packedindex manual

Thomas Jahns
Stefan Kurtz

Research Group for Genomeinformatics
Center for Bioinformatics
University of Hamburg
Bundesstrasse 43
20146 Hamburg
Germany

kurtz@zbh.uni-hamburg.de

August 24, 2012

1 Introduction

The *packedindex* tools of the *GenomeTools* package facilitate the creation of sequence index files which implement the FM index[1] concept for biological sequences with some more recent features incorporated[2].

The *packedindex* routines and tools are written in `C` and are part of the *GenomeTools* library [3]. *packedindex* methods are called as part of the single binary named `gt`.

The *packedindex* tools depend on routines of the *Suffixerator* program to prepare the data stored in the index. Many command-line options are therefore the same.

2 Command line tools

Some text is highlighted by different fonts according to the following rules.

- **`toolname`** is used for the names of software tools.
- *`filename`* is used for file names.
- `-option` is used for program options.
- `argument` is used for the argument(s) of an option.

2.1 The *packedindex* tools

The creation and integrity checks of the *packedindex* tools are available as a sub-tool of `gt`. Each of those is called by giving **`packedindex`** as the first argument of the **`gt`** command-line.

2.1.1 Options common to all tools

`-help`
: Prints a summary of available options to standard output and terminates with exit code 0.

`-v`
: Turns on verbose progress information.

`-version`
: Displays **`genometools`** version information.

2.1.2 The creation of packedindex files

The command to build a *packedindex* is

`gt packedindex mkindex` [Options] `-db` *`sequenceinput...`*

Table 1: Overview of the *packedindex* **mkindex** options.

-bsize	specify the number of symbols joined in a block
-blbuck	specify number of blocks to join in a bucket
-locfreq	specify interval at which locate marks will be inserted
-locbitmap	force/deny use of the bitmap tagging method
-sprank	build source rank table so that regeneration of original sequence is possible without the usual gaps
-sprankilog	define log of interval used for the intermediate table

The following options are specific to the *packedindex* **mkindex** tool:

Except for the output options -suf, -lcp, and -bwt, all other options are the same as used for the **suffixerator** tool.

-bsize *u*

Group *u* symbols in one block. While bigger blocks can speed up index access, the block size should be chosen so that

$$\alpha^u u \lceil \log_2 \alpha \rceil$$

does not exceed the relevant cache size (typically of the 2nd level chache) of the system.

The default of a block size of 8 results in a lookup-table size of 128kiB for the DNA alphabet. Systems with larger 2nd level cache like 4MiB on the Intel Core2 Duo might show improved performance when choosing block size 10.

-blbuck *k*

Group this many blocks so that every atomic query of the index will on average require $\frac{k}{2}$ lookups of the block table (see option -bsize). While smaller blocks decrease query times, the corresponding increase in memory usage of the index will eventually defeat this effect.

-locfreq *f*

Put location marks at interval f into the index. For a sequence of length n, $\frac{n}{f}\lceil \log_2 n \rceil$ bits of storage will be required.

Locate queries (i.e. finding the position of matches) will require $\frac{f}{2}$ operations. Again having an index that does not fit main memory will eventually defeat the speed gains of lower intervals.

-locbitmap *yes* or *no*

Force/disable the use of a bitmap for locate marks. Finding marked positions will be faster if a bitmap is used but will require n bits for a sequence of length n. If no locate bitmap is used, extra space usage will be $\frac{n}{f}\lceil \log_2 uk \rceil$.

By default the optimal value in terms of space usage will be chosen.

-sprank *yes* or *no*

Normally special characters (N/X, separators and terminator) are encoding so that the reverse mapping required for context regeneration cannot pass over such symbols. If this option is given, an in-memory structure will be built to facilitate queries which allow to store necessary extra information in the index so that full context (the entire original sequence actually) can be regenerated.

`-sprankilog` s

Set logarithm of interval used for the intermediate table (see `-sprank` option). Index creation might be faster if a lower value than the default $s = \log_2 \log_2 n$ is used and sufficient memory is available.

Since every increment by 1 doubles the memory usage of the table (which is not stored itself in the index) during index construction two high values might actually slow index construction.

`-ctxilog` c

Set logarithm of interval to use for the creation of a reverse lookup table. Said table will be stored alongside the index in a file with the same name prefix but suffix *.ccxm* so that multiple reverse lookup accelerator tables can be stored for one index. So it's possible to load the one fitting available memory best later.

The reverse lookup table is restricted to indices computed with the `-sprank` option. In those cases the index can replace the encoded sequence (stored in *prefix.esq*) completely.

The reverse suffix indices will be sampled at interval length 2^c

2.1.3 Building reverse lookup tables after index creation

The **mkctxmap** sub-tool facilitates late building of a reverse lookup table, it can use the *prefix.suf* table as well as the locate information inside the packedindex structure (i.e. of any packedindex not constructed with `-locfreq` set to 0).

The **mkctxmap** sub-tool only accepts the `--ctxilog`, `--v`, `--version`, and `--help` options already described for the **packedindex mkindex** tool.

Given a previously computed suffix array or packedindex project *chr01*, one can build an additional reverse lookup-table with interval 32 like this:

```
$ gt packedindex mkctxmap -ctxilog 5 chr01
```

2.1.4 Deriving packedindex files from a suffix array

The **trsuftab** sub-tool will translate an index by reading suffix array and encoded sequence files. Thus the options specific to the packedindex properties are the same as previously shown 2.1.2 but the suffixerator options are not available. Instead the latter are implicit in the properties of the suffix array and encoded sequence indices used as input for this tool.

Thus to produce a packedindex with default options where a suffix array has already been created in chr01.{prj|suf|esq} use:

```
$ gt packedindex trsuftab chr01
```

2.1.5 packedindex integrity check tool

The integrity check tool is intended to verify that an index correctly represents the encoded BWT sequence. Therefore the .bwt file for the index project in question must have been generated as a prerequisite for this check.

To check a *packedindex* index file BWT sequence representation issue

gt packedindex chkintegrity [Options] *indexname*

where *indexname* refers to the name given previously on index creation (or automatically derived from the sequence input file set). The .bwt table file must have previously been generated with the **suffixerator** tool. See section 3 for an example.

Table 2: Overview of the *packedindex* **chkintegrity** options.

-skip	specify the number of symbols to skip before starting the comparison
-ticks	print a dot after this many symbols processed correctly
-ext-rank-check	do additional checks of rank query results

-skip *k*
: Start the comparison *k* has to be a positive integer. If this option is not selected by the user, then L_{ex} is set to 30 by default.

-ticks *k*
: As a form of progress meter, print a dot after k symbols of the packedindex and the reference file (*indexname*.*bwt*) compared equal and rank information was computed correctly.

-ext-rank-check *yes*||*no*
: If yes, enables additional checks of the rank results, i.e. for all positions rank queries will be computed for all symbols of the alphabet and not only for the symbol for which the rank value changed in the reference.

2.1.6 Check *packedindex* search accuracy tool

This check verifies that a packedindex correctly represents both the original sequence and correctly delivers the exact same search results as the suffix array.

To check a *packedindex* index file issue

gt packedindex chksearch [Options] *indexname*

where *indexname* refers to the name given previously on index creation (or automatically derived from the sequence input file set).

The encoded sequence (*.esq*) and suffix array table (*.suf*) must have been previously generated with the **suffixerator** tool.

Table 3: Overview of the *packedindex* **chksearch** options.

-minpatlen	shortest pattern length
-maxpatlen	longest pattern length
-ticks	print a dot after this many symbols processed correctly
-nsamples	generate this many patterns to search for
-chksfxarray	do additional checks of stored suffix array data

`-minpatlen` l
: Perform searches for patterns that are at least of length l.

`-maxpatlen` m
: Perform searches for patterns that are at most of length l.

`-nsamples` k
: Generate k patterns to search for, defaults to 1000.

`-chksfxarray` *yes*||*no*
: Verify integrity of stored suffix array positions and test the regenerated context functionality if the index was constructed in a reversible fashion (option `-sprank` on index construction).

  In the latter case the test will take considerable time.

`-ticks` k
: Print a dot after this many searches completed correctly. If `-chksfxarray` is set, also prints a dot for every k suffix array values compared.

3 Examples

3.1 Create an index for sequence data from the *gttestdata* repository[4]

Amongst the more voluminous sequence data sets there is the *Saccharomyces cerevisiae* genome in the *ltrharvest/s_cer* sub-directory. This example will show how to build an index for chromosome 1 (file *chr01.19960731.fsa.gz*).

First compute the *packedindex* index file (this assumes *chr01.19960731.fsa.gz* has been copied/linked to the current directory and the gt binary is in the PATH):

```
$ gt packedindex mkindex -db chr01.19960731.fsa.gz -indexname chr01 -tis -des
# TIME determining sequence length and number of special symbols 0.04
# TIME computing sequence encoding 0.04
[...]
# TIME overall 0.71
$
```

Then build the reference files that the index subsumes:

```
$ gt suffixerator -db chr01.19960731.fsa.gz -indexname chr01 -bwt -suf -des
# TIME determining sequence length and number of special symbols 0.04
# TIME computing sequence encoding 0.04
[...]
# TIME overall 0.31
$
```

This index can then be verified or used in one of the tools supporting it. For integrity check use:

```
$ gt packedindex chkintegrity chr01
# Using index over sequence 230210 symbols long.
[...]
```

```
..
$ gt packedindex chksearch chr01
Using pre-computed sequence index.
[...]
Finished 1000 of 1000 matchings successfully.
$
```

References

[1] Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications, September 03 2000.

[2] Paolo Ferragina, Giovanni Manzini, Veli Mäkinen, and Gonzalo Navarro. Compressed representations of sequences and full-text indexes. *ACM Transactions on Algorithms*, 3(2), 2007.

[3] Gordon Gremme. The GENOMETOOLS genome analysis system. `http://genometools.org`.

[4] GENOMETOOLS test data set. `git://genometools.org/gttestdata.git`.