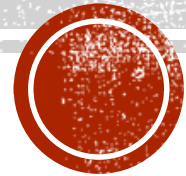


# PRINCIPLES OF DATA ORGANISATION

Membership tester



# MOTIVATION



Hashing

What is not inside a page?



# MEMBERSHIP TESTERS

- ⌘ Answers membership queries for a set: is a query key contained in a set?
- ⌘ Suitability checking **only for existence** and not the content itself
  - ⌘ Weaker than hashing
  - ⌘ Motivation: spell checking – testing whether a word is correct
- ⌘ Each (external memory) bucket can have a membership tester associated
- ⌘ Membership tester should be small concerning the entire set size
  - ⌘ Keep **in primary memory**
- ⌘ Membership tester **CAN** return **false positives** (“the record can be there”), but not many
- ⌘ Membership tester **CAN NOT** return false negatives



# EXACT MEMBERSHIP TESTERS

- ✂ Let the set come from a finite universe  $U$  containing  $u$  elements
- ✂ The tester has to represent  $2^u$  subsets
- ✂ At least  $\log_2 2^u$  bits needed for encoding
  - ✂ Encodes any state = exact membership tester (no false positives)
  - ✂ Too much memory
  
- ✂ If we restrict ourselves to a fixed subset of  $n$  elements we need to represent only all subsets of size  $n \sim n \log u$  for  $n \ll u$
- ✂ Several membership testers of almost that size have been proposed
  - ✂ Carter et al. (1978)
  - ✂ Brodник and Munro (1999)



# APPROXIMATE MEMBERSHIP TESTERS

- Originally proposed by Bloom, 1970 – **Bloom filter**
- Bit string of length  **$b$**
- $k$**  hash functions  **$h_i : U \rightarrow \{1, \dots, b\}$**

*$x \in X$ : set the bits corresponding to  $h_1(x), h_2(x), \dots, h_k(x)$  to 1*



# MOTIVATION EXAMPLE

✂ Hash functions:

- ✂  $k \bmod 6$
- ✂  $(k \operatorname{div} 3) \bmod 6$
- ✂  $(k \gg 2) \bmod 6$

	0	1	2	3	4	5	... bits
✂ Insert 7: 1, 2, 1	0	1	1	0	0	0	
✂✂ Insert 22: 4, 1, 5	0	1	1	0	1	1	
✂✂✂ Insert 34: 4, 5, 2	0	1	1	0	1	1	
✂✂✂✂ Insert 42: 0, 2, 4	1	1	1	0	1	1	

✂ Is 34 there? 4, 5, 2 ... may be (after storing 22 it is not yet)

✂✂ Is 3 there? 3, 2, 0 ... NO



# APPROXIMATE MEMBERSHIP TESTERS

- ❧ Bloom (1970), Carter et al. (1978) and Mullin (1983)
  - ❧ How many bits we should choose
  - ❧ The more hash functions and bits we have, the better
- ❧ In order for  $y$  to belong to  $X$  it has to be true that  $h_1(y) = h_2(y) = \dots = h_k(y) = 1$
- ❧ Can return false positives
- ❧ Under the uniform hashing assumption
  - (  $h_i(x)$  are independent and uniformly distributed )
  - $|X| = n, b = (\log_2 e) kn$
  - The error rate is upper bounded by  $2^{-k}$ 
    - ❧  $k$  = parameter
    - ❧ We specify error rate => formula says how many bits to use
- ❧ Deletion not available since that might modify other inserted values
  - ❧ We do not know the other values to unset bits used only by me

