Clark's Select

Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

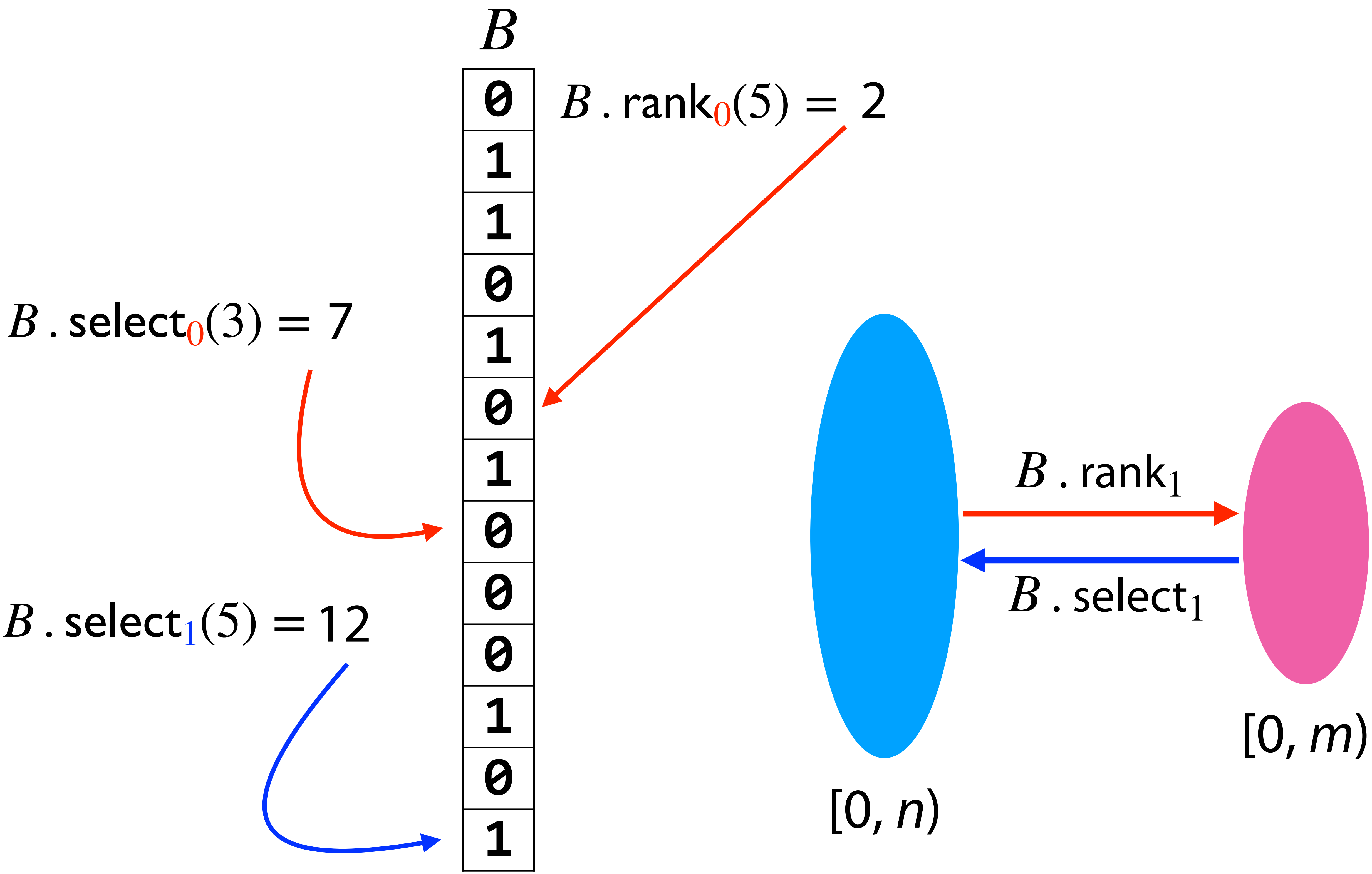
Department of Computer Science

Clark's select

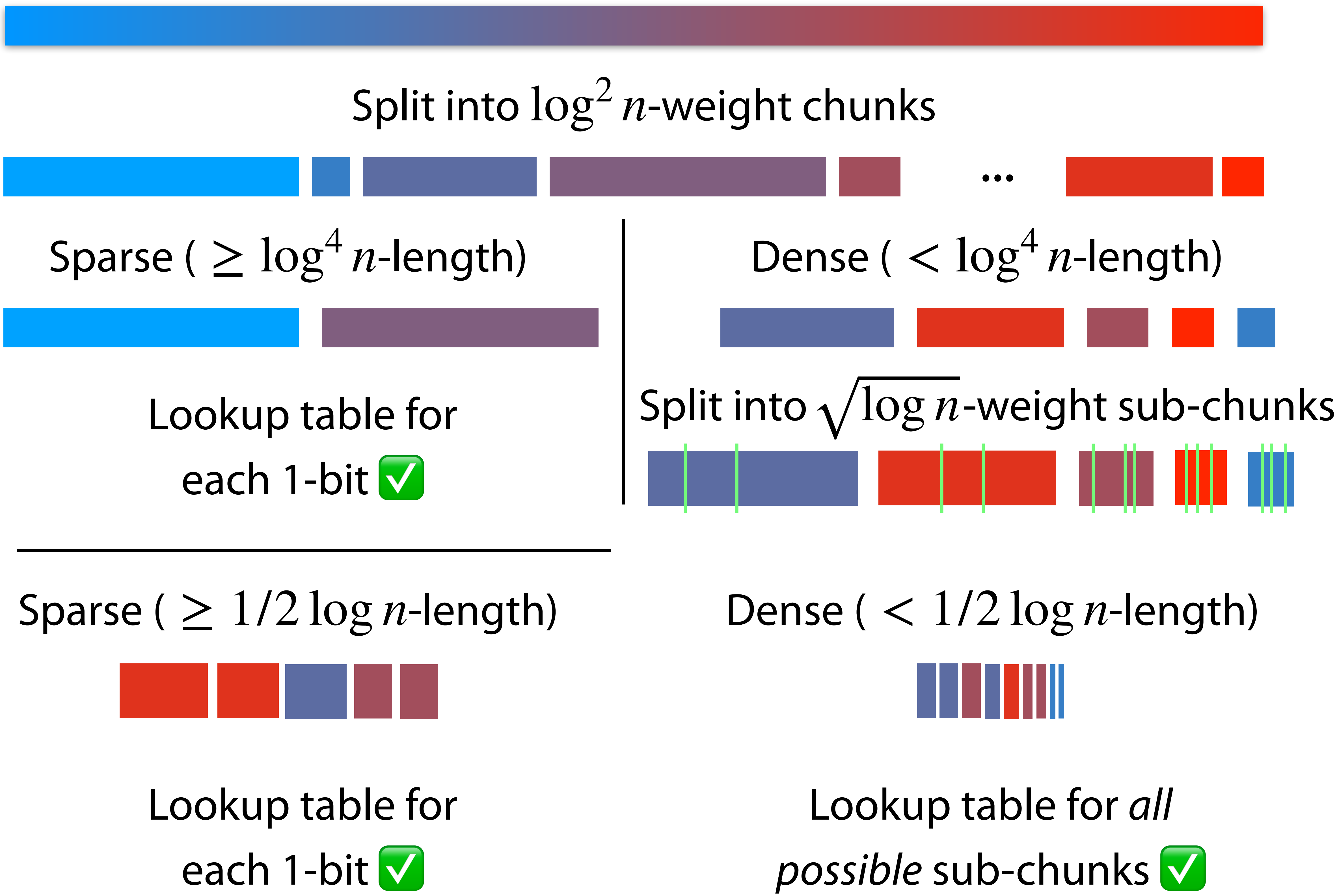
Unlike rank:

- Chunks are defined by # 1s, not # bits
- Two layers of “sparsity”
- Answer is an offset into bitvector

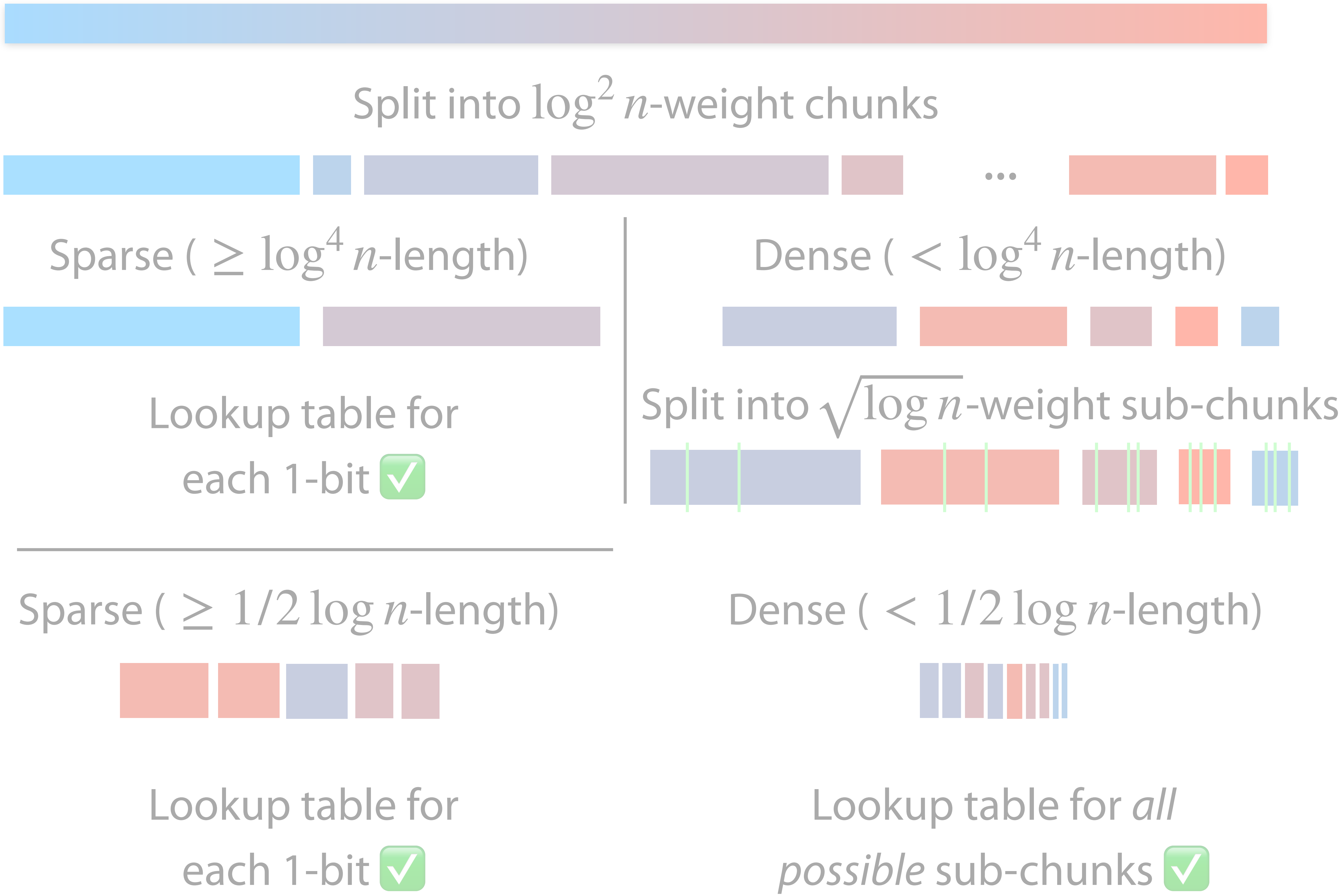
Bitvectors


B
0
1
1
0
1
0
1
0
0
0
1
0
1
B . rank0(5) = 2
B . select0(3) = 7
B . select1(5) = 12
B . rank1
B . select1
[0, n)
[0, m)

Clark's select



Clark's select



Clark's select

Split into $\log^2 n$-weight chunks

...

Sparse ($\geq \log^4 n$-length)

Lookup table for
each 1-bit ✅

Dense ($< \log^4 n$-length)

Split into $\sqrt{\log n}$-weight sub-chunks

Sparse ($\geq 1/2 \log n$-length)

Lookup table for
each 1-bit ✅

Dense ($< 1/2 \log n$-length)

Lookup table for *all*
possible sub-chunks ✅

Clark's select

Split the string into chunks each containing $\log^2 n$ 1-bits

Larger chunks are ***sparse***; 1's spread out

Shorter chunks are ***dense***; 1's packed together

Clark's select

Each chunk contains $\log^2 n$ 1-bits

We store offset of each chunk start

This takes:

Clark's select

Each chunk contains $\log^2 n$ 1-bits

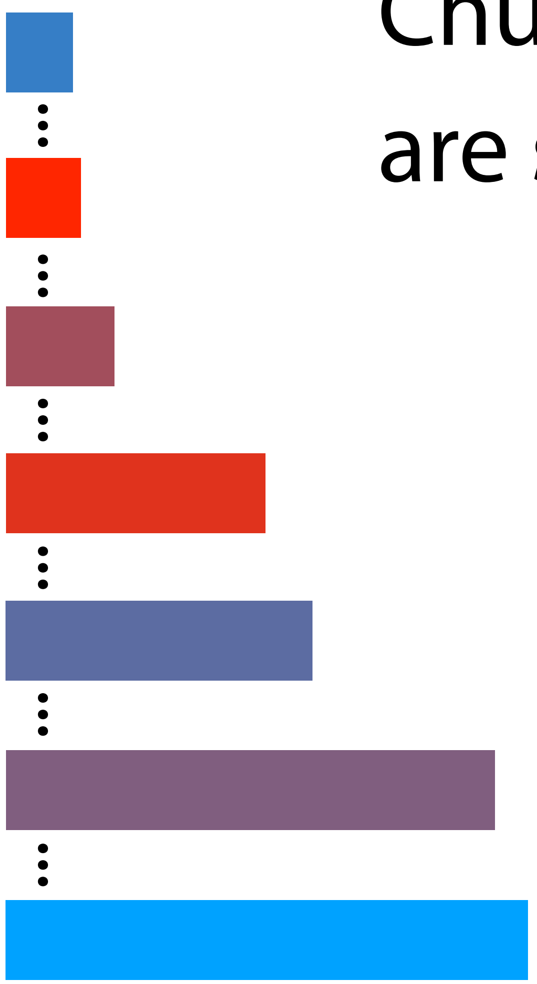
We store offset of each chunk start

This takes:

$$O\left(\frac{n}{\log^2 n}\log n\right) = O\left(\frac{n}{\log n}\right) = \check{o}(n) \text{ bits}$$

Worst case:
every bit set

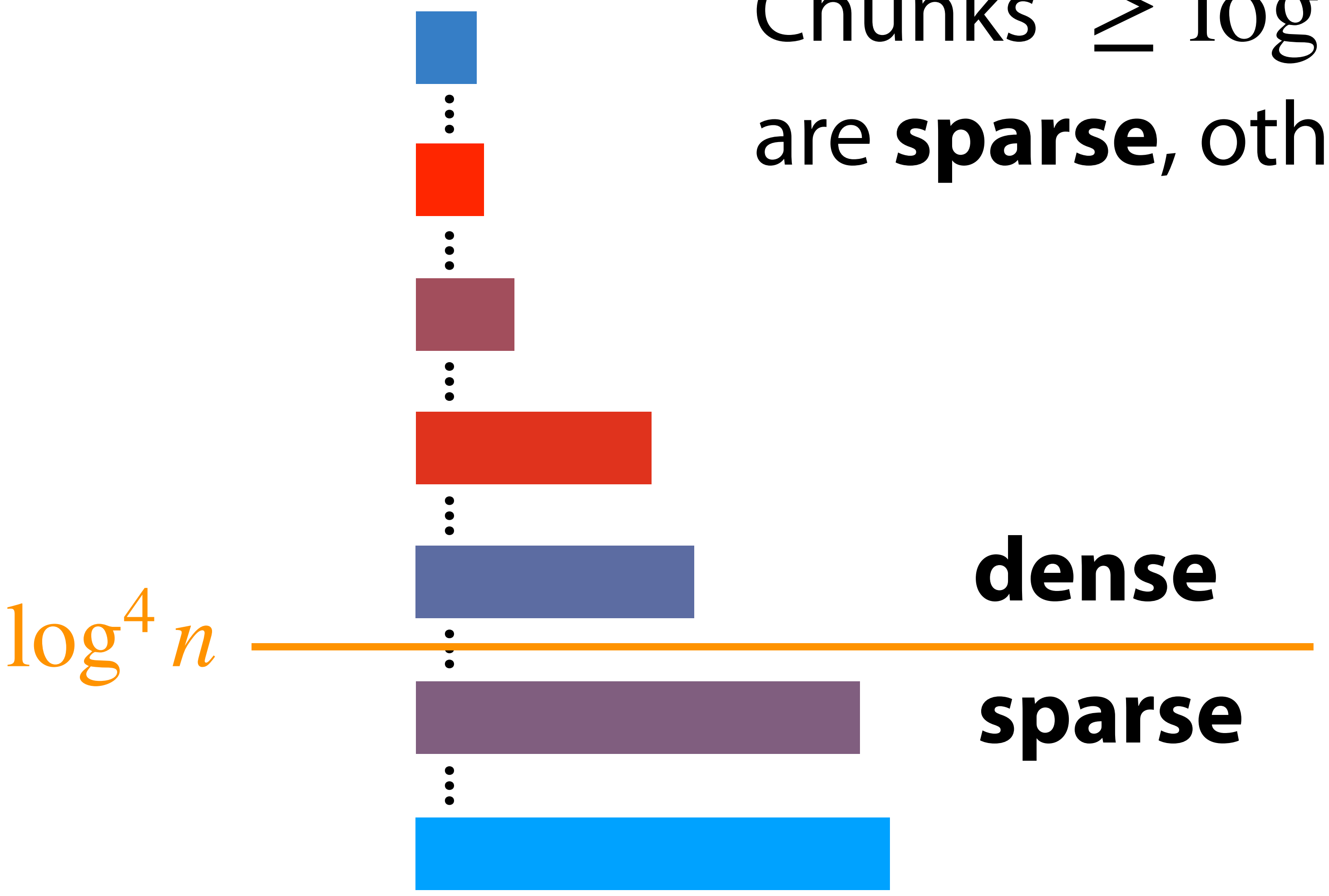
Clark's select



Chunks $\geq \log^4 n$ bits in length are **sparse**, others **dense**

Clark's select

Chunks $\geq \log^4 n$ bits in length are **sparse**, others **dense**



$\log^4 n$ is square of the # of set bits per chunk, $\log^2 n$

("sparse" roughly means "less than $\sqrt{\text{\# bits}}$ are set")

Clark's select: sparse case

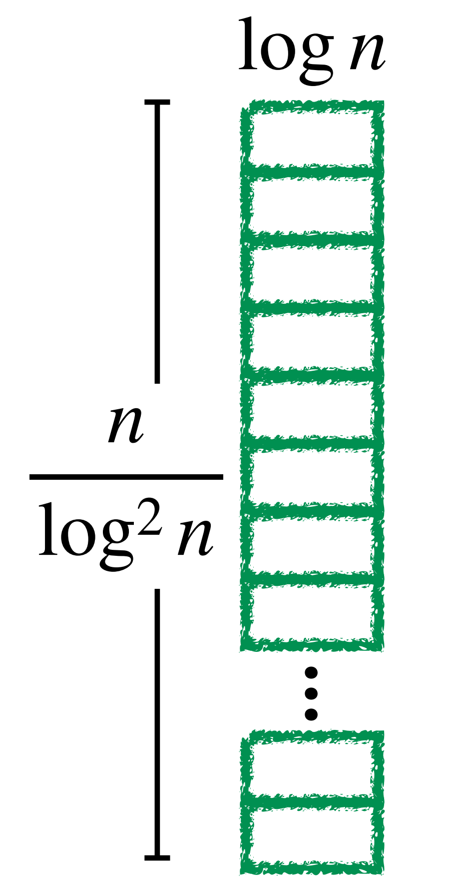
Pre-calculate B . select_1 for 1-bits in sparse chunks

Clark's select: sparse case

Pre-calculate B . select$_1$ for 1-bits in sparse chunks

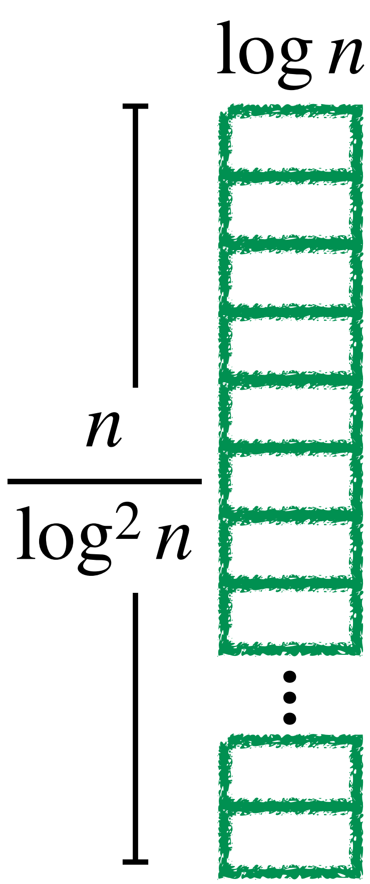
$$O\left(\underbrace{\frac{n}{\log^4 n}}_{\text{Max \# sparse chunks}} \cdot \underbrace{\log n}_{\text{\# bits to store 1 answer}} \cdot \underbrace{\log^2 n}_{\text{\# answers per chunk}}\right)$$

$$= O\left(\frac{n}{\log n}\right) = \check{o}(n)$$



Offsets for 1-bits in sparse chunks

Clark's select



Offsets for 1-bits in sparse chunks

So far, strategy for select is:

(a) find what chunk it's in (division)

(b) if chunk is **sparse** ($\geq \log^4 n$ bits)

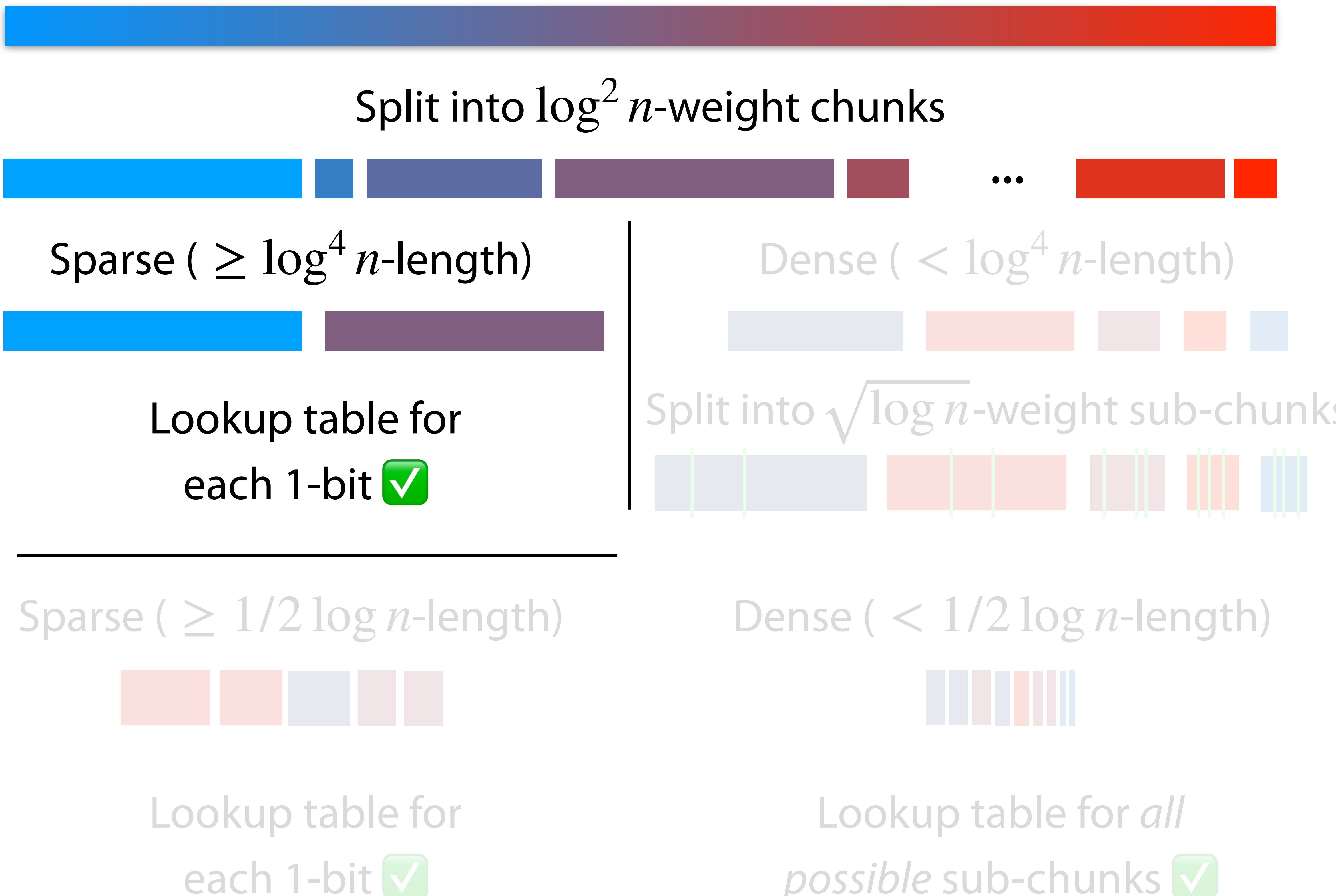
(b.i) look up in sparse offset table

(c) if chunk is **dense** ($< \log^4 n$ bits)

TODO

Space is $\breve{o}(n)$ so far

Clark's select



Clark's select

Split into $\log^2 n$-weight chunks

...

Sparse ($\geq \log^4 n$-length)

Lookup table for each 1-bit ✅

Dense ($< \log^4 n$-length)

Split into $\sqrt{\log n}$-weight sub-chunks

Sparse ($\geq 1/2 \log n$-length)

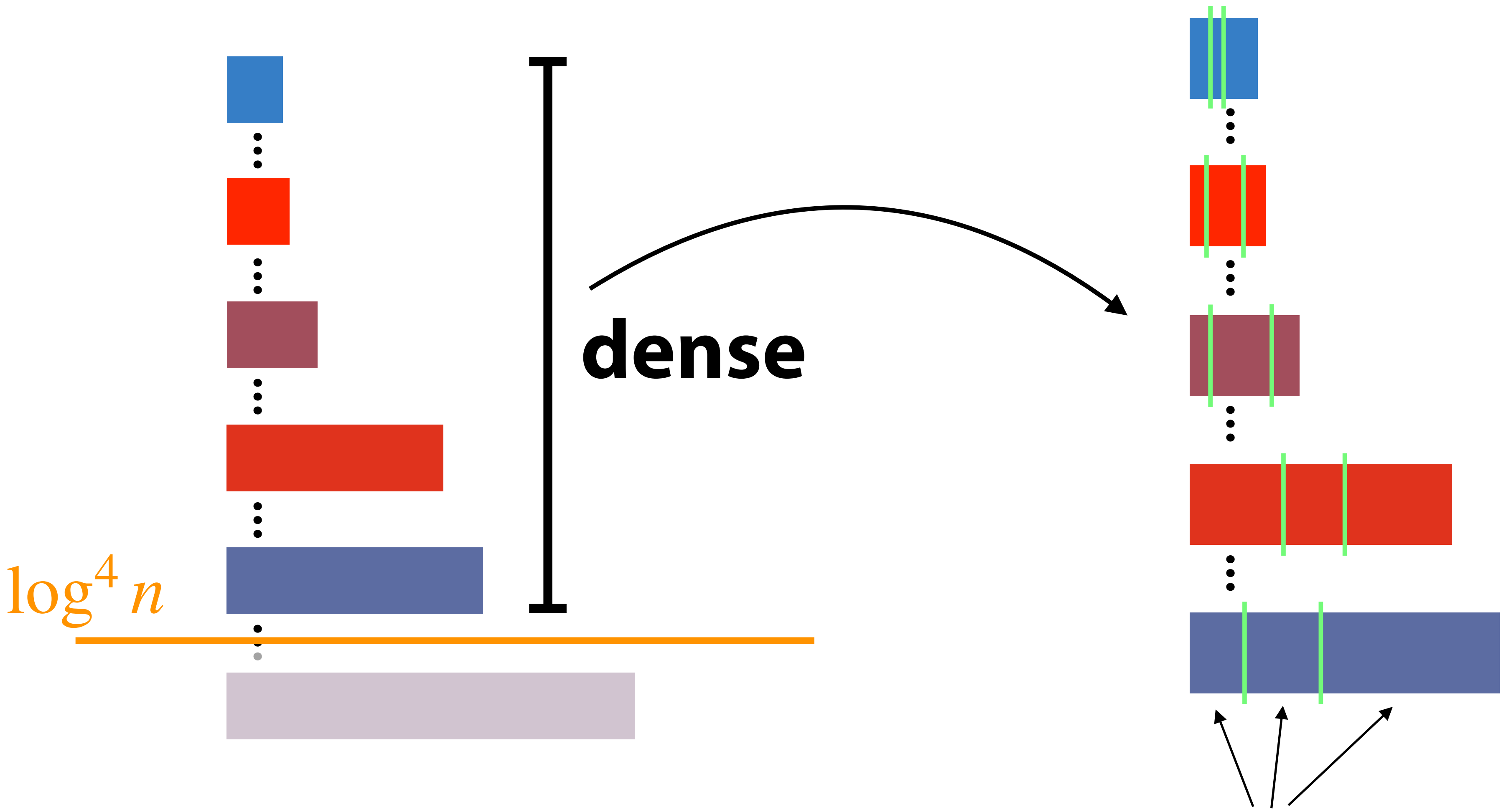
Lookup table for each 1-bit ✅

Dense ($< 1/2 \log n$-length)

Lookup table for *all possible* sub-chunks ✅

Clark's select: dense case

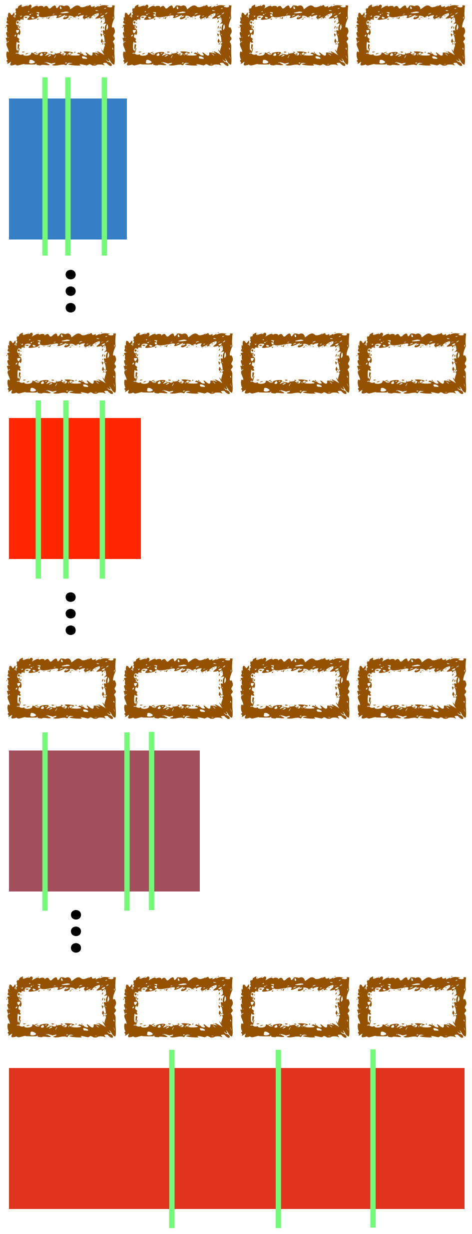
Dense chunks are shorter than $\log^4 n$ bits; further subdivide to sub-chunks of $\sqrt{\log n}$ 1-bits each



Chunks with $\log^2 n$ 1s

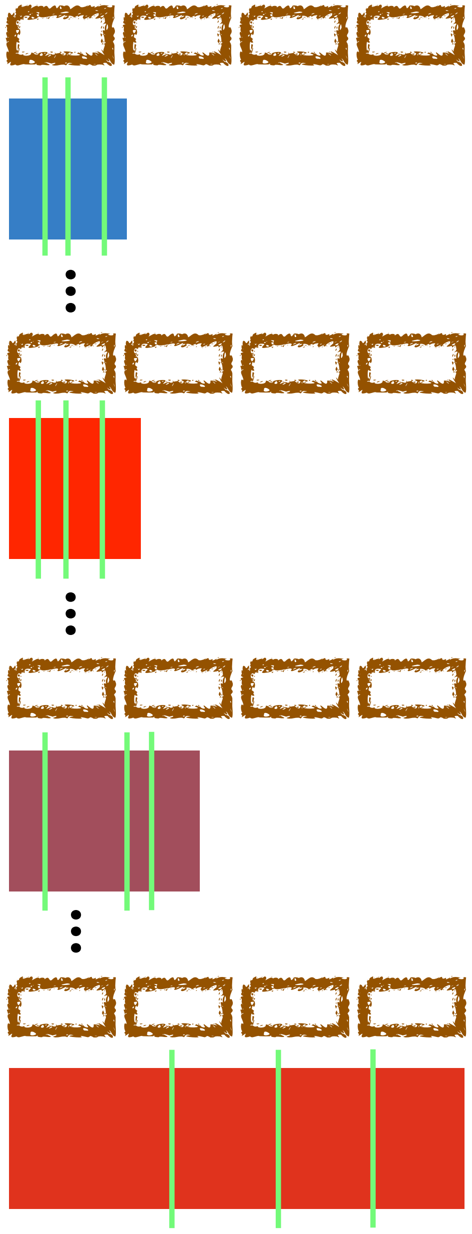
Sub-chunks with $\sqrt{\log n}$ 1s

Clark's select: dense case



Store relative offset per sub-chunk

Clark's select: dense case

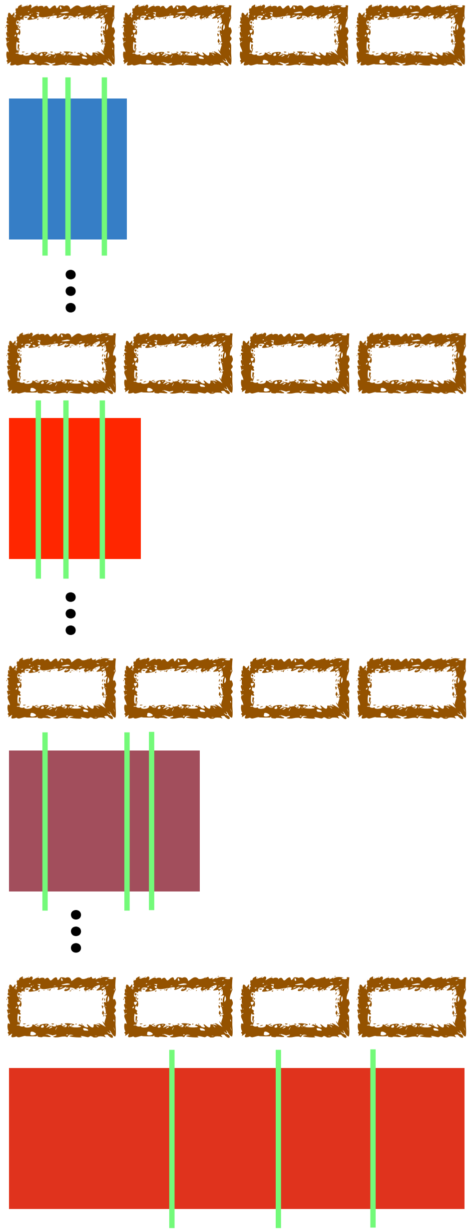


Store relative offset per sub-chunk

At most $n/\sqrt{\log n}$ sub-chunks

Containing chunk has $< \log^4 n$ bits, so relative offset fits in

Clark's select: dense case

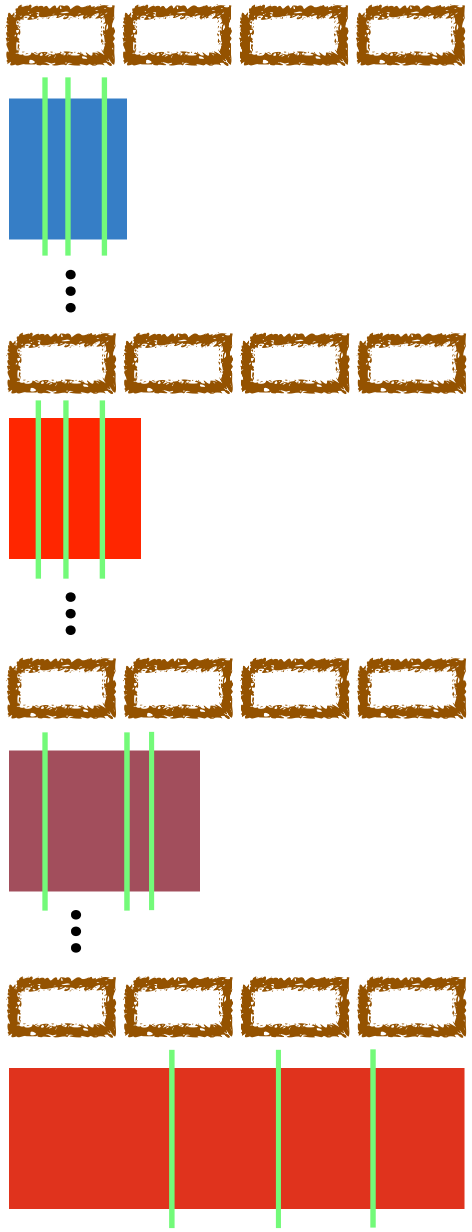


Store relative offset per sub-chunk

At most $n/\sqrt{\log n}$ sub-chunks

Containing chunk has $< \log^4 n$ bits, so relative offset fits in $O(\log \log^4 n) = O(\log \log n)$ bits

Clark's select: dense case



Store relative offset per sub-chunk

At most $n/\sqrt{\log n}$ sub-chunks

Containing chunk has $< \log^4 n$ bits, so relative offset fits in $O(\log \log^4 n) = O(\log \log n)$ bits

Overall: $O\left(\frac{n \log \log n}{\sqrt{\log n}}\right) = \check{o}(n)$

Clark's select

So far:

(a) find what chunk it's in (division)

(b) if chunk is **sparse**

(b.i) look up in sparse offset table

(c) if chunk is **dense**

(c.i) look up chunk's offset

(c.ii) find what sub-chunk it's in (division by $\sqrt{\log n}$)

(c.iii) look up sub-chunk's relative offset

TODO: need to look *within* sub-chunks

Clark's select

So far:

(a) find what chunk it's in (division)

(b) if chunk is **sparse**

(b.i) look up in sparse offset table

(c) if chunk is **dense**

(c.i) look up chunk's offset

(c.ii) find what sub-chunk it's in (division by $\sqrt{\log n}$)

(c.iii) look up sub-chunk's relative offset

TODO: need to look *within* sub-chunks

Clark's select

So far:

(a) find what chunk it's in (division)

(b) if chunk is **sparse**

(b.i) look up in sparse offset table

(c) if chunk is **dense**

(c.i) look up chunk's offset

(c.ii) find what sub-chunk it's in (division by $\sqrt{\log n}$)

(c.iii) look up sub-chunk's relative offset

TODO: need to look *within* sub-chunks

Clark's select: dense/sparse case

Sub-chunks with $\geq 1/2 \log n$ bits are **sparse**; we simply store relative offsets for every 1-bit

Clark's select: dense/sparse case

Sub-chunks with $\geq 1/2 \log n$ bits are **sparse**; we simply store relative offsets for every 1-bit

Overall: $O\left(\underbrace{\frac{n}{1/2 \log n}} \; \underbrace{\log \log n} \; \underbrace{\sqrt{\log n}}\right)$

Max # sparse sub-chunks

bits to store 1 answer (rel. to chunk)

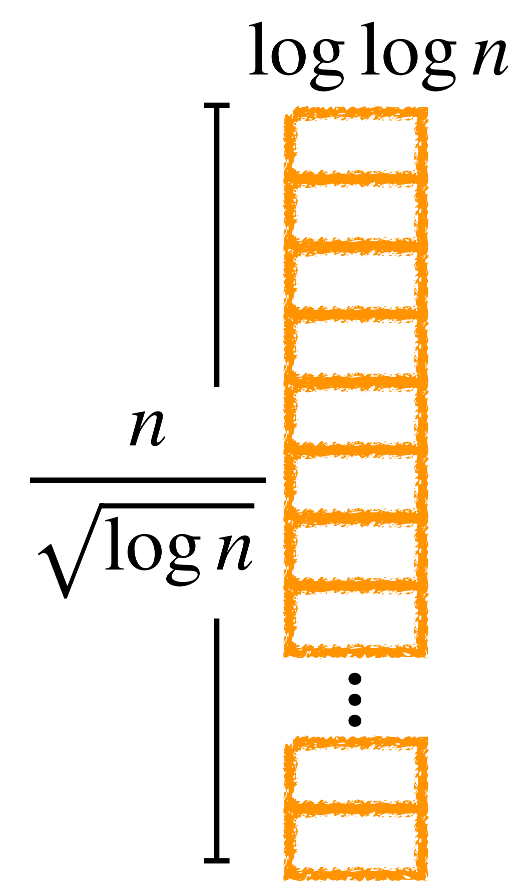
1-bits per chunk

Clark's select: dense/sparse case

Sub-chunks with $\geq 1/2 \log n$ bits are **sparse**; we simply store relative offsets for every 1-bit

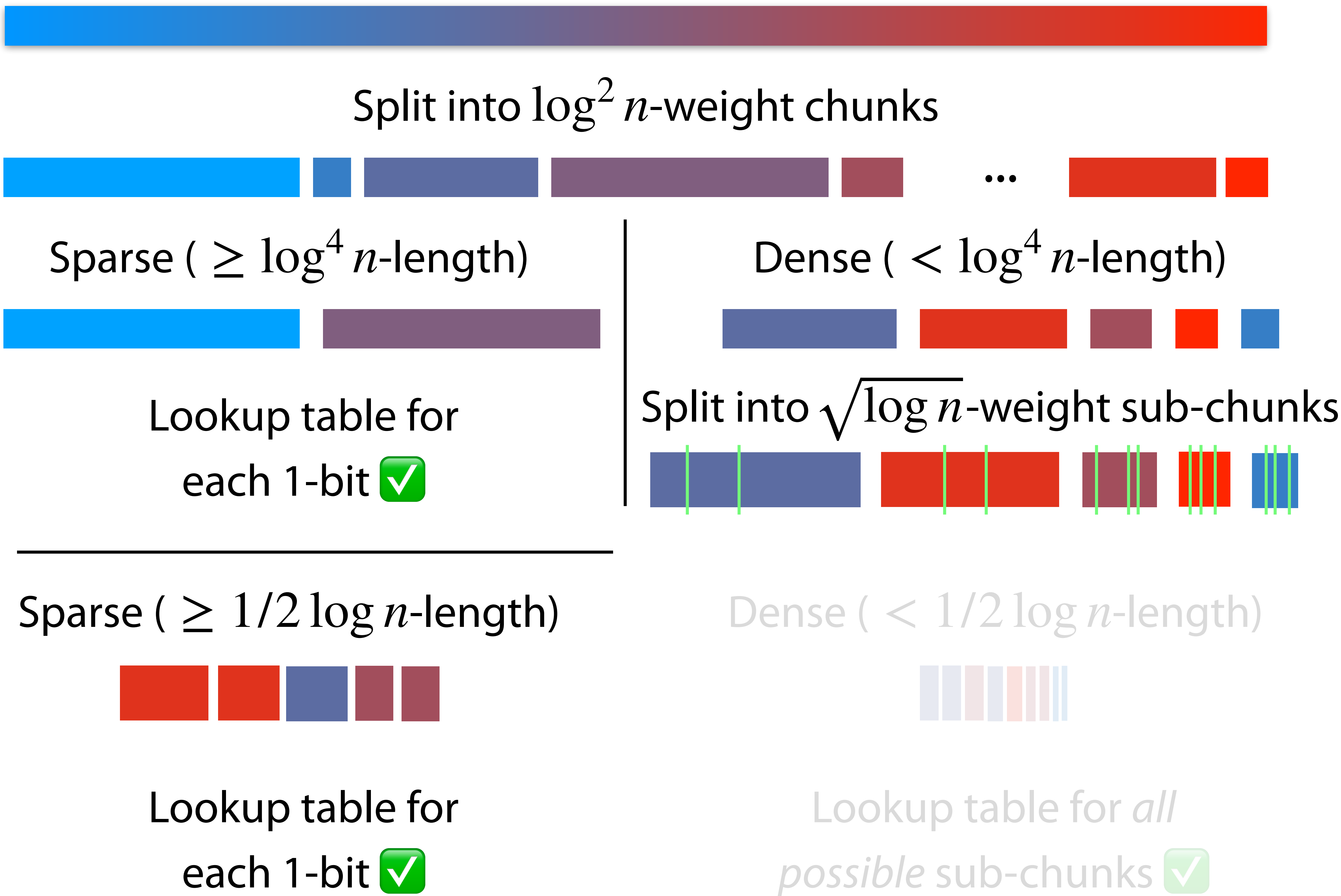
Overall: $$O\left(\frac{n}{1/2 \log n} \log \log n \sqrt{\log n}\right)$$

- $\frac{n}{1/2 \log n}$: Max # sparse sub-chunks
- $\log \log n$: # bits to store 1 answer (rel. to chunk)
- $\sqrt{\log n}$: # 1-bits per chunk



$$= O\left(\frac{n\sqrt{\log n} \log \log n}{\log n}\right) = O\left(\frac{n \log \log n}{\sqrt{\log n}}\right) = \check{o}(n)$$

Clark's select



Clark's select

Split into $\log^2 n$-weight chunks

...

Sparse ($\geq \log^4 n$-length)

Lookup table for each 1-bit ✅

Dense ($< \log^4 n$-length)

Split into $\sqrt{\log n}$-weight sub-chunks

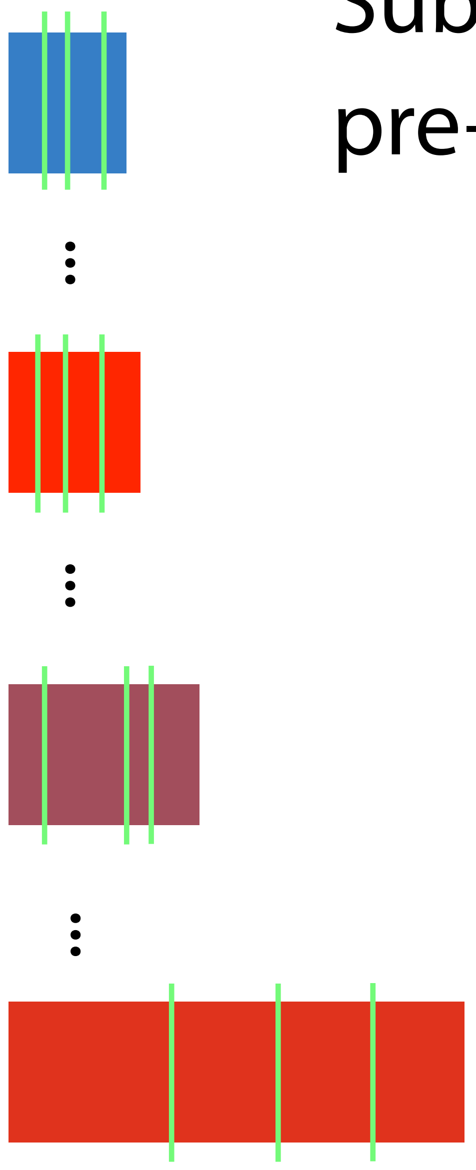
Sparse ($\geq 1/2 \log n$-length)

Lookup table for each 1-bit ✅

Dense ($< 1/2 \log n$-length)

Lookup table for *all possible* sub-chunks ✅

Clark's select: dense/dense case

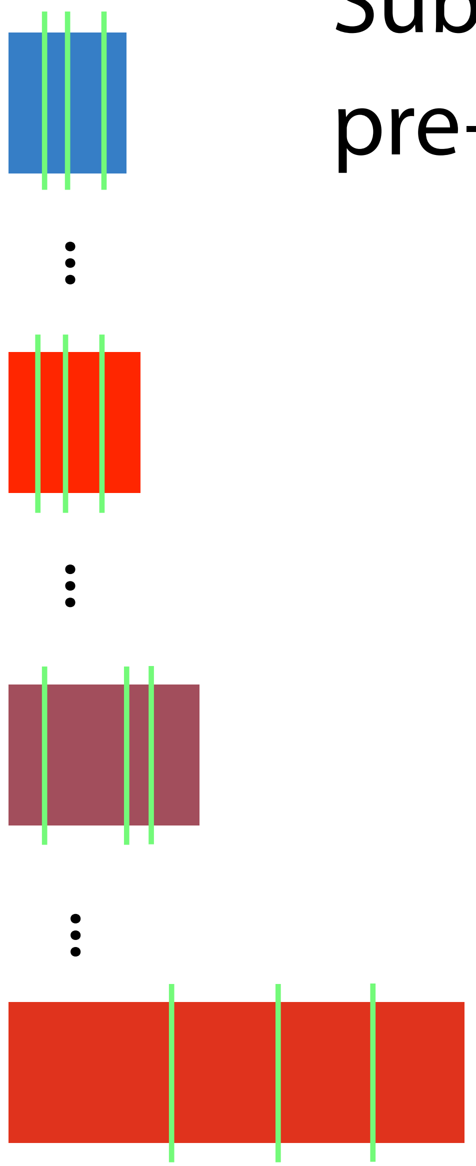


Sub-chunks $< 1/2 \log n$ bits are **dense**;
pre-calculate answers for all such chunks, like rank:

$$2^{1/2 \log n}$$

possible
bitvectors

Clark's select: dense/dense case



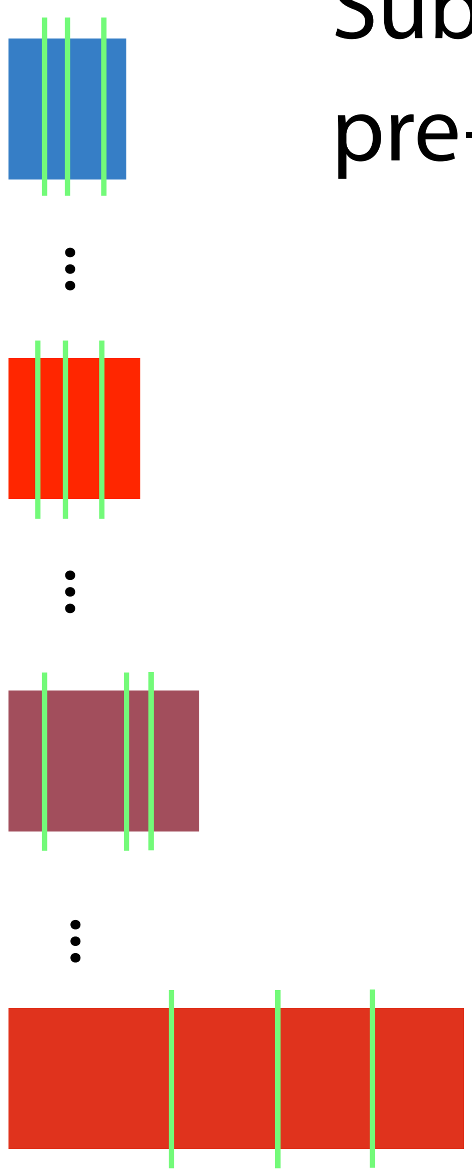
Sub-chunks $< 1/2 \log n$ bits are **dense**;
pre-calculate answers for all such chunks, like rank:

$$2^{1/2 \log n} \cdot \sqrt{\log n}$$

possible bitvectors

possible 1-bits

Clark's select: dense/dense case

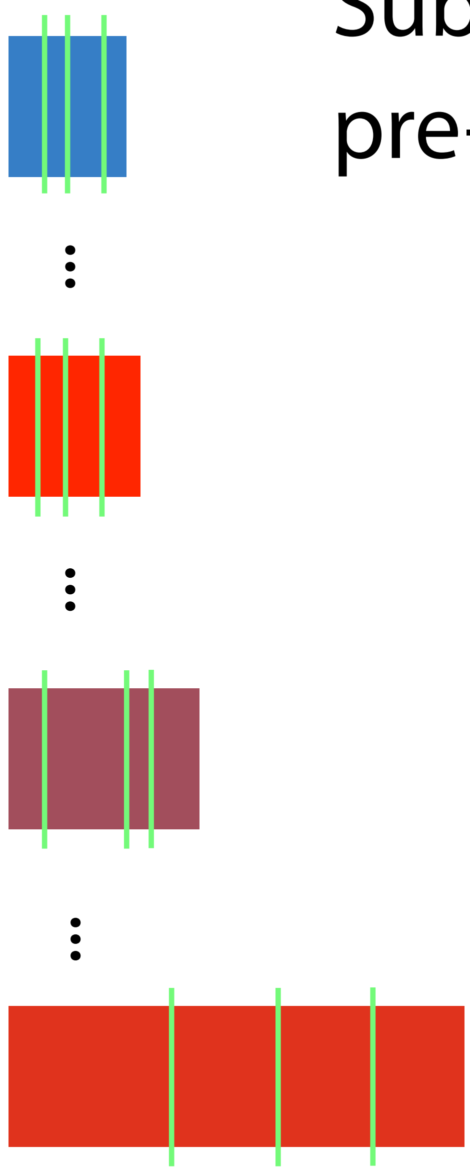


Sub-chunks $< 1/2 \log n$ bits are **dense**;
pre-calculate answers for all such chunks, like rank:

$$2^{1/2 \log n} \cdot \sqrt{\log n} \cdot \log \log n$$

possible bitvectors · possible 1-bits · answer

Clark's select: dense/dense case



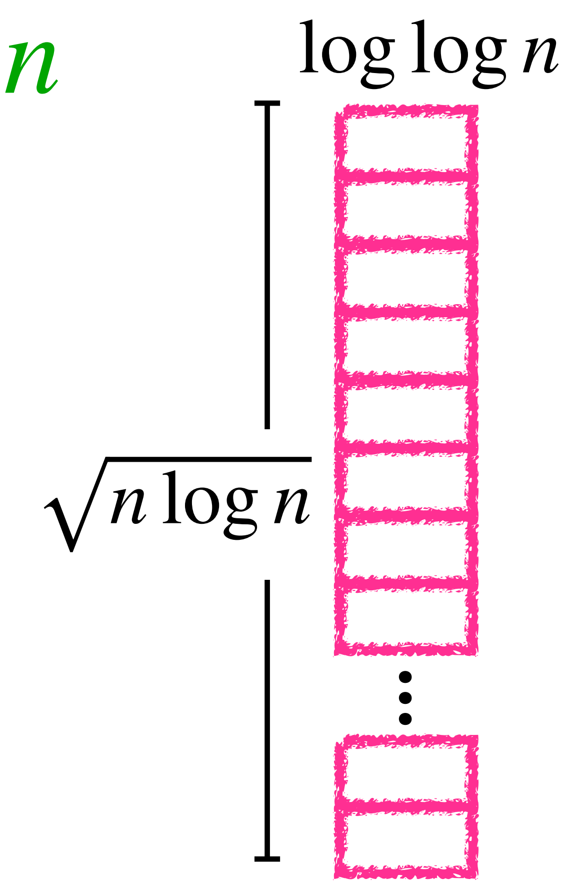
Sub-chunks $< 1/2 \log n$ bits are **dense**;
pre-calculate answers for all such chunks, like rank:

$$2^{1/2 \log n} \cdot \sqrt{\log n} \cdot \log \log n$$

possible bitvectors · possible 1-bits · answer

$$= O\left(\sqrt{n \log n} \log \log n\right)$$

$$= \check{o}(n)$$



Clark's select

(a) find what chunk it's in (division by $\log^2 n$)

(b) if chunk is **sparse** ($\geq \log^4 n$ bits)

(b.i) look up answer in sparse offset table

(c) if chunk is **dense** ($< \log^4 n$ bits)

(c.i) look up chunk's offset

(c.ii) find what sub-chunk it's in (divide by $\sqrt{\log n}$)

(c.iii) look up sub-chunk's relative offset

(c.iv) if sub-chunk is **sparse** ($\geq 1/2 \log n$ bits)

(c.iv.A) look up answer in sparse 1-bit table

(c.iv.B) return **(c.i)** + **(c.iii) + (c.iv.A)**

(c.v) if sub-chunk is **dense**

(c.v.A) look up answer in all possible dense/dense table

(c.v.B) return **(c.i)** + **(c.iii) + (c.v.A)**

Clark's select

(a) find what chunk it's in (division by $\log^2 n$)

(b) if chunk is **sparse** ($\geq \log^4 n$ bits)

(b.i) look up answer in sparse offset table

(c) if chunk is **dense** ($< \log^4 n$ bits)

(c.i) look up chunk's offset

(c.ii) find what sub-chunk it's in (divide by $\sqrt{\log n}$)

(c.iii) look up sub-chunk's relative offset

(c.iv) if sub-chunk is **sparse** ($\geq 1/2 \log n$ bits)

(c.iv.A) look up answer in sparse 1-bit table

(c.iv.B) return **(c.i)** + **(c.iii)** + **(c.iv.A)**

(c.v) if sub-chunk is **dense**

(c.v.A) look up answer in all possible dense/dense table

(c.v.B) return **(c.i)** + **(c.iii)** + **(c.v.A)**

Clark's select

(a) find what chunk it's in (division by $\log^2 n$)

(b) if chunk is **sparse** ($\geq \log^4 n$ bits)

(b.i) look up answer in sparse offset table

(c) if chunk is **dense** ($< \log^4 n$ bits)

(c.i) look up chunk's offset

(c.ii) find what sub-chunk it's in (divide by $\sqrt{\log n}$)

(c.iii) look up sub-chunk's relative offset

(c.iv) if sub-chunk is **sparse** ($\geq 1/2 \log n$ bits)

(c.iv.A) look up answer in sparse 1-bit table

(c.iv.B) return **(c.i)** + **(c.iii)** + **(c.iv.A)**

(c.v) if sub-chunk is **dense**

(c.v.A) look up answer in all possible dense/dense table

(c.v.B) return **(c.i)** + **(c.iii)** + **(c.v.A)**

Clark's select

(a) find what chunk it's in (division by $\log^2 n$)

(b) if chunk is **sparse** ($\geq \log^4 n$ bits)

(b.i) look up answer in sparse offset table

(c) if chunk is **dense** ($< \log^4 n$ bits)

(c.i) look up chunk's offset

(c.ii) find what sub-chunk it's in (divide by $\sqrt{\log n}$)

(c.iii) look up sub-chunk's relative offset

(c.iv) if sub-chunk is **sparse** ($\geq 1/2 \log n$ bits)

(c.iv.A) look up answer in sparse 1-bit table

(c.iv.B) return **(c.i)** + **(c.iii) + (c.iv.A)**

(c.v) if sub-chunk is **dense**

(c.v.A) look up answer in all possible dense/dense table

(c.v.B) return **(c.i)** + **(c.iii)** + **(c.v.A)**

Clark's select

(a) find what chunk it's in (division by $\log^2 n$)

(b) if chunk is **sparse** ($\geq \log^4 n$ bits)

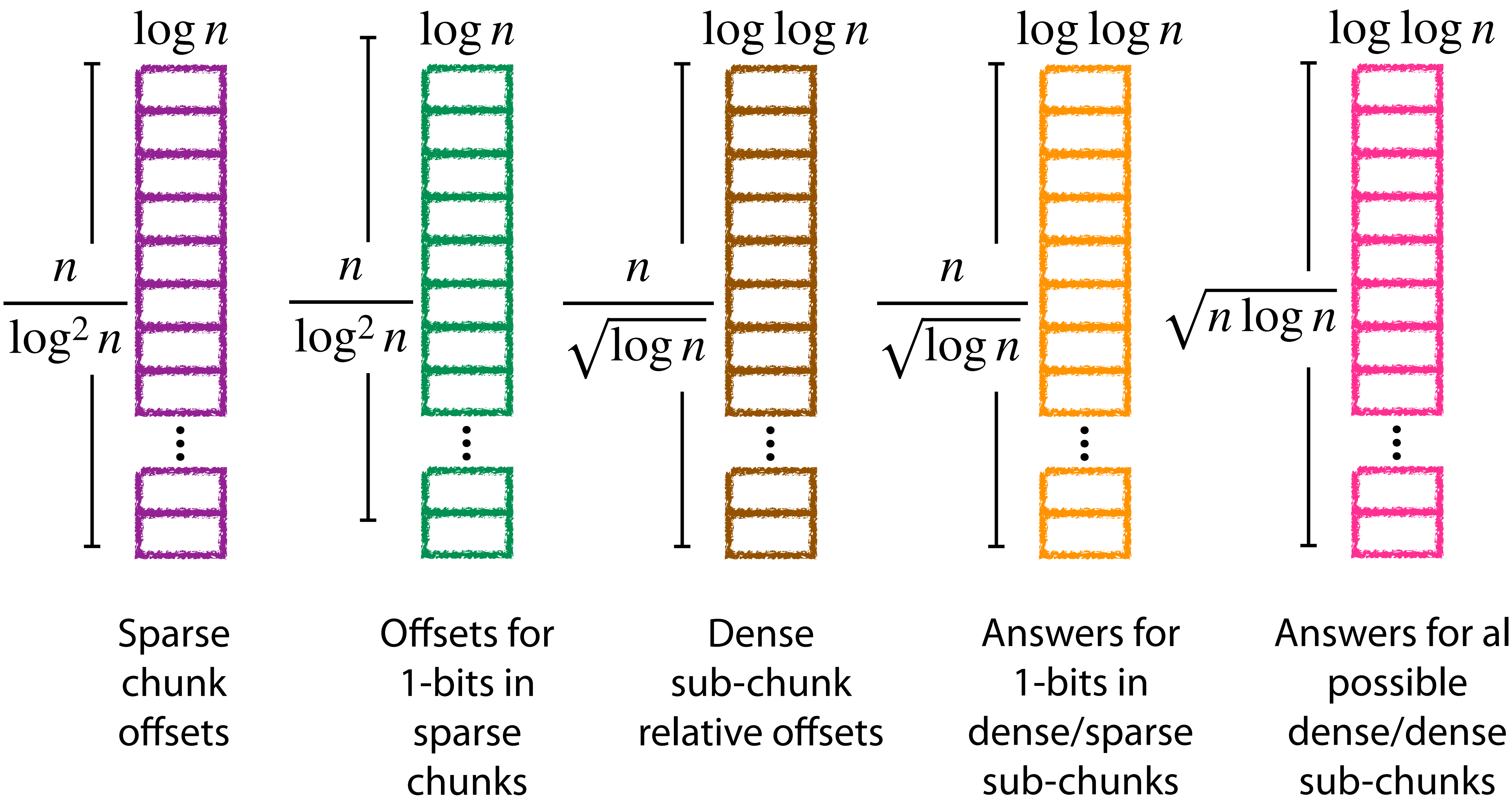
- **(b.i)** look up answer in sparse offset table

(c) if chunk is **dense** ($< \log^4 n$ bits)

- **(c.i)** look up chunk's offset
- **(c.ii)** find what sub-chunk it's in (divide by $\sqrt{\log n}$)
- **(c.iii)** look up sub-chunk's relative offset
- **(c.iv)** if sub-chunk is **sparse** ($\geq 1/2 \log n$ bits)
  - **(c.iv.A)** look up answer in sparse 1-bit table
  - **(c.iv.B)** return **(c.i)** + **(c.iii) + (c.iv.A)**
- **(c.v)** if sub-chunk is **dense**
  - **(c.v.A)** look up answer in all possible dense/dense table
  - **(c.v.B)** return **(c.i)** + **(c.iii) + (c.v.A)**

Clark's select

Overall, space is $\check{o}(n)$



Bitvectors

The dream! 🛌 💭 The reality! 🎆🥳🎉

	Time	Space (bits)	Note
$B.\text{access}$	$O(1)$	n	Lookup
$B.\text{select}_1$	$O(1)$	$\check{o}(n)$	✅ Clark
$B.\text{rank}_1$	$O(1)$	$\check{o}(n)$	✅ Jacobson