CoCoALib - Feature #202

MatrixView/function for viewing a single row or column (RowMat, ColMat)

04 Jul 2012 10:17 - Anna Maria Bigatti

Status:	Closed	**Start date:**	04 Jul 2012
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Data Structures	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99850	**Spent time:**	2.00 hours

Description

We should make a special view for a row or column since the C++/STL for calling **submat** is so bad (vector containing all row indices has no easy C++ constructor)

The main difficulty is finding a good name to highlight the result is a **MatrixView** and non a vector....
- **SingleRow**?
- **OneRow**?
- **OneRowSubmat**?

2024-03 functions called **RowMat**/**ColMat** (calling submat)

Related issues:

Related to CoCoALib - Bug #1014: RowMat, ColMat with arg an empty list/vector	**In Progress**	**24 Feb 2017**
Related to CoCoALib - Feature #312: LongRange(a,b) returning vector of long a...	**Closed**	**14 Feb 2013**
Related to CoCoALib - Design #64: submat takes only vector<long>	**Closed**	**15 Dec 2011**
Related to CoCoALib - Feature #1788: New MatrixView/function "FirstRows/First...	**Closed**	**08 Mar 2024**

History

#1 - 04 Jul 2012 18:33 - Anna Maria Bigatti

another proposal

```
RowMat(M, i);
ColMat(M, j);
```

comments?

#2 - 01 Aug 2014 08:59 - Anna Maria Bigatti

- Target version set to CoCoALib-1.0

#3 - 13 May 2015 09:56 - Redmine Admin

- Category set to Data Structures

#4 - 24 Feb 2017 17:42 - John Abbott

Another possibility: uglier, but possibly clearer?

- **RowMatView**
- **ColMatView**

The intention is that the result should be a matrix 1xC or Rx1, right?

#5 - 24 Feb 2017 17:49 - John Abbott

- Related to Bug #1014: RowMat, ColMat with arg an empty list/vector added

#6 - 24 Feb 2017 17:51 - John Abbott

What should these functions do if the matrix is 0-by-C or R-by-0?
Return a 0-by-1 or 1-by-0 matrix?

#7 - 02 Mar 2017 15:22 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 50

Implemented (using submat).
Still have to write tests and doc.

#8 - 06 Nov 2017 14:56 - Anna Maria Bigatti

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99600

#9 - 14 Jun 2018 15:53 - John Abbott

I have just checked the code: it is in MatrixView (as one would expect).

I am not so happy about the names **RowMat** and **ColMat** because these names exist also for some completely different functions: ones which allow a C++ vector of RingElem to be viewed as a row-matrix or a col-matrix.

How about the names **RowOfMat** and **ColOfMat**?
Or even **submat1row** and **submat1col**? (uglier but maybe clearer?)

#10 - 31 Jul 2018 13:20 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.99600 to CoCoALib-0.99650 November 2019

#11 - 01 Oct 2019 12:04 - John Abbott

- Target version changed from CoCoALib-0.99650 November 2019 to CoCoALib-0.99700

#12 - 18 Jan 2020 19:59 - John Abbott

We could even use the names **row** and **col**, so sample uses would be

```
  MatrixView FirstRow = row(M,0);
  MatrixView LastRow = row(M, NumRows(M)-1);
```

One odd aspect of this implementation is that if we want to get an element from the resulting row/column, we must supply **two** indices (because the result is really just a sub-matrix. For instance:

```
  MatrixView FirstCol = col(M,0);
  if (FirstCol(0,0) == 1) cout << "Top left entry is 1" << endl;
```

This is rather weird! Usually we try to avoid too much weirdness.

Personally I would expect the result of a "get-row" or "get-col" function to produce an object which requires just a single index.

An advantage of the current impl (as a submatrix) is that we can directly use the chosen row/col in matrix multiplication.

Perhaps the correct result type(s) should be new objects which require just a single index to access the values, but if used in a context where a matrix is required are automatically converted to (or viewed as) a matrix.

#13 - 18 Jan 2020 20:05 - John Abbott

Should the "get-row" or "get-col" function make a copy or just refer to the original matrix?

Making a copy could be expensive, and possibly unwanted in many contexts.
If a copy is required, it can be made explicitly by calling DenseMat (or some similar fn).

Refering to the original matrix might leave a "dangling pointer":

```
  M = 3x3 matrix;
  auto LastRow = row(M,2);
  M = 2x2 matrix;
  cout << LastRow(2) << endl;  /// What happens here?
```

If the matrix impl is ref counted then the object produced by **row** (or **col**) could maintain a counted pointer to the impl. This would then appear to the user as though a copy had been made.

Also, can elements in a row/col object be assigned to?   [probably **yes**]
Should this then change the entry in the original matrix? [probably **yes**]
Assigning an element in a row object may cause an element in a col object to change value:

```
  M = 3x3 matrix;
  auto FirstRow = row(M,0);
  auto FirstCol = col(M,0);
  FirstRow[0] = -99;
  cout << FirstCol[0] << endl;  // should print -99!
```

#14 - 12 Feb 2020 16:43 - Anna Maria Bigatti

- Target version changed from CoCoALib-0.99700 to CoCoALib-0.99800

#15 - 04 Feb 2022 21:27 - John Abbott

- Target version changed from CoCoALib-0.99800 to CoCoALib-0.99850

#16 - 08 Feb 2024 21:55 - John Abbott

What is the status of this issue? With luck we can soon close it!

#17 - 08 Mar 2024 09:04 - Anna Maria Bigatti

- Status changed from In Progress to Resolved

- % Done changed from 50 to 80

John Abbott wrote:

> I have just checked the code: it is in MatrixView (as one would expect).
>
> I am not so happy about the names **RowMat** and **ColMat** because these names exist also for some completely different functions: ones which allow a C++ vector of RingElem to be viewed as a row-matrix or a col-matrix.

I checked: it is implemented as **RowMat** and I think it is good because, in both meanings, it is a MatrixView.   I wonder what I originally wanted to use it for... I could not find any such call of RowMat nor awkward call of submat :-/

#18 - 08 Mar 2024 09:05 - Anna Maria Bigatti

- Related to Feature #312: LongRange(a,b) returning vector of long a..b (included) added

#19 - 08 Mar 2024 09:11 - Anna Maria Bigatti

- Status changed from Resolved to Feedback

- % Done changed from 80 to 90

I found a good use in ex-IdealOfPoints1.C.
The comparison in readability between before and after is indeed quite striking:

```
// matrix M1 = NewDenseMat(submat(M, LongRange(0,0), LongRange(0,2)));
   matrix M1 = NewDenseMat(RowMat(M, 0));
```

Similarly in test-MatrixForOrdering2.C

#20 - 08 Mar 2024 09:15 - Anna Maria Bigatti

- Related to Design #64: submat takes only vector<long> added

#21 - 08 Mar 2024 10:13 - Anna Maria Bigatti

- Related to Feature #1788: New MatrixView/function "FirstRows/FirstCols"? added

#22 - 08 Mar 2024 10:13 - Anna Maria Bigatti

- Subject changed from MatrixView for viewing a single row or column to MatrixView for viewing a single row or column (RowMat, ColMat)

#23 - 08 Mar 2024 10:15 - Anna Maria Bigatti

- Description updated

- Assignee set to John Abbott

#24 - 08 Mar 2024 17:20 - Anna Maria Bigatti

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

#25 - 20 Mar 2024 14:07 - Anna Maria Bigatti

- Description updated

#26 - 20 Mar 2024 14:08 - Anna Maria Bigatti

- Subject changed from MatrixView for viewing a single row or column (RowMat, ColMat) to MatrixView/function for viewing a single row or column (RowMat, ColMat)

- Description updated

#27 - 20 Mar 2024 14:09 - Anna Maria Bigatti

- Description updated