Indexable Compressed Bitvectors

02-714

Slides by Carl Kingsford

Ramen, Ramen, Rao. Succinct Indexable Dictionaries with Applications to Encoding k-ary Trees, Prefix Sums and Multisets, SODA 2002: 233-242

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.

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

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

0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 0

w_1 w_2 w_3 w_4 w_5 …

s_1 s_2 s_3 s_4 s_5 …

p_1 p_2 p_3 p_4 p_5 …

w_i = number of 1s in block i

s_i = space to represent p_i

p_i = index into tables of bit patterns

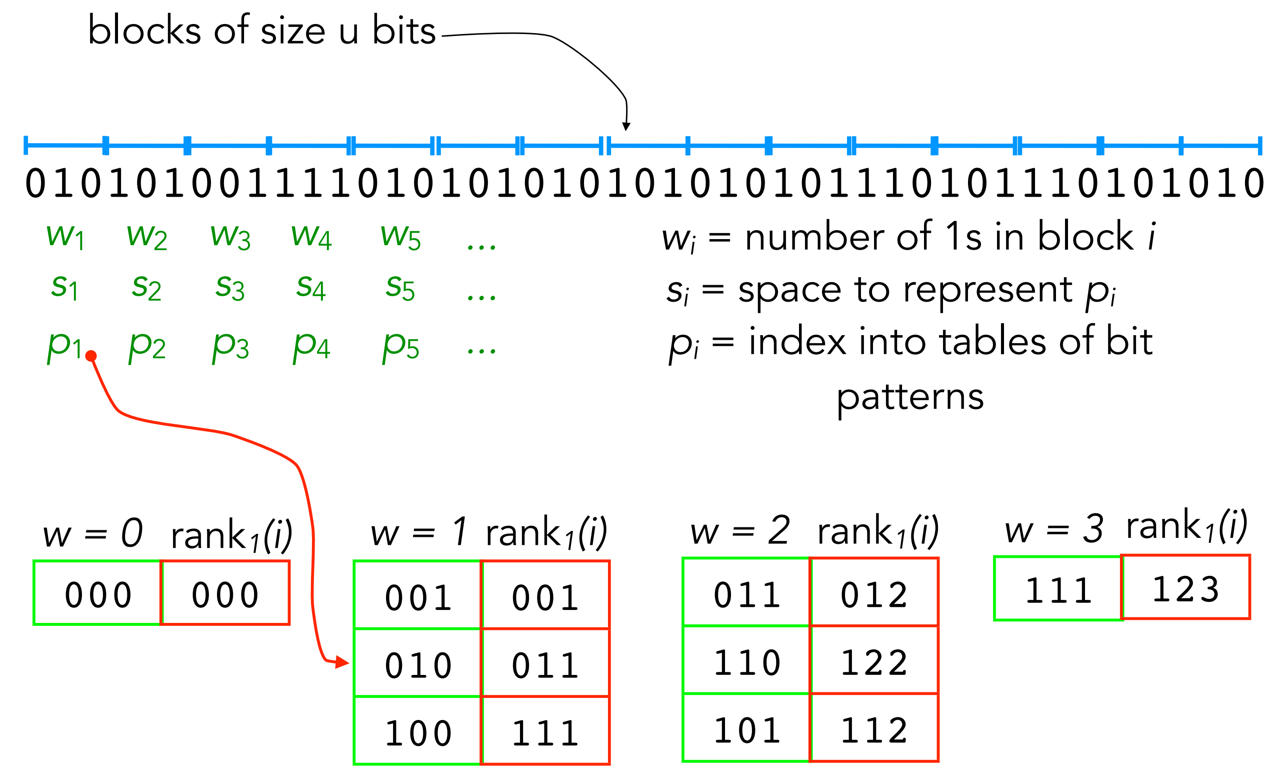
w = 0	rank$_1$(*i*)
000	000

w = 1	rank$_1$(*i*)
001	001
010	011
100	111

w = 2	rank$_1$(*i*)
011	012
110	122
101	112

w = 3	rank$_1$(*i*)
111	123

RRR Space So Far



Each w_i is ≤ 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 $\left\lceil \log \binom{u}{w_i} \right\rceil$ bits.

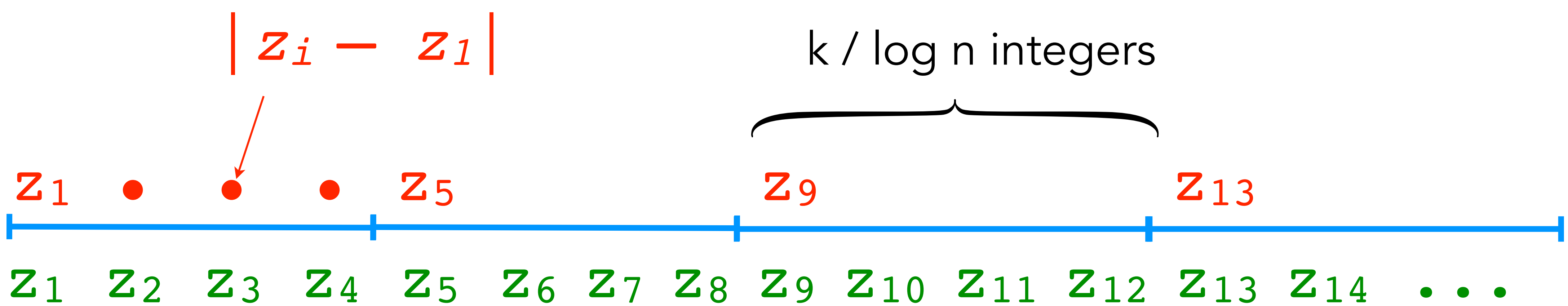
Each s_i is ≤ u, so can be represented in $\lceil \log u \rceil$ bits.

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:



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

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

$\Longrightarrow$ O(k + klog log n) = O(k log log n) bits total. $\square$

Prefix Sum Data Structure, 2

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.

0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 0 1 0

f_1 f_2 f_3 f_4 f_5 …

f_i = number of 1s up through the end of block i

Condition 1: $f_i \leq n$

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

$\Longrightarrow k = n / u = n / \log n$

$\Longrightarrow$ 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 $O((n / \log n) \log \log n)$ 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 + 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

010101001111010101010101010101011101011101010

w_1	w_2	w_3	w_4	w_5	…	w_i = number of 1s in block i
s_1	s_2	s_3	s_4	s_5	…	s_i = space to represent p_i
p_1	p_2	p_3	p_4	p_5	…	p_i = index into tables of bit patterns
f_1	f_2	f_3	f_4	f_5	…	Prefix sums of w_i as in previous slides
q_1	q_2	q_3	q_4	q_5	…	Prefix sums of s_i as in previous slides

$w = 0$	$\text{rank}_1(i)$
000	000

$w = 1$	$\text{rank}_1(i)$
001	001
010	011
100	111

$w = 2$	$\text{rank}_1(i)$
011	012
110	122
101	112

$w = 3$	$\text{rank}_1(i)$
111	123

Space Usage, 1

blocks of size u = O(log n) bits

n / log n blocks

0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 0 1 0

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

$s_i = B(w_i, u) < u$ b/c the plain u-long bit vector could store the subset.

Space Usage, 2

p_1 p_2 p_3 p_4 p_5 ... p_i = index into tables of bit patterns

$$\sum_{i=1}^{s} \left\lceil \log_2 \binom{u}{w_i} \right\rceil < s + \sum_{i=1}^{s} \log_2 \binom{u}{w_i} \le s + \log_2 \binom{\sum_{i=1}^{s} u}{\sum_{i=1}^{s} w_i} \le s + \left\lceil \log_2 \binom{n}{w} \right\rceil$$

the floor

the sums

Space Usage, 2

p_1 p_2 p_3 p_4 p_5 … p_i = index into tables of bit patterns

$$\sum_{i=1}^{s} \left\lceil \log_2 \binom{u}{w_i} \right\rceil < s + \sum_{i=1}^{s} \log_2 \binom{u}{w_i} \le s + \log_2 \binom{\sum_{i=1}^{s} u}{\sum_{i=1}^{s} w_i} \le s + \left\lceil \log_2 \binom{n}{w} \right\rceil$$

the floor

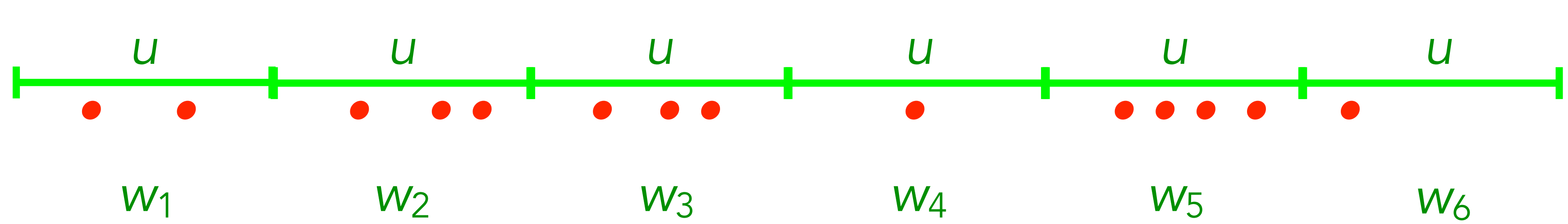
$$\sum_{i=1}^{s} \log \binom{u}{w_i} = \log \prod_{i=1}^{s} \binom{u}{w_i}$$

the sums

$\square \le \square$

$\square$ = # of ways to pick $\sum w_i$ objects from this line

$\square$ = # of ways to pick $\sum w_i$ objects from this line where we take w_i from segment i



Space Usage, 3

p_1 p_2 p_3 p_4 p_5 ...

p_i = index into tables of bit patterns

$$\sum_{i=1}^{s} \left\lceil \log_2 \binom{u}{w_i} \right\rceil < s + \sum_{i=1}^{s} \log_2 \binom{u}{w_i} \leq s + \log_2 \binom{\sum_{i=1}^{s} u}{\sum_{i=1}^{s} w_i} \leq s + \left\lceil \log_2 \binom{n}{w} \right\rceil$$

space to store p array

minimum space to select set of *w* 1 bits out of *n*.

$$s = m/u = m/\log m$$

So the total space for the *p* array is: $B(w,n) + m/\log m$

Space Usage, 4: The Tables

w = 0	rank$_1$(*i*)
000	000

w = 1	rank$_1$(*i*)
001	001
010	011
100	111

w = 2	rank$_1$(*i*)
011	012
110	122
101	112

w = 3	rank$_1$(*i*)
111	123

The tables are tiny:

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

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

$$= u \log u \sum_{w=0}^{u} \binom{u}{w}$$

$$= u 2^u \log u$$

$$= O\left((\log n)(\log \log n)(\log n)\right)$$

$$= O\left(\log^2 n \log \log n\right)$$

for every weight, we this have many entries

Summary: Space Usage

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

So: how do we solve $\text{rank}_1(S, i)$?

$rank_1(S, i)$

1. Find the block $x = \lfloor i/u \rfloor$ that i is in.
2. $rank_1(S, i) = prefixSum(x) + T[w_x][i - xu]$

compute in constant time → prefixSum(x)

The rank table for weight w_x → $T[w_x]$

The appropriate bit in the x^{th} block. → $[i - xu]$

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