Indexable Compressed Bitvectors

02-714
Slides by Carl Kingsford

Operations on bit vectors

- \( \text{rank}_1(S, i) := \) the number of 1 bits at or before position \( i \) in \( S \).
- \( \text{select}_1(S, j) := \) the position of the \( j \)th 1 bit in \( S \).
- \( \text{rank}_0(S, i) \) and \( \text{select}_0(S, j) \) are defined analogously.

\[
S[i] = \text{“access bit } i\text{”} = \text{rank}_1(S, i) - \text{rank}_1(S, i - 1)
\]

Note: \( \text{rank}_1(S, \text{select}_1(S, j)) = j \), so rank and select are inverses of each other.

**Goal:** rank and select in \( O(1) \) time while using small space.
RRR

Ramen, Ramen, Rao, FOCS 2002

blocks of size $u$ bits

$w_1 \ w_2 \ w_3 \ w_4 \ w_5 \ ...$  \hspace{1cm} $w_i = \text{number of 1s in block } i$

$s_1 \ s_2 \ s_3 \ s_4 \ s_5 \ ...$  \hspace{1cm} $s_i = \text{space to represent } p_i$

$p_1 \ p_2 \ p_3 \ p_4 \ p_5 \ ...$  \hspace{1cm} $p_i = \text{index into tables of bit patterns}$

$w = 0 \ \text{rank}_1(i)$

$000 \ 000$

$w = 1 \ \text{rank}_1(i)$

$\begin{array}{cc}
001 & 001 \\
010 & 011 \\
100 & 111 \\
\end{array}$

$w = 2 \ \text{rank}_1(i)$

$\begin{array}{cc}
011 & 012 \\
110 & 122 \\
101 & 112 \\
\end{array}$

$w = 3 \ \text{rank}_1(i)$

$\begin{array}{cc}
111 & 123 \\
\end{array}$
Each \( w_i \) is \( \leq u \), so can be represented in \( \lceil \log u \rceil \) bits.

Each \( p_i \) is an index into a table with \( \binom{u}{w_i} \) entries, so can be represented with \( \lceil \log \binom{u}{w_i} \rceil \) bits.

Each \( s_i \) is \( \leq u \), so can be represented in \( \lceil \log u \rceil \) bits.

Tables contain \( 2^u \) entries. The rank vectors in table \( w = k \) are of size \( u \log k \)
Prefix Sum Data Structure

**Thm** (Tarjan & Yao; Pagh; simplified). Let \( z_1, \ldots, z_k \) be integers such that \(|z_i| = n^{O(1)}\) and \(|z_i - z_{i-1}| = O(\log n)\), then the list \( z_1, \ldots, z_k \) can be represented in \( O(k \log \log n) \) bits allowing for constant access.

**Proof.** Use the following representation:

\[
\begin{align*}
|z_1 - z_i| & \quad \text{k / log n integers} \\
\end{align*}
\]

The “key frame” integers take at most \( O\left(\frac{k}{\log n} \log n\right) = O(k) \) bits.

The \( \approx k \) delta integers take total \( O(k \log \log n) \) bits because each takes \( O(\log \log n) \) bits.

\[\implies O(k + k \log \log n) = O(k \log \log n) \text{ bits total.} \]
Thm (Tarjan & Yao; Pagh; simplified). Let $z_1, ..., z_k$ be integers such that $|z| = n^{O(1)}$ and $|z_i - z_{i-1}| = O(\log n)$, then the list $z_1, ..., z_k$ can be represented in $O(k \log \log n)$ bits allowing for constant access.

Condition 1: $f_i \leq n$

Condition 2: $|f_{i+1} - f_i| = O(\log n)$ if $u = O(\log n)$

$$\implies k = \frac{n}{u} = \frac{n}{\log n}$$

$$\implies$$ prefix sums can be represented in $(n / \log n) \log \log n$ bits.
Summary: Prefix Sum Data Structure

**Thm.** The prefix-sum data structure used in RRR takes $O\left(\frac{n}{\log n} \log \log n\right)$ space. It can answer prefix-sum queries in constant time.

**Proof:** To answer a prefixSum($x$) query:

1. find the $z_i$ that is just before index $x$.

2. return $z_i + \text{the } z_x - z_i$ that is stored at position $x$.

Each step takes $O(1)$ time.
(Nearly) Complete RRR Data Structure

blocks of size \( u = O(\log n) \) bits

\[
\begin{array}{cccccccccccccccc}
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\
\end{array}
\]

\( w_1 \ w_2 \ w_3 \ w_4 \ w_5 \ldots \) \( w_i \) = number of 1s in block \( i \)

\( s_1 \ s_2 \ s_3 \ s_4 \ s_5 \ldots \) \( s_i \) = space to represent \( p_i \)

\( p_1 \ p_2 \ p_3 \ p_4 \ p_5 \ldots \) \( p_i \) = index into tables of bit patterns

\( f_1 \ f_2 \ f_3 \ f_4 \ f_5 \ldots \) Prefix sums of \( w_i \) as in previous slides

\( q_1 \ q_2 \ q_3 \ q_4 \ q_5 \ldots \) Prefix sums of \( s_i \) as in previous slides

\[
\begin{array}{c|c}
\text{w = 0} & \text{rank}_1(i) \\
000 & 000 \\
\end{array}
\begin{array}{c|c|c}
\text{w = 1} & \text{rank}_1(i) \\
001 & 001 \\
010 & 011 \\
100 & 111 \\
\end{array}
\begin{array}{c|c|c}
\text{w = 2} & \text{rank}_1(i) \\
011 & 012 \\
110 & 122 \\
101 & 112 \\
\end{array}
\begin{array}{c|c}
\text{w = 3} & \text{rank}_1(i) \\
111 & 123 \\
\end{array}
\]
blocks of size $u = O(\log n)$ bits

$w_1 \ w_2 \ w_3 \ w_4 \ w_5 \ ... \ \ w_i < u \Rightarrow (n/\log n) \log \log n$

$s_1 \ s_2 \ s_3 \ s_4 \ s_5 \ ... \ \ s_i < u \Rightarrow (n/\log n) \log \log n$

$p_1 \ p_2 \ p_3 \ p_4 \ p_5 \ ... \ \ p_i = \text{index into tables of bit patterns}$

$f_1 \ f_2 \ f_3 \ f_4 \ f_5 \ ... \ (n/\log n) \log \log n$

$q_1 \ q_2 \ q_3 \ q_4 \ q_5 \ ... \ (n/\log n) \log \log n$

$w_i < u$ because there are at most $u$ 1s in a block of size $u$.

Let $B(w_i, u) = \left\lceil \log_2 \binom{u}{w_i} \right\rceil = \# \text{ of bits needed to select a subset of } w_i \text{ elements from a universe of } u \text{ elements.}$

$s_i = B(w_i, u) < u \text{ b/c the plain } u\text{-long bit vector could store the subset.}$
Space Usage, 2

$p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad ... \quad p_i = \text{index into tables of bit patterns}$

$$\sum_{i=1}^{s} \left[ \log_2 \left( \frac{u}{w_i} \right) \right] < s + \sum_{i=1}^{s} \log_2 \left( \frac{u}{w_i} \right) \leq s + \log_2 \left( \frac{\sum_{i=1}^{s} u}{\sum_{i=1}^{s} w_i} \right) \leq s + \left\lfloor \log_2 \left( \frac{n}{w} \right) \right\rfloor$$

the floor \hspace{1cm} the sums
Space Usage, 2

\[ p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad \ldots \quad p_i = \text{index into tables of bit patterns} \]

\[
\sum_{i=1}^{s} \left\lfloor \log_2 \left( \frac{u}{w_i} \right) \right\rfloor < s + \sum_{i=1}^{s} \log_2 \left( \frac{u}{w_i} \right) \leq s + \log_2 \left( \frac{\sum_{i=1}^{s} u}{\sum_{i=1}^{s} w_i} \right) \leq s + \left\lceil \log_2 \left( \frac{n}{w} \right) \right\rceil
\]

the floor \[ \sum_{i=1}^{s} \log \left( \frac{u}{w_i} \right) = \log \prod_{i=1}^{s} \left( \frac{u}{w_i} \right) \]

the sums

\[ \boxed{=} \leq \boxed{=} \]

\[ \boxed{=} = \text{# of ways to pick } \sum w_i \text{ objects from this line} \]

\[ \boxed{=} = \text{# of ways to pick } \sum w_i \text{ objects from this line where we take } w_i \text{ from segment } i \]
Space Usage, 3

\[ p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad ... \quad p_i = \text{index into tables of bit patterns} \]

\[
\sum_{i=1}^{s} \left\lceil \log_2 \left( \frac{u}{w_i} \right) \right\rceil < s + \sum_{i=1}^{s} \log_2 \left( \frac{u}{w_i} \right) \leq s + \log_2 \left( \sum_{i=1}^{s} u \right) \leq s + \left\lfloor \log_2 \left( \frac{n}{w} \right) \right\rfloor
\]

- \( s = \frac{m}{u} = \frac{m}{\log m} \)
- minimum space to select set of \( w \) 1 bits out of \( n \).

So the total space for the \( p \) array is: \( B(w, n) + \frac{m}{\log m} \)
### Space Usage, 4: The Tables

<table>
<thead>
<tr>
<th>$w = 0$</th>
<th>$\text{rank}_1(i)$</th>
<th>$w = 1$</th>
<th>$\text{rank}_1(i)$</th>
<th>$w = 2$</th>
<th>$\text{rank}_1(i)$</th>
<th>$w = 3$</th>
<th>$\text{rank}_1(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000</td>
<td>001</td>
<td>001</td>
<td>011</td>
<td>012</td>
<td>111</td>
<td>123</td>
</tr>
<tr>
<td>010</td>
<td>011</td>
<td>110</td>
<td>122</td>
<td>101</td>
<td>112</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The tables are tiny:

The rank vector is $u$-long, and each entry is $O(\log w)$

\[
\sum_{w=0}^{u} \binom{u}{w} u \log w \leq u \sum_{w=0}^{u} \binom{u}{w} \log u
\]

\[
= u \log u \sum_{w=0}^{u} \binom{u}{w}
\]

\[
= u 2^u \log u
\]

\[
= O \left( (\log n)(\log \log n)(\log n) \right)
\]

\[
= O \left( \log^2 n \log \log n \right)
\]
Summary: Space Usage

**Thm.** The RRR data structure takes $O\left(B(w,n) + m \log \log m / \log m\right)$ space.

So: how do we solve $\text{rank}_1(S, i)$?
1. Find the block \( x = \lfloor i/u \rfloor \) that \( i \) is in.

2. \( \text{rank}_1(S, i) = \text{prefixSum}(x) + T[w_x][i - xu] \)

Each step takes constant time, so the entire rank computation takes \( O(1) \) time.
Summary

- Can store bit vector in minimum space
  \( B(w,m) + O(m \log \log m / \log m) \)

- Despite using asymptotically less space than the naive representation, you can answer:
  - rank
  - select
  - access
  queries in constant time