
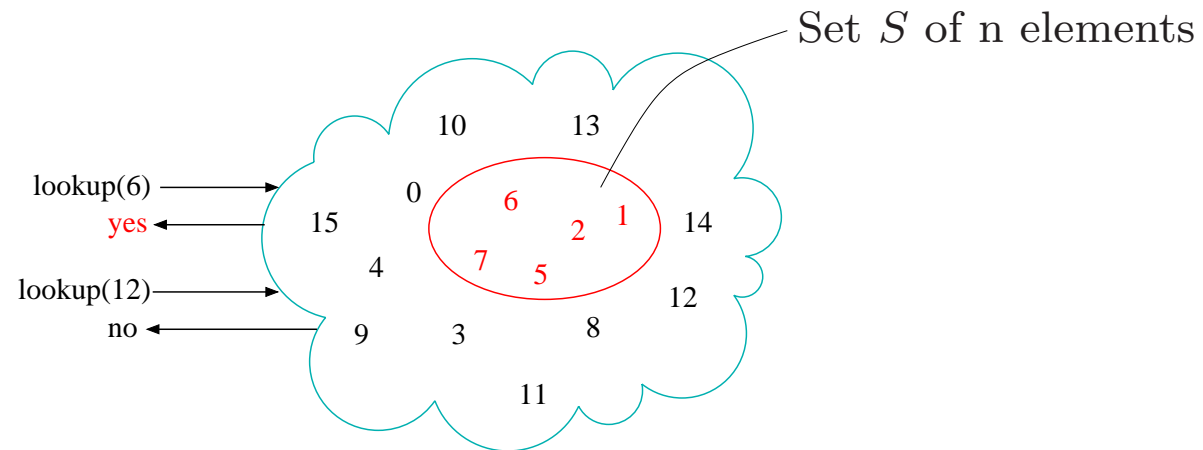


# LOSSY DICTIONARIES

Rasmus Pagh and Flemming Friche Rodler (speaking)  
 BRICS, University of Aarhus, Denmark

ESA, Aarhus, August 30, 2001

# Motivation/Setting



## Interest:

Fast static dictionary and membership schemes using little memory.

## Problem:

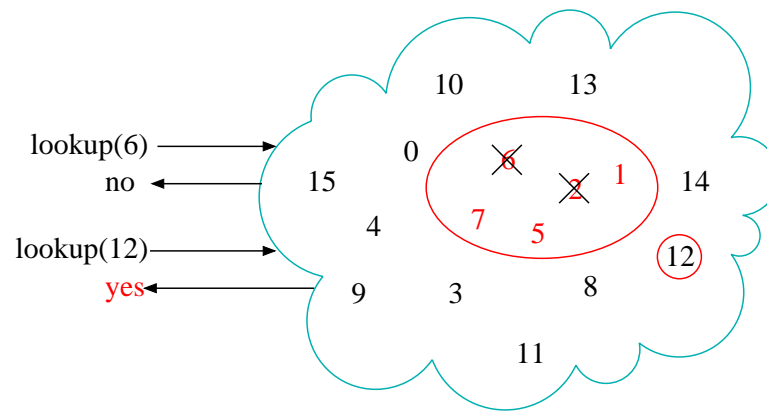
Current schemes using asymptotically optimal space are impractical.

## Model of computation:

- Unit cost RAM with word size  $w$  and  $U = \{0, 1\}^w$

# Concept: Lossy Dictionaries

By storing a slightly different set than intended new possibilities arise.



*Lossy Dictionary*

X: False Negatives ( $\leq \gamma n$ ).

O: False Positives ( $\leq \epsilon 2^w$ ).

## Previous Work

### Bloom Filtering [Bloom '70]:

- Stores  $S' \supseteq S$ , such that  $|S' \setminus S| \leq \epsilon 2^w$ .
- For  $n \ll 2^w$  about  $n \log_2(1/\epsilon)$  bits are sufficient to store  $S'$ .

### Bits needed for exact membership:

$$B = \log_2 \binom{2^w}{n} \approx n \log_2 \left( \frac{2^w e}{n} \right) .$$

### Note:

- Uses  $O(\log(1/\epsilon))$  memory accesses.
- Does not provide a solution to the dictionary problem.

# Applications

## Caching (static):

- A cache stores a small subset of a large key set.
- A small number of false negatives is often acceptable.

## Web cache sharing (for example Web proxies):

- Each proxy keeps a summary of the other proxies to reduce network traffic.
- To reduce space the summary can be lossy.

## Lossy compression:

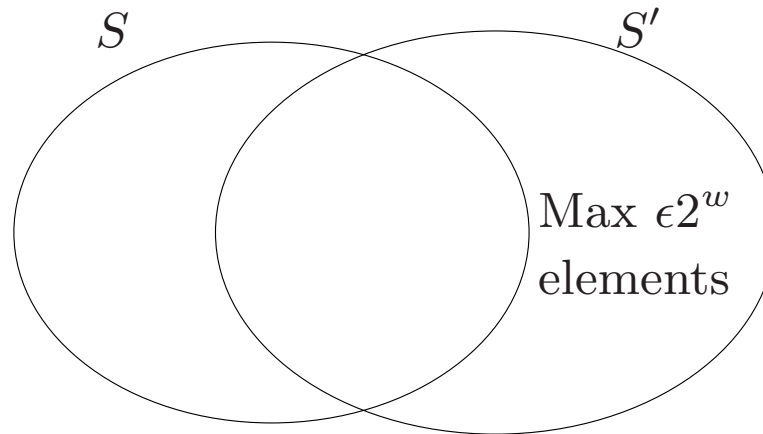
- Store (e.g. wavelet) coefficients with high magnitude.
- Shown very practical [Rodler & Pagh '01].

## Highlights

### In the paper we present:

- A lower bound on the space needed to store a set  $S'$  with  $\epsilon 2^w$  false positives and  $\gamma n$  false negatives.
- A simple static lossy dictionary with the following properties:
  - Very efficient lookups.
  - Near optimal space usage:
    - Theoretical: Within 43% of lower bound –  $O(n)$ .
    - Empirical: Within 20% of lower bound –  $O(n)$ .
  - Can be constructed in time  $O(n \log^* n)$ .

# Lossy Dictionaries



Each key in  $S$  has a weight.

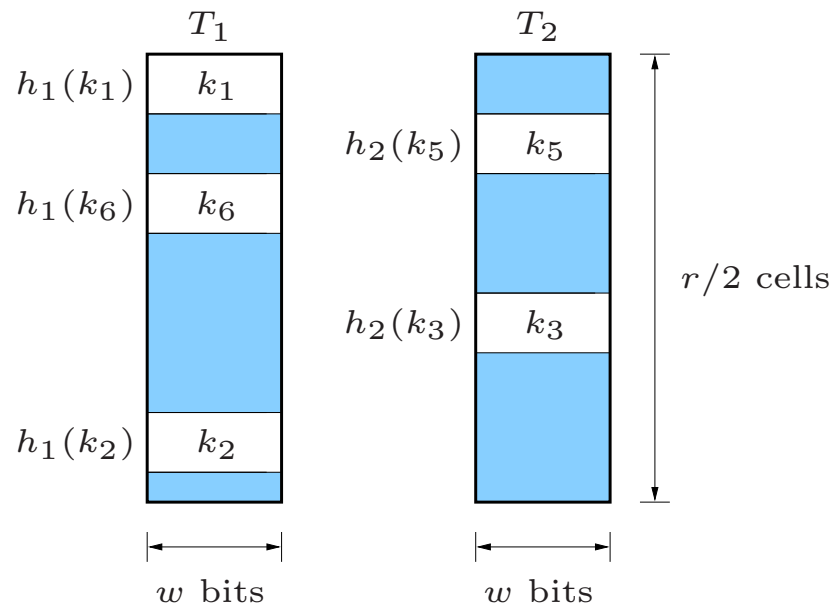
Aim:

Under a given space constraint maximize weight of keys in  $S' \cap S$ .

## Our Data Structure

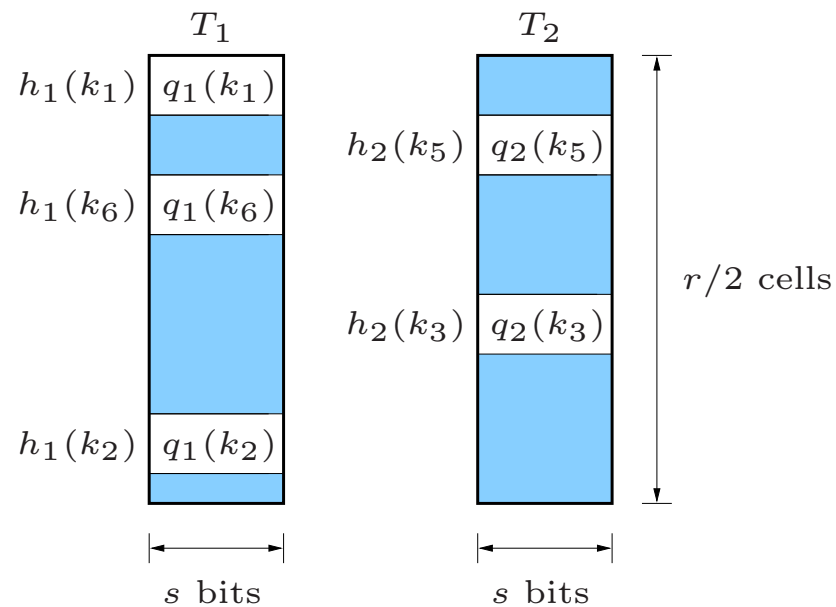
Our starting point is the data structure of [Pagh '01]:

- Use tables  $T_1$ ,  $T_2$  and hash functions  $h_1$ ,  $h_2$ .
- Store key  $k$  in one of  $T_1[h_1(k)]$  and  $T_2[h_2(k)]$  (if possible).



Space:  $r \cdot w$  bits.

# Quotient Functions

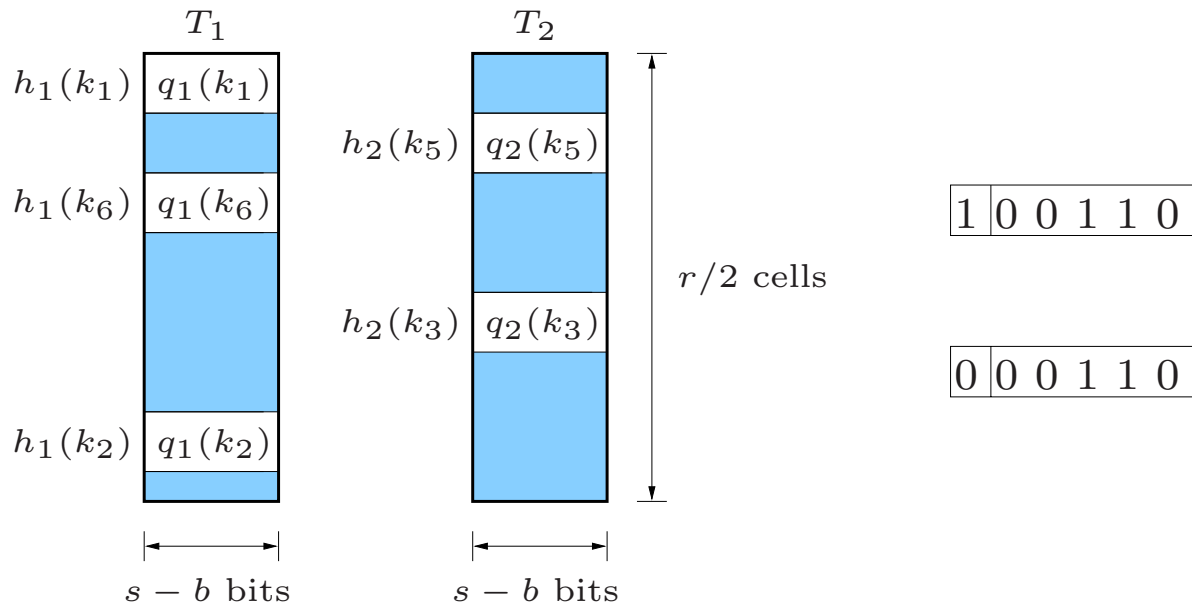


Quotient function  $q(k)$  [Pagh '99]:

- $q : \{0, 1\}^w \rightarrow \{0, 1\}^s$ ,  $k \mapsto (h(k), q(k))$  is one-one.
- $s \approx w - \log_2 r$ .

Space:  $r \cdot s$  bits.

# False Positives



The number of false positives is bounded by:

$$\epsilon 2^w \leq (2^b - 1)r.$$

Space:  $r \cdot (s - b)$  bits.

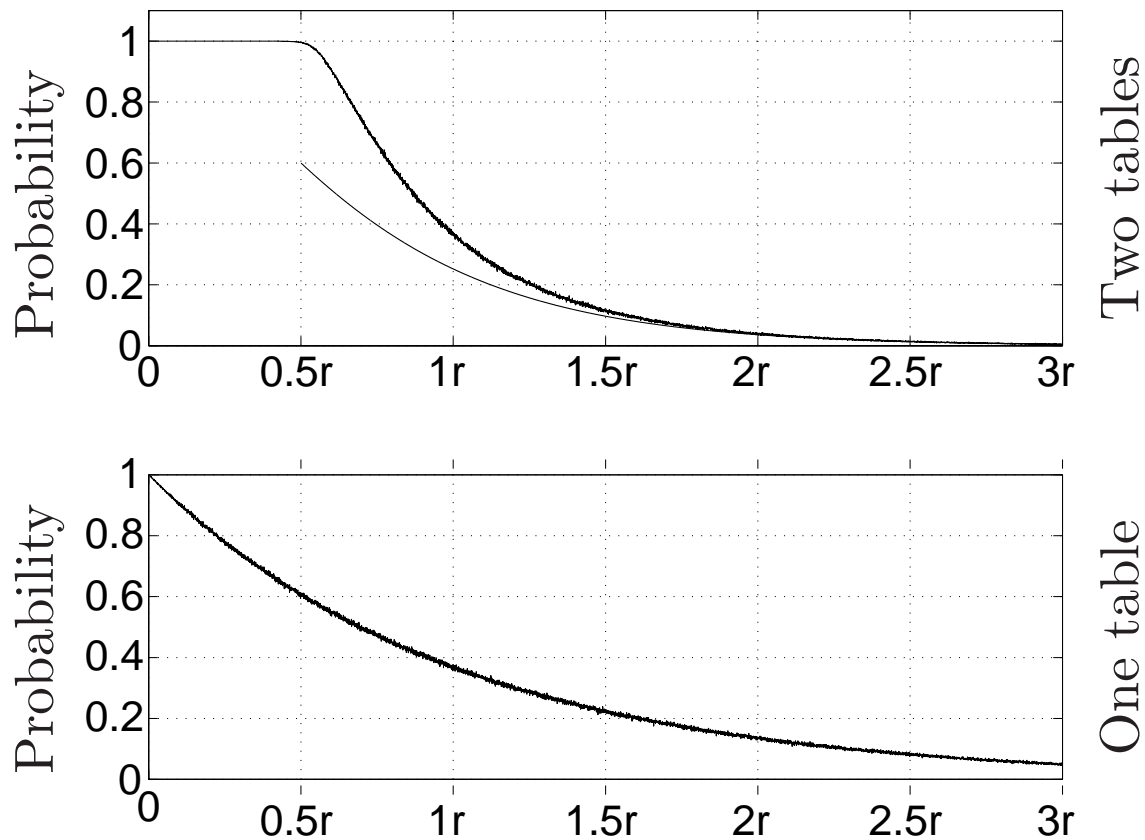
## Construction Algorithm

### Algorithmic facts:

- Greedy – Inserts keys sorted according to weight starting with the largest key.
- Efficiently decides whether to include key  $k_i$  or not, by trying to insert it.
- The algorithm is optimal for all “two table schemes”.
- Running in time:  $O(n \log^* n)$ .

# Experiments

Probability of an element being included in the dictionary.



## Conclusion

We have introduced the concept of lossy dictionaries.

We have presented a static lossy dictionary with:

- Very efficient lookups.
- Near optimal space usage.
- $O(n \log^* n)$  construction time.

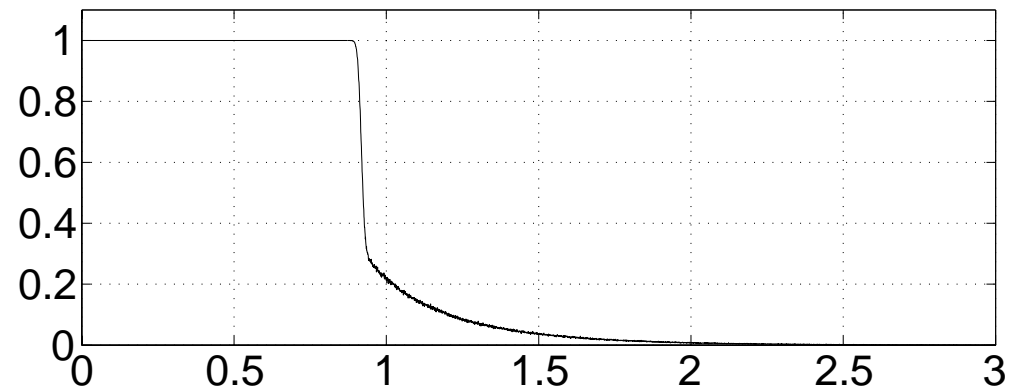
To explore:

- Finding practical hash functions that provably work.
- Improving the theoretical bound for included keys.

## Using More Tables

### Algorithm:

- Variant of CUCKOO HASHING.
- Based on a heuristic.



## Lower Bound

Space needed for:

Lossy dictionary with  $\epsilon 2^w$  false positives and  $\gamma n$  false negatives?

Answer:

$$(1 - \gamma)n \log_2 \left( \frac{1}{\epsilon + n/2^w} \right) - O(n)$$

Proof technique (for  $\gamma = 0$  similar to [Carter et al. '78]):

Based on counting.

Space for  $s = w - \log_2 r + O(1)$  and  $b = \lfloor \epsilon 2^w / r + 1 \rfloor$ :

$$n \log_2 \left( \frac{1}{\epsilon + n/2^w} \right) + O(n)$$