Bitvectors and RSA queries

Ben Langmead

Johns Hopkins
Whiting School of Engineering

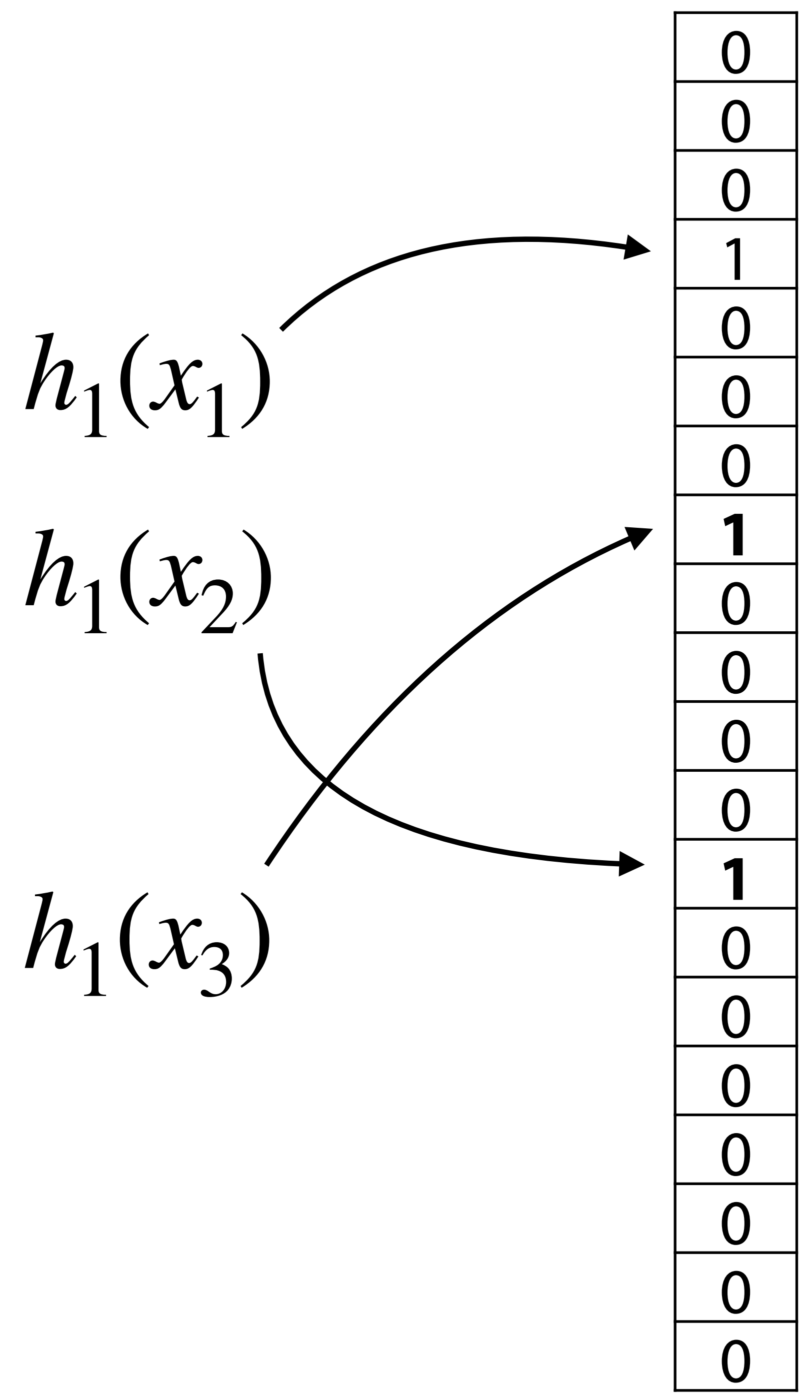
Department of Computer Science

Bitvectors

Bitvectors are no stranger to us; Bloom filters!

Now we consider bitvectors where slots have *meaning*

Navigating between slots also meaningful



Bitvectors

Does this bitvector have a "meaning?"

What if its name was is_prime? 😉

How might we query it?

E.g. next-highest-prime

E.g. designing a 2-universal hash, we want smallest prime (leftmost 1) greater than some number

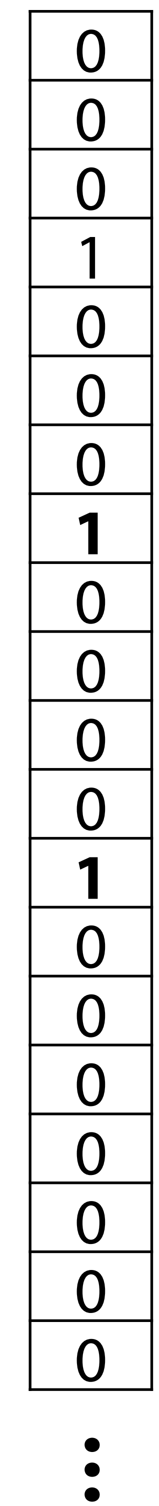
0
1
1
0
1
0
1
0
0
0
1
0
1
⋮

Bitvectors

What if the vector really was a Bloom filter?

Why might want to "navigate" it?

> Say we are counting 1s to estimate cardinality
>
> Might want to "jump" between 1s, ask how spaced out they are (k^{th} minimum value)



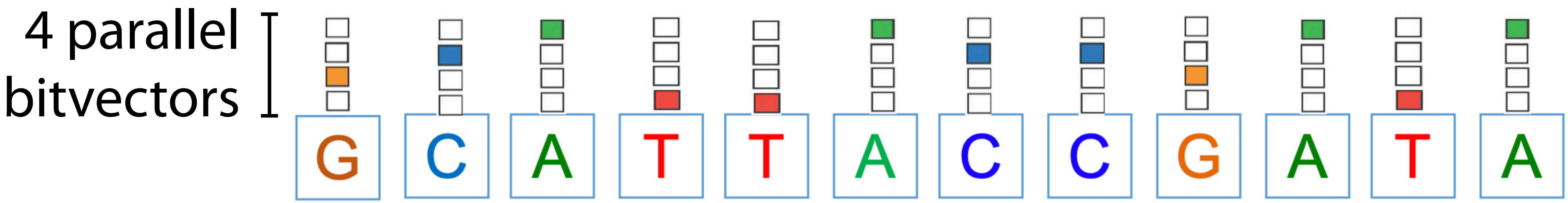
Bitvectors

Might represent words in a document

0	abinoam
1	abiogenesis
0	abiological
0	abiosis
0	abiotic
0	abiotically
0	abiotrophy
0	abirritate
0	abishag
0	abit
0	abitibi
0	abiu
1	abject
⋮	

Bitvectors

Could be a "one-hot" encoding of string



Navigating bitvectors = navigating the occurrences of characters in the string

https://www.encodeproject.org/documents/5284a1d9-7eb6-44d9-8979-b90408332b12/@@download/attachment/Anshul_slides.pdf

Bitvectors

How do we navigate / query bitvectors?

Proposal: "RSA" (Rank, Select, Access)

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

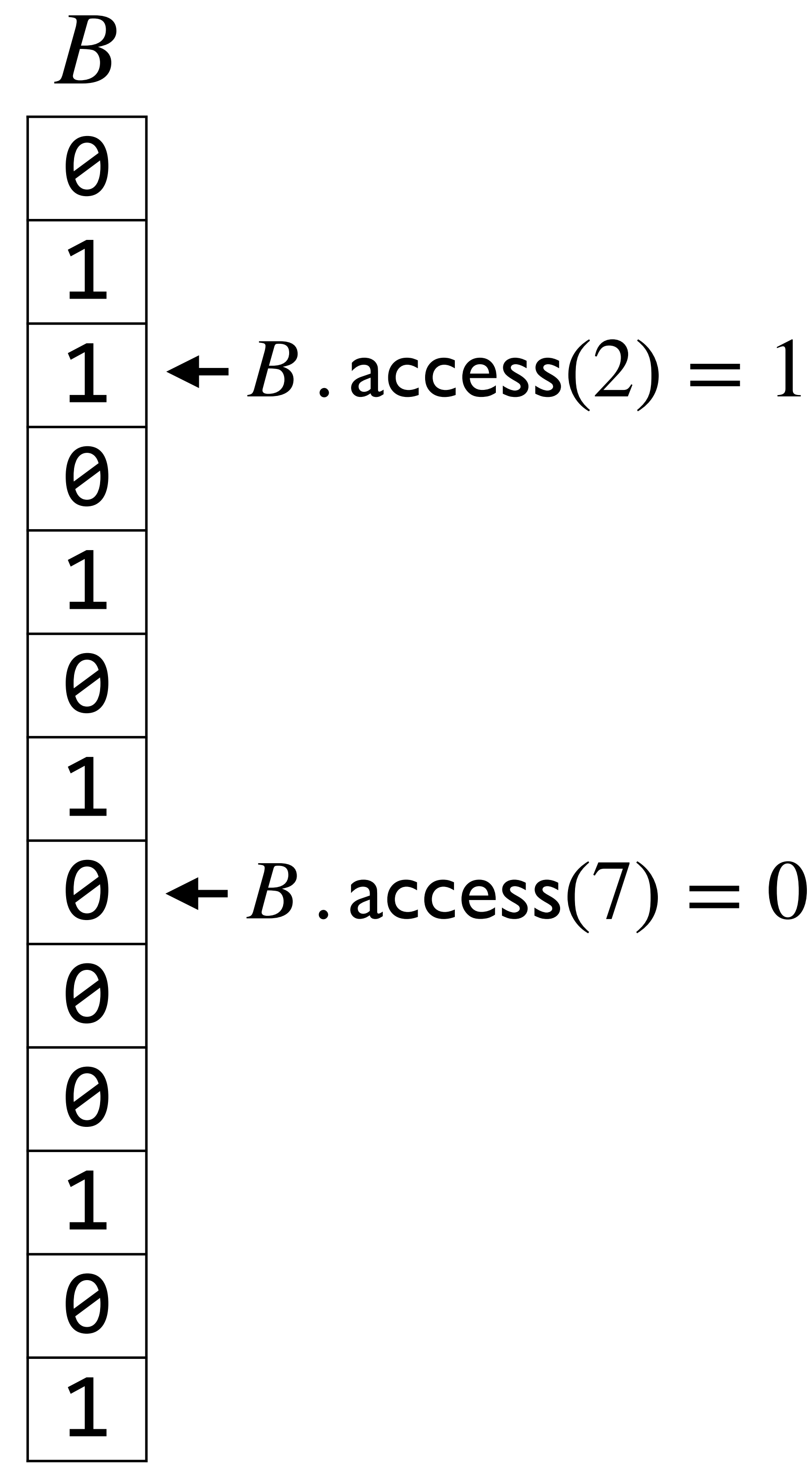
Bitvectors

$B\,.\,\text{access}(i) = B[i]$

Conceptually trivial, but harder if we compress B

(more later)

Indexing starts at 0

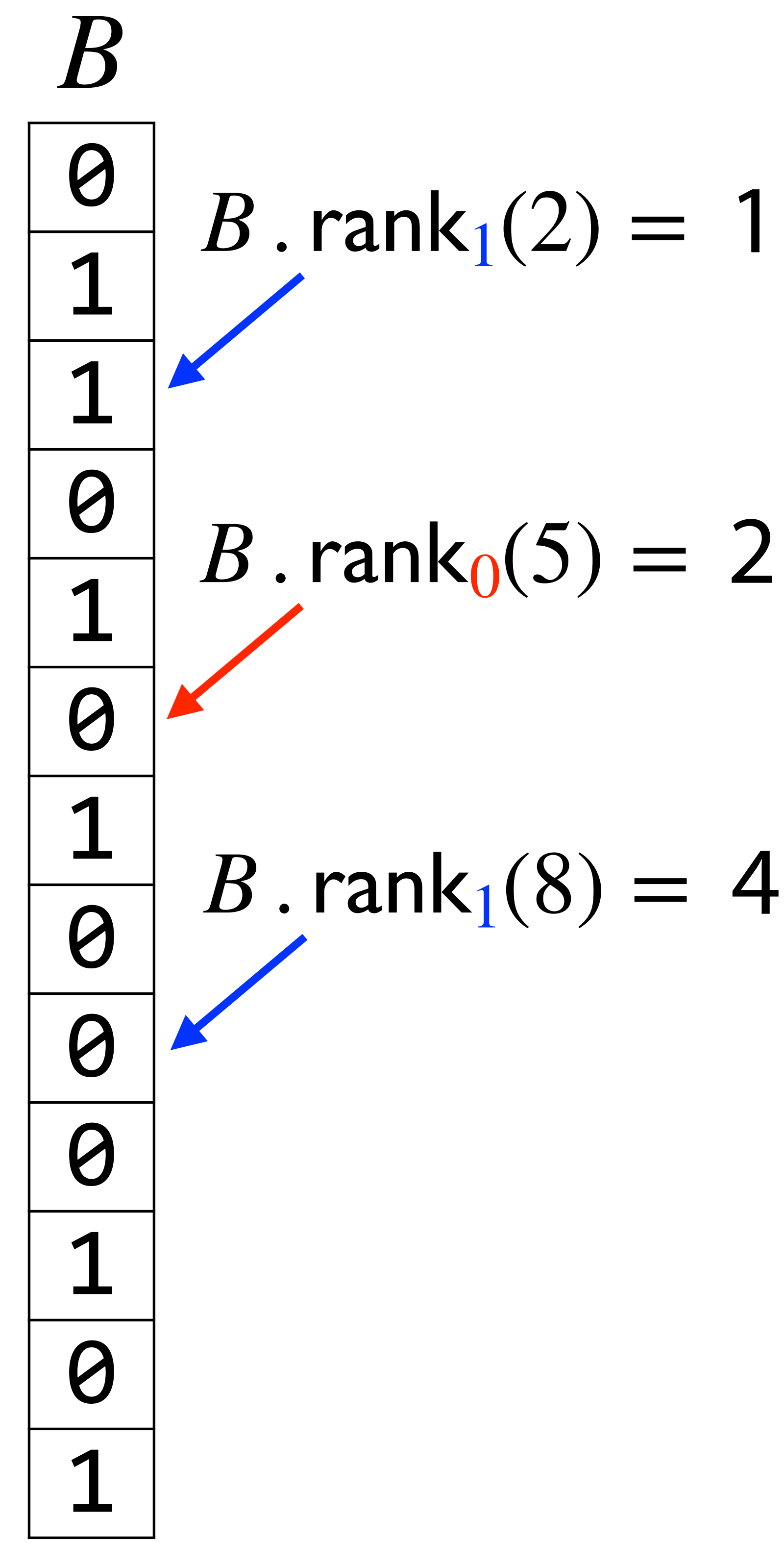


Bitvectors

$$B\,.\,\mathrm{rank}_1(i) = \sum_{j=0}^{i-1} B[j]$$

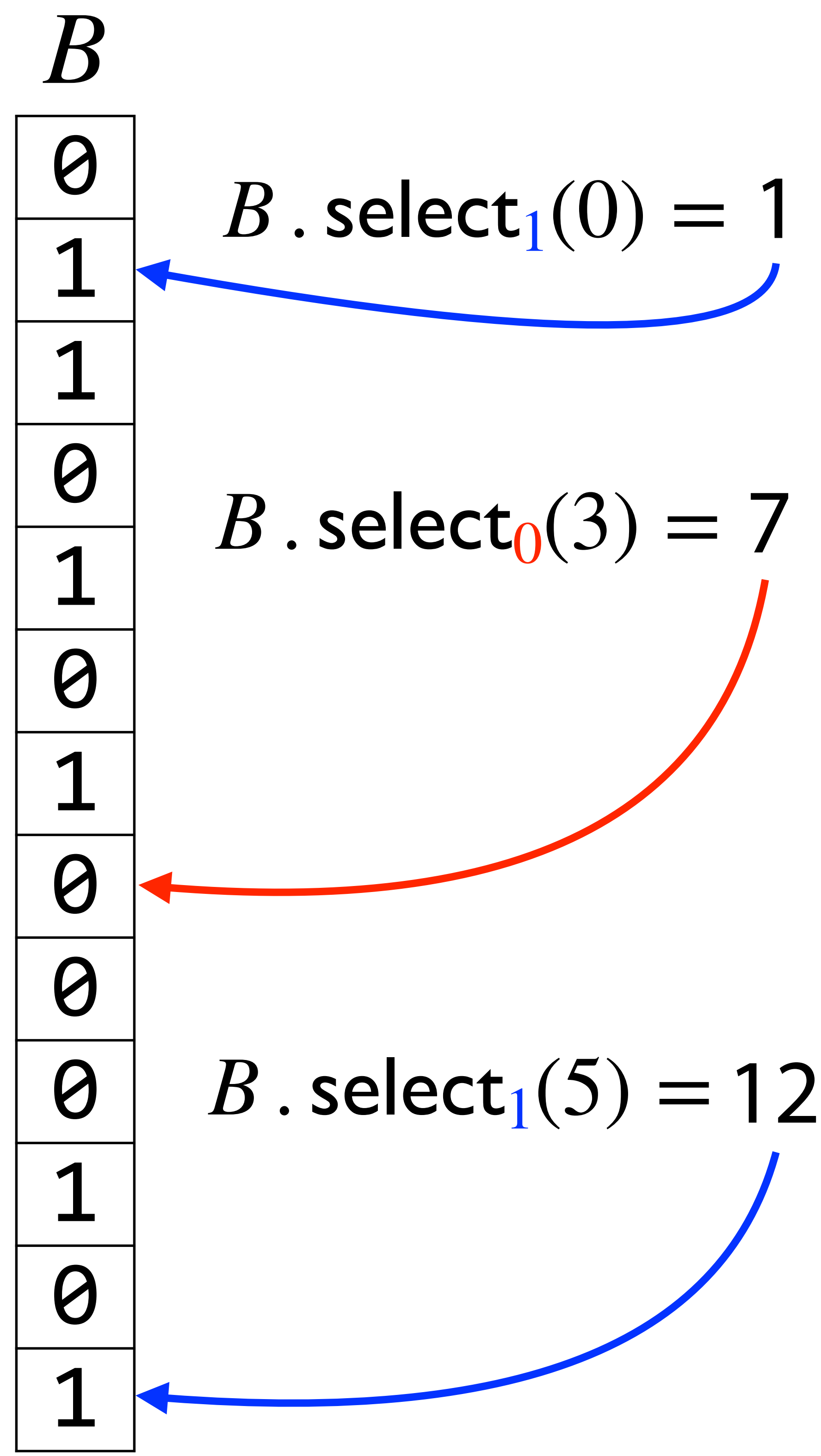
$$B\,.\,\mathrm{rank}_0(i) = i - B\,.\,\mathrm{rank}_1(i)$$

Rank counts up to
but not including offset i



Bitvectors

$$B\,.\,\mathrm{select}_1(i) =$$
$$\max\{\, j \mid B\,.\,\mathrm{rank}_1(j) = i \,\}$$
$$B\,.\,\mathrm{select}_0(i) =$$
$$\max\{\, j \mid B\,.\,\mathrm{rank}_0(j) = i \,\}$$



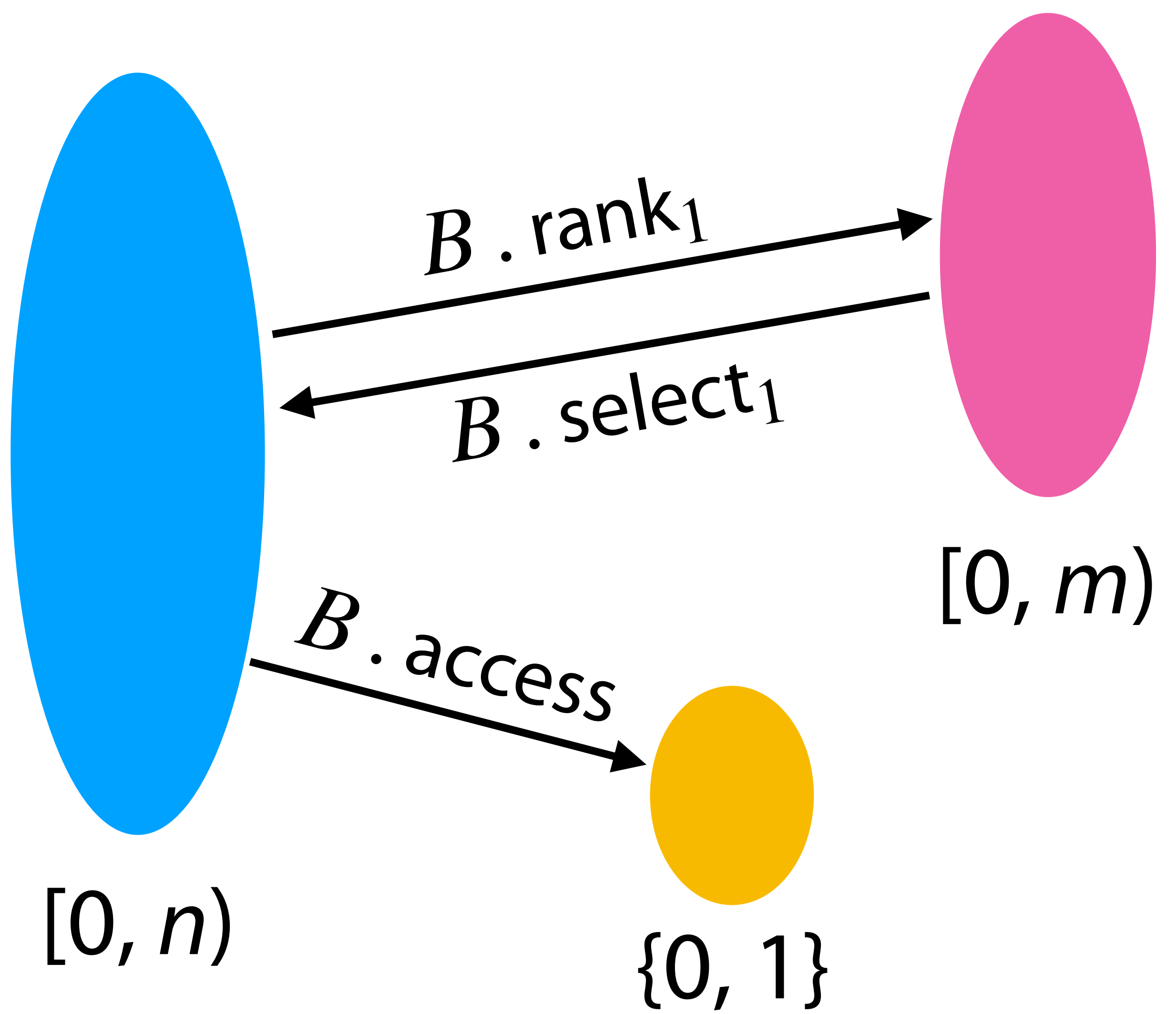
Bitvectors

$B\,.\,\text{access}(\ldots)$

$B\,.\,\text{rank}(\ldots)$

$B\,.\,\text{select}(\ldots)$

Let $|B| = n$ and let m equal the number of set bits



Bitvectors

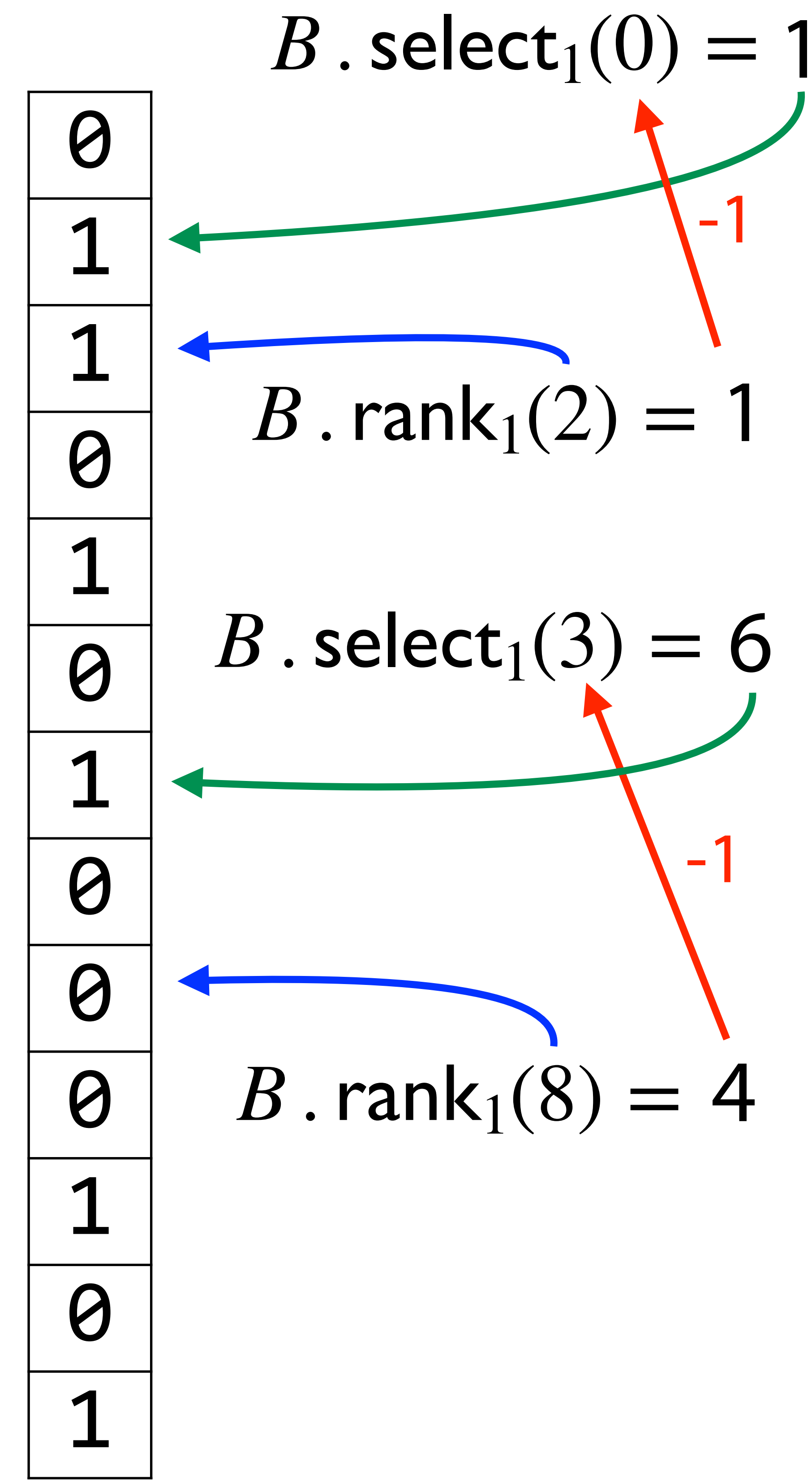
What does this do?

$$B\,.\,\mathrm{select}_1(B\,.\,\mathrm{rank}_1(i) - 1)$$

$$B\,.\,\mathrm{rank}_1(i) = \sum_{j=0}^{i-1} B[j]$$

$$B\,.\,\mathrm{select}_1(i) = \max\{\, j \mid B\,.\,\mathrm{rank}_1(j) = i \,\}$$

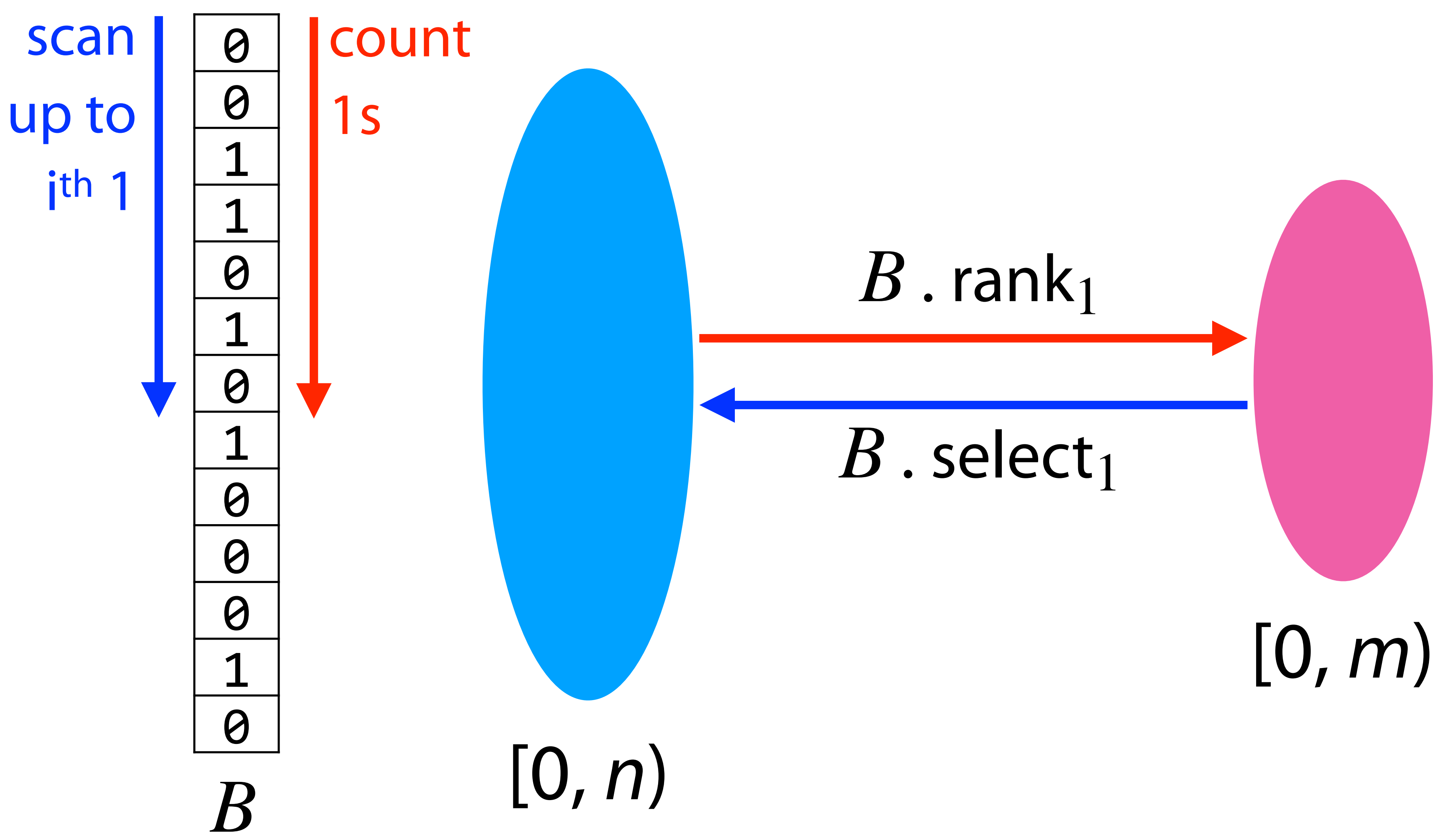
Gives offset of next-earliest set bit -- *predecessor*



Bitvectors

How to implement $B.\text{rank}_1$ & $B.\text{select}_1$?

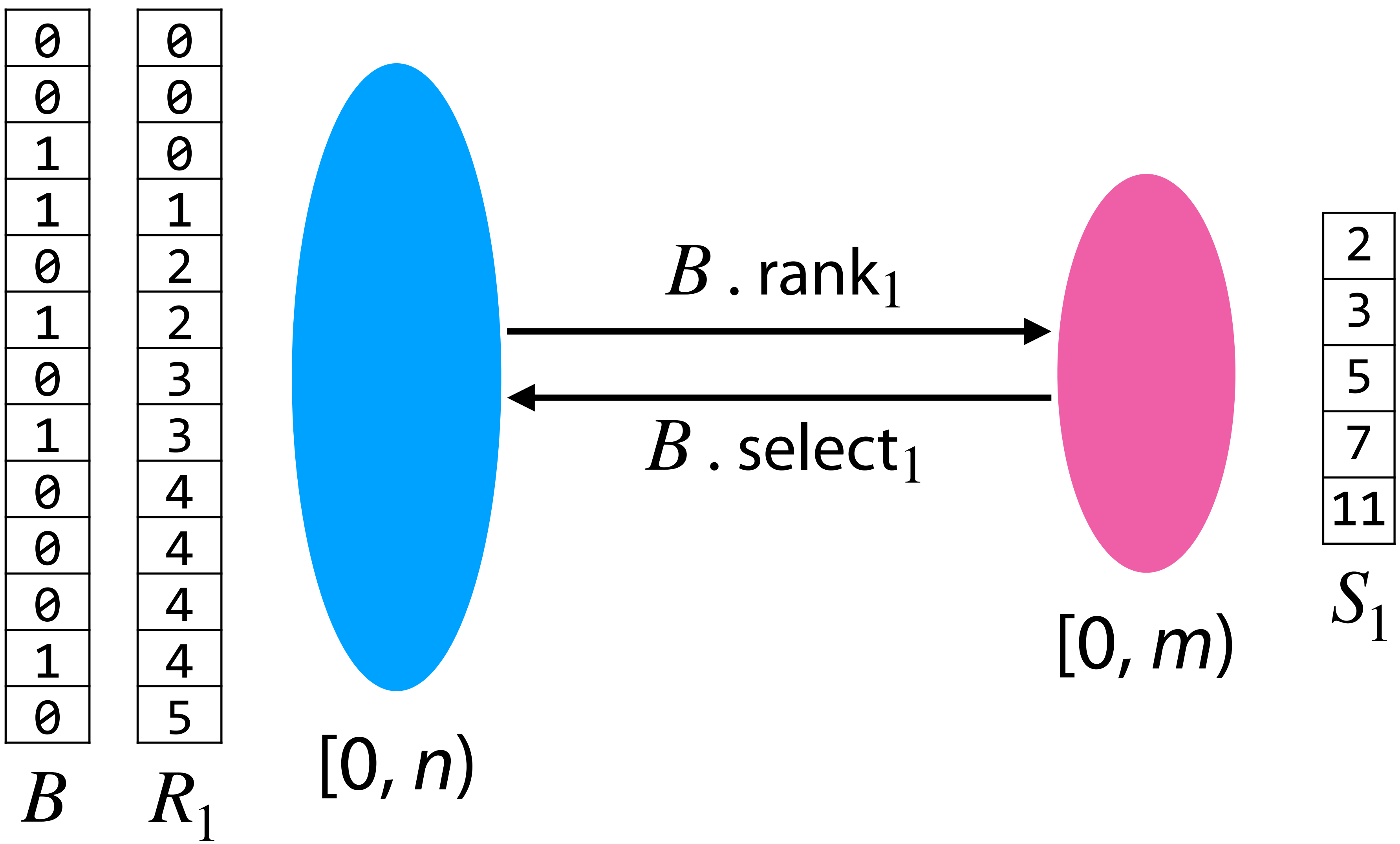
Idea 0: linear scans over B



Can we be more efficient?

Bitvectors

Idea 1: Pre-calculate all answers



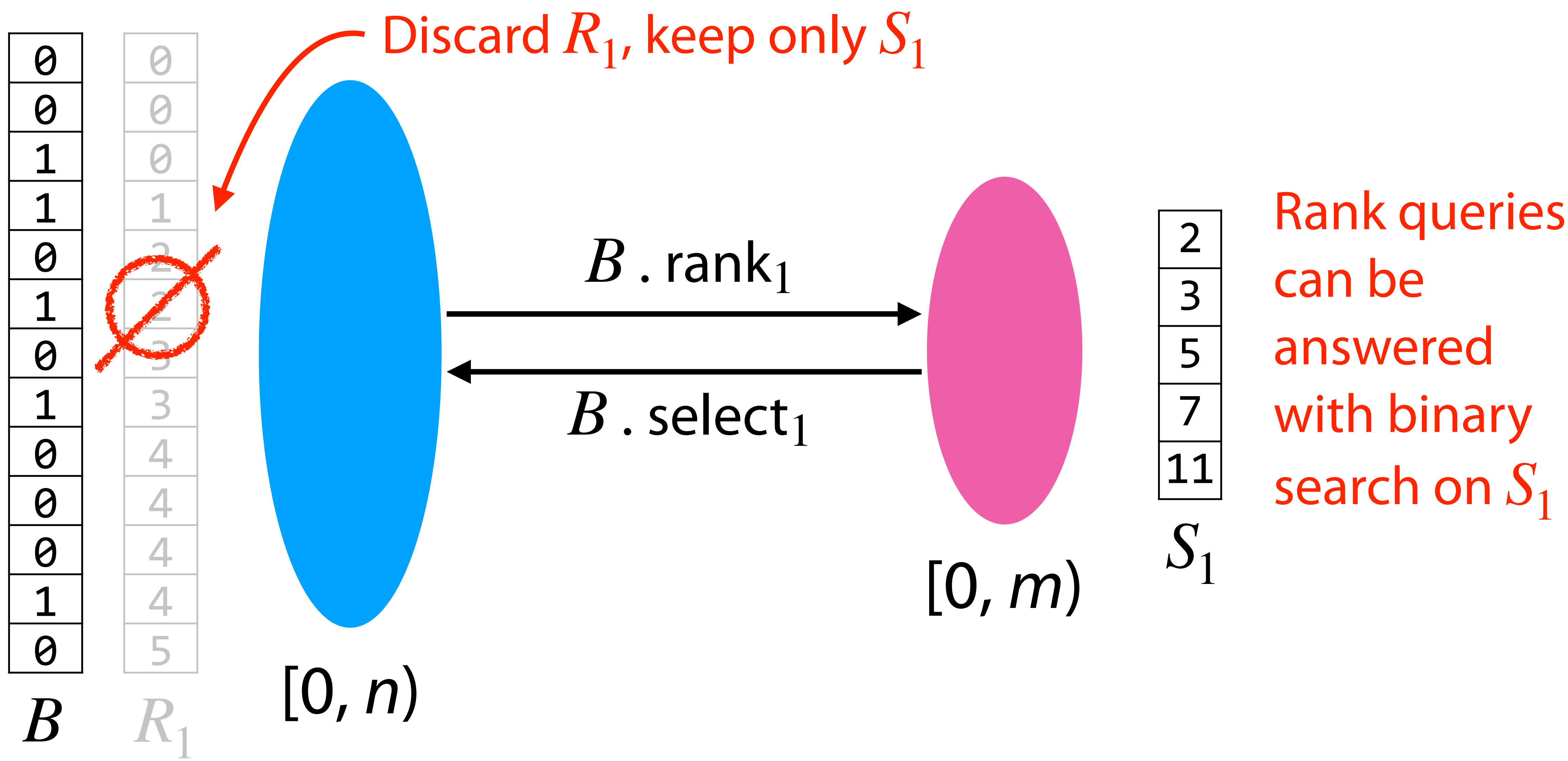
Bitvectors

Idea 1: Pre-calculate all answers

	Time	Space (bits)	Note
B . access	$O(1)$	n	Lookup
B . select_1	$O(1)$	$O(m \log n)$	Pre-calculate S_1
B . rank_1	$O(1)$	$O(n \log m)$	Pre-calculate R_1

Bitvectors

Idea 2: Pre-calculate all answers for B . select$_1$



$O(m \log n)$ bits. B . rank$_1$ is $O(\log m)$ time.

Bitvectors

Idea 2: Pre-calculate all answers for B . select$_1$

	Time	Space (bits)	Note
B . access	$O(1)$	n	Lookup
B . select$_1$	$O(1)$	$O(m \log n)$	Pre-calculate S_1
B . rank$_1$	$O(\log m)$	$O(m \log n)$	Binary search on S_1

Bitvectors

Coming soon:

	Time	Space (bits)	Note
$B\,.\,\text{access}$	$O(1)$	n	Lookup
$B\,.\,\text{select}_1$	$O(1)$	$\check{o}(n)$	?🧚🦄🧚‍♀️?
$B\,.\,\text{rank}_1$	$O(1)$	$\check{o}(n)$	?🧚🦄🧚‍♀️?