CoCoA-5 - Feature #622

New function: RandomSubset

12 Sep 2014 10:38 - Anna Maria Bigatti

Status:	Closed	**Start date:**	12 Sep 2014
Priority:	Urgent	**Due date:**	
Assignee:	Anna Maria Bigatti	**% Done:**	100%
Category:	CoCoA-5 function: new	**Estimated time:**	6.01 hours
Target version:	CoCoA-5.1.2 summer 2015	**Spent time:**	5.95 hours

Description

Sometimes "all subsets" are far too many (compute how many! ;-) to fit into memory.
So checking a property on a suitable number of random subsets can be interesting.

Related issues:

Related to CoCoALib - Feature #379: Iter for subsets/tuples	**Closed**	**19 Jun 2013**
Related to CoCoA-5 - Feature #531: Package protected variables should know wh...	**Closed**	**09 Apr 2014**
Related to CoCoALib - Feature #715: RandomSubsetIndices, RandomTupleIndices?	**Closed**	**20 May 2015**

History

#1 - 12 Sep 2014 10:43 - Anna Maria Bigatti

- % Done changed from 0 to 20

- Estimated time set to 1.50 h

done. still missing: documentation, tests.
Would it be useful in CoCoALib? maybe just a random subset of 1..n?

#2 - 24 Apr 2015 11:20 - John Abbott

I wanted a RandomSubset function which generated a random subset of 1..n of cardinality r. I was surprised when CoCoA reported that I could not define the fn as the name was already a package exported variable.

I have a slight preference for my fn over the existing one which generated a random subset of a set represented as a list. The fns are closely related:

```
RandomSubset(L,r) == [L[i] | i in RandomSubsetJAA(len(L),r)];
RandomSubsetJAA(n,r) == RandomSubset(1..n, r)
```

Why do I prefer my fn? It somehow "feels simpler", and might be slightly easier to implement efficiently in C++; the current defn requires copying some/all elements of the input list L which could be expensive if the elements are large.

For completeness here is my defn (taken from Wikipedia "Reservoir sorting"?)

```
define RandomSubset(n,r)
  subset := 1..r;
  for k := r+1 to n do
    i := random(1,k);
    if i > r then continue; endif;
    subset[i] := k;
  endfor;
  return sorted(subset);
enddefine; -- RandomSubset
```

#3 - 24 Apr 2015 11:23 - John Abbott

We could simply have both; that way the user can choose which is more convenient (or efficient).

If we do choose to have both, should they have the same name or different names?

Maybe my fn should be called RandomIndexSubset? (long and a rather ugly)

#4 - 24 Apr 2015 17:59 - Anna Maria Bigatti

- % Done changed from 20 to 30

Let's start with just yours, for many reasons:
- it is simpler, as you said
- indeed that's also what I needed ;-)
- it can be implemented with the same philosophy in cocoalib (without fears for copies)
- you will write the documentation and tests (won't you? ;-)

#5 - 08 May 2015 13:32 - John Abbott

- Status changed from New to In Progress

Following CoCoALib conventions the return type should be vector<long>.

What should the values in the vector be? Should it be a subset of 0..(n-1) or a subset of 1..n? If we regard the values as indices then the former makes more sense in C++, but it might be more convenient/natural in CoCoA-5 to adopt the latter convention.

It would even be possible to implement both: in CoCoALib the range is 0..(n-1) while the "same function" in CoCoA-5 uses the range 1..n. Is this a good or a bad idea?

#6 - 08 May 2015 14:59 - John Abbott

- Priority changed from Normal to Urgent

#7 - 20 May 2015 10:38 - Anna Maria Bigatti

I suggest having also RandomTuple(n,r) which is indeed pretty simple: in CoCoA

```
define RandomTuple(L, r)
  return [ L[random(1,len(L))] | i in 1..r ];
enddefine; -- RandomSubset
```

Now I think in CoCoALib we should have RandomSubsetIndices(n,r) and RandomTupleIndices(n,r)
(and also RandomSubsetIndices(vec, r) and RandomTupleIndices(vec,r)?)

#8 - 20 May 2015 11:41 - Anna Maria Bigatti

I added to NotBuiltin.cpkg the functions

```
Export RandomSubset;
Export RandomSubsetIndices;
Export RandomTuple;
Export RandomTupleIndices;
```

now I write the documentation...

#9 - 20 May 2015 12:04 - Anna Maria Bigatti

documentation done (and cvs'ed)

When we implement the "Indices" in CoCoALib we'll remove them from "NotBuiltin".
(Remember: shift indices by 1)

#10 - 20 May 2015 12:48 - Anna Maria Bigatti

- Status changed from In Progress to Feedback

- % Done changed from 30 to 90

- Estimated time changed from 1.50 h to 3.00 h

#11 - 20 May 2015 13:50 - John Abbott

I have added defns to CoCoALib of RandomSubsetIndices and RandomTupleIndices. These names are very long :-/

Not entirely convinced by the usefulness of RandomTupleIndices.

Also added defns to BuiltinFunctions.C

Not yet: doc, examples, tests.

#12 - 20 May 2015 15:22 - John Abbott

Where should the new functions go?

Currently they are in a new file combinatorics.C because I could not see any other good place tro put them. Is this OK?

#13 - 20 May 2015 15:28 - Anna Maria Bigatti

John Abbott wrote:

> Where should the new functions go?
>
> Currently they are in a new file combinatorics.C because I could not see any other good place tro put them. Is this OK?

yes

#14 - 20 May 2015 15:40 - John Abbott

Checked in.
You'll have to remove interrupt.C from src/AlgebraicCore/Makefile until I can check those files in... hot choc break now! :-)

#15 - 20 May 2015 17:20 - Anna Maria Bigatti

John Abbott wrote:

> I have added defns to CoCoALib of RandomSubsetIndices and RandomTupleIndices. These names are very long :-/
>
> Not entirely convinced by the usefulness of RandomTupleIndices.

Nor am I, for cocoalib. But I know someone who should make good use of RandomTuple in cocoa-5 ;-)

> Also added defns to BuiltinFunctions.C

cvs?

> Not yet: doc, examples, tests.

I can do that.

#16 - 20 May 2015 17:57 - John Abbott

I have checked in RandomSubsetIndices and RandomTupleIndices (in CoCoALib); also the interface fns in BuiltinFunctions.C.

I have not implemented RandomSubset in C++; nor RandomTuple.

#17 - 30 Jun 2015 14:19 - Anna Maria Bigatti

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

- Estimated time changed from 3.00 h to 6.01 h