CoCoALib - Slug #1643

rref slower than expected (maybe) [[reduced row echelon form]]

12 Jan 2022 13:54 - John Abbott

Status:	New	**Start date:**	12 Jan 2022
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Improving	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99880	**Spent time:**	0.25 hour

Description

I am preparing a demo for the students, and have written a naive impl of gaussian reduction to triangular form (in CoCoA-5).

I was surprised to find that my naive impl is faster than rref; it is true that rref does more work (by reducing "upwards" as well), and also most of the time is BigRat arithmetic. Still my naive code took 23s against 30s for rref.

Perhaps have a look at rref to see if it can be made more efficient?

History

#1 - 12 Jan 2022 13:57 - John Abbott

Here is the test I made:

```
-- A simplistic implementation of gaussian reduction to
-- triangular form (assuming full row rank).
define triang(M)
  // Assume input is full row rank matrix (over a field).
  ncols := NumCols(M);
  nrows := NumRows(M);
  for c := 1 to ncols do
    PivotRow := c;
    while PivotRow <= nrows and M[PivotRow,c] = 0 do
      PivotRow := PivotRow+1;
    endwhile;
    if PivotRow > nrows then return "Det is zero"; endif;
    SwapRows(ref M,c,PivotRow); // ignore that this negates the det
    println "Pivot ",c," is ", M[c,c];
    for r := c+1 to nrows do
      q := M[r,c]/M[c,c];
      SetRow(ref M, r, GetRow(M,r) - q*GetRow(M,c)); // M[r] := M[r] - q*M[c];
    endfor;
  endfor;
  return M;
enddefine; -- triang

// Let's try the code on a non-trivial example:
n := 200;
M := mat([[random(-99,99) | j in 1..n] | i in 1..n]);
t0 := CpuTime(); T := triang(M); TriangTime := TimeFrom(t0); --> about 23s
t0 := CpuTime(); R := rref(M); RrefTime := TimeFrom(t0); --> about 30s
```

#2 - 01 Mar 2024 09:54 - John Abbott

- Target version changed from CoCoALib-0.99850 to CoCoALib-0.99880