UniPi past and ongoing activity round-up
[Tasks T₁, T₂, T₃, T₄]

Paolo FERRAGINA
Giuseppe PRENCIPE

9 February 2022
One new PhD student

Antonio Boffa
PhD Student
LinkedIn · Website

Francesco Tosoni
PhD Student
LinkedIn

Andrea Guerra
PhD Student
Team

One new PostDoc

Giorgio Vinciguerra
PhD Student
LinkedIn · Website

Università di Pisa
Department of Computer Science

Ph.D. Thesis
Learning-based compressed data structures

Giorgio Vinciguerra
giorgio.vinciguerra@phd.unipi.it

Supervisor
Paolo Ferragina

Internal committee
- Luca Oneto
  Università di Genova

External referees
- Stratos Idreos
  Harvard University
- Gonzalo Navarro
  Universidad de Chile

Salvatore Ruggieri
Università di Pisa

October 31, 2021
Why worst-case bounds are important?

Tuned Hybrid RMI implementation from Marcus et al. [VLDB 2021]
- PGM is faster for **insert-heavy workloads** (< 25% queries)
- PGM and ART are faster for **balanced workloads** (50% queries)
- ART is faster for **query-heavy workloads** (75% queries)
- PGM, ART and ALEX are faster for **query-only workloads** (100% queries)
Dynamic scenario: overall memory usage

Overall = keys (8 bytes) + values (8 bytes) + index, including space due to half empty nodes/slots

1. PGM is the most memory-efficient (12.9 GB)
2. B-tree is the second-best (16.5 GB)
3. ALEX is +15% than B-tree, and +47% than PGM
4. ART is the most memory-hungry (34.6 GB)

Take-away msg:
- some learned indexes are larger than traditional ones
- PGM is fast in query/ins ops and very space succinct
The PGM software library

Variants of the PGM

- CompressedPGM
- EliasFanoPGM
- BucketingPGM
- Big integers (up to 256 bytes)

MultidimensionalPGM

- Orthogonal range queries
- k-NN queries (thanks DBlab @ Nagoya Univ.)
Our Algorithm Engineering Achievements

Compressed and Learned Data Structures

Software & Datasets

**PGM-index**
An optimal learned data structure that enables fast point and range searches in arrays of billions of items using orders of magnitude less space than traditional indexes.
GitHub • Website

**LA-vector**
Compressed learned bitvector supporting efficient rank and select queries.
GitHub

**FM-index v2**
A full-text index data structure that combines compression and indexing by encapsulating in a single compressed file both the original file plus some indexing Information.
Website

---

**Block-ε tree**
Compressed rank/select dictionary exploiting approximate linearity and repetitiveness.
GitHub

**Pizza & Chili**
Datasets for compressed indexes and test collections benchmarking
GitHub • Website
Our Theory Achievements
~


Second round of review @ Journal


Two new results on Trie compression (subm. SIGIR ‘22), Compressed Matrix for Linear Algebra (subm. VLDB ‘22), Efficiency of properly designed NN wrt tries (on going with UniMI)