γ - CODE

encode: $\gamma(x) = \underbrace{0 \dots 0}_{\# |bin(x)|-1} \underbrace{\qquad\qquad}_{bin(x)}$

decode: read # of zeros until you reach a 1

=> convert the number from $(\dots)_2$ to $(\dots)_{10}$

es

- encode 9

$$\gamma(9) = \underbrace{000}_{4-1=3} \underbrace{1001}_{|4|}$$

- decode

$$000\underset{4}{1}001 \qquad (1001)_2 = 2^3 + 1 = 8 + 1 = 9$$ ✓

— o —

- encode 15

$$\gamma(15) = \underbrace{000}_{4-1=3} \underbrace{1111}_{4}$$

- decode: $000\underbrace{1111}$ $\qquad (1111)_2 = 15$ ✓

— ~ —

- encode 36

$$\gamma(36) = \underbrace{00000}_{6-1=5} \underbrace{100100}_{6}$$

decode: $00000\underbrace{100100} = 36$ ✓

δ - CODE $\quad x > 0$

$$\delta(x) = \underbrace{\gamma(e)}\ \underbrace{bin(x)}_{e}$$

$$\delta(14) = \underbrace{00100}_{\gamma(4)}\ \underbrace{1110}_{e}$$

$$\hookrightarrow \underbrace{00}_{3-1}\ \underbrace{100}_{e=3}$$

- decode δ

$$\underbrace{00}_{2}\underbrace{100}_{4}\underbrace{1110}_{8+4+2=14} \Rightarrow 14$$

$$\delta(10) = \underbrace{\gamma(e)}\ \underbrace{bin(10)}_{e} = \underbrace{00100}_{\gamma(4)}\ \underbrace{1010}_{e}$$

$$\gamma(4) = \underbrace{00}_{\#0 = |bin(4)|-1}\ \underbrace{100}_{bin(4)}$$

- decode with δ code the sequence:

$$(\underbrace{00}_{2}\underbrace{100}_{3}\underbrace{1100}_{8+4=12})_2 \Rightarrow (12)_{10}$$

- decode with δ code

$$\underbrace{0}_{1}\underbrace{10}_{2}\underbrace{11}_{2+1=3}\underbrace{00}_{2}\underbrace{100}_{3 \Rightarrow 4}\underbrace{1010}_{8+2=10}\underbrace{1}_{1}\underbrace{1}_{1}$$

$3 \quad 10 \quad 0 \quad 1$

$\Rightarrow 3, 10, 1$

RICE-CODING: $x > 0$

$$R = \text{REMINDER} = x - 1 - 2^k \cdot q$$
$$q = \text{QUOTIENT} = \left\lfloor \frac{x-1}{2^k} \right\rfloor$$

$U(0) = 0$
$U(1) = 01$ $\quad U(2) = 001$
$U(3) = 0001$

$$R_k(q) = \underbrace{\text{UNARY}(q)}_{q+1 \text{ bits}} \underbrace{\text{BIN}(R)}_{k\text{-bits}}$$

es.: $k = 4$

$$R_4(20) = \begin{cases} R = 20 - 1 - 2^4 \cdot 1 = 19 - 16 = 3 \\ q = \left\lfloor \frac{20-1}{2^4} \right\rfloor = \left\lfloor \frac{19}{16} \right\rfloor = 1 \end{cases}$$

$$\Rightarrow \underbrace{01}_{U(1)} \underbrace{11}_{bin(R) = bin(3)}$$

$$R_4(20) \Rightarrow q = \left\lfloor \frac{x-1}{2^k} \right\rfloor = \left\lfloor \frac{19}{2^4} \right\rfloor = \left\lfloor \frac{19}{16} \right\rfloor = 1$$
$$R = x - 1 - 2^k \cdot q = 20 - 1 - 2^4 \cdot 1 =$$
$$19 - 16 = 3$$

$U(q) = 01$
$Bin(3) = 11$

$$R_4(20) = 0111$$

• Encode S with Delta & Rice coding → $k=3$

$S = (1, 6, 15, 18, 21, 24, 30)$

$S' = (1, 5, 9, 3, 3, 3, 6)$

$\delta(x) = \underbrace{\gamma(e)}\ \underbrace{\text{bin}(x)}_{e}$

$\delta(1) = 1\ \underbrace{1}_{e=1}$

$\gamma(1) = \underbrace{}_{\#0\ e-1}\ \underbrace{\text{bin}(1)}_{e} = \cancel{1}\ \underbrace{1}$

$\delta(5) = \underbrace{011}_{\gamma(3)}\ \underbrace{101}_{e=3}$

$\gamma(3) = \underbrace{0}_{\#0{:}\,e-1}\ \overset{\text{bin}(3)}{\underbrace{11}_{e}}$

$\delta(9) = \underbrace{00100}_{\gamma(4)}\ \underbrace{1001}_{4}$

$00\ \underbrace{100}$

$\delta(3) = \gamma(2)\ \underbrace{11}_{2} = 01011$

$\gamma(2) = 0\ \underbrace{10}$

$\delta(6) = \gamma(3)\ \underbrace{110}_{3} = 011110$

$\gamma(3) = 011$

$S' = (1, 5, 9, 3, 3, 3, 6)$ Rice coding $k=3$

$R_3(1) = 1000$

$R_3(1) \Rightarrow R = x - 1 - 2^k \cdot q = 0 - 2^3 \cdot 0 = 0$

$q = \left\lfloor \frac{x-1}{2^k} \right\rfloor = \left\lfloor \frac{0}{2^3} \right\rfloor = 0$

$\underbrace{U(q)}_{1}\ \underbrace{\text{Bin}(r)}_{000}$ $k=3$ bits

$R_3(5) \Rightarrow R = x - 1 - 2^k \cdot q = 5 - 1 - 0 = 4$

$q = \left\lfloor \frac{x-1}{2^k} \right\rfloor = \left\lfloor \frac{4}{8} \right\rfloor = 0$

$\underbrace{U(0)}\ \underbrace{\text{Bin}(4)}_{3\,=\,k\text{ bits}} = \underbrace{1}\ \underbrace{100}$

$R_3(9) \Rightarrow R = x - 1 - 2^k \cdot q = 9 - 1 - (8) = 0$

$q = \left\lfloor \frac{x-1}{2^k} \right\rfloor = \left\lfloor \frac{8}{8} \right\rfloor = 1$

$\overset{U(1)}{\underbrace{01}_{q+1\text{ bits}}}\ \overset{\text{bin}(0)}{\underbrace{000}_{3\text{ bits}}}$

$R_3(3) \Rightarrow R = x - 1 - 2^k \cdot q = 3 - 1 - 8 \cdot 0 = 2$

$q = \left\lfloor \frac{x-1}{2^k} \right\rfloor = \left\lfloor \frac{2}{8} \right\rfloor = 0$

$\underbrace{1}_{}$ $\underbrace{010}_{3 \text{ bits}}$ — $U(0)$ bin(2)

$R_3(6) \Rightarrow R = x - 1 - 2^k \cdot q = 6 - 1 - 8 \cdot 0 = 5$

$q = \left\lfloor \frac{x-1}{2^k} \right\rfloor = \left\lfloor \frac{5}{8} \right\rfloor = 0$

$\underbrace{1}_{}$ $\underbrace{101}_{3 \text{ bits}}$ — $U(0)$ bin(5)

P FOR DELTA

we work with integers, represented between: $0 \le i < 2^w$

but most of the integers are small, in the range: $\boxed{0 \le 1 < 2^b - 1}$

=> reduce integer represented in b-bits ($b << w$) $[0, 2^b - 1]$

es. $w = 32$ bits, most of integer have most 4 bits $b = 4$

- Represent my INPUT SEQUENCE with 2 arrays of bits
  1st) array were each integer has size b-bits ($b = 4$)
  2nd) array of additional w-bits ($w = 32$)

=> Take each input symbol, encode first in the first array ($b = 4$)
if the # will be small enough to represent them in this range stop
else if # too large (cannot be represented with b-bits => I will represent in the 1ST array with an ESCAPE VALUE and its value stored in the 2ND array → shorter then input sequence

example: $w = 32$, $b = 4$

$i = 0, \dots, 14$ → 1ST array cuz I can represent $0 \le i < 2^b - 1 = 2^4 - 1 = 15$

if $i = 15, 16, \dots$ → 1ST array I store 15 and in the 2ND array I store the value i (Better $i - 15$)

S = 1 3 17 4 15 12

1ST ARRAY 4 bits

1	3	15	4	15	12

$\underbrace{}_{4 \text{ bits}}$

2ND ARRAY 32 bit

17	15

$\underbrace{}_{32 \text{ bits}}$

es. p for delta 2/11/2021

$b = 3$, base not given, specify by yourself

base: value respect to which you encode all the other numbers

$S = (11, 14, 16, 17, 19, 20, 21, 31)$
$-11 \quad -11 \quad -11 \quad -11 \quad -11 \quad -11$

$S' = (0, 3, 5, 6, 8, 9, 10, 20)$

$b = 3$
$\Rightarrow$ max
$111 = 7 \Rightarrow$ ESCAPE
$\# \geq 7$
go outside

base = 11

base = $\min\{s_i\}$

$\Rightarrow$ 000, 011, 101, 110, 111, 111, 111, 111

OUTSIDE: 8, 9, 10, 20

$\Rightarrow$ BUT NO ADVANTAGE, can exploit GAP encoding (by −)

$\Rightarrow S'' = (\underset{\text{copy}}{\underline{0}}, 3, 2, 1, 2, 1, 1, 10)$

$\Rightarrow$ 000, 011, 010, 001, 010, 001, 001, $\underset{\text{escape}}{\underline{111}}$ OUT: 10

Decompress: used, gap-encoding; base = 11

$S''_{\text{compressed}}$	000	011	010	001	010	001	001	111 (escape)	OUT: 10
S':	0	3	3+2= 5	6	8	9	10	20	
									+11
S:	11	14	16	17	19	20	21	31	

VARIABLE BYTE CODING

- given a number i
- partition the binary representation of i in blocks of 7 bits

  - i - $Bin(i)$ - $i \geq 0$ - Little endian: START from right

$Bin(i)$: $\underbrace{1\,0\,0\,0\,1}_{5}\ \underbrace{0\,1\,0\,0\,1\,0\,1}_{7\text{ bits}}$ 12 bits

- put blocks of 7 bits in 1 byte (1 extra bit)
  ↳ used to make decoding prefix free

- LAST byte in the encoding
has the most significant bit = 1
others to 0

=> $\underbrace{0\,\underbrace{0\,0}_{\text{ADDED}}\,\underbrace{1\,0\,0\,0\,1}_{5}}_{8\text{ bits}}\ \underbrace{1\,|\,1\,0\,0\,1\,0\,1}_{8\text{ bits}}$

FERRAGINA FA AL CONTRARIO

- #s 0 - 127 requires 1 byte => $\boxed{1}\underbrace{\qquad\qquad}_{7\text{ bits encoding}}$
- #s 128 = 2^7 - $2^{14} - 1$ 2 bytes => $\boxed{0}\underbrace{\qquad\qquad}_{7\text{ bits}}\boxed{1}\underbrace{\qquad\qquad}_{7\text{ bits}}$

=> with 2 bytes I can encode until $\boxed{2^{14} - 1}$ => 14 bits available 4 encoding

- # 3 bite
$2^{14} - 2^{21} - 1$ 3 bytes => $\boxed{0}\underbrace{\qquad\qquad}_{7\text{ bits}}\boxed{0}\underbrace{\qquad\qquad}_{7\text{ bits}}\boxed{1}\underbrace{\qquad\qquad}_{7\text{ bits}}$

es. like Elias-Fano's style VARIABLE byte: padding always

$$Bin(i) = \underbrace{0\,0\,0\,1\,1\,0\,1}_{7 \text{ bits}}\ \underbrace{1\,0\,0\,1\,1\,1\,1}_{7 \text{ bits}}$$

MSB 1 if current byte
is not the last in encoding

$$\Rightarrow \underbrace{\boxed{1 : 0\,0\,0\,1\,1\,0\,1}}_{8 \text{ bits}}\ \underbrace{\boxed{0 ; 1\,0\,0\,1\,1\,1\,1}}_{8 \text{ bits}}$$

S - C DENSE CODE (byte oriented)

↳ but in example we'll use blocks of 4 bits

- S = stoppers
- C = continuers

$S + C = 2^b$ $b = 8 \Rightarrow$ 128 STOPPERS, 128 continuers $\Rightarrow$

- $\boxed{0}\ldots$ $\Rightarrow$ stoppers: 0...000 ; 0111...1
- $\boxed{1}\ldots$ $\Rightarrow$ continuers: 100...0 ; 11...1

$\Rightarrow$ values from 0 to 127 $\Rightarrow$ all stoppers

$\Rightarrow$ values from ~~127~~ to 256 $\Rightarrow$ continuers

→ looking at MSbit (most significant) I know if is a S or a C

but we can choose the range of C & S

at most we have $S + C = 2^8 = 256$

- we can encode how many distinct values with the choice:

$\Rightarrow$ with 1 byte $\Rightarrow$ STOPPER $\Rightarrow$ S possible value

2 bytes $\Rightarrow$ CONTINUER, STOPPER $\Rightarrow C \cdot S$ possible values

3 bytes $\Rightarrow$ CONTINUER, CONTINUER, STOPPER $\Rightarrow C^2 \cdot S$ possible values

...

using up to 3 bytes, the total # of possible values is $S + CS + C^2 S = S\underbrace{(1 + C + C^2)}_{\text{geometric progression}}$

$$= S\left(\frac{C^3 - 1}{C - 1}\right)$$

in general, using up to k-bites, =>
we can encode

$$\frac{s(c^k-1)}{c-1} \quad \text{different values}$$

- Suppose encoding all positives (≥ 0) integers: given i, I have to find the value k such that:

$$\frac{s(c^{k-1}-1)}{c-1} \leq i \leq \frac{s(c^k-1)}{c-1}$$

=> i will be encoded with k bites

- encode with S-C coding $S=2$ $C=2$

$S+C = 2+2 = 4$

$2^b = 4$

$b = \log_2 4 = 2$ => 2 bits

$S = 00, 01$ $\quad C = 10, 11$

$\boxed{S}$	encode	$(\#)_{10}$
	00	0
	01	1

--- o ---

$\boxed{C}\boxed{S}$	encode
10 00	0
10 01	1
11 00	2
11 01	3

encode 3 => 1101

CCS

• S-C dense code   encode 8 $S \cdot C =$
$= 5 \cdot 3 = 15$

$S = 5 \quad C = 3$

$S + C = 8$

$2^b = 8$

$b = \log_2 8 = 3$

$S = \underbrace{\begin{cases} 000 \\ 001 \\ 010 \\ 011 \\ 100 \end{cases}}_{5}$

$C = \underbrace{\begin{cases} 101 \\ 110 \\ 111 \end{cases}}_{3}$

0	101	000	5	110	000	10	111	000
1	101	001	6	110	001	11	111	001
2	101	010	7	110	010	12	111	010
3	101	011	8	110	011	13	111	011
4	101	100	9	110	100	14	111	100

$(8)_{10} = (110011)_{\text{S-C code}}$

decode = 111011

$b = 3$

$$\begin{array}{c|c} c & s \\ 111 & 011 \end{array} = 13$$

S-C dense code   definire le prime 8 codeword

$s = 3$

$c = 1$

$s + c = 4$ $2^b = 4$

$b = \log_2 4 = 2$

$S = \left.\begin{matrix} 00 \\ 01 \\ 10 \end{matrix}\right\} 3$

$C = 11 \} 1$

	$\boxed{S}$		$\boxed{C}\boxed{S}$		$\boxed{C}\boxed{C}\boxed{S}$
0	00	3	11 00	6	11 11 00
1	01	4	11 01	7	11 11 01
2	10	5	11 10	8	11 11 10

Show the (1, 3) codewords of the integer 8 over 2 bits "words" => using 2 bits

$s = 1$ $S = \{00\}$ $s + c = 4$

$c = 3$ $C = \left\{\begin{matrix} 01 \\ 10 \\ 11 \end{matrix}\right\}$ $2^b = 4$

$\log_2 4 = 2$ bits ✓

$\boxed{S}$		$\boxed{C}\boxed{S}$		$\boxed{C}\boxed{C}\boxed{S}$
(0) 00	(1)	01 00	(4)	01 01 00
	(2)	10 00	(5)	01 10 00
	(3)	11 00	(6)	01 11 00
			(7)	10 01 00
			(8) =>	10 10 00
			(9)	10 11 00

↓ s integers encoded with 1 word ↓ group of 2 bits

↓ c · s integers encoded with 2 words, from 1, 3

↓ $c^2 \cdot s$ integers encoded with 3 cw from 4 to $4 + c^2 \cdot s - 1 =$
$= 4 + 9 \cdot 1 - 1 =$
12

✓ ok by Unciprene

• How many bits for the 36TH codeword?

from 12 to $13 + c^3 \cdot \overset{=1}{s} = 3^3 \cdot 1 = 27$

=> ok range $c_2\, c_2\, c_2\, s_2$ => 8 bits $13 + 27 - 1 = 39$

S-C code

$S = 3 \; ; \; C = 1$

=> Encode of 7

$C = 11 \qquad S = \{00, 01, 10\}$

$\boxed{S} \qquad \boxed{C \mid S} \qquad \boxed{C \mid C \mid S}$

0	00	3	11 00	6	11 11 00
1	01	4	11 01	7	11 11 01 ←
2	10	5	11 10		11 11 10

$S + C = 3 + 1 = 4$

$2^b = 4$

$\Rightarrow b = \log_2 4 = 2$

=> total configs:

$\{\underbrace{00, 01, 10}_{S}, \underbrace{11}_{C}\}$

INTERPOLATIVE CODE

Monotonic increasing sequence

$S = \{ s_1, s_2, s_3, \dots, s_n \}$

$\rightarrow$ S'

$S' : s'_i = \sum_{j=1}^{i} s_j$

$S = \{ 1, 3, 1, 2, \dots \}$ $\xrightarrow{\text{copy the } 1^{st}}$ $S' = \{ 1, 2, 5, 6, 8 \dots \}$

$1+1 \quad 2+3 \quad 5+1 \quad 6+2$

START

$l = 1$

$R = n$

$low = S'_1$

$high = S'_n$

GENERIC STEP

$m = \lfloor \frac{l+r}{2} \rfloor$

COMPRESS S'_m (the middle) given $< l, r, low, high >$

$\rightarrow low \leq S'_m \leq high$

COMPUTE $0 \leq S'_m - low \leq high - low$

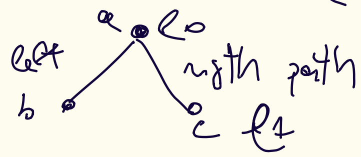
and compress in the # of bits $= \lceil \log_2 (high - low + 1) \rceil$

es: one level of recursion of interpolative code:

$S = \{ 11, 14, 16, \boxed{19}, 20, 21, 22 \}$ => already monotonic increasing

$1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7$

root of rec. tree



=> eventually encode 3 integers: l_0, l_1 of rec. tree

a) $l = 1 \quad r = 7 \quad low = 11 \quad high = 22$

$m = \lfloor \frac{l+r}{2} \rfloor = \lfloor \frac{1+7}{2} \rfloor = 4 \qquad S[m] = 19$

• $S[m] = 19$ lives in the range of integers: $[low + m - \ell;\ high - R + m]$
$11 + 4 - 1;\ 22 - 7 + 4$
$\Rightarrow [14;\ 19]$

encode $S[m] - 14 = 19 - 14 = 5$

in |range| where the middle falls:
$\lceil \log_2 (19 - 14 + 1) \rceil = \lceil \log_2 6 \rceil = 3$ bits

OUTPUT $(5)_{10} = (101)_2$

go left in the recursion

b) $\ell = 1$ $R = 3$ low $= 11$ high $= S[\text{previous } m] - 1 =$
$= S[4] - 1 = 19 - 1 = 18$

$m = \lfloor \frac{\ell + r}{2} \rfloor = \lfloor \frac{4}{2} \rfloor = 2$

$S[m] = S[2] = 14$ 14 lives in range $[low - \ell + m;\ high - R + m] =$
$= [11 - 1 + 2;\ 18 - 3 + 2] =$
$= [12;\ 17]$

• encode $S[m] - 12 = 14 - 12 = 2$

with # of bits = |interval| where $S[m]$ lives

$\Rightarrow \lceil \log_2 17 - 12 + 1 \rceil = \lceil \log_2 6 \rceil = 3$

OUTPUT $(2)_{10} = (010)_2$

---

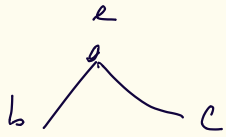
c) $\ell = 5$ low $= S[\text{prev. } m] + 1 = S[4] + 1 = 19 + 1 = 20$
$R = 7$ high $= 22$
$m = \lfloor \frac{\ell + r}{2} \rfloor = \lfloor \frac{5 + 7}{2} \rfloor = \frac{12}{2} = 6$
$S[m] = S[6] = 21 \rightarrow$ lives in $[low - \ell + m;\ high - r + m]$
$20 - 5 + 6;\ 22 - 7 + 6$
$[21;\ 21]$

encode $S[m] - 21 = 21 - 21 = 0$

using # bits $= \lceil \log_2 21 - 21 + 1 \rceil = \lceil \log_2 1 \rceil = 0$ bits

OUTPUT = "" empty string

Do Interpolative, 1 level of e



$S = \{11, 14, 16, \boxed{19}, 20, 21, 22\}$

$\quad\;\; 1 \quad\; 2 \quad\; 3 \quad\; 4 \quad\; 5 \quad\; 6 \quad\; 7$

$l = 1 \qquad low = 11 \qquad m = \left\lfloor \frac{l+r}{2} \right\rfloor = \left\lfloor \frac{1+7}{2} \right\rfloor = 4$

$r = 7 \qquad high = 22$

range $= [low - l + m;\ high - r + m) =$

$= [11 - 1 + 4;\ 22 - 7 + 4) =$

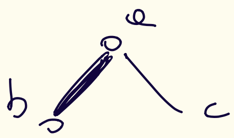
$= [14;\ 19]$

aka a

encode $S[m] - 14 = 19 - 14 = 5$

in $\lceil \log_2 \overbrace{19 - 14 + 1}^{\text{size of interval}} \rceil = \lceil \log_2 6 \rceil = 3$ bits

$\boxed{a = (5)_{10} = (101)_2}$



go left so $\Rightarrow$ $l = 1 \qquad r = 3$

$low = 11 \qquad high = (S[\text{old } m] - 1) =$

$= 18$

$m = \left\lfloor \log_2 \frac{3+1}{2} \right\rfloor = 2$

$S[m] = 14$

RANGE$[low - l + m;\ high - r + m) =$

$= [11 - 1 + 2;\ 18 + 2 - 3]$

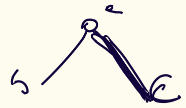
$= [12;\ 17]$

encode

$S[m] - 12 = 14 - 12 = 2$

in $\lceil \log_2 17 - 12 + 1 \rceil = \lceil \log_2 6 \rceil = 3$ bits

$\boxed{(2)_{10} = (010)_2}$

c)  $k=5$ $r=7$ $high=22$

$low = \text{Old } S[m]+1 = 19+1 = 20$

$m = \lfloor \frac{12}{2} \rfloor = 6$ $[20-5+6 ; 22-7+6] = [21 ; 21]$

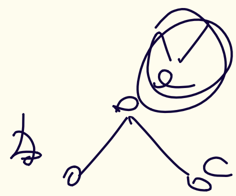
encode

$S[m] = 21-21 = 0$

in $\lceil \log_2 0 \rceil = 1$

$\Rightarrow 0 \Rightarrow$ empty string

One level of recursion of interpolative code



$S = (1, 2, 3, \boxed{4}, 6, 8, 9)$
$\quad\;\; 1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6 \;\; 7$

$l=1$ $r=7$
$low=1$ $high=9$ $m = \left\lfloor \frac{l+r}{2} \right\rfloor = \frac{8}{2} = 4$

$S[m] = 4$ 4 lies in range $[1-1+4 ; 9-7+4] = [4, 6]$

encode $4-4=0$ in

$\lceil \log_2 6-4+1 \rceil = \lceil \log_2 3 \rceil = 2$ bits

$\boxed{(0)_{10} = (00)_{10}}$

b) $l=1$ $r=3$
$low=1$ $high = \text{old } S[m]-1 = 4-1=3$ $m = \lfloor \frac{4}{2} \rfloor = 2$

$S[m]=2$ lies in range $[1-1+2 ; 3-3+2] = [2,2]$

$\Rightarrow$ NO BITS OUTPUTTED

c) $l = 5$ $\qquad r = 7$

$low = 4+1 = 5$ $\qquad high = 8$

$m = \left\lfloor \frac{12}{2} \right\rfloor = 6$ $\qquad S[m] = 8$

lies in range

$[5-5+6 ; 8-7+6] =$

$[6 ; 7]$

encode $S[m] - low' =$

$= 8 - 6 = 2$

in $\lceil \log_2 8 - 6 + 1 \rceil$ bits $= \lceil \log_2 3 \rceil = 2$ bits

$(2)_{10} = 10$

ELIAS - FANO CODE ⇒ Monotonic increasing sequence

- n items (integers) $[0, u)$ • $u = S'_m + 1$
- $b = \lceil \log_2 u \rceil$ ⇒ can encode the n integers $\in [0, u)$ with b bits
- $\ell = \lceil \log_2 \frac{u}{m} \rceil$ (least significant bits)
- $h = b - \ell$ (most significant bits)
- $h = \lceil \log_2 m \rceil$

⇒ we compress h not ℓ of each integer

$$S' = \{1, 4, 7, 18, 24, 26, 30, 31\}$$

unary(5) = 000001
¬ UNARY(5) = 111110

- $m = 8$
- $u = S'_m + 1 = 31 + 1 = 32$
- $b = \lceil \log_2 u \rceil = \lceil \log_2 32 \rceil = 5$ bits
- $h = \lceil \log_2 m \rceil = \lceil \log_2 8 \rceil = 3$ bits
- $\ell = b - h = 5 - 3 = 2$

- ARRAY L: CONCATENATE the low part
- ARRAY H: negative UNARY
  ↳ HOW MANY TIME I SEE A CONFIGURATION

S'_i	h	ℓ
1	000	01
4	001	00
7	001	11
18	100	10
24	110	00
26	110	10
30	111	10
31	111	11

L = 01 00 11 10 00 10 10 11

compressed → H =

H =	10	110	0	0	10	0	110	110
bucket:	0	1	2	3	4	5	6	7

elias-fano OUTPUT: L, H

+ DS supporting the operation:

return the position of the i^{th} zero in H ← $select_0(i, H)$

$select_1(i, H)$

# of buckets $= \frac{u}{2^\ell}$

• in each bucket I've 2^ℓ integers, cut in each bucket I've ℓ bits:

$$\begin{array}{c|cccc|cccc|cc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \dots \quad u \\
\ell & 00 & 01 & 10 & 11 & 00 & 01 & 10 & 11 & 00 & \dots
\end{array}$$

$\underbrace{00\ 01\ 10\ 11}_{2^\ell}$

• in H: $\#1 = n$

$\#0 = \#$ buckets

⇒ I'm writing 0 at the end of each bucket

• length of H $= |H| = \#_1 + \#_0 = n + \#\text{buckets} = n + \frac{u}{2^\ell} = n + \frac{u}{2^{\lceil \log \frac{u}{n} \rceil}}$

but $\ell = \lceil \log \frac{u}{n} \rceil$

$$\Rightarrow \#0 \le \frac{u}{2^{\lceil \log_2 \frac{u}{n} \rceil}} = \frac{u}{\left(\frac{u}{n}\right)} = n$$

$$\Rightarrow |H| \le 2n$$

• length of L $= |L| = n \cdot \ell = n \cdot \log_2 \frac{u}{n}$

$\ell = \lceil \log_2 \frac{u}{n} \rceil$

• TOTAL SPACE $|H| + |L| = 2n + n \lceil \log_2 \frac{u}{n} \rceil + o(n)$

for select DS

NEXT-GEQ(X) : returns the smallest integer in the sequence $\geq X$

es. next-geq(17) = 18

s_i	h	l
1	0 0 0	0 1
4	0 0 1	0 0
7	0 0 1	1 1
18	1 0 0	1 0
24	1 1 0	0 0
26	1 1 0	1 0
30	1 1 1	1 0
31	1 1 1	1 1

L = 01 00 11 10 00 10 10 11

compressed → H = 10 | 110 | 0 | 0 | 10 | 0 | 110 | 110

bucket: 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

next_geq (20)

IDEA: write 20 in binary using b-bits = 5

1 0 1|0 0

h(20) = 101 use h ⇒ $(101)_2 = (5)_{10}$

l(20) = 00

- if 20 is in our sequence, it will belong to the 5^{TH} bucket

BUT if not, next integer will be in bucket 5, 6, 7...

(or in a following bucket 5)

In general:

- given X
- compute h(x) and l(x)
- look into bucket h(x):

1) Non-empty: ⇒ go to L to look for the l(x) of integers in Bucket h(x)

- Compare these integer with the l(x)

⇒ if they match ⇒ nextgeq(x) is x itself

if I'm look for 10

will be here

next geq's l(x)

L: 00 01 11

- if they don't match => look next pair

• if all bits in L are smaller than $\ell(x)$ => value will be at the end of the bucket => next non-empty bucket

2) $h(x)$ empty: next_geq(x) will be the 1st value after that bucket

next_geq(20)

$(20)_{10} = 101|00$

$h(x): 101 \longrightarrow 5$

$\ell(x): 00$

↳ look for 5TH zero in H

L = 01 00 11 10 00 10 10 11

compressed => H = 10 | 110 | 0 | 0 | 10 | 0 | 110 | 110

bucket:	0	1	2	3	4	5	6	7
H =	10	110	0	0	10	0	110	110

$\text{select}_0(5)$ = ending position of bucket 4 = 9 (pos)

$\text{select}_0(5+1)$ = ending position of bucket 5 = 10 (pos)

everything that is between these 2 positions will be the integers in bucket 5

=> since they're consecutive => no elements in bucket 5 => empty

=> REACH FOR NEXT VALUE after the end of bucket 5 (1 immediately after bucket 5)

=> count #1 before bucket 5 => 4 and the answer will be the 1 in bucket 6

count 1 before

(*) => $\text{select}(5+1) = 10 - 6 = 4$ → # of 1 up to 6TH zero

10: tot # of bits up to 6TH 0

6: # of 0 up to the 6TH 0

$\Rightarrow h$ number of 1 up to bucket 5

$\Rightarrow h+1=5$ is the 1 in H corresponding to next page (20)

. ENCODE with Elias-Fano:

$$S' = \{1, 3, 4, 5, 9, 16, 23, 27, 28, 31, 40\}$$

and show how to answer:

1) access (4)
2) next-geq (8)
3) next-geq (32

$n = 11$

$u = 41$

$b = \lceil \log_2 u \rceil = \lceil \log_2 41 \rceil = 6$

$h = \lceil \log_2 n \rceil = \lceil \log_2 11 \rceil = 4$

$l = 6 - 4 = 2$

S'_i	h	l
1	0000	01
3	0000	11
4	0001	00
5	0001	01
9	0010	01
16	0100	00
23	0101	11
27	0110	11
28	0111	00
31	0111	11
40	1010	00

For H we have 2^h buckets
$2^4 = 16$ buckets

L = 01 11 00 01 01 00 11 11 00 11 00

H =	110	110	10	0	10	10	10	110	0	0	10	00000
	0	1	2	3	4	5	6	7	8	9	10	11 12 13 14 15

access(4): 1) 4^{TH} groups of zeros in L = 01

2) search for 4^{TH} 1 in H

and make the index of that

$select_1(4) - 4 =$ $\Leftarrow$ 1 - value I'm looking for (4)

$= 5 - 4 = 1$ $\Rightarrow 5 - 4 = 1$

$bin(1) = 0001$

$\Rightarrow access(4) = (000101)_2 = 1 + 4 = 5$ ✓

2) Next_geq(8)

$(8)_{10} = \overbrace{0\,0\,1\,0}^{h} \mid \overbrace{0\,0}^{\ell}$

$\to 00$

$(0010)_2 = 2$

L = 01 11 00 01 01 00 11 11 00 11 00

H = 110 110 10 0 10 10 10 110 0 0 10 0

0 1 2 3 4 5 6 7 8 9 10 11

$\Downarrow$

look in bucket 2

STARTING position in H of the BUCKET 2

$p = select_0(2) + 1 = 6 + 1 = 7$

$\Rightarrow$ BUCKET HAS 1 ELEMENT

$H[p] = 1$ $\Rightarrow$ bucket's not empty

$\Rightarrow$ SCAN ITS elements starting from position

ℓ: lower part: $p - H(8) =$

$= p - 2 =$

$= 7 - 2 = 5$

$\Downarrow$

a 5^{TH} GROUP in $L \Rightarrow$ 01

00 10

Return = 9

$= (0010 01) = 9$

$\text{next\_geq}(32) = \text{next\_geq}\Big(\big(\underbrace{100\,0}_{h}\,|\,\underbrace{0\,0}_{\ell}\big)_2\Big)$

$\Downarrow$

8^{TH} bucket

$L = 01\ 11\ 00\ 01\ 01\ 00\ 11\ 11\ 00\ 11\ \overset{11}{00}$

$H = 110\ 110\ 10\ 0\ 10\ 10\ 10\ 110\ 0\ 0\ 10\ 0\ 0\ 0\ 0\ 0$

$\quad\ \ 0\quad 1\quad 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10\ 11\ 12\ 13\ 14\ 15$

$19 \qquad 21$

$\Rightarrow 8^{TH}$ bucket's empty

$p = \text{select}_0(8) + 1 = 18 + 1 = 19$

$H[19] = 0$

$\Rightarrow$ scan until you find a non-empty bucket

$\Rightarrow$ return next element which is:

$\text{access}(p - H(32)) = \text{access}(19 - 8) =$

$= \text{access}(11) =$

$21 - 11 = 10$

by encoding 0 in $h = \lceil \log_2 n \rceil = 3$ bits as $H(x_5) = 110$. We have thus obtained $x_5 = 110 \cdot 00 = 24$.

NextGEQ(x) is implemented as follows. First we observe that $\mathsf{Select}_0(H(x))$ gives the position (counting from 1) of the bit 0 ending the negative unary sequence representing $H(x) - 1$ (we assume that the position is 0 if $H(x) = 0$). Thus, $p = \mathsf{Select}_0(H(x)) + 1$ is the starting position in array H of the negative unary sequence representing $H(x)$ and thus of the elements whose highest bits are equal to $H(x)$ (if any). If the bit $H[p] = 0$, then it is empty the bucket containing elements in S' with the same highest bits as x, and thus we have to take the first element of the next non empty bucket (which has highest bits larger than $H(x)$): this is the element of rank $p - H(x)$ in S', which may be retrieved with $\mathsf{Access}(p - H(x))$. Otherwise the bucket is non empty, its elements have the same highest bits as x, and hence the element answering NextGEQ(x) is identified by scanning the elements at position $p - H(x)$ and beyond, by comparing the corresponding lowest bits in L until an element y with $L(y) \geq L(x)$ is found in the bucket of x or the next bucket is reached (hence a bit 0 in H is passed over). And in this latter case, the first element of the next bucket is returned being surely larger than x. These elements are $2^\ell = \Theta(u/n)$, so their scan would take $O(u/n)$ time in the worst case. We could speed up this search by performing a binary search over that bucket, thus taking $O(\log(u/n))$ time.

nextgeq(x) $\qquad p = \text{Select}_0(H(x)) + 1$

if $H[p] = 0$ $\rightarrow$ empty bucket

$\Rightarrow$ go to first element of an non-empty bucket

retrieve $\text{Access}(p - H(x))$

else (bucket's not empty)

scan element at position $p - H(x)$

and compare the corresponding lowest bit in L until you get all y with

$L(y) \geq L(x)$

S_i	h	l
1	0000	01
3	0000	11
4	0001	00
5	0001	01
9	0010	01
16	0100	00
23	0101	11
27	0110	11
28	0111	00
31	0111	11
40	1010	00

L = 01 11 00 01 01 00 11 11 00 11 00

H = 110 110 10 0 10 10 10 110 0 0 10 00000

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

ALGO NEXT_GEQ

$p = select_0(H(x)) + 1$

$i = p - (h \text{ di } x)$ → h del numero da cercare in sinistra)

ripeti

$y = access(i)$

$i{+}{+}$

finché $(y < x)$ && $(i < m)$ → h piccolo di x → se si verifica, esco

ENCODE WITH ELIAS-FANO:

$$S = (1, 4, 7, 18, 24, 26, 30, 31)$$

$u = 31 + 1 = 32$

$n = 8$

$b = \lceil \log_2 u \rceil = \lceil \log_2 32 \rceil = 5 \Rightarrow 5$ bits

$h = \lceil \log_2 n \rceil = \lceil \log_2 8 \rceil = 3$ bits

$\ell = 5 - h = 5 - 3 = 2$ bits

For H we have 2^h buckets $\Rightarrow$ $2^3 = 8$

s_i	h	ℓ
1	000	01
4	001	00
7	001	11
18	100	10
24	110	00
26	110	10
30	111	10
31	111	11

L = 01 00 11 10 00 10 10 11

H =	10	110	0	0	10	0	110	110
	0	1	2	3	4	5	6	7

$(25)_{10} = \underbrace{1\,1\,0}_{h\,=\,5}\ \underbrace{0\,1}_{\ell}$

next_geq(25)

$p = select_0(H(5)) + 1 =$

$= 9 + 1 = 10$

if $H[p] == 0 \Rightarrow$ return Access$(p - H(x))$

else $i = p - H(x)$

jump to the i-th group of 1 bits in L

and scan comparing with L(x)

$2^h = 2^3 = 8$

S'_i		
1	000	01
4	001	00
7	001	11
18	100	10
24	110	00
26	110	10
30	111	10
31	111	11

L = 01 00 11 10 00 10 10 11

H = 10 110 0 0 10 0 110 110
0 1 2 3 4 5 6 7

$(29)_{10} = 111|01$, $111 = 7$

next-geq(29)

$p = select_0(7) + 1 = 13 + 1 = 14$

$i = 14 - 7 = 7$

go to 7^{TH} pos in L = 10
and compare with $\ell(x) = 01$

$10 > 01$

$\Rightarrow$ return 10

$\Rightarrow (11110)_2 = 30$

next-geq(8)

$0\,10|00$ (h: 010 $= 2$, ℓ: 00)

$p = select_0(2) + 1 = 6$

$i = 6 - 2 = 4$

repeat
$y = access(i)$
$i{+}{+}$
finché $(y \geq x)$

$\Rightarrow access(4) = \underbrace{100}_{select_1(x) - x = 8 - 4 = 4}\ \underbrace{10}_{4^{TH}\text{ pos in L}}$

✓ $(10010)_2 = (18)_{10}$

next-geq(30)

$(30)_{10} = 111|10$, $111 = 7$

$P = select_0(7) + 1 = 13 + 1 = 14$

$i = 14 - 7 = 7$

repeat
$y = access(i) = access(7) \Rightarrow$
$i{+}{+}$
finché $(y \geq x)$

$access(7) = \underbrace{\quad}_{select_1(7) - 7 = 14 -}\ \underbrace{10}$

$\Rightarrow i{+}{+}$

$access(8) = \underbrace{111}_{select_1(8) - 8 = 15 - 8 = 7}\ \underbrace{11}$

$\Rightarrow$ 31 ok ✓

next-geq(19) $\qquad (19)_2 = \underset{\overset{\shortparallel}{4}}{100}|11$

$p = select_0(4) + 1 = 7 + 1 = 8$

$i = 8 - 4 = 4$

ripeti $y = access(i)$ $\Rightarrow$ $access(4) = 18$ misb(?)

$i{+}{+}$

finche $(y \geq X)$ $\qquad$ $i{+}{+}$ $\quad access(5) = \underbrace{110}\ \underbrace{00} = 24$ ✓

$select_1(5) - 5 =$

$= 11 - 5 = 6$

next-geq(7) $\qquad (7) = \underset{\overset{\shortparallel}{1}}{001}|11$

$p = select_0(1) + 1 = 3 = 2 + 1$

$i = 3 - 1 = 2$

ripeti

$y = access(i)$ $\Rightarrow access(2) = 4 \Rightarrow$ misba

$i{+}{+}$

finche $(y \geq x)$ $\qquad \Rightarrow i{+}{+}$

$\Rightarrow access(3) = \underbrace{001}\ \underbrace{11}$ $\quad$ res = 7

$\cancel{\ }$ $\quad$ 3 ns

$select_1(3) - 3 =$ $\quad$ 1 ... 7 in L

$= 4 - 3 = 1 = 1$

MUTUAL PARTITIONING

DOUBLING SEARCH

trie

Elias - Fano: $n = 8$ $u = 31 + 1 = 32$ $b = \lceil \log_2 32 \rceil = 5$

$h = \lceil \log_2 n \rceil = \lceil \log_2 8 \rceil = 3$

$\ell = 5 - 3 = 2$

array H => $2^3 = 8$ buckets (posizioni)

S_i	h	e
11	0 1 0	1 1
14	0 1 1	1 0
16	1 0 0	0 0
17	1 0 0	0 1
19	1 0 0	1 1
20	1 0 1	0 0
21	1 0 1	0 1
31	1 1 1	1 1

L = 11 10 00 01 11 00 01 11

10 => my # will be in the next bucket

11 → this # gives me 2 uno => still same bucket

H = 0 0 10 10 1110 110 0 10
0 1 2 3 4 5 6 7

for high part ⇓ access (8)

access(4):

$select_1(4) - 4 = 8 - 4 = 4$ => $\underbrace{100}\ \underbrace{01}_{4^{TH} \text{ pair in } L} = 2^4 + 1 = 17$ ✓

next_geq(18) $(18)_{10} = \underbrace{100}_{H}\ \underbrace{10}_{L}$

GIUSTA NOBRA

$p = select_0(H(x)) + 1 = select_0(4) + 1 = 6 + 1 = 7$

$i = p - H(x) = 7 - 4 = 3$ => 3 uno => go to L and jump to the 3rd item => 00

ripeti
$S = access(i)$
$i{+}{+}$
finché $(y \geq x)$

compare until you find in L a couple $\geq$ di 10

=> $(10011)_2 = 19$ ✓

retrieve 11

next_geq (15) $\quad (15)_{10} = 0\ 1\ 1 \mid 1\ 1$ $\quad \Downarrow 3$

$15 = 8 + 4 + 2 + 1$

$p = \text{select}_0(3) + 1 = 4 + 1 = 5$

$i = p - H(x) = 5 - 3 = 2$

end of L, number is < 15 $\Rightarrow$ need to advance

$\Rightarrow$ bucket $4 \Rightarrow 100\ 00$ (end $L = 00$)

$= 16$

---

next_geq(22) $\quad (22)_2 = (1\ 0\ 1 \mid 1\ 0)_2$

$= 5$ bucket

$p = \text{select}_0(5) + 1 = 10 + 1 = 11$ → Mi dice nel bucket che dus andare

$i = 11 - 5 = 6$

Jump to the 6^{TH} number, the 1^{ST} one of the bucket interested in (6^{TH} PAIR in L)

L =	11	10	00	01	11	00	01	11
H =	0	0	10	10	1110	110	0	10

next_geq (15) $\quad (15)_{10} = 0\ 1\ 1 \mid 1\ 1$ $\quad = 3$

15	1
7	1
3	1
1	1
0	

$p = \text{select}_0(3) + 1 = 4 + 1 = 5$

5^{TH} GROUP

$11 = 11 \Rightarrow$ number will be in the next bucket

$h \Rightarrow (5)_2 = 101$

$\ell \Rightarrow 00 = 00$

$\Rightarrow (10100)_2 = 2^2 + 2^4 = 16 + 4 = 20$

yes!!

given blind:

$L = 0111\,0001\,0100\,1111\,0011\,00$

$H = 110110100101010111000100000$

$|H|$: 16 buckets

Find: ℓ, h, n?

$n = \#$ of 1 in $H = 11$

$h = \lceil \log_2 n \rceil \Rightarrow h = \lceil \log_2 11 \rceil$ $\quad h = 4$

$\ell = \dfrac{|L|}{n} = \dfrac{22}{11} = 2$

$\mathrm{acen}(8) = $

$\nearrow$ $L = 11$

$\searrow$ 0110

$H = \mathrm{select}_1(8) - 8 = 14 - 8 = (6)_{10} = 0110$

$\Leftarrow$

$(011011)_2 = 1 + 2 + 2^3 + 2^4 = 1 + 2 + 8 + 16 = 27$ ✓

~~Do~~ elias - ~~pf~~ fano

ENCODE WITH ELIAS-FANO:

$S := (1, 4, 7, 18, 24, 26, 30, 31)$ $\quad n = 8$

- $u = 31 + 1 = 32$
- $b = \lceil \log_2 32 \rceil = 5$ bits $\quad h = \lceil \log_2 8 \rceil = 3 \quad l = 2$

	h	l
1	000	01
4	001	00
7	001	11
18	100	10
24	110	00
26	110	10
30	111	10
31	111	11

L = 01 00 11 10 00 10 10 11

H = 10 110 0 0 10 0 110 110

0 1 2 3 4 5 6 7

2^h BUCKETS

$2^3 = 8$

$access(4) = select_1(4) - 4 =$

$= 8 - 4 = 4$ in binary

and 4TH groups of L

$(100\ 10)_2 = 2^4 + 2 = 18$

$access(6) = select_1(6) - 6 =$

$= 12 - 6 = 6 \Rightarrow (110\ 10)_2 =$

$= 2^4 + 2^3 + 2 =$

$= 16 + 8 + 2 = 26$ ✓

next-geq(15) =>

8 4 2 1

$(15)_2 = \underbrace{011}_{h}\ \underbrace{1\ 1}_{e}$

3

$p = select_0(3) + 1 = 7$

4TH groups in H cont

others in increasing

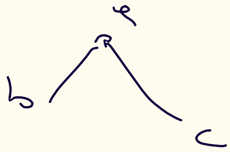
are empty

$(4)_2 = 10010$ & 1ST ~~pri~~ ~~BOF~~ BITS in L cont

two ~~2~~ clean in the next buckets

$S = (1, 2, 4, 6, 7, 8, 9)$
indices: 1 2 3 (4) 5 6 7

Interpolative Coding



$l = 1 \quad low = 1$

$r = 7 \quad high = 9$

a) $m = \left\lfloor \frac{7+1}{2} \right\rfloor = 4 \qquad S[m] = 4$

4 lives in range $[1-1+4;\ 9-7+4]$
$[4;\ 6]$

encode $S[m] - 4 =$
$6 - 4 = 2$

in $\left\lceil \log_2 \underbrace{6-4+1}_{=3} \right\rceil$ bits $= \lceil \log_2 3 \rceil = 2$ bits

(10)

b) $l = 1 \quad low = 1$
$r = 3 \quad high = 6 - 1 = 5$

$m = \left\lfloor \frac{3+1}{2} \right\rfloor = 2 \qquad S[m] = 2$

lives in $[0+2;\ 5-3+2] =$
$= [2;\ 4]$

encode $S[m] - 2 = 0$
in $\lceil \log_2 4-2+1 \rceil =$
$= \lceil \log_2 3 \rceil = 2$ bits

$\Rightarrow 00$

c) $r = 7 \quad high = 9$
$l = 5 \quad low = 6 + 1 = 7$

$m = \left\lfloor \frac{7+5}{2} \right\rfloor = 6$ in range $[7-5+6;\ 9-7+6] =$
$= [8, 8]$
$\Rightarrow$ no bits emitted

$S = (1, 2, 4, 6, 7, 8, 9)$ Elias-Fano

$n = 9 \quad m = 7 \quad$ in $b = \lceil \log_2 9 \rceil = 4$ bits

$h = \lceil \log_2 7 \rceil = 3 \quad \ell = 4 - 3 = 1$

	h	l
1	000	1
2	001	0
4	010	0
6	011	0
7	011	1
8	100	0
9	100	1

$L = 1 \; 0 \; 0 \; 0 \; \underline{1} \; 0 \; \underline{1}$

$2^h = 2^3 = 8$

$$H = \underset{0}{10} \quad \underset{1}{10} \quad \underset{2}{10} \quad \underset{3}{110} \quad \underset{4}{110} \quad \underset{5}{0} \quad \underset{6}{0} \quad \underset{7}{0}$$

$access(3) = select_1(3) - 3 = 5 - 3 = (2)_{10} =$

$0\underbrace{10}_{=}0$

$4 \; pos$

$next\_geq(4) =)$

110

$4+2 = 6$

$(4)_{10} = \underbrace{10}_{h}\underbrace{0}_{\ell}$

$p = select_0(2) + 1 = \quad 2^{\ell}$

$4 + 1 = 5$

$(5)_2 = 101$