CoCoALib - Slug #1181

CpuTime is costly!

23 Apr 2018 12:21 - John Abbott

Status:	Closed	**Start date:**	23 Apr 2018
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Improving	**Estimated time:**	8.51 hours
Target version:	CoCoALib-0.99600	**Spent time:**	2.75 hours

Description

A call to **CpuTime()** is quite costly (on my machine with Fedora 25 Linux).

This means that a calls to CheckForTimeout() and to CheckForInterrupt are also costly!

Improve them!

Related issues:

Related to CoCoALib - Feature #638: Time limit: let user specify time limit f...	**Closed**	**27 Oct 2014**
Related to CoCoALib - Bug #1376: GBasisTimeout: not working as expected	**Closed**	**12 Dec 2019**

History

#1 - 23 Apr 2018 12:25 - John Abbott

I found the problem with an *ad hoc* program to search for 6x6 matrices with large ratio hadamard/det.
In the main loop I inserted a check for the CPU time consumed. The linux/shell time command indicated that around 20% of resources were "system"; when I modified the code to check every 100 iters, then the "system" time used dropped almost to 0.

I deduce that calling **CpuTime** is quite costly, so should not be done every iteration inside "cheap loops".

#2 - 23 Apr 2018 12:25 - John Abbott

- Related to Feature #638: Time limit: let user specify time limit for a computation added

#3 - 23 Apr 2018 12:30 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

It might be useful to have something like ProgressReporter which sets a flag when a time-limit has been exceeded?

Think about it!

#4 - 17 May 2018 17:58 - John Abbott

- Assignee set to John Abbott

- % Done changed from 10 to 30

I now have a prototype which works well in one small test.

The new design does not mimic the old interface... :-/

#5 - 18 May 2018 18:33 - John Abbott

Here is a brief comment about the design: I shall suppose that the class is called CpuTimeLimit

(1) simple use example:

```
CpuTimeLimit CheckTimeLimit(3.0); // limits is 3.0 seconds
for (int i=0; i < 1000000; ++i)
{
   CheckTimeLimit();  // will throw when limit is exceeded
   ...BODY OF LOOP...
}
```

(2) Use as fn arg:

```
vector<RingElem> GBasisTimeout(const vector<RingElem>& gens, const CpuTimeLimit& CheckTimeLimit)
{
  // Call CheckTimeLimit() inside any potentially long loops
}
```

```
// to call the fn do
GB = GBasisTimeout(gens, CpuTimeLimit(3.0));
```

NOTE: if we do this then we could even use a better name for the fn (such as GBasis)

#6 - 18 May 2018 18:34 - John Abbott

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

#7 - 14 Jun 2018 16:29 - John Abbott

- Status changed from In Progress to Feedback

- % Done changed from 30 to 90

Changing to **feedback** even though there is scope for improvement.

It seems to be difficult to do this properly in a multithreaded enviroment.

It is tedious to have to pass the CpuTimeLimit object as an optional parameter.

#8 - 31 Jul 2018 14:25 - Anna Maria Bigatti

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

- Estimated time set to 8.51 h

#9 - 12 Dec 2019 21:45 - John Abbott

- Related to Bug #1376: GBasisTimeout: not working as expected added