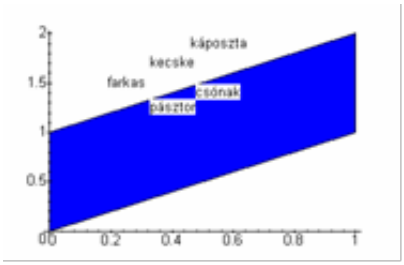
The Shepherd, the Wolf, the Goat and the Cabbage

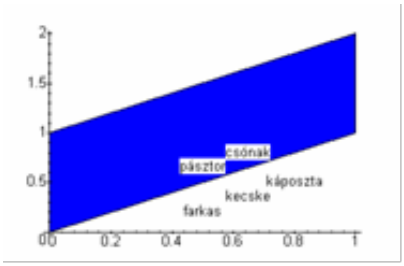
The shepherd, the wolf, the goat and the cabbage are on the riverbank. The shepherd's task is to take them to the other bank. For this purpose, he has a boat for two people, that is, besides the shepherd another passenger, the goat, the wolf or the cabbage, can travel in it Naturally we assume that only the shepherd can manage the boat so he has to be there at every crossing.

The situation of the shepherd is not all roses because the goat eats the cabbage and the wolf eats the goat unless the shepherd is present. Can the shepherd solve the task?

The present situation is the following



We have to reach this:



Can we solve the situation mentioned above with mathematical methods? Let's think about it. In the beginning, the four of them are on the left bank of the river. Then the shepherd takes the wolf to the other bank in the boat thus the goat and the cabbage stay on the left bank and the shepherd, the boat and the wolf are on the right bank.

Notice that it is enough to know who are on the left bank to describe the present situation because that determines the ones on the right bank. It is also important to notice that there are forbidden situations, for instance the situation mentioned above. The goat and the cabbage are on the left bank which is not allowed because the goat eats the cabbage without any supervision.

How can the situation be described with Maple objects? For example, by listing those who are on the left riverbank at an exact moment. The set data type is perfect for this purpose because the order of the characters on the riverbank does not matter. However, we still list those who are on the left riverbank in a list because in this way we can see the participants in the same order when displaying the situations.

Let's write the situations allowed. Although the number of all the situations is 25=32 it can be easily seen that only the following 10 cases exist out of the 32. If we systemize the writing of the situations not only do we get the situations allowed but we can also be sure that there are no more cases allowed.

> *restart*

> s_1 := [*csónak*, *pásztor*, *farkas*, *kecske*, *káposzta*]:

> s_2 := [*csónak*, *pásztor*, *farkas*, *kecske*] :

> s_3 := [*csónak*, *pásztor*, *farkas*, *káposzta*] :

> s_4 := [*csónak*, *pásztor*, *kecske*, *káposzta*] :

> s_5 := [*csónak*, *pásztor*, *kecske*] :

> s_6 := [*farkas*, *káposzta*] :

> s_7 := [*káposzta*] :

> s_8 := [*kecske*] :

> s_9 := [*farkas*] :

> s_{10} := [ ] :

We followed the descending order of the element number of the lists.

There is only one five-element case which is allowed.

The boat is always where the shepherd is. That's why we can only get a four-element list if it contains the shepherd and the boat as well. And since the shepherd is present, we can choose two of the other three participants besides the shepherd and the boat. We can get the cases described by the four-element list in this way.

There is only one situation that can be described with a three-element list. If the shepherd is not in the three-element list neither the boat is which is impossible because the wolf eats the goat which eats the cabbage although in this case this is only possible in a reversed order. So the shepherd must be on the three-element list but in this case the boat is there as well thus we can only choose one more participant. We have three choices but we can choose neither the wolf nor the cabbage because in the first case the goat eats the cabbage and in the second case the wolf eats the goat. Only the goat is allowed to be chosen.

Since there is not a situation that can be described by two three-element lists there is not a situation that can be described by two different two-element lists. Why? Similarly, because of symmetrical reasons, the number of the cases that can be described by a one-element list equals to the number of the situations described by four-element lists, that is, three. And finally, the most important case, which provides the solution of the task, is the situation that can be described by an empty list. In this case, everybody has arrived to the other bank.

After having written down the situations allowed, we have to write all the transitions allowed. There is no allowed transition from the s1 situation to the s8 because this would mean that the shepherd took the wolf and the cabbage to the other bank besides himself and the boat which is impossible according to the task.

We are going to create a set of two-element lists. The [i,j] pair of numbers is the element of the set only if there is an allowed transition from the si situation to the sj and i<j.

> *átmenetek* := {[2, 8], [2, 9], [3, 9], [3, 6], [4, 7], [4, 8], [5, 8], [5, 10], [1, 6], [3, 7]} :

It can be easily checked that 10 pairs of numbers of the transitions set satisfy the criterion above. Thus,

for example, there is a transition allowed from s1 to s6 because if the shepherd takes only the boat then the participants eat each other. This is impossible so the shepherd has to deliver something else besides himself and the boat. The present situation is a two-element list out of which there is only one situation and that is s6. The checking of the other transitions should be done by the readers.

An important remark: if there is a transition from the si situation to the sj then there is a transition from the sj to the si as well. That's why it was useful to specify the i<j assumption in the definition of the transition set.

After this, let's write the M transition matrix of the problem

```
> f:=(i,j)->if member([i,j],`átmenetek`) or member([j,i],
  `átmenetek`) then 1 else 0 fi;
```

$$f := (i,j) \rightarrow \textbf{if}\ member([i,j], átmenetek)\ \text{or}\ member([j,i], átmenetek)\ \textbf{then}\ 1\ \textbf{else}\ 0\ \textbf{end if} \qquad (1)$$

```
> M:=Matrix(10,10,f)
```

$$M := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (2)$$

The member(x,s); instruction examines whether the x expression is an element of the s set. If yes it gives a true otherwise a false value. According to this, the f is a function with two variables that assigns the 1 value to the (i,j) pair if the [i,j] or the [j,i] two-element list is the element of the transition set and the function value is 0.

We used this generator function to create the M matrix. It can be clearly seen that [képlet]. This is a very efficient way to create matrixes. What if we had had to type all the hundred elements? How many times would the readers have made mistakes during the input?

The transition matrix writes exactly whether there is a transition between the two situations or no. More precisely, the Mi,j is 1 only if there is a transition from the si situation to the sj. Maybe it is not surprising that the M is a symmetrical matrix. Now let's look at the square of the transition matrix. We have two choices. The first is to use the dot (.) infix operator which executes the multiplication of the matrixes, so we can write M.M or we can use the operation of raising to power: M2.

```
> M.M
```

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \quad (3)$$

In the square of the transition matrix we cannot find only 0 and 1 elements. What does the square of the M matrix describe? The [i,j]th element of the square matrix is

$$\sum_{k=1}^{10} M_{i,k} M_{k,j} \quad ,$$

which can be equal to one if there is only one k index for which [képlet]. And this means exactly that there is a transition from the si situation to sk from where there is a transition to the sj. In other words, we can get to the sj situation from the si in two steps.

It can be easily seen that the [i,j]th element of the nth power of the M matrix is not zero only if we can get from the ith situation to the jth exactly within n steps. According to this, we only have to look at the powers of the transition matrix to solve the problem of the shepherd, that is, the delivery of the participants to the other bank. When we find such an M power the [1,10] element of which is not zero then we can state that there is a solution for the task.

```
> B:=M: for n while B[1,10]=0 do B:=evalm(M^n) od:
```

```
> 'n' = n - 1
```

$$n = 7 \quad (4)$$

```
> evalm(B)
```

(5)

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 22 & 23 & 16 & 23 & 2 \\ 0 & 0 & 0 & 0 & 0 & 39 & 62 & 84 & 63 & 23 \\ 0 & 0 & 0 & 0 & 0 & 68 & 84 & 80 & 84 & 16 \\ 0 & 0 & 0 & 0 & 0 & 39 & 63 & 84 & 62 & 23 \\ 0 & 0 & 0 & 0 & 0 & 18 & 39 & 68 & 39 & 22 \\ 22 & 39 & 68 & 39 & 18 & 0 & 0 & 0 & 0 & 0 \\ 23 & 62 & 84 & 63 & 39 & 0 & 0 & 0 & 0 & 0 \\ 16 & 84 & 80 & 84 & 68 & 0 & 0 & 0 & 0 & 0 \\ 23 & 63 & 84 & 62 & 39 & 0 & 0 & 0 & 0 & 0 \\ 2 & 23 & 16 & 23 & 22 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (5)$$

As we can see, the shepherd should be familiar with the matrix multiplication and Maple to solve the task.

The set of the start and end value can be omitted in the cycle instruction. In this case, the start value and the step value are 1 and the cycle runs as long as the assumption subsequent to the while key word does not become false. However, this can be dangerous. What if our assumption is never going to become false? In this case, we have written an indefinite cycle which is not the best. The

$$B := M; \textbf{ for } n \textbf{ to } 10 \textbf{ while } B_{1,\,10} = 0 \textbf{ do } B := evalm\left(M^{n}\right); print\left(n, B_{1,\,10}\right) \textbf{ end do};$$

command would have been safer. In this case, the running of the cycle stops at n=11 value even if the while assumption never becomes false.

Why do we think that it is enough to look at the powers of the M matrix only until the n=10 value? Due to the Cayley-Hamilton theorem which states that an arbitrary square matrix, and similarly the M as well, is the root of its own characteristic polynomial. Let's see what this means in our case.

```
> sort(linalg[charpoly](M,lambda));
```

$$\lambda^{10} - 10\,\lambda^{8} + 33\,\lambda^{6} - 44\,\lambda^{4} + 24\,\lambda^{2} - 4 \quad (6)$$

```
> f:=unapply(%,lambda);
```

$$f := \lambda \rightarrow \lambda^{10} - 10\,\lambda^{8} + 33\,\lambda^{6} - 44\,\lambda^{4} + 24\,\lambda^{2} - 4 \quad (7)$$

```
> f(M);
```

(8)

$$\begin{bmatrix} 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0 \end{bmatrix} \tag{8}$$

The usage of the linalgcharpoly procedure may seem strange because previously we considered the LinearAlgebra package as the linear algebra package of Maple. In this package it is the CharateristicPolynomial procedure that creates the characteristic polynomial. In the beginning, there was only the linalg package with short, easy-to-remember names then the module paradigm was introduced thus the new linear algebra package was developed. But the compatibility considerations did not allow the changing of the previous package that's why in the new package the old procedures had to be given new names. So the extremely long names were created this way and maybe these could encourage us to use the instruction completion functionality. So the (6) output could have been generated by the $LinearAlgebra_{CharacteristicPolynomial}(M, \text{lambda})$ instruction.

Back to the Cayley-Hamilton theorem and its usage, the M matrix is the real solution of the f polynomial. However, if the [képlet] for an arbitrary K matrix then we get that the K11 is displayed as the linear combination of the powers of the K lower than 10 by multiplying both sides of the equation by the K and expressing the K11 from the equation created.

> $f(K) = 0$

$$K^{10} - 10\,K^8 + 33\,K^6 - 44\,K^4 + 24\,K^2 - 4 = 0 \tag{9}$$

> $map(x \to K\,x, \textbf{(9)})$

$$K\,(K^{10} - 10\,K^8 + 33\,K^6 - 44\,K^4 + 24\,K^2 - 4) = 0 \tag{10}$$

> $expand(\%)$

$$K^{11} - 10\,K^9 + 33\,K^7 - 44\,K^5 + 24\,K^3 - 4\,K = 0 \tag{11}$$

> $isolate(\%, K^{11})$

$$K^{11} = 10\,K^9 - 33\,K^7 + 44\,K^5 - 24\,K^3 + 4\,K \tag{12}$$

Similarly, it can be seen that the powers of the K11 of the K matrix larger than 10 can be created as the linear combination of the powers lower than 10. Naturally, the same is true for the M matrix itself.

In this case there are two possibilities. If there is a k<10 for which the Mk1,10 differs from zero then the task can be solved in k number of steps. In the opposite case, the [1,10] index element is zero in all the powers of the M lower than 10 which leads to the fact that the [1,10] element will be zero in the powers larger than 10. In this case there is no solution to the task. However, it is enough to check the first 10 powers of the M irrespectively of having a solution or no.

The seventh power of the M is

> M^7

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 22 & 23 & 16 & 23 & 2 \\ 0 & 0 & 0 & 0 & 0 & 39 & 62 & 84 & 63 & 23 \\ 0 & 0 & 0 & 0 & 0 & 68 & 84 & 80 & 84 & 16 \\ 0 & 0 & 0 & 0 & 0 & 39 & 63 & 84 & 62 & 23 \\ 0 & 0 & 0 & 0 & 0 & 18 & 39 & 68 & 39 & 22 \\ 22 & 39 & 68 & 39 & 18 & 0 & 0 & 0 & 0 & 0 \\ 23 & 62 & 84 & 63 & 39 & 0 & 0 & 0 & 0 & 0 \\ 16 & 84 & 80 & 84 & 68 & 0 & 0 & 0 & 0 & 0 \\ 23 & 63 & 84 & 62 & 39 & 0 & 0 & 0 & 0 & 0 \\ 2 & 23 & 16 & 23 & 22 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (13)$$

in which case the [1,10] index element is 2. So the shepherd can take the participants to the other bank within seven steps and he can execute it in two ways. But why and how? In other words, we already have an existence statement thus we know that there is a solution but we do not know the solution yet.

A graphic display is always useful so let's try to find the solution by the graphic representation of the task. During the creation of the mathematical model we entered situations and defined transitions between them. The graphs are perfect to illustrate these structures in a way that the vertices of the graph represent the situations and the edges stand for the transitions.

Maple offers the networks package for the handling of the graphs and networks. The procedures of the package are made available by the with(networks) instruction.

> $with(networks)$:

The first step is to create an empty graph with ten vertices which does not have any edges with the void procedure. The vertices displays the vertices of the graph given as parameters and the edges procedure displays the edges. As we can see, the void procedure labelled the vertices of the G graph with natural numbers because it had not got any instructions. This is perfect for us because we consider that the 1 vertex represents the s1 situation and the 2 vertex represents the s2 and so on.

> $G := void(10)$:

> $vertices(G)$

$$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \quad (14)$$

> $edges(G)$

$$\{\} \quad (15)$$

The output of the edges(G) is empty which indicates that for the time being G does not have a definite edge. Let's create the edges in a way that we connect the i and j vertices by the edges only if there is a transition from the si situation to the sj. We created the possible transitions at the beginning of the worksheet. The value of the transitions variable is the set containing the transitions allowed which can be used at the creation of the edges of the G graph.

> *átmenetek*

$$\{[2, 8], [2, 9], [3, 9], [3, 6], [4, 7], [4, 8], [5, 8], [5, 10], [1, 6], [3, 7]\} \quad (16)$$

```
> for i in átmenetek do
   connect(i[1],i[2],G)
  od:
```

The connect(i1,i2,G) instruction located in the body of the loop connects the i1 and i2 vertices of the G graph for the i element of the transition set.

The following commands illustrate the new properties of the G. Now it is the edges(G) that shows 10 edges. Naturally, we cannot find out from this which vertices the edges connect. The end procedure the first parameter of which is an edge and the second is a graph gives us a helping hand and we get the end points of the edge chosen as an output. According to this, the e3 edge connects the 3 and 9 vertices. If we give only one graph to the end procedure as a parameter then we will get the end points of all the edges. This may look fine but there is a little problem: we do not know with which edges the two-element vertices sets coincide.

Finally, we queried the adjacent matrix of the graph with the adjacency procedure. It must not be surprising that this corresponds with the M matrix (see (2)).

```
> edges(G)
```

$$\{e6, e7, e8, e3, e4, e1, e10, e2, e9, e5\} \quad \textbf{(17)}$$

```
> ends(e3, G)
```

$$\{3, 9\} \quad \textbf{(18)}$$

```
> ends(G)
```

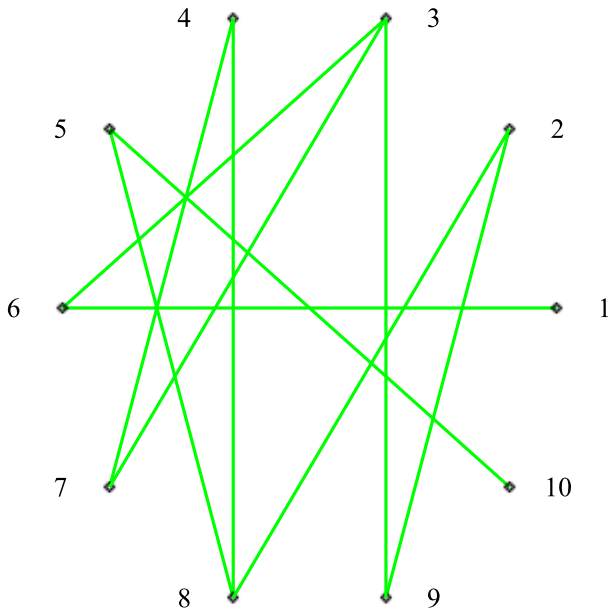
$$\{\{5, 10\}, \{5, 8\}, \{1, 6\}, \{2, 8\}, \{3, 6\}, \{3, 7\}, \{3, 9\}, \{4, 7\}, \{4, 8\}, \{2, 9\}\} \quad \textbf{(19)}$$

```
> adjacency(G)
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(20)}$$

Now it is time for us to begin the graphic representation of the task. Let's draw the G graph.

```
> draw(G)
```



That's a nice graph.

Since the edges of the graph represent transitions between each situation and we have to find a sequence of transitions from the 1 situation to the 10 we need to find a path to G the start point of which is the 1 situation and its end point is the 10. At this point, the task is similar to a quiz show, that is, we can find the path after some attempts but we would rather entrust Maple with this task.

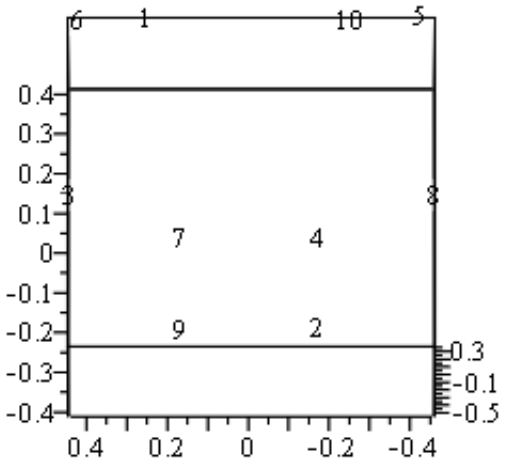
The spantree procedure uses the Prim algorithm to determine the spanning tree. This tree is such a sub-graph which contains all the vertices but it does not contain a circle. To tell the truth, this is needless for finding the path; its call is purely technical. The spantree prepares data for the successful operation of the path in the background.

```
> spantree(G, 1) :
> path([1, 10], %)
```

$$[1, 6, 3, 7, 4, 8, 5, 10] \quad (21)$$

So we have got a sequence of vertices, a path which leads from point 1 to point 10. This and the other solution of the task can be immediately seen in the 3-D graph of the G graph.

```
> draw3d(G, orientation =[90, 101])
```



This is a nice, clear-cut graph. We can easily find the two existing paths from vertex 1 to 10. These are [1, 6, 3, 7, 4, 8, 5, 10] és [1, 6, 7, 4, 8, 5, 10].

So we can conclude that it is highly recommended for the shepherd to acquire some knowledge about the matrix operations and the graphs if he wants to solve such difficult issues.

Let's finish this worksheet with the illustration of the solution found. First, list the situations mentioned in path (21). Don't forget that this shows the participants on the left side of the river.

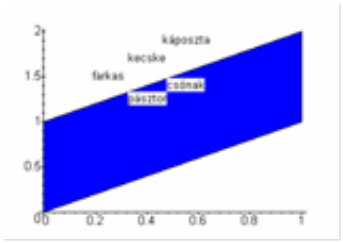
```
> for i in (21) do
   s[i]
  od;
```

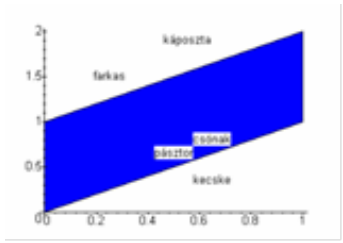
[*csónak*, *pásztor*, *farkas*, *kecske*, *káposzta*]
[*farkas*, *káposzta*]
[*csónak*, *pásztor*, *farkas*, *káposzta*]
[*káposzta*]
[*csónak*, *pásztor*, *kecske*, *káposzta*]
[*kecske*]
[*csónak*, *pásztor*, *kecske*]
[ ] **(22)**

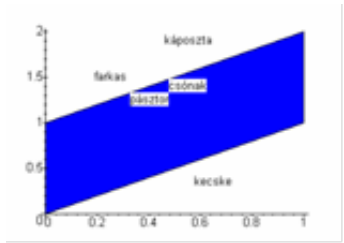
```
>
```

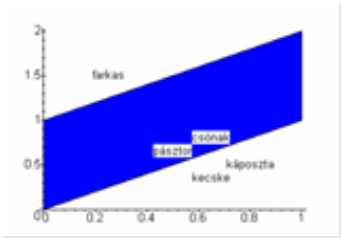
First the shepherd takes the goat to the other bank. We know this because the second situation contains only the wolf and the cabbage. After having delivered the goat to the bank, the shepherd comes back in the empty boat. Then he takes the wolf and takes back the goat. Then he takes the cabbage and puts it next to the wolf and comes back for the goat in the empty boat. At this point he knows for sure that he will succeed in solving the task because only the goat is to be delivered.

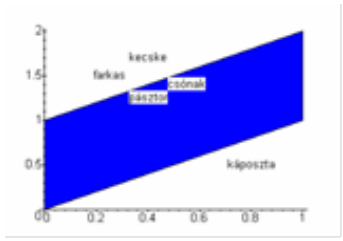
The following figures illustrate the seven situations created during the process.

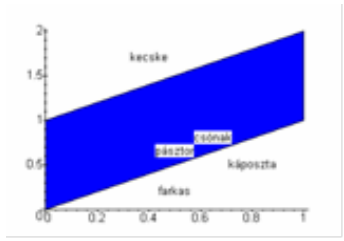


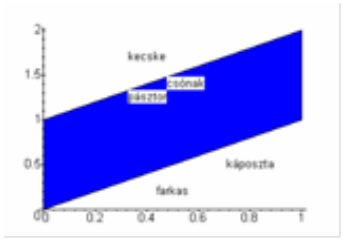


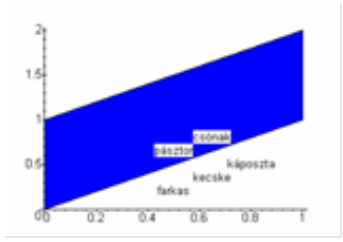












What Have You Learnt About Maple?

• If the f is a function with two variables then the M:=matrix(n, m, f) instruction creates such an M matrix the [i,j]th element of which is f(i,j). The f function is called a generator function

• The

$$T := table([index_1 = elem_1, index_2 = elem_2],...)$$

instruction creates a table data type object. We can refer to the elements of the table with their indexes, e.g. T_{index_1}. The sequence of the indexes in the table is provided by the indices procedure and the sequence of the elements is provided by the entries procedure.

• If the break name is evaluated in the body of the loop then the controlling is executed after the od

keyword which closes the body of the loop, that is, with the help of the break we can exit the cycle. Contrary to this, the next name is used to skip the remaining instructions of the body of the loop and to execute the cycle again with the next value (if there is one) of the cycle variable.

- The system executes repeatedly the sequence of instructions closed between the do and od keywords without checking.

Excersise

1. Generate an 8x8 arbitrary 0-1 matrix and denote it with M.

2. Interpret the elements of the M in a way that they should stand for the existing routes of a package delivery company. More precisely, assume that Mi,j=1 only if the company has a route from the si spot to the sj.

a) Determine the number of loads with which the company is able to deliver goods from the s1 spot to the s6. Try this for other situation pairs as well.

b) Determine those two spots where the most loads are needed for the goods to get from one place to the other.

c) Are there such two spots between which the goods cannot be delivered? Determine all these pairs.