Deterministic Dictionaries

[Hagerup, Miller, Pagli '91]

Goal: build a dictionary for $w$-bit keys in $O(n \log w)$ time on a word-RAM using $O(n)$ space deterministically!

History of deterministic dictionaries

- [Tarjan, Yao '79]: $O(n)$ construction if $w$ is $O(\log n)$
  
  \[ w \text{ is polynomially related to universe: } \{0, \ldots, 2^w - 1\} \]
  
  keep bit in a single word (for $w = 32$) table can be stored as int vector (many entries)

- [Fredman, Komlos, Szemeredi '83]: deterministic FKS needs $O(n^3 w)$

- [Raman '96]: Refinement of FKS idea: $O(n^2 w)$

Remark: we can build $O(n)$ tables in constant query time if

\[ W = O(n^2 \log n) \]

\[ \text{[Fusion Trees]} \]

\[ \text{few entries} \]

\[ \text{[Alon, Naor '96]: } O(n \log \log n) \text{ but queries need } \Theta(\frac{\sqrt{w}}{\log w}) \]
Today \( O(\log n) \) Construction \( \setminus \) deterministic
\( O(1) \) look-up space
\( O(n) \)

Abstract

Construction

1. Universe reduction \( (N = n^{O(1)}) \)
   (Error correcting codes)
2. Further universe reduction \( (N = n^2) \)
   (Trees)
3. Solution for \( N = n^2 \)
   (Displacement method)

1. Universe reduction to \( N = n^{O(1)} \)

We want \( \phi : \mathbb{R}^n \rightarrow \mathbb{R}^n \), such that
- \( t = O(\log n) \)
- \( \phi \) is \( \ell \)-1 on \( S \)

let \( \psi : \mathbb{R}^n \rightarrow \mathbb{R}^n \) be an error correcting code with relative minimum distance \( d > 0 \).
This means: \( H \times \gamma(x+y): H \gamma \neq 0 \)
\( \gamma(x) = \gamma(y) \) differ at at least
\( 45w \) bits

Remark: \( \delta \leq \frac{1}{2} \) can be picked for \( n \geq 2 \)

Lemma: There exist a set \( D \subseteq [4w] \) for \( \delta(\epsilon) \)
of distinguishable bit positions (i.e. \( \gamma(x) \neq \gamma(y) \), \( x+y \) differ at one of the bits of \( D \)) with \( |D| \leq 2 \log n \)
\( \frac{\log n}{\log \frac{1}{\delta}} \)

Proof: We construct \( D \) incrementally:
\( D_0 \subseteq D_1 \subseteq D_2 \ldots \quad D \subseteq D_k \subseteq \{1, \ldots, 4w\} \)

For \( D_j \) we have a natural partition of \( S \) into:
\( \{ \gamma(D, \nu e \in \mathbb{Z}_{100}) \} \)
\( \{ \gamma(CD, \nu) \}, \nu \in \mathbb{Z}_{100}, \nu \) with
\( C(D, \nu) = \{ \nu w \mid w \in S \}, \) the bits of \( w \) at the positions of \( D \) match \( \nu \)
\[ B(D) = \sum_{x \in \{0,1\}^{13}} (C(x)) = \text{Collisions} \]

Example

\[
\begin{array}{ccc}
\text{d}_1 & \text{d}_2 & \text{d}_3 \\
1 & 0 & 1 \\
1 & 1 & 0
\end{array}
\]

\[ D = 3d_1d_2d_3 \]

\[ \chi \]

\[ \gamma \]

\[ \gamma \]

\[ X \subseteq C(d_{1,2,3}, 100) \]

Fact: \( \gamma(x) \) & \( \gamma(y) \) can water on at most \( 240(1-\delta) \) positions

\[ \Rightarrow \]

For at most \( 10 \) \( x \) by \( y \) stays in the same cluster.

\[ \chi \gamma \chi \subseteq C(d_{1,2,3}) \Rightarrow \sum_{d \in D} B\left(D \cup 3d_3\right) \leq 20 \sum_{d \in D} B\left(D \cup 3d_3\right) \leq \frac{1}{2} B(D) (1-\delta) \]

(\text{merging principle})

\[ \exists d : B(D \cup 2d_3) \leq \frac{1}{2} B(D) (1-\delta), \]

Notice: it might be that \( D \cup 3d_3 = D \)

We use \( D_\text{in} = D \cup 3d_3 \)
Since \( D_0 = \emptyset \) \( \Rightarrow \) \( B(D_0) = \binom{u}{2} \leq u^2 \)
we reach \( B(D_k) < 1 \) for
\[ k = \frac{1}{2} \log \frac{u}{8u} \]
\[ \Rightarrow 101 < 0(u^2) \circ (\log u) \]

Lemma: At bit position \( \frac{1}{2} \):
\[ B(D \cup 3d) \leq (1-\delta)^{\frac{1}{2}} B(D) \]

Can be found in \( \Theta(u) \) time & space
deterministically

Without proof (Bit Tricks)

2. Universe reduction to \( n = \lceil n^2 \rceil \)

Split \( x \in U \) into log \( n \) chunks

\[ x \]

\[ \text{log log log log} \]

\[ \text{\# of log blocks} = n = \]

\( \Theta(1) \) many chunks
Store \( x \in S \) in a tree with \(|S| = n \)

- Fan-out \( \leq n \)
- \( n \) nodes: \( n \)

5. Every node gets an ID

To find a node we build a table that can answer queries of the form 
\( (ID, a) \rightarrow \) which node can I reach

from node ID following \( x \in S \)

\( O(n^2) \) tuples to store in hash-table

We also store in every node of the tree if it is terminal and \( x \in S \) and \( x \notin S \)

\( \Rightarrow \) We can find \( x \in S \) \( \notin S \) using the queries

\( \Rightarrow \) Search time is \( O(1) \) because the height of the tree is \( O(1) \)
3. Solution for a quadratic universe

We search a perfect hash function

\[ \tilde{h} : [u^2] \rightarrow [2^r] \], \quad r = \log n + O(1) \]

we first obtain a function

\[ \hat{h} : [u^2] \rightarrow [2^r]^2 \]

\[ \hat{h}(x) = (\text{low}(x), \text{high}(x)) \]

\( \hat{h}(x) \) is 1-1 on \( S \)

\[ \tilde{h}(x) = (a, b) \]

if every column would contain only 1 entry of \( \hat{h}(x) \) \( 1 \times S \)

We can use the column # as hash value

\[ \text{Idea: Scramble the Table} \]
For every column $i$, pick some $a_i \in \mathbb{Z}_2^{3r}$.

- Compute $\otimes_i a_i$ and add the vector as row $i$ in the new table.

**Formal:**

$(x, y) \in \text{Table} \Rightarrow (y \text{ xor } a_x, 1) \in \text{NewTable}$

- $\gamma = \text{collisions in the columns of Table}$
- $\gamma' = \text{collisions in the columns of NewTable}$

**Lemma:**

There is a set of $3a_3$ or $\gamma'$ such that $\gamma' \leq \min \{ r \cdot \frac{9}{2^{r-3}}, |u_2| \}$

**Proof:**

1. Reorder columns such that

$$1 \leq 2s_1 | 2s_2 | 2s_3 | \ldots \ldots \ldots | 2s_{2+1}$$

$S_i = \text{elements in the } i\text{th column of Table}$

$$S_i = 2 \times | (i | x) \in \cap \{ s \} |$$
Notice: $q = \sum_i 1$ \( \binom{i}{2} \)

(2) Pick $a_1, a_2, a_3, \ldots, a_{2^r}$ in increasing order.

Assume we want to find $a_i$.

- Pick $a_i$ uniformly at random from $\{a_1, a_2, \ldots, a_{2^r}\}$.
- Let $\{w_j\}$ be the collisions in NewTable at column $i$ so far.

New collisions by picking $a_i$:

$$\# \text{new collisions by } a_i = \sum_{\text{yes } i} \text{My xoe } a_i$$

$$E(\sum_{\text{yes } i} \text{My xoe } a_i) = \sum_{\text{yes } i} E(\text{My xoe } a_i)$$

$$= \sum_{\text{yes } i} \sum_{\text{yes } i} 1 \cdot \frac{1}{2^r}$$

$$= \sum_{\text{yes } i} \frac{1}{2^r} \cdot \frac{\sum_{\text{yes } i} 1}{2^r}$$

$$= \frac{1}{2^r} \cdot \frac{\sum_i s_i}{2^r} = \frac{\sum_i s_i}{2^r} \quad \text{(4)}$$
By Markov's inequality:
\[ P \left[ \text{new collisions} \leq 2^{-*(x)} \right] \geq \frac{1}{2} \]

In after 2 tries we found a \( a_i \) vector:
\[ \# \text{new collisions} \leq \left| \sum_{j \in S} \left| S_i \right| \sum_{j \in S} |S_j| \right| = M_i \]

\[ \text{floor, because } \# \text{ is even} \]

\[ \Rightarrow 9_1^i \leq \sum_{j \in S} |S_i| n \cdot 2^{-r+1} \leq n^2 2^{-r+1} \leq n \]

Consequence for \( r = \log n + 4 \)

A. \[ 9_1^i \leq 3 |S_i| n \cdot 2^{-r+1} \leq n^2 2^{-r+1} \leq n \]

B. \[ 9_1^i \leq 3 M_i \]

B.1 |S_i| = 1 \[ \Rightarrow M_i = \left| L^2 \cdot \sum_{j \in S} \left| S_i \right| \right| \leq \left| L^2 \cdot \frac{u}{2^r} \right| = 0 \]

B.2 |S_i| > 1 \[ \Rightarrow \forall j \in S: |S_j| \leq 4 \left( \frac{|S_i|}{2} \right) \]

\[ = \Rightarrow M_i \leq \left| L^2 \cdot \sum_{j \in S} \left| S_j \right|^2 \right| \leq L^2 \frac{\sum_{j \in S} |S_i|^2}{2^r} \]

\[ \Rightarrow 9_1^i \leq \sum_{j \in S} L^2 \cdot \frac{1}{2^{r+3}} \left| S_i \right| \leq n \cdot L^2 \cdot \frac{1}{2^{r+3}} \]
1st Stirling Number

Lemma

\[ q' \leq n \]

2nd Stirling Number

Lemma

\[ q'' \leq \left\lfloor \frac{n}{2^{k+3}} \right\rfloor \cdot n = \left\lfloor \frac{n}{2^{k+3}} \right\rfloor \cdot n = 0 \]

Derandomization of the Selection of \( a_i \)

- **Extension of notation**
  
  \[ \Theta \in \mathcal{Z}, \gamma \in \mathcal{Z} \]

  \[ W_{\gamma} = \bigoplus_{\omega \in \mathcal{Z} \setminus \gamma} W_{\omega} \]

- We want to keep track of all \( W_{\gamma} \) during the search for \( a_i \)

  - Store current \( W_{\gamma} \) values inside a binary tree

  ![Binary Tree Diagram]

  - Depth = 1
Adding new ac value -> update trie

by inserting all yα-Si, w ∈ the trie by computing a\*oy and increase
all w<br>of the associated
root -> leaf path

40(\|Si\|*1) for every ac

⇒ O(R·N) = O(m\cdot\log\cdot4) total

Derandomize by the technique of

**CONDITIONAL EXPECTATIONS**

IDEA: We know that E(q') is "good" (≤*) for some ai: we have to pick

⇒ Either

E(q'/a_i starts with z) is also

good for either z=0 or z=1

⇒ We use the trie to determine

the value of z and determine

a_i bit by bit
\[ E(q') = \sum \sum_{y \in \Sigma, \mathcal{w} \in \mathcal{L}_y, \mathcal{w}_0} \frac{1}{2^n} \mathcal{w}_{\mathcal{w}_0} \mathcal{L}_y \]

\[ E(q'| q_c \text{ starts with } 0) = \sum \sum_{y=1, \mathcal{w} \in \mathcal{L}_y, \mathcal{w}_0} \frac{1}{2^n} \mathcal{w}_{\mathcal{w}_0} \mathcal{L}_y \]

\[ E(q'| q_c \text{ starts with } 1) = \sum \sum_{y=0, \mathcal{w} \in \mathcal{L}_y, \mathcal{w}_0} \frac{1}{2^n} \mathcal{w}_{\mathcal{w}_0} \mathcal{L}_y \]

Check which one is larger and use the corresponding bit in \( q_c \)

\( q \) iterate

\( y \) in the end

\[ E(q'| q_c = 01001\ldots 01) \geq (*) \]

\( \Rightarrow q \) for \( q_c = 01\ldots \) \( \leq (*) \)
Idea: dynamic perfect hashing

[Dietzfelbinger et al 1997]

- After
  - Every \( n' = cn \) insert/delete Opu rebuild

- Delete: mark a node as deleted if is stored

- Insert: lookup node: if marked as deleted, mark
  If not there: add in table
    - If collision, rehash 2nd level table
    - If too many elements in the 2nd level table
    - Rehash every thing

\( \Rightarrow O(n) \) space, \( O(1) \) per Opu

- expectation & amortized