Using Maple to perform Row Reduction

Often, it is handy to have a computer algebra system perform row operations so that we don't have to worry about making arithmetic errors.

In Maple, you'll want to type the following before doing the row ops:

```
with(LinearAlgebra);
```

In the `LinearAlgebra` package, we can perform the three elementary row operations using the `RowOperation` command. Here's how (and this is in the Maple help file, too). In these sample commands, we assume that matrix A has been entered.

- Row Operation I: Swap rows i and j

  ```
  RowOperation(A,[i,j]);
  ```

- Row Operation II: Multiply row i by k

  ```
  RowOperation(A,i,k);
  ```

- Row Operation III, using pivot row j acting on row i: $kR_j + R_i \to R_i$

  ```
  RowOperation(A,[i,j],k);
  ```

As an example, let's enter the following matrix and row reduce it:

$$A = \begin{bmatrix} 1 & -2 & -1 & 3 & 0 \\ -2 & 4 & 5 & -5 & 3 \\ 3 & -6 & -6 & 8 & 2 \end{bmatrix}$$

In Maple, we'll enter this column-wise:

```
with(LinearAlgebra):
A:=<<1,-2,3>|<-2,4,-6>|<-1,5,-6>|<3,-5,8>|<0,3,2>>;
```

Now we'll row-reduce. We'll pivot in the $(1,1)$ position. The row operations will be:

$$2R_1 + R_2 \to R_2 \qquad -3R_1 + R_3 \to R_3$$

In Maple:

```
A1:=RowOperation(A,[2,1],2);
A2:=RowOperation(A1,[3,1],-3);
```

We now have the matrix:

$$\begin{bmatrix} 1 & -2 & -1 & 3 & 0 \\ 0 & 0 & 3 & 1 & 3 \\ 0 & 0 & -3 & -1 & 2 \end{bmatrix}$$

Therefore, we pivot in the $(2,3)$ position and use the row ops:

$$\frac{1}{3}R_2 \to R_2 \qquad R_2 + R_1 \to R_1 \qquad 3R_2 + R_3 \to R_3$$

In Maple:

```
A3:=RowOperation(A2,2,1/3):
A4:=RowOperation(A3,[1,2],1):
A5:=RowOperation(A4,[3,2],3);
```

We now have:

$$\begin{bmatrix} 1 & -2 & 0 & 10/3 & 1 \\ 0 & 0 & 1 & 1/3 & 1 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

Programming Note

Instead of writing, for example,

```
A1:=RowOperation(A,[2,1],2):
A2:=RowOperation(A1,[3,1],-3):
A3:=RowOperation(A2,2,1/3);
```

It is possible to type, instead:

```
A:=RowOperation(A,[2,1],2):
A:=RowOperation(A,[3,1],-3):
A:=RowOperation(A,2,1/3):
```

Why might this be a bad idea? In the second example, you are *overwriting* your original matrix A. If you happen to type an error in one of the commands, you would have to start over.

On the other hand, the first case assigns a new variable to each version of the matrix- This makes it easy to check on the row reduction process.

Finally, if you were to actually write software to do all this automatically, you might only keep the original matrix, but then create a second matrix and overwrite it as you go.

I should also mention that Maple does have a built in solver for linear programs, but we're learning how to do the simplex algorithm by hand first. The built-in command is `LPSolve` if you want to check it out.

Using Maple to Row Reduce a Tableau

Tableaux work in the same way, of course. Here's a short maximization tableau:

$$\begin{array}{rrr|rr|r} -1 & -2 & -2 & 0 & 0 & 0 \\ \hline 2 & 1 & 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 0 & 1 & 10 \end{array}$$

Here are the Maple commands to get the final tableau:

```
B:=<<-1,2,0>|<-2,1,0>|<-2,0,1>|<0,1,0>|<0,0,1>|<0,8,10>>;
B1:=RowOperation(B,[1,2],2);
B2:=RowOperation(B1,[1,3],2);
```

Using Maple for Big M

The command we will be using is `RowOperation`. Here's how it works, with a sample tableau:

$$\begin{array}{rrrrrrr|r} x_1 & x_2 & e_1 & e_2 & s_1 & a_1 & a_2 & rhs \\ -1 & 2 & 0 & 0 & 0 & M & M & 0 \\ \hline 1 & 1 & -1 & 0 & 0 & 1 & 0 & 2 \\ -1 & 1 & 0 & -1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 3 \end{array}$$

In Maple, to enter the matrix, go column-wise (don't break the line when you're typing into Maple):

```
with(LinearAlgebra)
A:=<<-1,1,-1,0>|<2,1,1,1>|<0,-1,0,0>|<0,0,-1,0>|
                <0,0,0,1>|<M,1,0,0>|<M,0,1,0>|<0,2,1,3>>;
```

Now we'll perform our pivots. The row operations to get rid of M under a_1, a_2 will be:

$$-MR_2 + R_1 \to R_1 \qquad \text{and} \qquad -MR_3 + R_1 \to R_1$$

In Maple, this is:

```
A1:=RowOperation(A,[1,2],-M);
A2:=RowOperation(A1,[1,3],-M);
```

I like to assign the results to an intermediate variable, so that I can backtrack if I mis-type something along the way- Otherwise, you have to start over again.