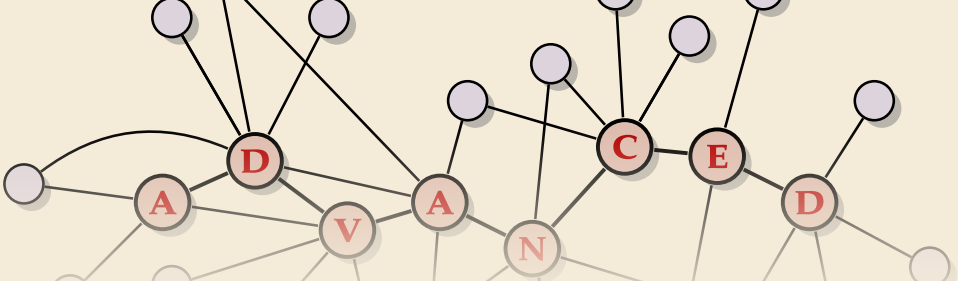


9 Random tricks

24 June 2025

Prof. Dr. Sebastian Wild

CS627 (Summer 2025)
Philipps-Universität Marburg
version 2025-06-25 09:25

Outline

9 Random tricks

Uses of Randomness

- Since it is likely that BPP = P, we focus on the more fine-grained benefits of randomization:
  - simpler algorithms (with same performance)
  - improving performance (but not jumping from exponential to polytime)
  - improved robustness

- Here: Collection of examples illustrating different techniques
  - fingerprinting / hashing
  - exploiting abundance of witnesses
  - random sampling

9.1 Hashing

Fingerprinting / Hashing

- Often have elements from huge universe $U = [0..u)$ of possible values, but only deal with few actual items $x_1, \ldots, x_n$ at one time.
  Think: $n \ll u$
- Fingerprinting can help to be more efficient in this case
  - fingerprints from $[0..m)$
  - $m \ll u$
  - *Hash Function* $h : U \to [0..m)$

$\in \mathcal{U}$

h will have collisions

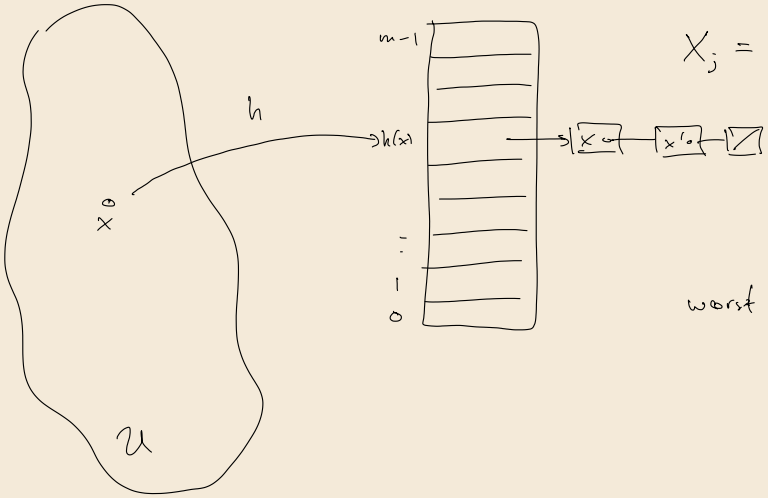
$(x, y \in \mathcal{U} \; ; \; h(x) = h(y))$

Fingerprinting / Hashing

- Often have elements from huge universe $U = [0..u)$ of possible values, but only deal with few actual items $x_1, \ldots, x_n$ at one time.
  Think: $n \ll u$
- Fingerprinting can help to be more efficient in this case
  - fingerprints from $[0..m)$
  - $m \ll u$
  - *Hash Function* $h : U \to [0..m)$
- Classic Example: hash tables and Bloom filters

Hash Tables

indirect chaining



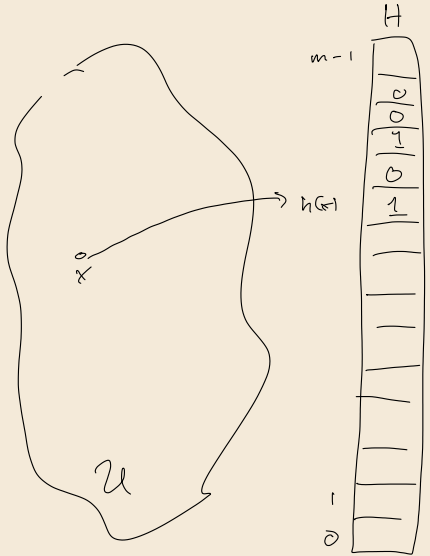
Performance:

How big are buckets?

X_j = # keys x with $h(x) = j$ in our HT

worst case $X_1 = n$

Bloom Filters



insert(x) : $H[h(x)] := 1$

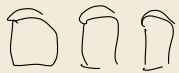
query(x) : $H[h(x)]$

↳ output 1 (Yes) can be a false positive!

output 0 (No) correct

(reduce false positive rate using independent $h_1, h_2, h_3, h_4 \ldots$)

application: segmented database "cheap first checks"



Uniform – Universal – Perfect

Randomness is essential for hashing to make any sense! Three very different

1. *uniform hashing assumption*: (optimistic, often roughly right in practice!)
   How good is hashing if input is "as nicely random" as possible?

Uniform hashing assumption:

All m^n possible hash seq. $h(x_1), h(x_2), h(x_3), \ldots, h(x_n)$ are equally likely.

Uniform – Universal – Perfect

Randomness is essential for hashing to make any sense! Three very different

1. *uniform hashing assumption*: (optimistic, often roughly right in practice!)
   How good is hashing if input is "as nicely random" as possible?

2. Since fixed h is prone to "algorithmic complexity attacks" (worst case inputs)
   $\rightsquigarrow$ *universal hashing*: pick h at random from class H of suitable functions

universal class of hash functions

Uniform – Universal – Perfect

Randomness is essential for hashing to make any sense! Three very different

1. *uniform hashing assumption*: (optimistic, often roughly right in practice!)
   How good is hashing if input is "as nicely random" as possible?

2. Since fixed h is prone to "algorithmic complexity attacks" (worst case inputs)
   $\rightsquigarrow$ *universal hashing*: pick h at random from class H of suitable functions

   universal class of hash functions

3. For given keys, can construct collision-free hash function
   $\rightsquigarrow$ *perfect hashing*