Minimal perfect hash functions in large scale bioinformatics Problem
Antoine Limasset, Camille Marchet, Pierre Peterlongo, Lucie Bittner

To cite this version:
Antoine Limasset, Camille Marchet, Pierre Peterlongo, Lucie Bittner. Minimal perfect hash functions in large scale bioinformatics Problem. JOBIM 2016, Jun 2016, Lyon, France. <hal-01341718>

HAL Id: hal-01341718
https://hal.archives-ouvertes.fr/hal-01341718
Submitted on 4 Jul 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Minimal perfect hash functions in large scale bioinformatics

Antoine Limasset, Camille Marchet and Pierre Peterlongo
INRIA/IRISA/GenScale, Campus de Beaulieu, 35042 Rennes Cedex, France
antoine.limasset@irisa.fr, camille.marchet@irisa.fr, pierre.peterlongo@irisa.fr

**Problem**

Quasi-dictionary

- No membership operation
- A ‘stranger’ key can be associated to a value

**References**

Bowtie2

BWA

Starcode

BLAST

**Results**

Still have to assess the qualitative aspects of our methods ...

**Team**

- Antoine Limasset
- Camille Marchet
- Pierre Peterlongo

**Links**

Bbhash library:
github.com/rizkg/BBHash

Quasi-dictionary:
github.com/pierrepeterlongo/quasi_dictionary

Short Read Connector:
github.com/GATB/rconnector

**Hash functions**

- a.b.c.d.e : hashable elements (e.g. strings, images, etc.)
- hash function
- image [0-0] of hash function (e.g. indices of buckets in a hash table)

**BBhash library**

- Memory efficient (less than 3bits per key)
- Fast query (200ns)
- Fast to construct (even for billions elements)

**Quasi-dictionary**

- Put a fingerprint in the value and check it at the query
- False positive rate :
  \[
  \frac{2^{2k} - 1}{2^{2k}} = \frac{1}{1} - \frac{1}{2^k}
  \]
- Memory consumption :
  Bit/elements : 10 FP rate: 1/10²
  Bit/elements : 20 FP rate: 1/10⁵
  Bit/elements : 30 FP rate: 1/10⁸

**Short Read Connector tools**

Given A and B sets of reads:

**SRC Linker**
Output reads in A that has T Kmers that appear in set B.

**SRC Counter**
Output reads in A that has T Kmers in common with the reads of B and estimate their coverage.

**Questions**

Dataset comparison:
Detection of similar reads inter or intra datasets

**Problem**

Indexing huge set of elements

- Memory efficient (less than 3bits per key)
- Fast query (200ns)
- Fast to construct (even for billions elements)

**Results**

Time for indexing and querying 1M reads with SRC Linker

Time for indexing and querying 100M reads with SRC Linker

Memory used for indexing 100M reads with SRC Counter

Less pressure on your machine!