

Probability and Computing – Retrieval Data Structures

Stefan Walzer | WS 2024/2025



Zusammenfassung der Befragung ($n = 8$)

- Inhalt eher schwer... (3.8 / 5)
- ... aber anschaulich (1.5 / 5)
- Aufwand eher hoch... (3.4 / 5)
- ... aber sehr lehrreich (1.3 / 5)
- Aufwand Vorbereitung/Nachbereitung
 - 0h-1h (× 2)
 - 1h-2h (× 4)
 - 2h-3h (× 2)
- sehr geeignete Materialien (1.4 / 5)
- insgesamt noch sehr gut (Note 1.4)



Weitere Rückmeldungen oder Verbesserungsvorschläge?

Verbleibende Veranstaltungen & Prüfungstermine

Do 23.1. reguläre Vorlesung Peeling
Di 28.1. reguläre Übung
Do 30.1. Vorlesung Retrieval mit Übung
Do 6.2. Q&A, weitere Übungsaufgaben, könnte ausfallen
Di 11.2. Übung fällt definitiv aus
Do 13.2. *Gastvorlesung Hans-Peter Lehmann: Perfect Hashing*
15.2. *offizielles Ende der Vorlesungszeit*

Mi 26.2. Prüfungstermine
Do 27.2. Prüfungstermine
Fr 28.2. Prüfungstermine

Mi 26.3. Prüfungstermine
Do 27.3. Prüfungstermine
Fr 28.3. Prüfungstermine

Andere Prüfungstermine auf Anfrage vermutlich möglich.

Prüfung (mündlich, 20 Minuten)

- Anmeldung über das Sekretariat:
 - Anja Blancani (blancani@kit.edu)
 - cc an mich (stefan.walzer@kit.edu)
 - Bitte angeben:
 - Vollständiger Name
 - Matrikelnummer
 - Studienfach
 - Version der Prüfungsordnung
- Abmeldung auch über das Sekretariat
- Ort: Stefans Büro (50.34, Raum 209).
- Zeitslots jeweils:
 - 10:00, 10:25, 10:50, 11:15
 - 14:00, 14:25, 14:50, 15:15
- Inhalte von Vorlesung *und* Übung

1. The Retrieval Problem

- Definition
- Motivation

2. Retrieval based on Fingerprinting

3. Cuckoo-Style Retrieval

- Using Peeling
- In General Using Linear Algebra
- Teaser: Ribbon Retrieval

4. Summary

The Retrieval Problem

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

construct(f):

input: function $f : S \rightarrow [k]$ // $f \subseteq D \times [k]$
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 $x \in S$

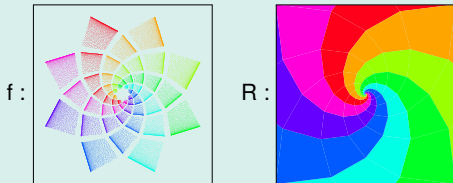
The price to pay

- 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 $n \lceil \log_2(k) \rceil + o(n)$
 - $\triangle!$ 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



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

Motivation for Retrieval

Task: Predict gender based on first name

First name:

Last name:

Gender:
 F M other

- want $\geq 90\%$ accuracy
- client side only
- lightweight

Solution using retrieval

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

Have large data base:

Annotated list of 10000 most common first names.

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

≈ 10 bytes per name, too large to send to client.

Weaknesses:

May guess incorrectly if

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

Exercise: Filters from Retrieval

Good retrieval data structures yield good static filters.

1. The Retrieval Problem

- Definition
- Motivation

2. Retrieval based on Fingerprinting

3. Cuckoo-Style Retrieval

- Using Peeling
- In General Using Linear Algebra
- Teaser: Ribbon Retrieval

4. Summary

Fingerprint-Based Retrieval

Idea 0: Use dictionaries after all

Idea 1: Store fingerprint-value pairs instead of key-value pairs

- Sample fingerprint hash function $fp \sim \mathcal{U}([\ell]^D)$ for small ℓ .

Idea 2: Partition into b small buckets (Example: $b = 3$)

- Sample partition hash function $part \sim \mathcal{U}([b]^D)$.
- Use many dictionaries $dict_1, \dots, dict_b$ of small capacity c (example: $c = 4$).
- Store $fp(x) \mapsto f(x)$ in $dict_{part(x)}$.

Idea 3: *Bump* inconvenient keys to fallback data structure

Recursively constructed, for small fraction of keys.

problem

$(x, f(x))$ too large
fingerprint collisions
dict_i might overflow

how to address?

fingerprinting: $(fp(x), f(x))$ is small
reduced by partitioning, solved by bumping colliding keys
bump excess keys



In expectation: construction time $\mathcal{O}(n)$, query time $\mathcal{O}(1)$, space $\mathcal{O}(\log_2 k)$ bits per key (Müller et al, 2014).

x	$fp(x)$	$f(x)$	$part(x)$
Inception	I	👤	3
Life of Brian	L	👤	3
Avatar	A	👤	2
Titanic	T	👤	3
The Matrix	T	👤	1
Gladiator	G	👤	1
Forrest Gump	F	👤	1
Interstellar	I	👤	2
The Godfather	T	👤	2
Jurassic Park	J	👤	3
The Lion King	T	👤	1
Frozen	F	👤	3

Data Structure:

bucket 1	F	G	T	-	👤	👤	👤	-
bucket 2	A	I	-	-	👤	👤	-	-
bucket 3	I	J	L	T	👤	👤	👤	👤

Algorithm $eval(x)$:

```

i ← part(x)
if dicti.contains(fp(x)) then
    return dicti[fp(x)]
else
    return fallback.eval(x)
    
```

1. The Retrieval Problem

- Definition
- Motivation

2. Retrieval based on Fingerprinting

3. Cuckoo-Style Retrieval

- Using Peeling
- In General Using Linear Algebra
- Teaser: Ribbon Retrieval

4. Summary

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

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^Δ
- $h_1, h_2, h_3 \sim \mathcal{U}([m]^D)$ //SUHA
- $\text{eval}_R(x) := (A[h_1(x)] + A[h_2(x)] + A[h_3(x)]) \bmod k$

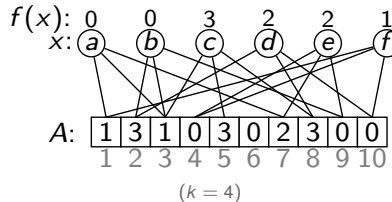
Performance

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

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.

- can forget about x_i “for now” and focus on the rest
- if configuration is peelable, this takes care of all keys



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 $\mathbb{F}_2 = \{0, 1\}$

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

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

$$\begin{array}{l}
 r = 3 \\
 m = 7 \\
 n = 5
 \end{array}
 \left(\begin{array}{c}
 \text{---} h(x_1) \text{---} \\
 \text{---} h(x_2) \text{---} \\
 \text{---} h(x_3) \text{---} \\
 \text{---} h(x_4) \text{---} \\
 \text{---} h(x_5) \text{---}
 \end{array} \right)
 \begin{pmatrix}
 011 \\
 010 \\
 000 \\
 001 \\
 000 \\
 110 \\
 101
 \end{pmatrix}
 \stackrel{!!}{=}
 \begin{pmatrix}
 f(x_1) \\
 f(x_2) \\
 f(x_3) \\
 f(x_4) \\
 f(x_5)
 \end{pmatrix}$$

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)

The Retrieval Problem
○○○

Retrieval based on Fingerprinting
○○

What more could we hope for?

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

Cuckoo-Style Retrieval
○○●○

Summary
○○

Summary

~~John~~ \mapsto σ
~~Mary~~ \mapsto φ
~~Lisa~~ \mapsto φ
~~Carl~~ \mapsto σ
~~Jane~~ \mapsto φ

Retrieval

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

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

- insert & delete not supported

Dictionary / Hash Table

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

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

- insert & delete usually supported

Constructions discussed here

- fingerprint-based approaches
- cuckoo-style retrieval using peeling
- cuckoo-style retrieval using linear algebra with \mathbb{F}_2

The Retrieval Problem
○○○

Retrieval based on Fingerprinting
○○

Cuckoo-Style Retrieval
○○○○

Summary
●○

Remark: There is more...

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

- Was ist der Funktionsumfang einer Retrieval Datenstruktur?
- Was sind die Vorteile und Nachteile im Vergleich zu einer normalen Hashtabelle?
- Welche Anwendungen für Retrieval Datenstrukturen haben wir kennengelernt?
- Zu Retrieval Datenstruktur mit dem Fingerprint-Ansatz:
 - Wie ist die Datenstruktur aufgebaut? Wie funktioniert der Query Algorithmus?
 - Was ist „Bumping“ und wieso brauchen wir es?
 - Was sind Konstruktions- und Zugriffszeiten sowie Speicherverbrauch?
- Zu Retrieval Datenstruktur basierend auf dem Schälalgorithmus:
 - Wie ist die Datenstruktur aufgebaut? Wie funktioniert der Query Algorithmus?
 - Was sind Konstruktions- und Zugriffszeiten sowie Speicherverbrauch?
- Zu Retrieval Datenstruktur basierend auf linearer Algebra mit dem Körper \mathbb{F}_2 :
 - Was ist das allgemeine Schema? Inwiefern passt auch der Ansatz mit dem Schälalgorithmus in dieses Schema?
 - Welche Ziele sollte ich bei der Wahl der Funktion h im Auge behalten?
- Ribbon Retrieval ist nicht prüfungsrelevant.