Efficiency for Real-World Applications

Giulio Ermanno Pibiri

giulio.ermanno.pibiri@isti.cnr.it
http://pages.di.unipi.it/pibiri

04/03/2021
PhD in Computer Science from the University of Pisa (November 2015 - October 2018)

Thesis defended on 08/03/2019

PostDoc at HPCLab since November 2018

giulio.ermanno.pibiri@isti.cnr.it

http://pages.di.unipi.it/pibiri
https://github.com/jermp
A good programmer cares about data structures and their relationships.

Linus Torvalds

1. **Data Structures**
   - how quickly a program does its work - **faster** work

2. **Algorithms**
   - how much work is required by a program - **less** work

3. **Data Compression**
What is Efficiency?

**Space efficiency** means storing the data in **compressed** format.

**Time efficiency?**

You can use the theory. Time efficiency means **“low” asymptotic complexity**.

You can (also) run **experiments**. Time efficiency means:
- cache-friendliness
- few data dependencies
- predictable branches
- super scalar execution

Yet, no textbook can teach you this.
Goal

Design time/space efficient algorithms and data structures with:
- appealing theoretical guarantees;
- a significant impact in practice, i.e., applications to problems at an industrial scale.

Inverted indexes
- Google
- Yahoo!
- Bing
- Ontotext
- RDF indexing

Databases
- IBM
- Dropbox
- Oracle

E-Commerce
- eBay
- Amazon

Geo-spatial data
- Google Maps
- Pokémon Go

Graph-compression
- Facebook
- Twitter
- LinkedIn
<table>
<thead>
<tr>
<th>Problem</th>
<th>Publications</th>
<th>Github Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDF Triples Indexing</td>
<td>TKDE 2020</td>
<td><a href="https://github.com/jermp/rdf_indexes">https://github.com/jermp/rdf_indexes</a></td>
</tr>
<tr>
<td>Query Auto-Completion</td>
<td>SIGIR 2020</td>
<td><a href="https://github.com/jermp/autocomplete">https://github.com/jermp/autocomplete</a></td>
</tr>
<tr>
<td>Prefix-Sums</td>
<td>SPE 2020</td>
<td><a href="https://github.com/jermp/psds">https://github.com/jermp/psds</a></td>
</tr>
<tr>
<td>Rank/Select Over Bitmaps</td>
<td>INFOSYS 2021</td>
<td><a href="https://github.com/jermp/mutable_rank_select">https://github.com/jermp/mutable_rank_select</a></td>
</tr>
</tbody>
</table>
Inverted Indexing

You have a large collection of Web pages, like several millions. Given $k$ words, how to find all Web pages where these words occur?

Build an inverted index data structure.

On Gov2 (~5 billion integers):
- Use PEF (SIGIR 2014) for 3.12 bits/int and 3 ms/query
- Use Slicing (DCC 2021) for 4.31 bits/int and 1 ms/query
You have a large collection of q-grams, like 11 billions. How, given a q-gram, return its context probability as fast as possible?

Build a compressed trie data structure.

On GoogleBooksV2 (~11 billion q-grams):
- Use EF-Trie (SIGIR 2017, TOIS 2019) for **1.31 bytes/q-gram** and **2 µs/query**
- KenLM (best alternative) is much larger with scalability problems
RDF Triples Indexing

You have a large collection of RDF triples (S,P,O), like 350 millions. Given a wildcard query like (? ? O) or (? P ?), how to return all matching triples?

Build a compressed trie data structure.

On DBpedia (~350 millions of triples):
- Use HDT (W3 standard) for **77 bits/triple**
- Use 2T/3T (TKDE 2020) for **54 bits/triple** but **3-5X** faster queries on average
Query Auto-Completion

Given a collection $S$ of scored strings and a partially completed user query $Q$, how to find the top-$k$ strings that “match” $Q$ in $S$?
You work at Facebook. Your task is to find which users didn't ever create a post with an emoji. You have the list of users (2 billions), which you can scan several times, and a huge list of posts (user and text), which you can scan only once. Also, you have one computer, with 1 GB of RAM. Can you do it? How?
You work at Facebook. Your task is to find which users didn't ever create a post with an emoji. You have the list of users (2 billions), which you can scan several times, and a huge list of posts (user and text), which you can scan only once. Also, you have one computer, with 1 GB of RAM. Can you do it? How?
Take-home messages

• Efficiency to deliver better services by using less resources. Impact is far-reaching and implies substantial economic gains.

• Compression is mandatory if your data are “big”.

• Experiments are primary: design driven by numbers.

Drop me a line if you are interested in this stuff!
Thanks for your attention!

Any questions?