CoCoALib - Slug #1569

IsInRadical too slow (test-RadicalMembership)

22 Jan 2021 15:09 - John Abbott

Status:	Closed	**Start date:**	22 Jan 2021
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:	Improving	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99850	**Spent time:**	5.45 hours

Description

From a demo I showed my students...
I computed a "nasty" 0-dim radical, then wanted to show MinPowerInIdeal, but it took too long.
The culprit is the check IsInRadical which returns -1, if the poly is not in the radical.

```
/**/ use QQ[x,y,z,w];
/**/ I := ideal([x^3 +8*x*y*z -z*w^2,  y^3 -x^2*z +9*y^2*z,  z^3 -x*z*w +8*y*w^2,  w^3 +7*x*y +y^2
]);
/**/ radI := radical(I);
/**/ RGBrad := ReducedGBasis(radI);
/**/ g := last(RGBrad);
/**/ MinPowerInIdeal(g,I);  --> stopped after about 10 mins
```

Related issues:

Related to CoCoALib - Bug #1565: IsInRadical: gives "weird" error	**Closed**	**19 Jan 2021**
Related to CoCoA-5 - Bug #1610: IsInRadical: some more little bugs	**Closed**	**27 Sep 2021**
Related to CoCoALib - Slug #1136: IsInRadical: sometimes a bit slow	**Closed**	**06 Dec 2017**
Related to CoCoALib - Feature #1103: Pseudo-zero-dim ideals	**In Progress**	**19 Sep 2017**

History

#1 - 22 Jan 2021 15:09 - John Abbott

- Related to Bug #1565: IsInRadical: gives "weird" error added

#2 - 22 Jan 2021 16:22 - John Abbott

Not surprisingly the computation mod p is very quick.

BTW the answer to the computation as given is (almost certainly) 9... that is what I got mod p.

#3 - 27 Sep 2021 09:52 - Anna Maria Bigatti

- Related to Bug #1610: IsInRadical: some more little bugs added

#4 - 19 Jan 2024 16:57 - Anna Maria Bigatti

- Related to Slug #1136: IsInRadical: sometimes a bit slow added

#5 - 21 Jan 2024 20:33 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

The "stupid approach" gets the answer in 10s (on my Linux laptop).

```
/**/ t0 := CpuTime();
/**/ [g^k isin I | k in 1..9];
[false,  false,  false,  false,  false,  false,  false,  false,  true]
```

```
/**/ TimeFrom(t0);
9.970
```

#6 - 15 Feb 2024 16:41 - Anna Maria Bigatti

John Abbott wrote:

> The "stupid approach" gets the answer in 10s (on my Linux laptop).

Even faster like this.

```
/**/ g := last(RGBrad);
/**/ t0 := CpuTime();
/**/ for i := 1 to 9 do G := NF(G*g,I); print " ", i; endfor;
 1 2 3 4 5 6 7 8 9/**/ G;
0
/**/ TimeFrom(t0);
0.003
```

I tried this approach in the code in /SparsePolyOps-ideal-RadicalMembership.C but doesn't seem good in general: maybe for 0-dim?
I'll investigate.

#7 - 15 Feb 2024 22:35 - John Abbott

As far as I can see **MinPowerInIdeal** is already implemented, using the method Anna described above.
See near line 166 in SparsePolyOps-ideal-RadicalMembership.C

If char is 0, we could try reducing modulo a large-small prime. That would be fast(er), but the result is not guaranteed to be correct. Maybe there should be a VerificationLevel parameter?

It would also make sense to convert the input poly to its prim (*i.e.* make a good scalar multiple to clear denominators and avoid content.

I suppose the code ought to try computing IsInRadical and try NF of powers "in parallel"...

PS I let the original example run to completion: MinPowerInIdeal took about 1200s (20mins) on my computer

#8 - 16 Feb 2024 16:06 - Anna Maria Bigatti

- % Done changed from 10 to 60

John Abbott wrote:

> As far as I can see **MinPowerInIdeal** is already implemented, using the method Anna described above.
> See near line 166 in SparsePolyOps-ideal-RadicalMembership.C

Thanks.
I implemented IsInRadical as test on powers if ideal is 0-dim and "small" (l=len(QB)<=512):
if I is 0-dim then f is in radical(I) iff f^l is in I.

Now this example is fast.
Let's keep an eye on this with SetVerbosityLevel(40)

#9 - 16 Feb 2024 20:46 - John Abbott

- Related to Feature #1103: Pseudo-zero-dim ideals added

#10 - 16 Feb 2024 20:47 - John Abbott

I have confirmed that Anna's 0-dim check makes it much faster. I have added a link to issue #1103 which is about pseudo-0-dim ideals.

#11 - 23 Feb 2024 12:27 - Anna Maria Bigatti

there is a new problem (in test-exbugs.C).

```
use QQ[x];
IsInRadical(x, ideal(x^2)); --> false ?!?
```

#12 - 23 Feb 2024 12:28 - Anna Maria Bigatti

Anna Maria Bigatti wrote:

> there is a new problem (in test-exbugs.C).
> [...]

Fixed, it was an "off-by-one" error for the 0-dim case

#13 - 23 Feb 2024 12:29 - Anna Maria Bigatti

Implement special version for monomial ideals

#14 - 01 Mar 2024 09:40 - Anna Maria Bigatti

- Subject changed from IsInRadical too slow to IsInRadical too slow (test-RadicalMembership)

#15 - 01 Mar 2024 09:44 - Anna Maria Bigatti

There is a slow test in test-RadicalMembership1 which is slower than the others and very slow with debugging on (20mins).
Find which one it is and add code to skip it when compiling with debugging on.... or make it fast! ;-)

#16 - 01 Mar 2024 17:12 - Anna Maria Bigatti

Anna Maria Bigatti wrote:

> There is a slow test in test-RadicalMembership1 which is slower than the others and very slow with debugging on (20mins).
> Find which one it is and add code to skip it when compiling with debugging on.... or make it fast! ;-)

There were 6 tests on non-homog ideal which were slow (1s each) and fairly redundant.
I added some tests with homogeneous ideal (algorithm is the same, but runs faster), and commented out 5 of the "slow" ones.

I added some code to get the timings, and left it there, commented out.
I think it is useful to have some pre-inserted code for benchmarking when we modify some algorithm.

#17 - 01 Mar 2024 18:38 - Anna Maria Bigatti

Anna Maria Bigatti wrote:

> Implement special version for monomial ideals

Moved to issue #1786

#18 - 01 Mar 2024 18:40 - Anna Maria Bigatti

- Status changed from In Progress to Closed

- % Done changed from 60 to 100