Karlsruhe Institute of Technology

Probability and Computing – Retrieval Data Structures

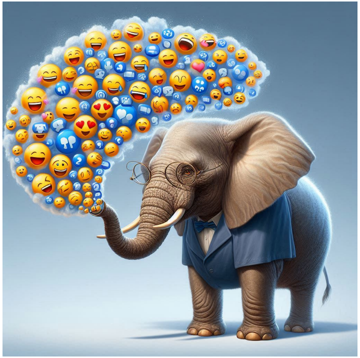
Stefan Walzer | WS 2025/2026

Results of the Course Evaluation

Karlsruhe Institute of Technology

Summary of Results ($n = 17$)

- content somewhat difficult... (3.4 / 5)
- ... but mostly clear (1.8 / 5)
- Amount of work is medium... (3.1 / 5)
- ... but you learn a lot (1.6 / 5)
- learn materials are good (1.9 / 5)
- overall grade is 1.4



Full evaluation results are available on the website.

Content

The Static Retrieval Problem

The retrieval data type (for universe D, range $[k]$)

construct(f):

input: function $f : S \to [k]$ //$f \subseteq D \times [k]$
where $S \subseteq D$ has size $n = |S|$

output: data structure R.

eval$_R(x)$:

input: $x \in D$

output: some value in $[k]$

requirement: **eval**$_R(x) = f(x)$ for all $x \in S$

The price to pay

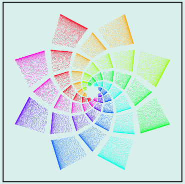
- R cannot be used to decide "is $x \in S$?"
- **eval**$_R(x)$ is *unspecified* if $x \notin S$.

Goals

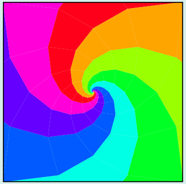
- space requirement of R is $\mathcal{O}(n \log k)$ bits
  - possibly even $n \log_2(k) + o(n)$
  - ⚠ naively storing f needs $\Omega(n(\log(k) + \log(|D|)))$
- ideally running time of **eval**$_R$ is $\mathcal{O}(1)$
- ideally running time of **construct** is $\mathcal{O}(n)$

Intuition

f :



R :

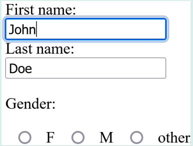


- R is a *continuation* of f
- information about the domain S is lost.

Motivation for Static Retrieval (somewhat contrived)

Karlsruhe Institute of Technology

Task: Predict gender based on first name



- want $\geq 99\%$ accuracy
- client side only
- lightweight

Have large data base:

Annotated list of 10000 most common first names.

$$f : \{\text{Dave} \mapsto M, \text{Joanna} \mapsto F, \text{Christina} \mapsto F, \ldots\}$$

$\approx$ 10 bytes per name, too large to send to client.

Solution using retrieval

- send $R = \textbf{construct}(f)$ to client
  $\hookrightarrow \approx 1$ bit per name
- prefill gender with $\textbf{eval}_R(\text{firstName})$

Weaknesses:

May guess incorrectly if

- name is ambiguous (“Kim”, “Chris”)
- user is *other* / prefers not to say
- name not listed in f (e.g. “Crhistina”, “Inghean”)
  $\hookrightarrow$ would be better to *refrain from guessing*

Static Filters / AMQs

Exercise: Filters from Retrieval

Good retrieval data structures yield good static filters.

Content

Cuckoo-Style Retrieval using Peeling for $f : S \to \{0, \dots, k-1\}$

Retrieval Data Structure $R = (h_1, h_2, h_3, A)$

- $A \in \{0, \dots, k-1\}^m$ is array of cleverly chosen values
- $m = \frac{n}{0.81} = 1.23n$ //0.81 is peeling threshold $c_3^{\triangle}$
- $h_1, h_2, h_3 \sim \mathcal{U}([m]^D)$ //SUHA
- $\textbf{eval}_R(x) := (A[h_1(x)] + A[h_2(x)] + A[h_3(x)]) \bmod k$

Performance

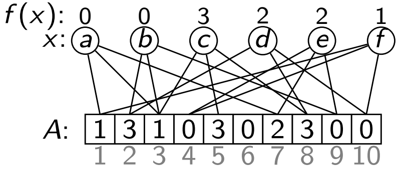
- space $1.23n\lceil \log_2(k) \rceil$ bits
- eval in $\mathcal{O}(1)$
- construct in $\mathcal{O}(n)$

How does **construct**(f) choose A?

If $A[j]$ is only used by x_i then setting $A[j]$ *in the end* takes care of x_i without affecting other keys.

$\hookrightarrow$ can forget about x_i "for now" and focus on the rest

$\hookrightarrow$ if configuration is peelable, this takes care of all keys



$(k = 4)$

Equations (mod k for $k = 4$)

c: $A[5] := 3 - A[3] - A[8]$

d: $A[8] := 2 - A[2] - A[10]$

b: $A[2] := 0 - A[3] - A[9]$

a: $A[3] := 0 - A[1] - A[7]$

e: $A[7] := 2 - A[4] - A[9]$

f: $A[1] := 1 - A[4] - A[10]$

Cuckoo-Style Retrieval using Linear Algebra over the field $\mathbb{F}_2 = \{0,1\}$

Karlsruhe Institute of Technology

Cuckoo-style retrieval for $f : S \to \mathbb{F}_2^r$ with $|S| = n$

Pick $m \geq n$. Data structure is pair $R = (h : D \to \mathbb{F}_2^m, \vec{z} \in \mathbb{F}_2^{m \times r})$ such that $h(x)^T \cdot \vec{z} = f(x)$ for all $x \in S$.

$r = 3$
$m = 7$
$n = 5$

$$\begin{pmatrix} \text{---}h(x_1)\text{---} \\ \text{---}h(x_2)\text{---} \\ \text{---}h(x_3)\text{---} \\ \text{---}h(x_4)\text{---} \\ \text{---}h(x_5)\text{---} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \end{pmatrix} \overset{!}{=} \begin{pmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \\ f(x_5) \end{pmatrix}$$

Goals when Choosing h

- i **success probability**: rows of matrix $(h(x))_{x \in S}$ must be linearly independent
- ii **construction time**: linear system should be easy to solve
- iii **query time**: products $h(x) \cdot z$ should be fast to compute
- iv **space**: $\alpha = \frac{n}{m}$ should be close to 1

What the peeling-based approach achieves

- i $1 - \mathcal{O}(1/m)$ (success iff peelable)
- ii $\mathcal{O}(n)$ (time to run peeling)
- iii $\mathcal{O}(1)$ (three memory accesses two $\oplus$-additions)
- iv $\alpha = 0.81$ (peeling threshold)

What more could we hope for?

- i -
- ii better cache efficiency
- iii better cache efficiency
- iv $\alpha = 1 - o(1)$

Teaser: “Ribbon Retrieval”

Standard Ribbon Retrieval

Idea: $\boldsymbol{h(x)}$ = “randomly placed block of $\mathcal{O}(\frac{\log n}{1-\alpha})$ random bits”

$$
\begin{pmatrix}
 & & 0 & 1 & 1 & 0 & 0 & 1 & & & & & & & \\
1 & 0 & 1 & 0 & 0 & 1 & & & & & & & & & \\
 & & & & & 0 & 1 & 0 & 1 & 1 & 0 & & & & \\
 & & & & 1 & 0 & 0 & 0 & 1 & 0 & & & & & \\
 & 0 & 1 & 1 & 0 & 1 & 1 & & & & & & & & \\
 & & & & & & & & & 0 & 1 & 0 & 1 & 0 & 1 \\
 & & 0 & 1 & 0 & 1 & 1 & 1 & & & & & & & \\
 & & & & & & 1 & 1 & 1 & 0 & 0 & 0 & & &
\end{pmatrix}
\begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \\ z_8 \\ z_9 \\ z_{10} \\ z_{11} \\ z_{12} \\ z_{13} \\ z_{14} \\ z_{15} \end{pmatrix}
\overset{!}{=}
\begin{pmatrix} 010 \\ 011 \\ 101 \\ 111 \\ 101 \\ 000 \\ 010 \\ 100 \end{pmatrix}
$$

$$\underbrace{1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0}_{\frac{\log n}{1-\alpha}}$$

Bumped Ribbon Retrieval

Further tricks allow reducing the width of the blocks to $\log n$. End result after a lot of effort[a]:

- construction $\mathcal{O}(n \log n)$
  // cache efficient
- query $\mathcal{O}(r)$ // cache efficient
- bits per key: $r + \mathcal{O}(\frac{\log \log n}{\log^2 n})$.

[a] *Ribbon: Fast Succinct Static Retrieval and Approximate Membership*, Martin Dietzfelbinger, Peter C. Dillinger, Lorenz Hübschle-Schneider, Peter Sanders, Stefan Walzer, 2026.

Summary

~~John $\mapsto$ ♂~~
~~Mary $\mapsto$ ♀~~
~~Lisa $\mapsto$ ♀~~
~~Carl $\mapsto$ ♂~~
~~Jane $\mapsto$ ♀~~

Static Retrieval

- constructed for $f : S \to [k]$
- goal: $\approx \log_2(k)$ bits per key
- does not store S

$$\textbf{eval}_R(x) = \begin{cases} f(x) & \text{if } x \in S \\ \textit{unspecified} & \text{if } x \notin S \end{cases}$$

- insert & delete not supported

Dictionary / Hash Table

- constructed for $f : S \to [k]$ // $S \subseteq D$
- goal: $\approx \log_2(D) + \log_2(k)$ bits per key
- stores S

$$\textbf{lookup}_R(x) = \begin{cases} f(x) & \text{if } x \in S \\ \bot & \text{if } x \notin S \end{cases}$$

- insert & delete usually supported

Constructions discussed here

- cuckoo-style retrieval using peeling
- cuckoo-style retrieval using linear algebra with $\mathbb{F}_2$
- approaches using perfect hashing (next lecture)

Remark: There is more...

- compressed retrieval data structures
- learned retrieval data structures
- active research @ITI Sanders!

Appendix: Possible Exam Questions

- What functionality does a retrieval data structure provide?
- What are the advantages and disadvantages compared to a standard hash table?
- What are applications of retrieval data structures?
- Regarding retrieval data structures using the fingerprint approach:
  - How is the data structure constructed? How does the query algorithm work?
  - What is “bumping” and why do we need it?
  - What are construction and access times, and what is the memory usage?
- Regarding retrieval data structures based on the peeling algorithm:
  - How is the data structure constructed? How does the query algorithm work?
  - What are construction and access times, and what is the memory usage?
- Regarding retrieval data structures based on linear algebra over the field $\mathbb{F}_2$:
  - What is the general framework? In what sense does the peeling-algorithm approach also fit into this framework?
  - What goals should one keep in mind when choosing the function h?
- Ribbon retrieval is not exam-relevant.