On-the-Fly Array Initialization in Less Space

Torben Hagerup
Universität Augsburg

Joint work with Frank Kammer

Dagstuhl, May 2, 2017
Desired Functionality

**Definition.** A *clearable word array* is a data structure that can be initialized with an integer \(n \geq 1\) and subsequently maintains a sequence \((a_0, \ldots, a_{n-1})\) of words (elements of \([0, \ldots, 2^w - 1]\)), initially \((0, 0, \ldots, 0)\), under the operations

- \(\text{read}(\ell)\) (\(\ell \in \{0, \ldots, n - 1\}\)): Returns \(a_\ell\)
- \(\text{write}(\ell, x)\) (\(\ell \in \{0, \ldots, n - 1\}\) and \(x \in \{0, \ldots, 2^w - 1\}\)): Replaces \(a_\ell\) by \(x\)
Desired Functionality

**Definition.** A *clearable word array* is a data structure that can be initialized with an integer \( n \geq 1 \) and subsequently maintains a sequence \((a_0, \ldots, a_{n-1})\) of words (elements of \(\{0, \ldots, 2^w - 1\}\)), initially \((0, 0, \ldots, 0)\), under the operations

\[
\text{read}(\ell) \quad (\ell \in \{0, \ldots, n-1\}) \text{: Returns } a_\ell
\]

\[
\text{write}(\ell, x) \quad (\ell \in \{0, \ldots, n-1\} \text{ and } x \in \{0, \ldots, 2^w - 1\}) \text{: Replaces } a_\ell \text{ by } x
\]

Redundancy: \(\#\text{ bits used} - nw\)
Folklore Method

0

0 1 2 3 4 5 6 7 8 9 10 11 12 13
Folklore Method

Redundancy: $2^{\lceil \log_2 n \rceil}$
Folklore Method

Access time: $O(1)$

Redundancy: $(n + 1) \lceil \log_2 n \rceil$
Folklore Method

Access time: $O(1)$

Redundancy: $(n + 1)\lceil \log_2 n \rceil$
Folklore Method

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>3</th>
<th>12</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

Redundancy: \( n + 1 \) ⌈ log_2 n ⌉
Folklore Method

\[ \text{Access time: } O(1) \]

\[ \text{Redundancy: } (n + 1) \lceil \log_2 n \rceil \]
Folklore Method

Access time: \( O(1) \)
Folklore Method

Access time: \( O(1) \)

Redundancy: \( (2n + 1) \lceil \log_2 n \rceil \)
Trie Method

read: $O(h)$

write: $O\left(\sum_{i=1}^{\lceil d_i/w \rceil}\right)$

Redundancy: # round nodes
Trie Method

\[ \text{read: } O(h) \]
\[ \text{write: } O(\sum_{i=1}^{d_i} \lceil \frac{d_i}{w} \rceil) \]

Redundancy: \# round nodes
Trie Method

\[
\begin{align*}
\text{read: } & \quad O(h) \\
\text{write: } & \quad O\left(\sum_{i=1}^{h} \left\lceil \frac{d_i}{w} \right\rceil \right)
\end{align*}
\]
Trie Method

read: $O(h)$
write: $O(\sum_{i=1}^{d} \left\lceil \frac{d_i}{w} \right\rceil)$

Redundancy: # round nodes
Trie Method

**Read:** \( O(h) \)

**Write:** \( O(\sum_{i=1}^{h} \lceil d_i/w \rceil) \)

**Redundancy:** \# round nodes
Navarro’s Method

\[ O(n + o(n)) \]

Folklore method

\[ \text{w} \]

\[ \text{w} \]
Navarro’s Method

Access time: \( O(1) \)
Redundancy: \( n + o(n) \)
New Result

Tradeoff from minimal time to minimal space
New Result

Tradeoff from minimal time to minimal space

Access time: $O(t)$

Redundancy: $\left\lceil n \left( \frac{t}{2w} \right)^t \right\rceil$
New Result

Tradeoff from minimal time to minimal space

Access time: $O(t)$

Redundancy: $n \left( \frac{t}{2w} \right)^t$

$t = O(1)$: Redundancy $\frac{n}{w^t}$
New Result

Tradeoff from minimal time to minimal space

Access time: \( O(t) \)

Redundancy:
\[
\left\lceil n \left( \frac{t}{2w} \right)^t \right\rceil
\]

\( t = O(1) \):

Redundancy \( n/w^t \)

(best previous: \( n + o(n) \))
New Result

Tradeoff from minimal time to minimal space

Access time: \( O(t) \)

Redundancy: \( \left\lceil n \left( \frac{t}{2w} \right)^t \right\rceil \)

- \( t = O(1) \): Redundancy \( n/w^t \)
  (best previous: \( n + o(n) \))

- \( t = \lceil \log_2 n \rceil \): Redundancy \( 1 \)
Redundancy 1 is Optimal

unless the initialization write to \( n \) words
Redundancy 1 is Optimal

unless the initialization write to $n$ words

**Proof:** Assume that a clearable word array of universe size $n$ is implemented in $N$ bits
Redundancy 1 is Optimal

unless the initialization write to $n$ words

**Proof:** Assume that a clearable word array of universe size $n$ is implemented in $N$ bits

- # states of $(a_0, \ldots, a_{n-1})$: $2^{nw}$
- # states of its representation: $2^N$
Redundancy 1 is Optimal

unless the initialization write to \( n \) words

**Proof:** Assume that a clearable word array of universe size \( n \) is implemented in \( N \) bits

\[
\begin{align*}
\# \text{ states of } (a_0, \ldots, a_{n-1}) & : \quad 2^{nw} \\
\# \text{ states of its representation} \quad & : \quad 2^N
\end{align*}
\]

\[\Rightarrow \text{ We must have } N \geq nw\]
Redundancy 1 is Optimal

unless the initialization write to \( n \) words

**Proof:** Assume that a clearable word array of universe size \( n \) is implemented in \( N \) bits

\[
\begin{align*}
\# \text{ states of } (a_0, \ldots, a_{n-1}) & : 2^{nw} \\
\# \text{ states of its representation} & : 2^N
\end{align*}
\]

⇒ We must have \( N \geq nw \)

Assume that \( N = nw \). Then every state of \((a_0, \ldots, a_{n-1})\) must be represented by *exactly one* bit pattern
Redundancy 1 is Optimal

unless the initialization write to \( n \) words

Proof: Assume that a clearable word array of universe size \( n \) is implemented in \( N \) bits

\[
\begin{align*}
\# \text{ states of } (a_0, \ldots, a_{n-1}) & : 2^{nw} \\
\# \text{ states of its representation} & : 2^N
\end{align*}
\]

\( \Rightarrow \) We must have \( N \geq nw \)

Assume that \( N = nw \). Then every state of \( (a_0, \ldots, a_{n-1}) \) must be represented by exactly one bit pattern

This is impossible unless the initialization fixes all \( N = nw \) bits
Smaller Array Elements ($b$ Bits)

$$\begin{array}{cccccc}
0 & 1 & 2 & \cdots & \frac{nb}{w} - 1 \\
\end{array}$$

$$\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 & \cdots & n-1 \\
\end{array}$$
Smaller Array Elements ($b$ Bits)

\[
\begin{array}{ccccccc}
0 & 1 & 2 & \cdots & \frac{nb}{w} - 1 \\
\end{array}
\]

\[w\]

\[
\begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & \cdots & n - 1 \\
\end{array}
\]

Reading:
Different Initial Values

Wish: Initialize $a_\ell$ with $f(\ell)$
Different Initial Values

Wish: Initialize $a_\ell$ with $f(\ell)$

Solution: $f(\ell)$ is represented as 0
0 is represented as $f(\ell)$
Everything else is represented as everything else
Different Initial Values

Wish: Initialize $a_\ell$ with $f(\ell)$

Solution: $f(\ell)$ is represented as 0

0 is represented as $f(\ell)$

Everything else is represented as everything else

\texttt{read}(\ell):
\begin{verbatim}
default := f(\ell);
value\_read := internal\_clearable.read(\ell);
if value\_read = 0 then return default;
if value\_read = default then return 0;
return value\_read;
\end{verbatim}
The Work-Horse

Idea: Store information in uninitialized array cells

Which information? How to find an uninitialized cell?
The Work-Horse

Idea: Store information in uninitialized array cells

Which information? How to find an uninitialized cell?
The Work-Horse

Idea: Store information in uninitialized array cells

Which information? How to find an uninitialized cell?
The Work-Horse

Idea: Store information in uninitialized array cells

Which information? How to find an uninitialized cell?
The Work-Horse

Idea: Store information in uninitialized array cells

Which information? How to find an uninitialized cell?
The Work-Horse

Idea: Store information in uninitialized array cells

Which information? How to find an uninitialized cell?
The *read* Operation

\[ \text{read}(\ell): \]

\[ u := r; \text{ (\textit{* start at the root \textit{\star}})} \]

**while** \( u \) is red **do**

\[ \text{if } u \text{ is a top node then (\textit{* switch to a new history \textit{\star}})} \]

\[ h := \text{leftmostleaf}(u); \text{ (\textit{* \( u \)'s historian \textit{\star}})} \]

\[ H := A[h]; \text{ (\textit{* \( u \)'s history \textit{\star}})} \]

\[ u := \text{viachild}(u, \ell); \text{ (\textit{* continue towards \( \ell \) \textit{\star}})} \]

**if** \( u \) is white **then return** \( 0; \text{ (\textit{* the initial value \textit{\star}})} \)

\((\textit{* now } u \textit{ is black \textit{\star}})\)

**if** \( u = r \text{ or } \ell \neq h \text{ then return } A[\ell]; \text{ (\textit{* not a historian \textit{\star}})} \)

\((\textit{* now } \ell \textit{ is a black historian \textit{\star}})\)

**return** \( A[p] \), where \( p \) is the leaf

\[(\text{at the end of the red path that contains } u \text{'s parent;})\]
Global Organization

Redundancy: 3 × 2 bits per tree
Global Organization

Folklore method
Global Organization

Redundancy: $3 \times 2 \text{ bits per tree}$
Fixing the Parameters

Height: $t$ ( execution time)
Degree: $d$
Fixing the Parameters

Height: \( t \) (\( = \) execution time)
Degree: \( d \)
A history:
\( (t \text{ levels}) \times (d \text{ children}) \times (2 \text{ bits per child}) \) (white, red, black)
We want \( 2dt \leq w \) (a history fits in a word)
Choose \( d = w/(2t) \)
Fixing the Parameters

Height: $t$ (= execution time)
Degree: $d$

A history:
- $(t \text{ levels}) \times$
- $(d \text{ children}) \times$
- $(2 \text{ bits per child})$ (white, red, black)

We want $2dt \leq w$ (a history fits in a word)

Choose $d = w/(2t)$

Handwaving: $d = 2w/t$
Fixing the Parameters

Height: \( t \) (\( \equiv \) execution time)
Degree: \( d \)
A history:
\( (t \text{ levels}) \times (d \text{ children}) \times (2 \text{ bits per child}) \) (white, red, black)
We want \( 2dt \leq w \) (a history fits in a word)
Choose \( d = \frac{w}{2t} \)
Handwaving: \( d = \frac{2w}{t} \)
\# trees: \( \left\lceil \frac{n}{dt} \right\rceil = \left\lceil n \left( \frac{t}{2w} \right)^t \right\rceil \)
Fixing the Parameters

Height:  $t$  (≡ execution time)
Degree:  $d$

A history:
  $(t \text{ levels}) \times$
  $(d \text{ children}) \times$
  $(2 \text{ bits per child})$ (white, red, black)

We want $2dt \leq w$ (a history fits in a word)

Choose $d = w/(2t)$

Handwaving:  $d = 2w/t$

# trees:  $\left\lceil \frac{n}{dt} \right\rceil = \left\lceil n \left(\frac{t}{2w}\right)^t \right\rceil$

Serious handwaving:  Redundancy $\left\lceil n \left(\frac{t}{2w}\right)^t \right\rceil$
Open Problems

- Do clearable word arrays have important applications?
Open Problems

- Do clearable word arrays have important applications?
- Is redundancy $\left\lceil n \left( \frac{t}{2w} \right)^t \right\rceil$ optimal for access time $O(t)$?