The last step performed on the next page shows that the branch below cannot provide a better solution than 107.

The optimal route is A-B-C-D-G-F-E-A with minimal cost 107.



The DERIVE procedure:

$$\#27:\quad costm := \begin{bmatrix} \infty & 20 & 33 & 23 & 12 & 18 & 16 \\ 19 & \infty & 24 & 31 & 20 & 14 & 20 \\ 35 & 25 & \infty & 26 & 23 & 22 & 18 \\ 24 & 28 & 28 & \infty & 16 & 15 & 11 \\ 13 & 23 & 22 & 17 & \infty & 6 & 5 \\ 20 & 13 & 22 & 18 & 7 & \infty & 6 \\ 19 & 19 & 17 & 12 & 6 & 6 & \infty \end{bmatrix}$$

#28: offs := [A, B, C, D, E, F, G]

#29: run2 := travel(costm, offs, 1000)

#30: minimum(run2) = 107

#31: sel(val) := SELECT(v, v = val, simul, DIM(simul↓1))

#32: sel(107) = [[A, B, C, D, G, F, E, A, 107]]

#33: [AVERAGE(simul↓↓9), STDEV(simul↓↓9)] = [128.66, 8.53107]

$$\#34:\quad NORMAL\left(\frac{107 - 128.66}{8.53}\right) = 0.00555415$$

The selection of all routes with a given value (107) is done by a function `sel(val)`. Comparing with my solution found in the last century we can see the correspondence.

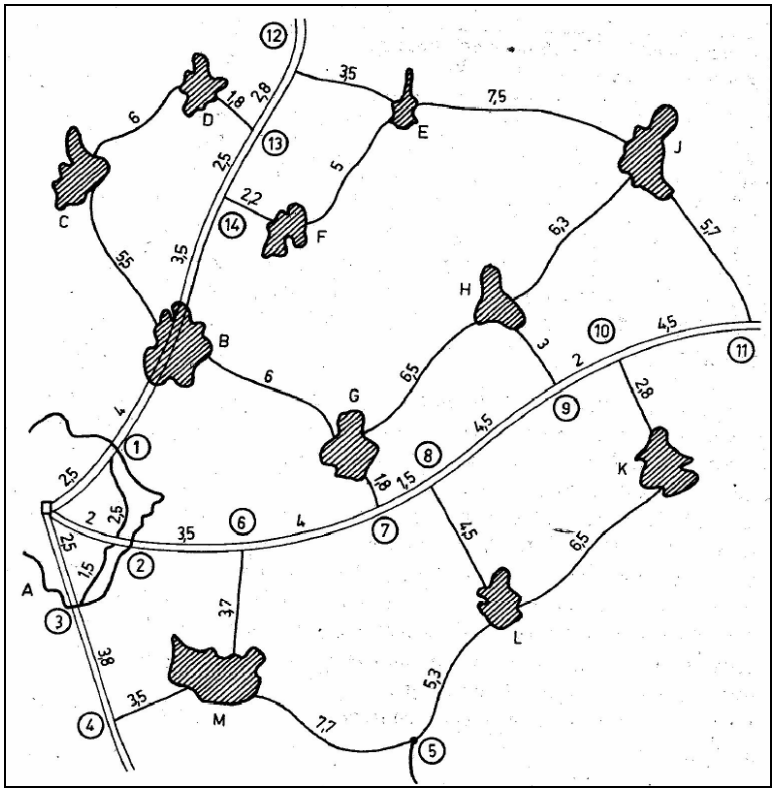
The probability of 0.5% finding a shorter path is sufficiently small to accept 107 miles at least as a good approximation – even without knowing the exact result – as we do here.

Finally let me show a more extended travel route as given in [2]:

This is the map with the distances between villages A to M. We need to find the shortest path connecting all places starting and ending in town A.

The first problem is to find out the shortest connections in pairs between all villages – considering all nodes. We will assume that this problem has been solved for using a special algorithm (Dantzig).

The distance matrix `trm` is given.



$$trm := \begin{bmatrix} \_ & 6.5 & 12 & 14.3 & 17.2 & 12.2 & 11.3 & 17.8 & 24.1 & 20.3 & 15.5 & 9.2 \\ 6.5 & \_ & 5.5 & 7.8 & 10.7 & 5.7 & 6 & 12.2 & 18.2 & 18.6 & 13.8 & 13.7 \\ 12 & 5.5 & \_ & 6 & 14.1 & 11.2 & 11.5 & 17.7 & 20.3 & 24.1 & 19.3 & 19.2 \\ 14.3 & 7.8 & 6 & \_ & 8.1 & 6.5 & 13.8 & 13 & 15.6 & 20.8 & 21.6 & 21.5 \\ 17.2 & 10.7 & 14.1 & 8.1 & \_ & 5 & 16.7 & 11.5 & 7.5 & 19.3 & 23.5 & 24.4 \\ 12.2 & 5.7 & 11.2 & 6.5 & 5 & \_ & 11.7 & 6.5 & 12.5 & 14.3 & 18.5 & 19.4 \\ 11.3 & 6 & 11.5 & 13.8 & 16.7 & 11.7 & \_ & 6.5 & 12.8 & 12.6 & 7.8 & 9.5 \\ 17.8 & 12.2 & 17.7 & 13 & 11.5 & 6.5 & 6.5 & \_ & 6.3 & 7.8 & 12 & 16 \\ 24.1 & 18.2 & 20.3 & 15.6 & 7.5 & 12.5 & 12.8 & 6.3 & \_ & 13 & 18.3 & 22.3 \\ 20.3 & 18.6 & 24.1 & 20.8 & 19.3 & 14.3 & 12.6 & 7.8 & 13 & \_ & 6.5 & 18.5 \\ 15.5 & 13.8 & 19.3 & 21.6 & 23.5 & 18.5 & 7.8 & 12 & 18.3 & 6.5 & \_ & 13 \\ 9.2 & 13.7 & 19.2 & 21.5 & 24.4 & 19.4 & 9.5 & 16 & 23.3 & 18.5 & 13 & \_ \end{bmatrix}$$

vills := [A, B, C, D, E, F, G, H, J, K, L, M]

11 villages give 1 814 400 possible routes. Which of them is the shortest one? I run three time 10000 simulations:

```
#38:  run3 := travel(trm, vills, 10000)
#39:  minimum(run3) = 111.5
#40:  1500 sec
#41:  minimum(run3) = 111
#42:  minimum(run3) = 104.5
#43:  sel(104.5) = [[C, B, E, J, H, L, K, G, M, A, F, D, C, 104.5]]
#44:  [AVERAGE(simul↓↓14), STDEV(simul↓↓14)] = [165.684, 15.7818]
```

$$\#45:\quad \text{NORMAL}\left(\frac{104.5 - 165.68}{15.78}\right) = 0.0000528632$$

The best result is 104.5 miles. Calculation gives a very small chance to find a better one.

Starting in A the route is: A – F – D – C – B – E – J – H – L – K – G – M – A.

Preparing the DNL I tried once more hoping to find a better approximation:

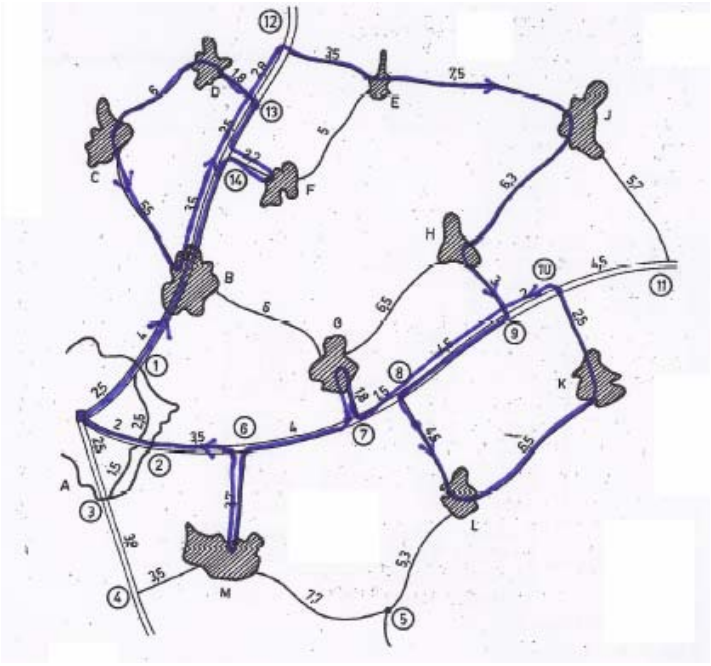
```
#47:  minimum(run3) = 100.1
#48:  [AVERAGE(simul↓↓14), STDEV(simul↓↓14)] = [165.433, 15.6341]
```

$$\#49:\quad \text{NORMAL}\left(\frac{100.1 - 165.43}{15.63}\right) = 0.0000145894$$

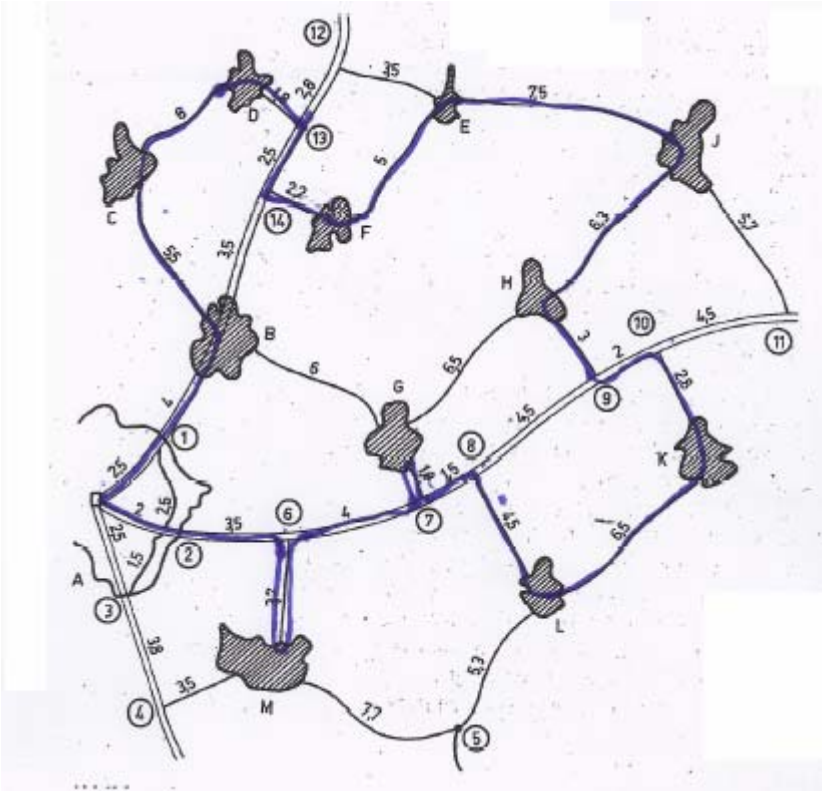
```
#50:  sel(100.1) = [[E, F, H, J, K, L, G, B, A, M, C, D, E, 100.1]]
```

The approximation given in [2] is much better. After some pages following a tiresome algorithm the approximation leads to a total distance of 84.1 when using the route A-B-C-D-F-E-J-H-K-L-G-M-A.

The first figure shows the graph connected with the route length 104.5.



The next one gives the route with total length 84.1.

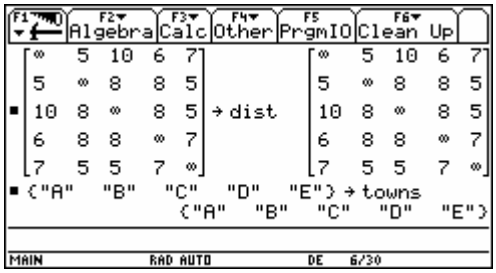


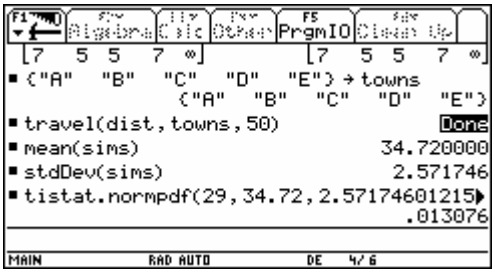
Even be inspection we recognize that this is a much better solution. So we would need more simulations to obtain a shorter path? Maybe that we would be lucky with the next single simulation. The reader is invited to follow the route with length 100.1.

The Travelling Salesman problem tackled with the TI-92 / Voyage 200:

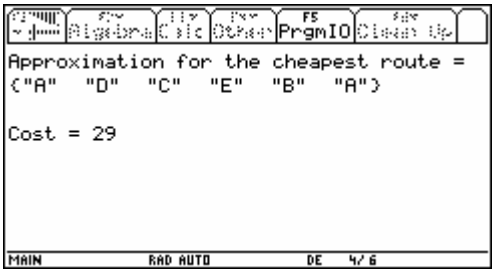
```
travel(dis,loc,numb)
Prgm
Local  i,p,c0,cost,way,li1,route
ClrIO
{}→sims
Σ(dis[j,j+1],j,1,dim(dis)[1]-1)→c0
c0+dis[dim(dis)[1],1]→c0
For  i,1,numb
seq(k,k,1,dim(loc))→p:seq(rand(),k,1,dim(loc))→li1
SortA  li1,p:augment(p,{p[1]})→p
Σ(dis[p[j],p[j+1]],j,1,dim(p)-1)→cost
augment(sims,{cost})→sims
If cost<c0 Then
   cost→c0:p→way
EndIf
EndFor
seq(loc[way[i]],i,1,dim(way))→way
Disp  "Approximation for the cheapest route = "
Disp  way
Disp  ""
Disp  "Cost = "&string(c0)
EndPrgm
```

As you can see it works pretty well.

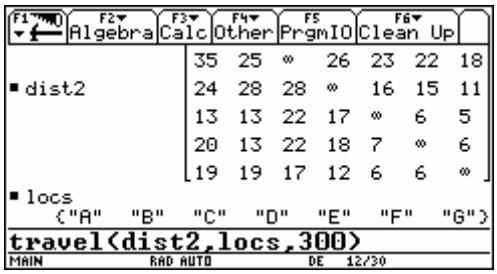


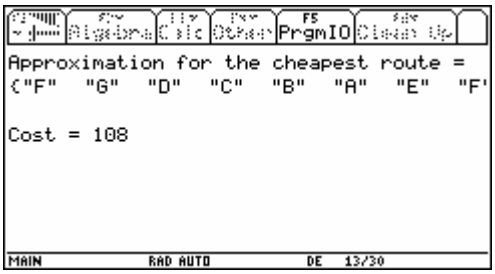


The route produced by this simulation (50 tries) gives the optimal solution.



I tried the second problem – with 7 locations:





Not so bad!!

[1] Autorenkollektiv, *Mathematische Standardmodelle der Operationsforschung*, Verlag Die Wirtschaft Berlin 1972.

[2] Autorenkollektiv, *Mathematische Standardmodelle der Operationsforschung, Aufgabensammlung*, Verlag Die Wirtschaft Berlin 1972.

http://en.wikipedia.org/wiki/Travelling_salesman_problem

http://de.wikipedia.org/wiki/Problem_des_Handlungsreisenden

http://www.tsp.gatech.edu/

http://mathworld.wolfram.com/TravelingSalesmanProblem.html

http://www.lancs.ac.uk/staff/letchfoa/talks/TSP.pdf

http://elib.zib.de/pub/UserHome/Groetschel/Spektrum/index2.html

2 The Assignment Problem (*Zuordnungsproblem*)

There are a number of agents and a number of tasks. Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment. It is required to perform all tasks by assigning exactly one agent to each task in such a way that the total cost of the assignment is minimized.

```
assign(ass, n, k, d, rows, cols, cn, rn, val, mx, mn, mxv, mnv, dummy) :=
  Prog
    [dummy := RANDOM(0), k := 1, d := DIM(ass), simul := []]
    [cn := REST(ass↓1), rn := REST(ass'↓1)]
    ass := VECTOR(VECTOR(ass↓i_↓j_, j_, 2, d), i_, 2, d)
    d := DIM(ass)
    Loop
      If k > n exit
      rows := VECTOR(k_, k_, d)
#1:   cols := (SORT(VECTOR([RANDOM(1), k_], k_, d))↓↓2
      val := Σ(ass↓i↓(cols↓i), i, 1, d)
      simul := APPEND(simul, [APPEND(cols, [val])])
      k :+ 1
    [mn := MIN(simul↓↓(d + 1)), mx := MAX(simul↓↓(d + 1))]
    mnv := (SELECT(v↓(d + 1) = mn, v, simul))↓1
    mxv := (SELECT(v↓(d + 1) = mx, v, simul))↓1
    mn := ["MIN:", [rn, VECTOR(cn↓k_, k_, mnv↓[1, ..., d])], mn]
    mx := ["MAX:", [rn, VECTOR(cn↓k_, k_, mxv↓[1, ..., d])], mx]
    [mn; mx; "all results in simul"]
```

The cost matrix – 6 projects are to be assigned to 6 agents is given by:

#2: ass1 :=

	P1	P2	P3	P4	P5	P6
A1	10	15	8	24	12	5
A2	4	20	6	10	8	5
A3	20	24	10	15	12	12
A4	15	20	12	15	10	14
A5	30	25	20	20	25	15
A6	12	15	15	12	15	10

We can also interpret the matrix as a profit matrix because the agents perform their tasks more or less efficient. Then the maximum assignment is required. My program `assign(matrix, number of simulations)` is looking for minimum and maximum as well.

Additionally I have included all steps leading to the final answers (fixing min and max, selecting the respective assignments and providing a nicely formatted output of the results).

#3: assign(ass1, 10) =

MIN:	A1	A2	A3	A4	A5	A6
	P5	P3	P6	P1	P2	P4
82						
MAX:	A1	A2	A3	A4	A5	A6
	P6	P5	P2	P4	P1	P3
97						
all results in simul						

Having performed 10 simulations which will very likely not give good approximations, I will try 2000 randomly chosen assignments. This needs only 9.7 seconds:

```
#4:   assign(ass1, 2000) = [ [MIN:, [ A1 A2 A3 A4 A5 A6 ], 64]
                           [       [ P6 P1 P3 P5 P4 P2 ]     ]
                           [MAX:, [ A1 A2 A3 A4 A5 A6 ], 118]
                           [       [ P4 P2 P1 P6 P5 P3 ]     ]
                           [          all results in simul   ]

#5:   needs 9.7 sec
```

This is much better now.

The probabilities to find better solutions are very small.

```
#6:   [AVERAGE(simul↓↓13), STDEV(simul↓↓13)]

#7:   [86.7745, 9.074074011]
```

$$\#8: \quad \text{NORMAL}\left(\frac{64 - 86.77}{9.07}\right) = 0.006028458473$$

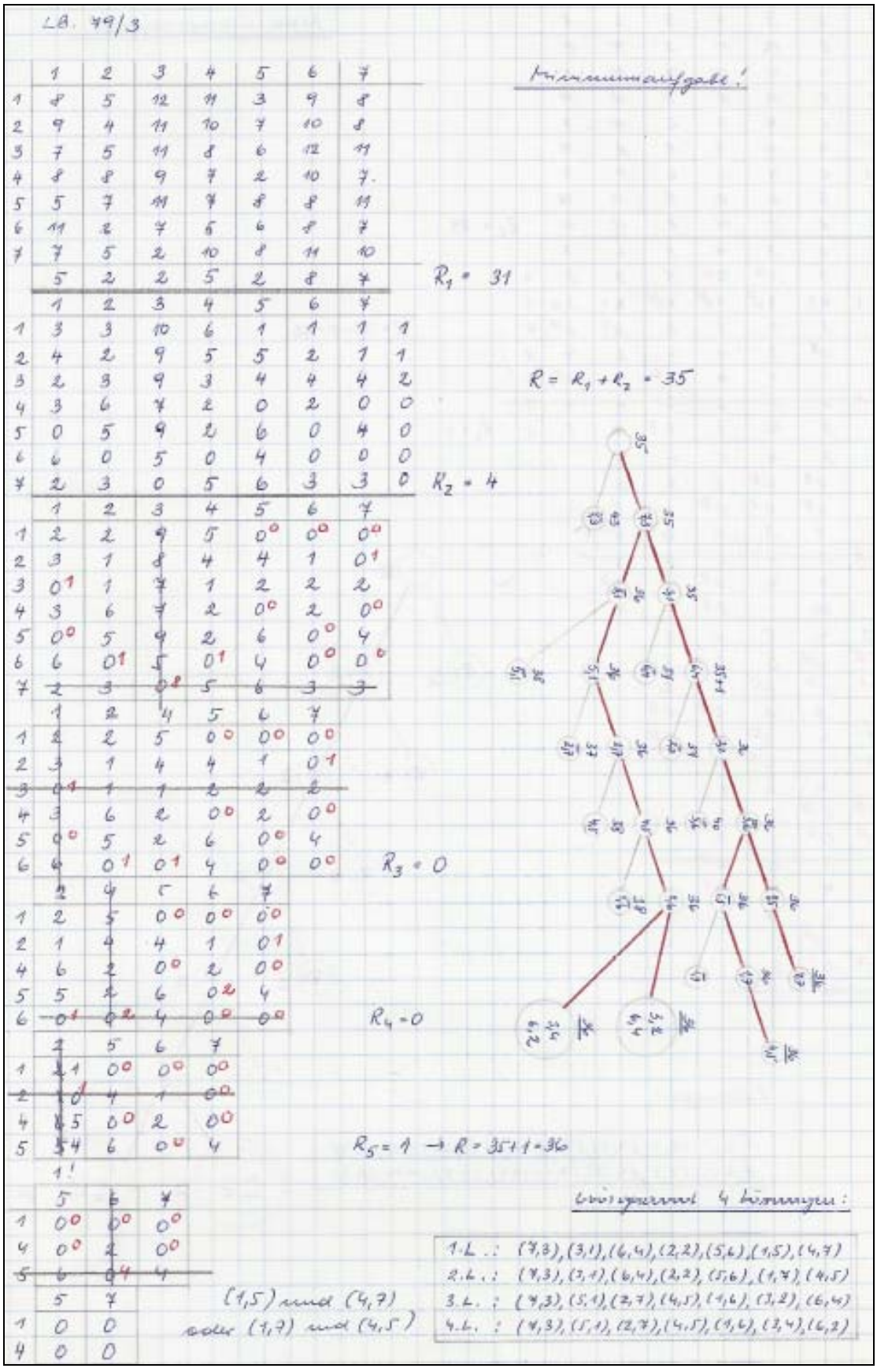
$$\#9: \quad 1 - \text{NORMAL}\left(\frac{118 - 86.77}{9.07}\right) = 0.0002874163774$$

Here is a second example from the 70ties-textbook. Seven jobs shall be assigned to seven machines.

The cost/profit-matrix is given by:

```
ass2 :=
[     J1  J2  J3  J4  J5  J6  J7 ]
[ M1   8   5  12  11   3   9   8 ]
[ M2   9   4  11  10   7  10   8 ]
[ M3   7   5  11   8   6  12  11 ]
[ M4   8   8   9   7   2  10   7 ]
[ M5   5   7  11   7   8   8  11 ]
[ M6  11   2   7   5   6   8   7 ]
[ M7   7   5   2  10   8  11  10 ]
```

Looking back to my 40 years old papers I see that there are more optimal assignments possible. This reminiscence gives the opportunity for demonstrating the Branch & Bound method once more.



We see that the manual procedure gives four solutions for the minimum problem – and there are two solutions for the maximum problem. I changed my program `assign` in order to return not only one solution – but I cannot be sure that we will find all possible solutions.

#16: assign_more(ass2, 10000) =

		M1	M2	M3	M4	M5	M6	M7	
MIN:		6	7	2	5	1	4	3	
		6	7	4	5	1	2	3	36
		5	2	1	7	6	4	3	
		7	2	1	5	6	4	3	

		M1	M2	M3	M4	M5	M6	M7	
MAX:		4	3	6	2	7	1	5	
		3	4	6	2	7	1	5	
		4	3	6	2	7	1	5	72
		4	3	6	2	7	1	5	
		3	4	6	2	7	1	5	

all results in simul

#17: 228 sec

This took me 228 seconds. Comparing with my papers from the past, I can say that all optimal assignments for minimum and maximum as well have been found by simulation. Look at the probabilities!

[AVERAGE(simul↓↓8), STDEV(simul↓↓8)] = [54.5964, 5.657005145]

$$\left[\text{NORMAL}\left(\frac{36 - 54.6}{5.66}\right),\ 1 - \text{NORMAL}\left(\frac{72 - 54.6}{5.66}\right)\right] = [0.0005077100026,\ 0.001055321911]$$

http://faculty.ksu.edu.sa/jkhan/Documents/OR/opt-lp5.pdf

http://cse.spsu.edu/bmorrison/cs4413_ppt/branch&bound.ppt

Josef, Karsten Schmidt and Carl Leinbach (Tartu 2012)

3 The Knapsack Problem (*Rucksackproblem*)

Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items.

Example:

Let's assume there is a burglar. He finds 6 items worthwhile to take them. They differ in weight and value. He is able to carry not more than 65 pounds. What is his best choice?

Item *i*	A1	A2	A3	A4	A5	A6
Weight w_i	30	20	8	25	12	15
Value v_i	25	18	10	20	15	8

There is a recursive algorithm. The respective recursion formula is given by:

$$f(k,g)=\begin{cases} f(k-1,g) & \text{if } w_k > g \\ \max(v_k + f(k-1,g-w_k), f(k-1,g)) & \text{if } w_k \le g \text{ and } k>0 \end{cases}$$

The last values are $k = n$ (which is 6 in our case, and $g = M = 65$).

The result of the procedure should be $f(6,65) = 63$.

```
#1:   simul :=

      knapsack(supply, restr, n, i, z, art, amt, vals, m, r, ind, sel, sg, sv, mx, dummy) :=
        Prog
          [dummy := RANDOM(0), simul := []]
          i := 1
          [art := supply↓1, amt := supply↓2, vals := supply↓3]
          [m := DIM(art), mx := 1]
          Loop
            ind := VECTOR(k, k, m)
            If i > n exit
            r := RANDOM(m - 1) + 2
            z := RANDOM(r) + 1
            sel := [z]
#2:         Loop
              If DIM(sel) = r exit
              ind := DELETE(ind, z)
              z := IF(DIM(ind) = 1, 1, RANDOM(DIM(ind)) + 1)
              sel := APPEND(sel, [ind↓z])
            [sg := Σ(VECTOR(amt↓k, k, sel)), sv := Σ(VECTOR(vals↓k, k, sel))]
            If sg ≤ restr ∧ sv ≥ mx
               Prog
                 simul := APPEND(simul, [[SORT(VECTOR(art↓k, k, sel)), sg, sv]])
                 mx := sv
            i :+ 1
          simul := SELECT(v↓3 = mx, v, simul)
          APPEND([["Goods", "Total Weight", "Total Value"]], simul)
```

See three series of simulations followed by the original binary B & B from my OR-teaching.

#3: example :=

A1	A2	A3	A4	A5	A6
30	20	8	25	12	15
25	18	10	20	15	8

#4: knapsack(example, 65, 10) =

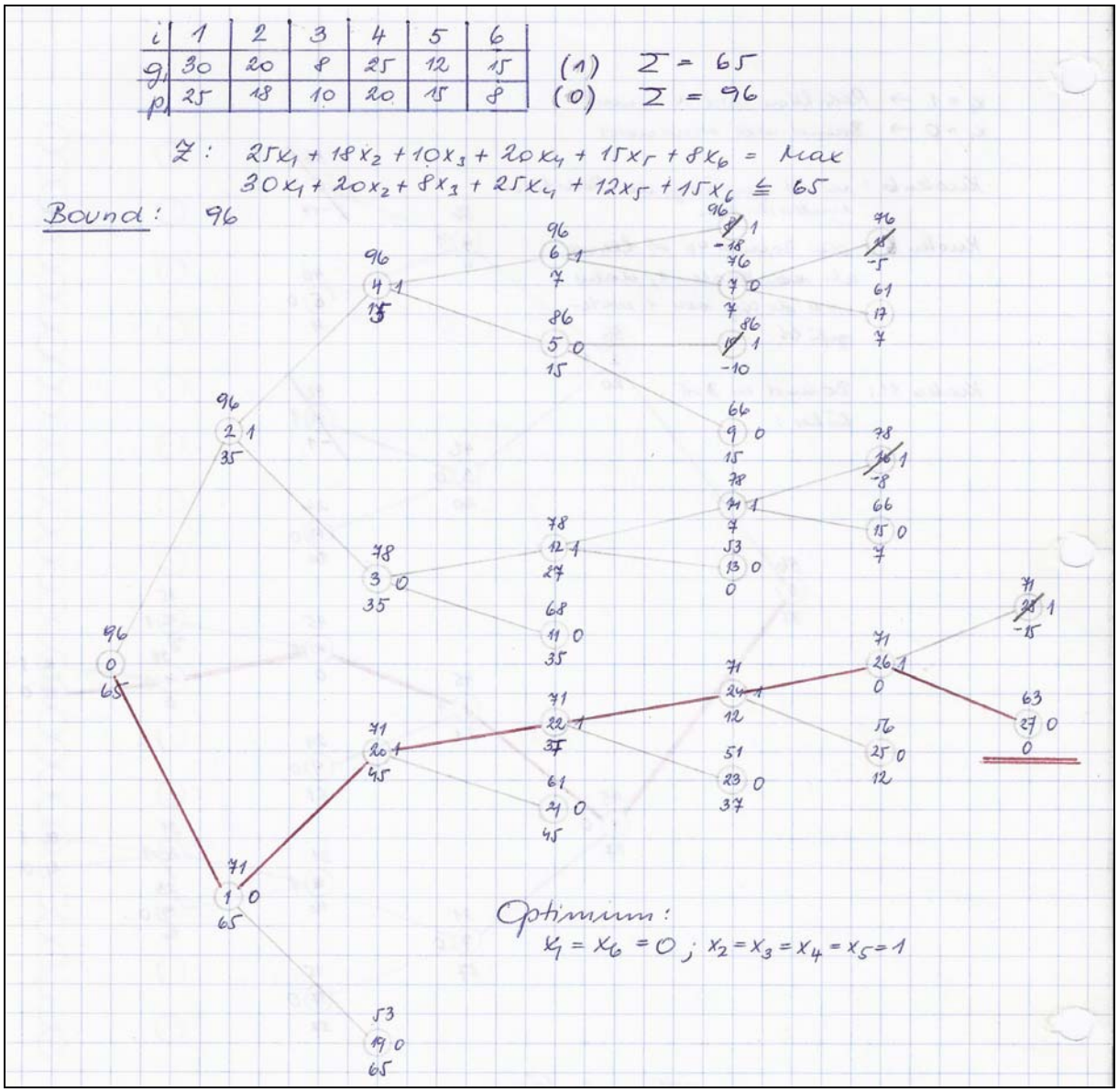
Goods	Total Weight	Total Value
[A1, A3, A4]	63	55

#5: knapsack(example, 65, 10) =

Goods	Total Weight	Total Value
[A2, A3, A5, A6]	55	51

#6: knapsack(example, 65, 100) =

Goods	Total Weight	Total Value
[A2, A3, A4, A5]	65	63
[A2, A3, A4, A5]	65	63
[A2, A3, A4, A5]	65	63



This is a so called (0,1)-problem: Take it (= 1) or do not (= 0). So we have 2^6 possible choices from (0,0,0,0,0,0) – leave all items – to (1,1,1,1,1,1) – take them all. It is left to the reader to write an enumerative procedure to check all possibilities (= end of all branches) for validity (total weight not more than 65 pounds) and for maximal total value.

I transferred the Travelling Salesman problem to the TI-92 / Voyage 200. Now I will tackle the Knapsack problem with TI-NspireCAS.

The problem is from my old textbook again. It is a really "Knapsack = Rucksack" problem:

A hiker packs his rucksack. He can to choose between eight victuals.

Beef	Pork	Lard	Butter	Cheese	Bread	Chocolade	Apples
2	2	1	1	2	4	0.5	2
2600	8000	9200	8000	8200	10000	2300	1200

The second row gives the weight (kg), the third one shows the calorie content. The hiker doesn't want to carry more than 10 kg.

Find the selection with maximum calorie content under the condition that the eatables are indivisible.

This is the TI-Nspire program:

knapsack

```
Define knapsack(supply,restr,n)=
Prgm
Local i,z,simul,art,amt,vals,ind,m,r,sel,sw,sv,sart,mx,sols
simul:=["Goods:"  "Total Weight"  "Total Value"]
sols:=simul
art:=mat▸list(supply[1]):amt:=mat▸list(supply[2]):vals:=mat▸list(supply[3])
mx:=1:m:=dim(art)
For i,1,n
ind:=seq(k,k,1,m):r:=randInt(2,m)
sel:=randSamp(ind,r,1):SortA sel
sw:=Σ(amt[sel[j]],j,1,dim(sel)):sv:=Σ(vals[sel[j]],j,1,dim(sel))
sart:=string(seq(art[sel[j]],j,1,dim(sel)))
If sw≤restr and sv≥mx Then
simul:=colAugment(simul,[sart  sw  sv]):mx:=sv
EndIf
EndFor
For i,2,dim(simul)[1]
If simul[i,3]=mx Then
sols:=colAugment(sols,simul[i])
EndIf
EndFor
Disp sols
EndPrgm
```

"{"A1","A3","A5","A6"}"	65	58
"{"A1","A2","A5"}"	62	58

Done

knapsack(*example*,65,100)

"Goods:"	"Total Weight"	"Total Value"
"{"A2","A3","A4","A5"}"	65	63
"{"A2","A3","A4","A5"}"	65	63
"{"A2","A3","A4","A5"}"	65	63

Done

hike:=

"Beef"	"Pork"	"Lard"	"Butter"	"Cheese"	"Bread"	"Chocolade"	"Apples"
2	2	1	1	2	4	0.5	2
2600	8000	9200	8000	8200	10000	2300	1200

"Beef"	"Pork"	"Lard"	"Butter"	"Cheese"	"Bread"	"Chocolade"	"Apples"
2	2	1	1	2	4	0.5	2
2600	8000	9200	8000	8200	10000	2300	1200

knapsack(*hike*,10,100)

"Goods:"	"Total Weight"	"Total Value"
"{"Beef","Pork","Lard","Butter","Bread"}"	10	37800

Done

knapsack(*hike*,10,100)

"Goods:"	"Total Weight"	"Total Value"
"{"Beef","Lard","Butter","Cheese","Bread"}"	10	38000

Done

knapsack(*hike*,10,500)

"Goods:"	"Total Weight"	"Total Value"
"{"Pork","Lard","Butter","Cheese","Bread"}"	10	43400
"{"Pork","Lard","Butter","Cheese","Bread"}"	10	43400

Done

And what is DERIVE recommending?

knapsack(hike, 10, 100) =

Goods	Total Weight	Total Value
[Beef, Bread, Butter, Lard, Pork]	10	37800
[Beef, Bread, Butter, Lard, Pork]	10	37800

knapsack(hike, 10, 500) =

Goods	Total Weight	Total Value
[Bread, Butter, Cheese, Lard, Pork]	10	43400

Following the result of the first run with only 100 simulations performed we would miss 5600 calories. Running 500 simulations needs only 1 second and gives a much better result.

```
      knapsack(supply, restr, n, i, z, art, amt, vals, m, r, ind, sel, sg, sv, mx, dummy) :=
        Prog
          [dummy := RANDOM(0), simul := []]
          i := 1
          [art := supply↓1, amt := supply↓2, vals := supply↓3]
          [m := DIM(art), mx := 0]
          Loop
            ind := VECTOR(k, k, m)
            If i > n exit
            r := RANDOM(m - 1) + 2
            z := RANDOM(r) + 1
            sel := [z]
#1:         Loop
              If DIM(sel) = r exit
              ind := DELETE(ind, z)
              z := IF(DIM(ind) = 1, 1, RANDOM(DIM(ind)) + 1)
              sel := APPEND(sel, [ind↓z])
            [sg := ∑(VECTOR(amt↓k, k, sel)), sv := ∑(VECTOR(vals↓k, k, sel))]
            If sg ≤ restr ∧ sv ≥ mx
               Prog
                 simul := APPEND(simul, [[SORT(VECTOR(art↓k, k, sel)), sg, sv]])
                 mx := sv
            i :+ 1
          simul := SELECT(v↓3 = mx, v, simul)
          APPEND([["Goods", "Total Weight", "Total Value"]], simul)
```

I found in Müller-Merbach[1] an economic application: An investor wants to invest 1300 monetary units (say 1000 €). He has the choice of 7 indivisible investments with different returns of investment (also given in 1000 €). How should he put together his portfolio?

```
#6:   stocks := ⎡  A    B    C    D    E    F    G  ⎤
                ⎢                                   ⎥
                ⎢ 500  500  400  300  200  100  100 ⎥
                ⎢                                   ⎥
                ⎣ 35   31   26   20   18    8    6  ⎦

#7:   knapsack(stocks, 1300, 50) = ⎡   Goods    Total Weight  Total Value ⎤
                                   ⎣ [A, B, D]      1300          86      ⎦

#8:   knapsack(stocks, 1300, 100) = ⎡     Goods     Total Weight  Total Value ⎤
                                    ⎣ [A, B, E, F]      1300          92      ⎦

#9:   knapsack(stocks, 1300, 300) = ⎡      Goods       Total Weight  Total Value ⎤
                                    ⎢                                          ⎥
                                    ⎢ [A, C, E, F, G]      1300          93      ⎥
                                    ⎢                                          ⎥
                                    ⎢ [A, C, E, F, G]      1300          93      ⎥
                                    ⎢                                          ⎥
                                    ⎣ [A, C, E, F, G]      1300          93      ⎦

      (knapsack(stocks, 1300, 1000))                                  = [[A, C, E, F, G], 1300, 93]
#10:                                 2
```

Our recommendation is: buy stocks A, C, E, F and G and make a profit of 93 000 €!!

[1] Müller-Merbach, *Operations Research*, Vahlen 1971

[2] *The Operations Research Problem Solver*, REA, New York 1985

http://www.diku.dk/users/pisinger/95-1.pdf

http://en.wikipedia.org/wiki/Knapsack_problem

http://www.es.ele.tue.nl/education/5MC10/Solutions/knapsack.pdf

http://www.diku.dk/users/pisinger/95-1.pdf

http://www.nuigalway.ie/mat/algorithms/knapsack.html
(a program!!)

http://www.cs.ucsb.edu/~jacopo/BB/
(a Branch & Bound Solver)

I'd like to add two comments:

(1) I really like to undertake extended hikes. But I never had the idea to carry lard with me. It might be difficult enough to have butter in the rucksack climbing up a mountain on its sunny side in summer time …. And I never had beef and pork with me. And why should I not be able to divide butter, bread, cheese, …? Believe it or not this is the original text from the textbook.

(2) I also like programming and exploring the advantages and disadvantages of various programming languages and various programmable computer algebra systems. I missed Nspire's **randSamp** function. So what to do? Easy answer: make your own DERIVE randSamp-function and benefit from DERIVE's feature to implement "real" random processes without needing a "randseed" using its RANDOM(0) command.

4 randSamp & co

Let's start with a random permutation of *n* elements. I wantd to avoid simplifying RANDOM(0) at the beginning of each session so I included this command in the function in order to have every time another initial value for the random number generating process.

```
#1:   test := [A, B, C, D, E, F, G, H]

      randp(l, dummy) :=
        Prog
#2:       dummy := RANDOM(0)
          VECTOR(l↓k_, k_, (SORT(VECTOR([RANDOM(1), k], k, DIM(l))))↓↓2)

#3:   randp(test) = [E, A, H, G, C, F, D, B]
```

This looked pretty good. I proceeded with a sequence of three random permutations of the set `test` in a row.

$$\#4:\quad \text{VECTOR(randp(test), k, 3)} = \begin{bmatrix} A & B & D & F & E & C & G & H \\ A & B & D & F & E & C & G & H \\ A & B & D & F & E & C & G & H \end{bmatrix}$$

I was very disappointed that the same permutation is appearing three times!

Ok, I'll come back to this problem later. We wanted to have random samples – with and without replacement (of the elements).

```
#6:   randsamp(test, 5) = [F, A, G, E, E]

#7:   randsamp(test, 5) = [E, H, F, E, F]

#8:   randsamp(test, 5, 1) = [G, D, H, B, A]

#9:   randsamp(test, 5, 1) = [E, F, G, B, C]
```

#10: VECTOR(randsamp(test, 5, 1), k, 3) =

B	A	E	H	C
B	A	E	H	C
B	A	E	H	C

The same problem as above occurred. One function call after the other worked properly, a sequence of random samples seemed not to be really randomly!! `randsamp(population,n)` generates a sample of *n* elements with repetition by default). Including 1 as third parameter avoids repetition of elements in the sample.

Do you know the reason for the repeated samples? I believe to know. The CPU works too fast. RANDOM(0) uses the internal CPU time as initial value for the random number generating algorithm. And it seems to be that this time does not change between two or even more calls of the function. What to do is, giving the machine some time for "thinking" or doing some useless work:

```
#11:  randsamp(l, n, s := 0, dummy, i, j, samp) :=
        Prog
          dummy := RANDOM(0)
          i := 1
          samp := []
          Loop
            If i > n
              RETURN samp
            k := IF(DIM(l) > 1, RANDOM(DIM(l)) + 1, 1)
            samp := APPEND(samp, [l↓k])
            If s = 1
              l := DELETE(l, k)
            j := 0
            Loop
              If j = 100 exit
              j :+ 1
            i :+ 1

#12:  randperm(l) := randsamp(l, DIM(l), 1)
```

I let DERIVE count from 1 to 100 in the last loop. Counting only to 10 or 20 is not sufficient (I tried).

I can use `randsamp` to produce a random permutation, too, because this is only a special case (taking a sample of all elements without replacement).

```
#13:  randsamp(test, 5) = [F, C, A, C, C]

#14:  randsamp(test, 5, 1) = [D, B, F, G, A]

#15:  randperm(test) = [C, H, G, B, F, E, A, D]
```

I try again generating a sequence of samples and permutations:

#16: VECTOR(randsamp(test, 6), k, 3) =

F	C	B	F	E	H
D	H	H	G	D	E
D	C	B	E	B	G

#17: VECTOR(randsamp(test, 6, 1), k, 3) =

H	B	G	F	E	C
D	F	G	C	E	B
D	A	B	E	F	H

#18: VECTOR(randperm(test), k, 5) =

A	G	B	H	E	C	F	D
H	D	B	E	G	C	A	F
G	A	E	C	B	H	F	D
G	H	D	B	E	F	C	A
D	E	H	C	F	B	A	G

Now everything is as expected. See other sequences of samples:

```
#19:  VECTOR(randsamp(test, k), k, 5)

#20:  [[H], [B, A], [B, A, H], [D, B, F, D], [A, F, B, F, H]]

#21:  VECTOR(randsamp(test, k, 1), k, 5)

#22:  [[A], [E, B], [E, B, D], [G, C, A, D], [D, H, E, C, G]]

#23:  VECTOR(randsamp(test, k), k, 8)
```

Speaking about permutations, I wanted to generate pairs of permutations where no single element remains on its position (permutations without a fix point).

```
      diffperms(l, p1, p2, i) :=
        Prog
          p1 := randperm(l)
          p2 := randperm(p1)
          Loop
            i := 1
#29:        Loop
              If p1↓i = p2↓i exit
              If i = DIM(p1)
                 RETURN [p1, p2]
              i :+ 1
            p2 := randperm(p1)
```

#30: diffperms(test) =

H	G	E	F	B	A	C	D
F	A	B	E	C	G	D	H

```
#31:  VECTOR(diffperms(test), k, 1, 2)
```

#32:

$$\left[\begin{bmatrix} F & A & H & C & E & D & B & G \\ E & B & F & G & D & C & H & A \end{bmatrix}, \begin{bmatrix} F & G & H & C & A & D & B & E \\ H & E & A & G & B & C & D & F \end{bmatrix}\right]$$

```
#33:  diffperms(VECTOR(k, k, 1, 15))
```

#34:

6	14	2	13	7	10	12	1	11	15	8	5	4	3	9
1	3	10	5	14	11	7	4	2	9	13	6	15	8	12

I intended to present a fourth Branch & Bound problem which needs the pairs which can be formed by numbers 1 to *n* without replacement and without considering the order (i.e. the combinations without repetitions)

```
#35:  pairs(l) := APPEND(VECTOR(VECTOR([l_i, l_j], j, i + 1, DIM(l)), i, 1, DIM(l) - 1))
```

```
#36:  pairs(VECTOR(k, k, 1, 4)) = [ 1 2 ]
                                  [ 1 3 ]
                                  [ 1 4 ]
                                  [ 2 3 ]
                                  [ 2 4 ]
                                  [ 3 4 ]
```

The number of possible pairs is given by $\binom{n}{2}$. So we should have 190 pairs formed of 20 elements.

```
#37:  DIM(pairs(VECTOR(k, k, 1, 20))) = 190
```

I save this file as `randsamp.mth` for possible use as utility file.

5 The Localisation Problem

We have the problem to distribute *n* offices (workstations, departments, …) among *n* rooms (factories, districts, towns, …). The distances between the localities are given and the intensity of necessary communications between the offices as well (number of travels, duration of communication, …).
How to distribute the offices among the rooms in order to minimize the sum of all communications (travel km, time needed, …)?

Example:

The distances r_{kl} between rooms R1, R2, R3, R4, and R5 (in minutes walking distance) are given in form of a matrix (`r_vals`) and the average number of communications b_{ij} between the offices A, B, C, D, and E is given in matrix `b_vals`.

r_{kl}	R1	R2	R3	R4	R5
R1	-	2	7	5	5
R2		-	3	2	6
R3			-	4	4
R4				-	3
R5					-

b_{ij}	A	B	C	D	E
A	-	15	20	12	14
B		-	10	15	12
C			-	8	10
D				-	6
E					-

For programming reasons I need one of the symmetric matrices in full form, and I chose the distance matrix `r_vals`.

$$\left[\text{b\_vals} := \begin{bmatrix} 0 & 15 & 10 & 12 & 14 \\ 0 & 0 & 10 & 15 & 12 \\ 0 & 0 & 0 & 8 & 10 \\ 0 & 0 & 0 & 0 & 6 \end{bmatrix},\ \text{r\_vals} := \begin{bmatrix} 0 & 2 & 7 & 5 & 5 \\ 2 & 0 & 3 & 2 & 6 \\ 7 & 3 & 0 & 4 & 4 \\ 5 & 2 & 4 & 0 & 3 \\ 5 & 6 & 4 & 3 & 0 \end{bmatrix}\right]$$

Read this matrices as follows:

15 communications between A and B if A is located in room 1 and B in room 3 will need totally $b_{12} \cdot r_{13} = 15 \cdot 7 = 105$ minutes, but if A and B were located in rooms 5 and 3 then the time needed for walking between A and B would be $b_{12} \cdot r_{53} = 15 \cdot 4 = 60$ minutes, etc.

For understanding my program let's assume that the present distribution is given by:

Offices	A	B	C	D	E
Rooms	R3	R4	R2	R1	R5

All necessary walks sum up as follows to 467 minutes.

(A,B)·(R3,R4)	+	(A,C)·(R3,R2)	+	(A,D)·(R3,R1)	+	(A,E)·(R3,R5)				
$b_{12} \cdot r_{34}$	+	$b_{13} \cdot r_{32} = r_{23}$	+	$b_{14} \cdot r_{31} = r_{13}$	+	$b_{15} \cdot r_{35}$				
$15 \cdot 4$	+	$10 \cdot 3$	+	$12 \cdot 7$	+	$14 \cdot 4$	=	230		
		$b_{23} \cdot r_{24}$	+	$b_{24} \cdot r_{14}$	+	$b_{25} \cdot r_{45}$				
		$10 \cdot 2$	+	$15 \cdot 5$	+	$12 \cdot 3$	=	131		
				$b_{34} \cdot r_{12}$	+	$b_{35} \cdot r_{25}$				
				$8 \cdot 2$	+	$10 \cdot 6$	=	76		
						$b_{45} \cdot r_{15}$				
						$6 \cdot 5$	=	30		

467

Now you can recognize the purpose of the list of pairs (named `co` in the following program).

I preload `randsamp.mth` as utility file to use `randperm` and `pairs` as well.

```
#1:   LOAD(D:\DOKUS\DNL\DNL87\randsamp.mth)

      locprob(bv, rv, numb, d, p, l, co, l_opt, cost, opt, i) :=
        Prog
          [d := DIM(rv), p := VECTOR(k, k, 1, d), opt := 100000]
          [i := 1, co := pairs(p), simul := []]
          Loop
            If i > numb exit
            l := randperm(p)
            cost := Σ(bv↓(co↓j↓1)↓(co↓j↓2)·rv↓(l↓(co↓j↓1))↓(l↓(co↓j↓2)), j, 1, DIM(co))
#2:         WRITE([i, cost])
            simul := APPEND(simul, [cost])
            If cost < opt
               Prog
                 opt := cost
                 l_opt := l
            i :+ 1
          DISPLAY("all results in simul")
          [l_opt, APPEND("Min = ", cost)]
```

The core line for finding the sum as described above is given by `cost:=`….

I run 100 simulations:

```
#4:   locprob(b_vals, r_vals, 100)
all results in simul
#5:   [[4, 2, 5, 1, 3], Min = 424]
#6:   [AVERAGE(simul), STDEV(simul)] = [459.8, 14.0028857]
#7:   NORMAL(424, 459.8, 14.0028857) = 0.005284792942
```

The recommended distribution is:

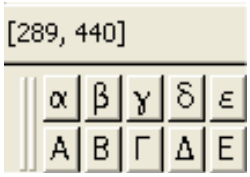
Offices	A	B	C	D	E
Rooms	R4	R2	R5	R1	R3

giving a total sum of 424 walking minutes per period. The probability to find a better solution is only 0.5%.

My paper & pen solution from 40 years ago is also 424 – but it needed some paper work!!

The program includes two little tricks which might not be well known to all DERIVIANs:

(1) The `WRITE([i,cost])` command presents the progress of the program. The number of the simulation just performed together with its result is given in the bottom left corner of the status line.



This is a part of a screenshot taken during a 500 simulations' run. We see that we have performed 289 simulations so far and the cost of this distribution is 440 minutes.

`WRITE` is very useful for debugging programs.

(2) At the end of the program I used the DISPLAY-command instead of including the text into the program output. This can be useful if you want to give additional information – and also for testing during developing a program.

I will close my expedition in the world of Branch & Bound with a more extended localisation problem: Matrix `activities` shows the number of activities between departments A to H and matrix `times` gives the transport times between sites R1 to R8. How to distribute the departments?

```
#8:  activities := [[0 3 4  5 6  2 10 7]
                    [0 0 1  8 5  4  2 9]
                    [0 0 0 11 3 12  1 8]
                    [0 0 0  0 6  5 10 0]
                    [0 0 0  0 0  7  2 6]
                    [0 0 0  0 0  0  9 1]
                    [0 0 0  0 0  0  0 5]], times := [[ 0 13 26 37 15  8 12 10]
                                                     [13  0  9 25 31 46 10  8]
                                                     [26  9  0 34 48 19 17  5]
                                                     [37 25 34  0  6 49 22 11]
                                                     [15 31 48  6  0 53 29 16]
                                                     [ 8 46 19 49 53  0 17  9]
                                                     [12 10 17 22 29 17  0 20]
                                                     [10  8  5 11 16  9 20  0]]
```

```
#9:   locprob(activities, times, 1000)

all results in simul

#10:  [[4, 2, 6, 8, 7, 1, 5, 3], Min = 2611]

#11:  [AVERAGE(simul), STDEV(simul)] = [3286.18, 240.7044416]

#12:  NORMAL(2611, 3286.18, 240.7044416) = 0.002515697936
```

My approximation is:

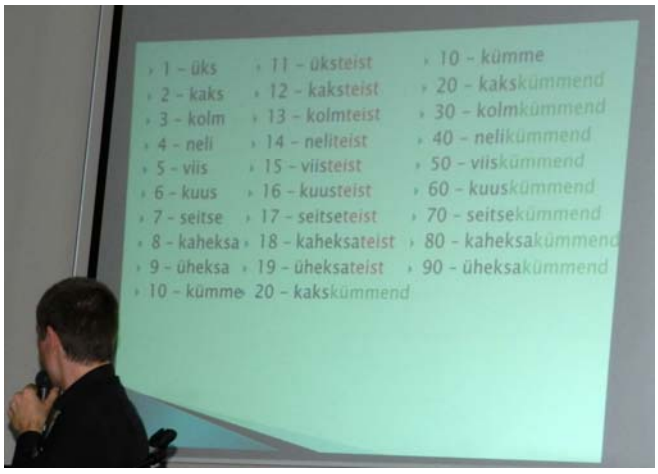
Departments	A	B	C	D	E	F	G	H
Localities	R4	R2	R6	R8	R7	R1	R5	R3

with a minimal amount of time which is 2611 time units. (You may double check the result!)

There is tiny probability of 0.2% to find a better way to distribute the departments, so we are pretty fine with the result.

I had much pleasure doing all the programming and I am quite sure that there are some ways to improve the procedures. I would like to invite you to improve the programs and to provide other B & B methods for simulation.

Many DUG members in front of the Kissing Students on the town square of Tartu (above).



Part of the TIME 2012 social program: Learning and practising counting in Estonian (right).

Comment on "Circles in a Point" (DNL#86)

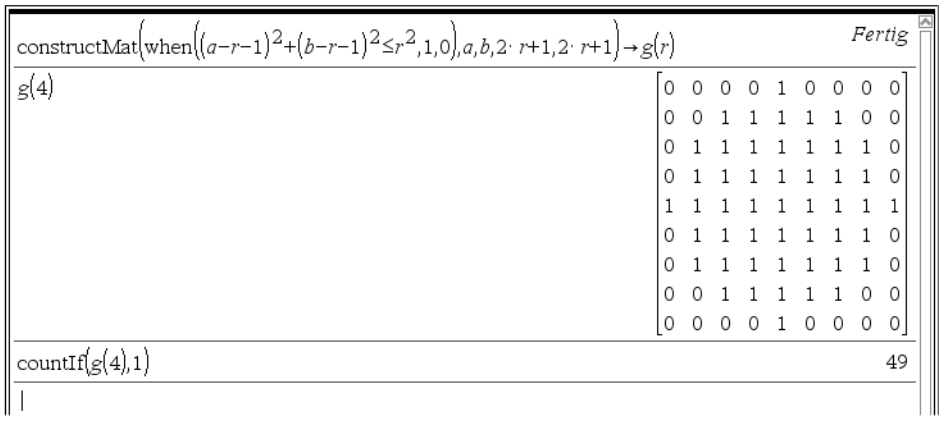
Lieber Herr Böhm,

vielen Dank für die neue Ausgabe des DNL!
Besonders gelungen finde ich diesmal den Artikel „Points in a Circle" von Roland Schröder.

Bei der Anpassung der Lösung für die TI-Nspire-Software haben Sie mangels VECTOR-Befehl eine andere Lösung vorgeschlagen.
Mit dem Nspire-Befehl *constructMat* lässt sich Herrn Schröders Vorgehensweise jedoch weitgehend identisch übertragen:

Many thanks for the new DNL issue. In particular I liked Roland Schöder's contribution on "Points in a Circle" very much.

Adapting the solution for TI-Nspire you proposed another approach because you missed the VECTOR command. But by applying the Nspire command *constructMat* it is possible to transfer Roland's procedure nearly identical.

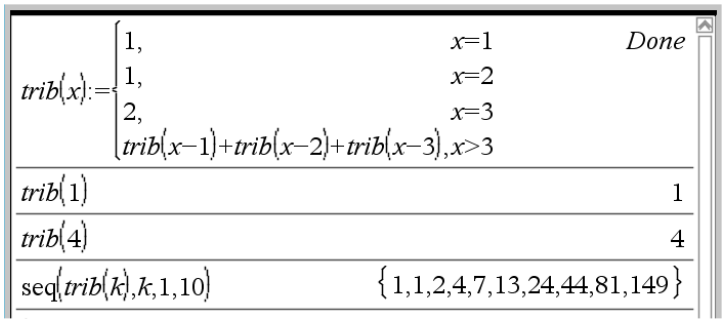


Herzliche Grüße / Best regards

Jürgen Wagner, DUG Member since 1992

The Tribonacci Sequence with TI-Nspire

In DNL#86 I did not use recursion for producing this sequence with TI-NspireCAS because I thought that this would be impossible – and I complained in Tartu about the lack of recursion. But Michel Beaudin gave a very valuable advice how to perform a recursion:



Many thanks to Michel. It is my pleasure to revise my opinion. **Recursion is possible**, Josef.

DERIVE 4 for DOS on HP 200LX

Minh Nguyen [mailto:mn1993@graffiti.net]

Hi, I am looking for derive 4 for dos to run on my HP 200LX. you know where i can get this now? Can't find it anywhere. Much appreciated. Thanks

Two problems provided by Fred Tydeman and Karsten Schmidt

Problem 1: Non Recursive Definition

Fred Tydeman wrote on 11 August 2012:

I have a set of values and two recursive definitions:

$$f(1)=0$$
$$f(2)=2*0+1=1*1=1$$
$$f(3)=3*1-1=2*1=2*(1+0)=2$$
$$f(4)=4*2+1=3*3=3*(2+1)=9$$
$$f(5)=5*9-1=4*11=4*(9+2)=44$$
$$f(6)=6*44+1=5*53=5*(44+9)=265$$
...
...

The generalisation reads as follows:

$$f(n)=n*f(n-1)+(-1)^n;\ f(1)=0$$
or
$$f(n)=(n-1)*(f(n-1)+f(n-2));\ f(1)=0, f(2)=1$$

Is there some way to get a non-recursive definition?

His comment on 12 August was:

Thanks to all that responded. They used either Maple or *MATHEMATICA* to solve this.

No one used DERIVE.

In looking at the various solutions sent to me, I came up with this:

$f(n)=\frac{n!}{e}$, rounded to the nearest integer.

This should be a challenge for the DERIVIANs – and the "Nspirians" as well to tackle this problem!

Problem 2: Eigendecomposition

… and for this I need the eigendecomposition (spectral decomposition = Spektralzerlegung) of a symmetric matrix:

Let A a symmetric matrix. Then A = Q·L·Q´. L is a diagonal matrix with the eigenvalues of A in the main diagonal and Q is the orthogonal matrix of the eigenvectors of A (see rule 4.1.11 in our Springer textbook[1]).
Do you know if anybody has done this with DERIVE before? This is, you know, very tiresome to write the eigenvalues into the main diagonal of a diagonal matrix. (The eigenvalues can appear multiple, i.e. the number of them which are returned by the EIGENVALUES function can be too small.) And it is much more laborious to calculate the respective eigenvectors using the EXACT_EIGENVECTORS-function, scale them to length 1, and then write them into matrix Q.

Best regards
Karsten Schmidt

I am no matrix algebra specialist at all. I took Karsten's textbook and studied carefully rule 4.1.11 presenting the spectral decomposition of a 2x2 matrix together with the following example 9. I wanted to apply this decomposition on a 3x3 example. On the web I found (I don't know which CAS was used):

Example: Spectral Decomposition of a Correlation Matrix[2]

```
C <- matrix(nrow=3, ncol=3, data=0.5)
C[1,2] = C[2,1] <- 0.7
C[2,3] = C[3,2] <- 0.9
diag(C) <- 1
print(C)
```

gives

```
      [,1]  [,2]  [,3]
[1,]   1.0   0.7   0.5
[2,]   0.7   1.0   0.9
[3,]   0.5   0.9   1.0
```

Eigenvalue Decomposition:

```
sz <- eigen(C)
# matrix of eigenvectors
Gamma <- sz$vectors
# diagonal matrix of eigenvectors
Lambda <- diag( sz$values )
# test, if it results as C again
test <- Gamma %*% Lambda %*% t(Gamma)
print(test)
```

gives

```
      [,1]  [,2]  [,3]
[1,]   1.0   0.7   0.5
[2,]   0.7   1.0   0.9
[3,]   0.5   0.9   1.0
```

I was curious and wanted to know if I were able to reproduce this with DERIVE, too.
This is how I did:

First of all I tried to find a DERIVE function like diag(row vector) from TI-Voyage 200 and TI-NspireCAS to make the procedure easier.

#1: diag(v) := VECTOR(VECTOR(IF(i = k, v_i, 0), i, DIM(v)), k, DIM(v))

#2: $a := \begin{bmatrix} 1 & 0.7 & 0.5 \\ 0.7 & 1 & 0.9 \\ 0.5 & 0.9 & 1 \end{bmatrix}$

#3: ev := EIGENVALUES(a)

#4: $ev := \left[\frac{\sqrt{465}\cdot\cos\left(\frac{\text{ATAN}\left(\frac{8\cdot\sqrt{1959}}{567}\right)}{3}\right)}{15} + 1,\ 1 - \frac{\sqrt{465}\cdot\sin\left(\frac{\text{ACOT}\left(-\frac{8\cdot\sqrt{1959}}{567}\right)}{3}\right)}{15},\ 1 - \frac{\sqrt{465}\cdot\cos\left(\frac{\text{ATAN}\left(\frac{8\cdot\sqrt{1959}}{567}\right)}{3} + \frac{\pi}{3}\right)}{15}\right]$

#5: $\left[w1 := \frac{\text{EXACT\_EIGENVECTOR}(a, ev_1)}{\left|\text{EXACT\_EIGENVECTOR}(a, ev_1)\right|},\ w2 := \frac{\text{EXACT\_EIGENVECTOR}(a, ev_2)}{\left|\text{EXACT\_EIGENVECTOR}(a, ev_2)\right|},\ w3 := \frac{\text{EXACT\_EIGENVECTOR}(a, ev_3)}{\left|\text{EXACT\_EIGENVECTOR}(a, ev_3)\right|}\right]$

#6: [w1 := [], w2 := [], w3 := []]

Imagine the simplified expression #5 - huge and bulky - which gives approximated #7.

#7: $\left[w1 := \begin{bmatrix} -0.51694 \\ -0.62723 \\ -0.58253 \end{bmatrix},\ w2 := \begin{bmatrix} -0.24961 \\ 0.76139 \\ -0.59830 \end{bmatrix},\ w3 := \begin{bmatrix} 0.81881 \\ -0.16388 \\ -0.55016 \end{bmatrix}\right]$

My first surprise was that the scaled vectors w1, w2 and w3 were approximated to empty vectors (#6). Then I simplified the expression [w1:= …] and received a huge expression full of roots and trigonometric expressions (the solutions of the characteristic equation which is here a cubic).

Then I approximated the exact result (resulting on #7) and built up the matrix of the scaled eigenvectors (= Q).
It is possible to form the matrix and its transpose in exact mode, too, but …

#8: Q := APPEND_COLUMNS(w1, w2, w3)

#9: $Q := \begin{bmatrix} -0.51694 & -0.24961 & 0.81881 \\ -0.62723 & 0.76139 & -0.16388 \\ -0.58253 & -0.59830 & -0.55016 \end{bmatrix}$

#10: Q·Q'

#11: $\begin{bmatrix} 1 & -3.8728\cdot10^{-11} & -2.8281\cdot10^{-11} \\ -3.8728\cdot10^{-11} & 1 & -3.9359\cdot10^{-11} \\ -2.8281\cdot10^{-11} & -3.9359\cdot10^{-11} & 1 \end{bmatrix}$

Following Karsten's rule matrix Q should be an orthogonal matrix, i.e. the product of Q and Q` should result in the 3x3 identity matrix.

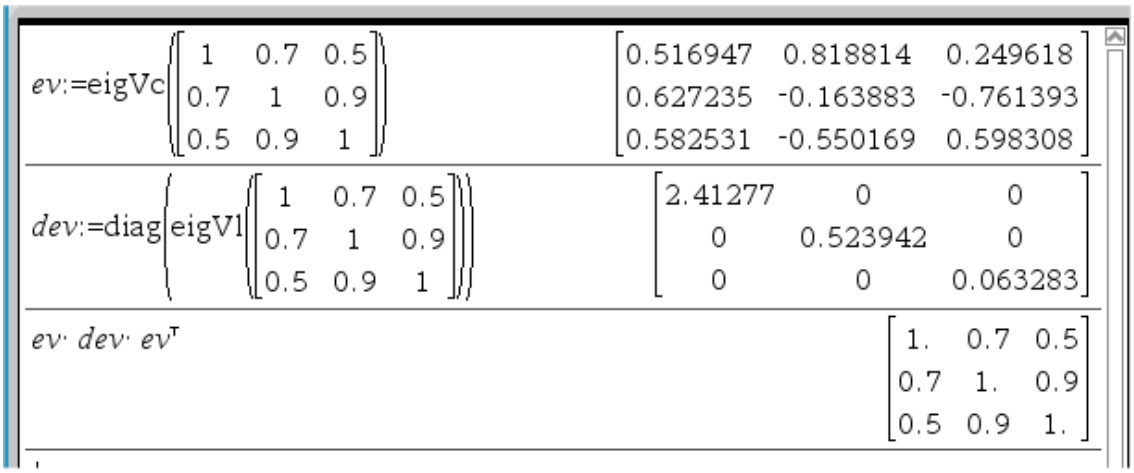
I tried simplifying Q·Q` but I could not wait for the result, so I approximated the product and was pretty sure to receive the unity matrix (#11) – more or less approximated – as expected. Finally I performed the "test" from above using the approximated matrices only.

$$\#12:\quad Q\cdot diag(ev)\cdot Q' = \begin{bmatrix} 1 & 0.7 & 0.5 \\ 0.7 & 1 & 0.9 \\ 0.5 & 0.9 & 1 \end{bmatrix}$$

I had expected a result close to the given matrix A because of my numerical procedure, but – and this was my second, but positive surprise – the result was exact matrix A (in decimal form!). To be honest, if you increase the number of decimal places you will recognize that the result is rounded. This is the story of my first successful eigendecomposition of a symmetric matrix.

[1] Schmidt, Trenkler, *Einführung in die Moderne Matrix-Algebra*, Springer 2006

I tried with TI-NspireCAS. I did not benefit from diag() only but also from the fact that eigVc(matrix) immediately returns the orthogonal matrix of eigenvectors. At the other hand TI-Nspire does not provide any exact results.



Karsten sent two additional DERIVE files:

`spectral2x2.dfw` (treating example 9 from [1]
`spectral 3x3F` (decomposition of a 3x3-matrix with multiple eigenvalues)

http://www.stat.uni-muenchen.de/~chris/rkurs/Rkurs6.pdf

http://www.ece.ucsb.edu/~roy/classnotes/210a/ECE210a_lecture15_small.pdf

http://cseweb.ucsd.edu/~dasgupta/291/lec7.pdf

http://www.youtube.com/watch?v=hmahoeta4KY