COMPRESSED DS: integers array

- given a binary array:

	1	2	3	4	5	6	7		m
B:	0	1	0	0	1	1	0	…	

$m = |B|$

$n = \# 1$

$\# 1$ # bits to represent the positions of the 1 (universe is m)

- STORE position of the 1 (STANDARD ENCODING) $O(n \cdot \log_2 m)$ bits space

(POINTER BASED PROGRAMMING)

⇓

$\log_2 m$ bits points to position of the once

- POINTERLESS PROGRAMMING

↳ TRY TO avoid pointers

- 2 OPERATIONS: $Rank_0(i) = i - Rank_1(i)$

1) $Rank_b(i)$: # of b in the prefix $B[1, i]$

position

$b \in \{0,1\}$

↳ how many b I have

$Rank_1(4) = 1$, $RANK_0(i) = i - RANK_1(i)$

(inverse of the rank)

2) $Select_b(i)$: position of the i^{TH} b in B

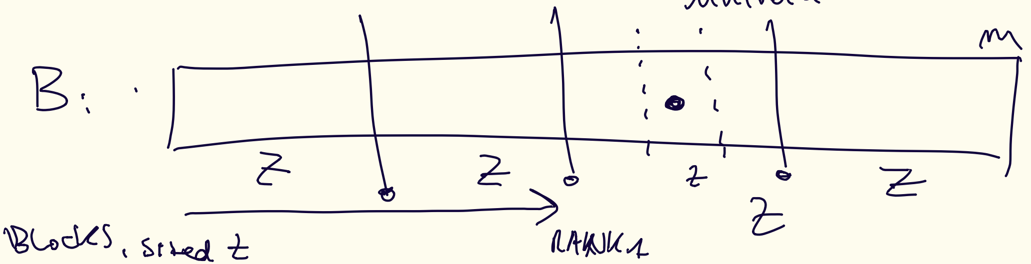
$select_1(3) =$ select the 3^{RD} 1 = 6

something asymptotically less than m (universe size)

TEO: ∃ a DS that takes $o(m)$ in addition to B and supports $Rank_1$ in $O(1)$ time

read only (array B not touched)

limit: sublinear in the Universe



1) Partition array in Blocks, sized Z

big blocks = $O\left(\frac{m}{Z}\right)$

① ∀ position ↓ (∀ Big block) I store $rank_1$ of all previous blocks

(absolute, counting 1s from the beginning)

• space for BIG BLOCKS = $O\left(\frac{m}{Z} \cdot \log_2 m\right)$ bits

↗ # of big block counter

(~~#~~ 1 up to the beginning, but
I ~~don't~~ don't want consider
just 1 but also the
zero => write $\log_2 m$

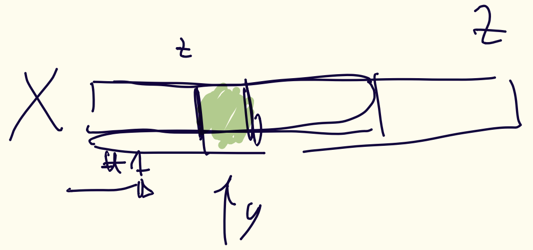
cut m >> m

② If I want to know # 1 in the middle of Z, need $rank_1$ previous
+ a scan for 1 => Z can be big

=> consider small blocks z

FOUR RUSSIAN TRICK : we're working in RAM model with word size $O(\log m)$ bits
=> every operation on a word of log m is constant

• when we do a rank:



quering this position:

↳ scan takes $O(z)$ time

=> want in constant => TABULATE ALL POSSIBLE ANSWER

=> • ∀ small block configuration x and for every position $y \in [1, z]$ => store #1 in $x[1, y]$

TABLE T

	0	1
00		
01		
10		
11		

ROWS 2^z

COLUMNS z

- $\text{Rank}_1(i) =$
  - → Counter STORED at Z (big block) including i
  - → " " " z (small ") " i
  - → let x be the small block including i
    let y = i mod z
    we get $T[x, y]$

SOLUTION WITH ELIAS ~ PHANO

THM ∃ DS that uses $o(m)$ bits in addition to B that supports select_1 & select_0 in constant time ($O(1)$)

The 2 THEOREMS doesn't depend on (n) (# of keys) but on the universe size

- In Elias-phano, I combine select_1 & select_0
  => I get a solution (a DS) that occupies

universe size

space = $2n + n\lceil \log \frac{m}{n} \rceil + o(n)$ bits

# of keys