1. Reduce N to 1 :→ Min. steps Required to convert N to 1.

Date: 28th January 2022

- ~~Reduce N To 1~~
- Pepcoder And Bits
- ~~Xor Of Sum Of All Pairs~~
- Abbreviation 1 - Using Bits
- Utf - 8 Encoding

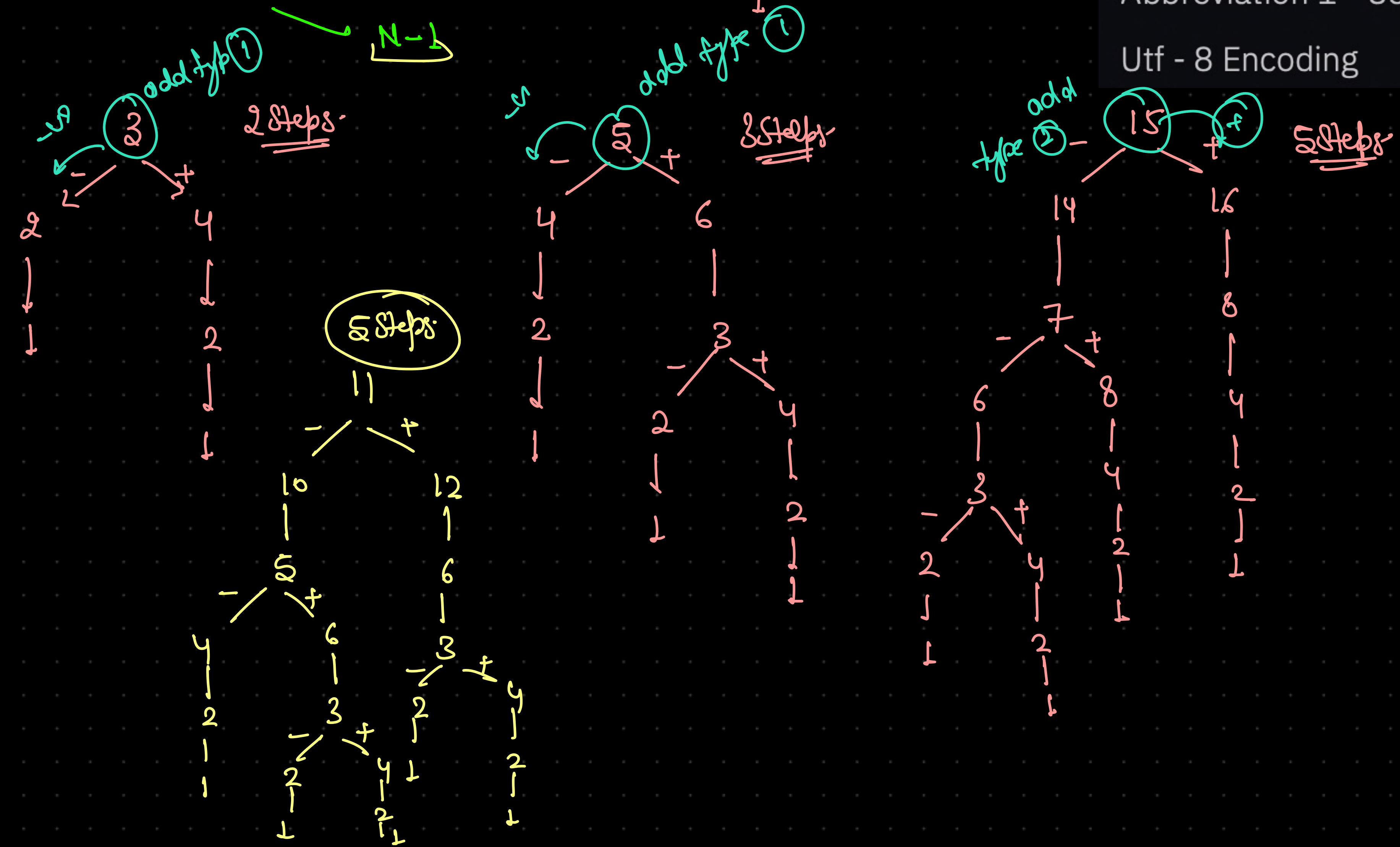
Condition →

N Even → N/2

N odd → N+1
N odd → N-1

4
↓
2
↓
1

2 Steps.



Even → Reduce N to N/2

odd → there are two types of odd.

we have x, and $x+1$

$4x+1$

$-$ $4x$ | $+$ $4x+2$

$2x$ m steps $2x+1$

$-$ $2x$ | $+$ $2x+2$ m+1

$\boxed{x}=15$ ↳ k steps

$x+1$ m+1 steps

x to 1 → k steps

x ↳ k steps

$\boxed{x+1}$ ↳ (k+1) steps

Assumption ①

→ suppose x is advantageous

$-$ve → Beneficial ?
from $4x+1$

~~+ve → Beneficial ?~~
~~from $4x+1$~~

Assumption 2

→ suppose $x+1$ is advantageous.

$-$ve → Benficial ?

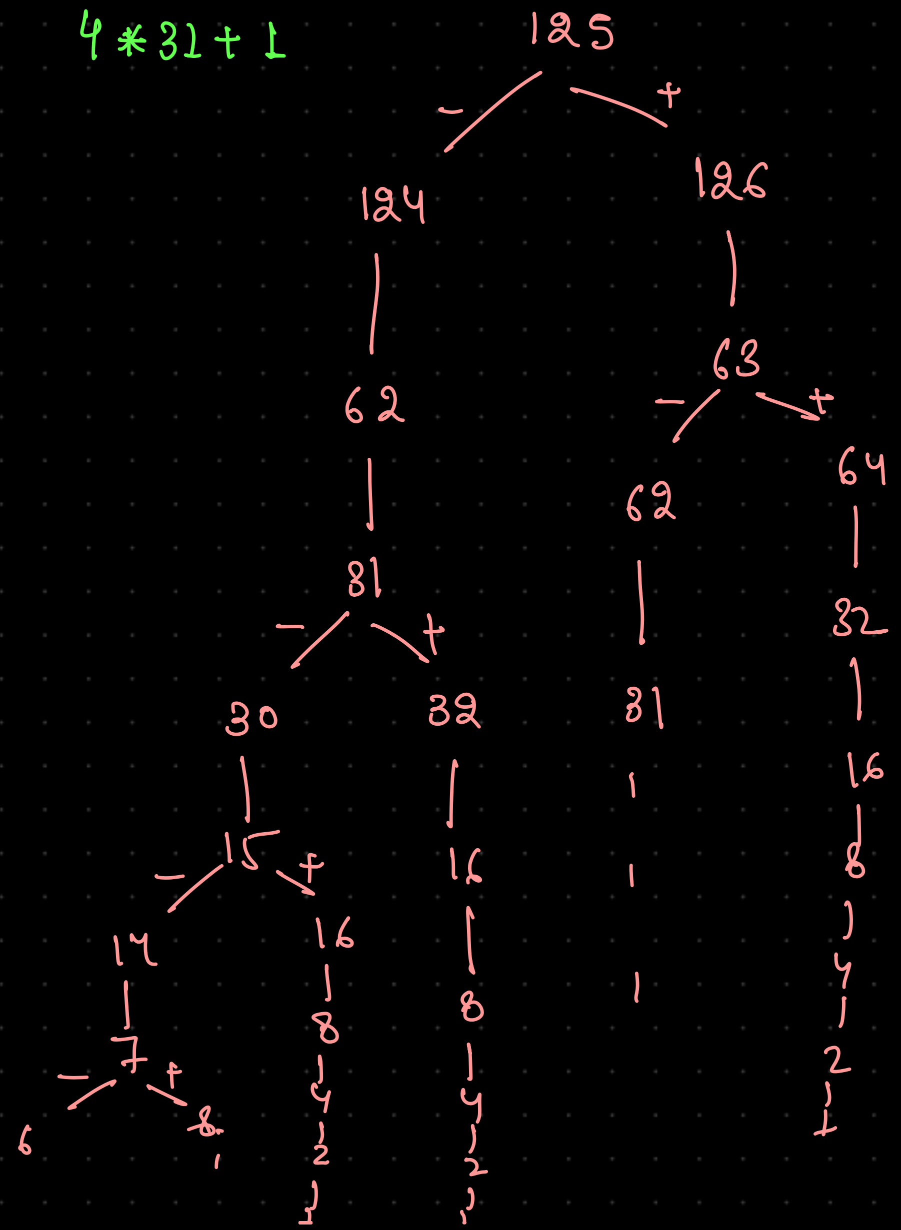
+ve → Beneficial ?

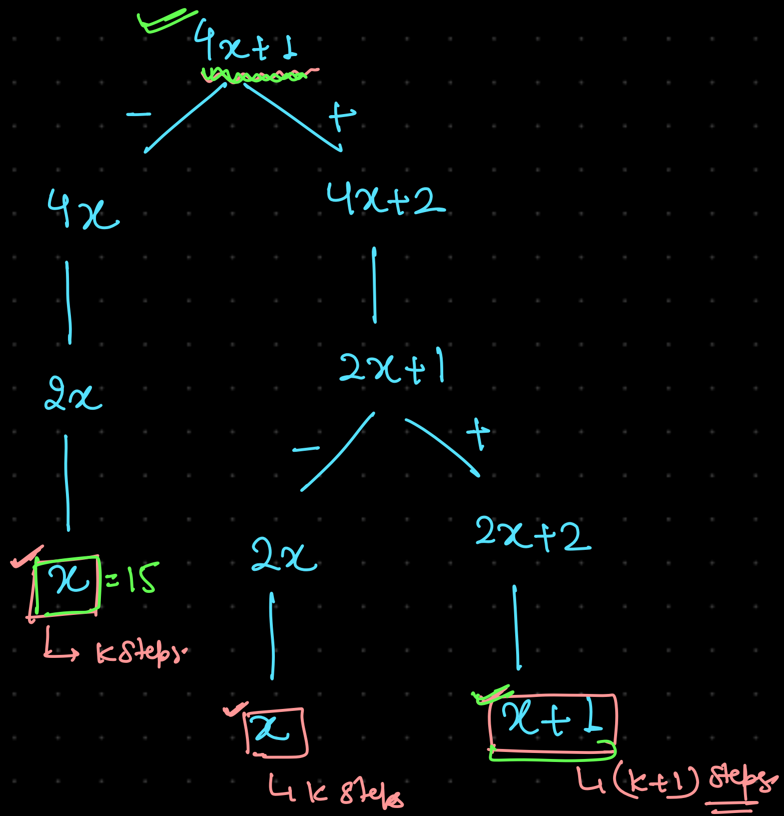
[Both are equal

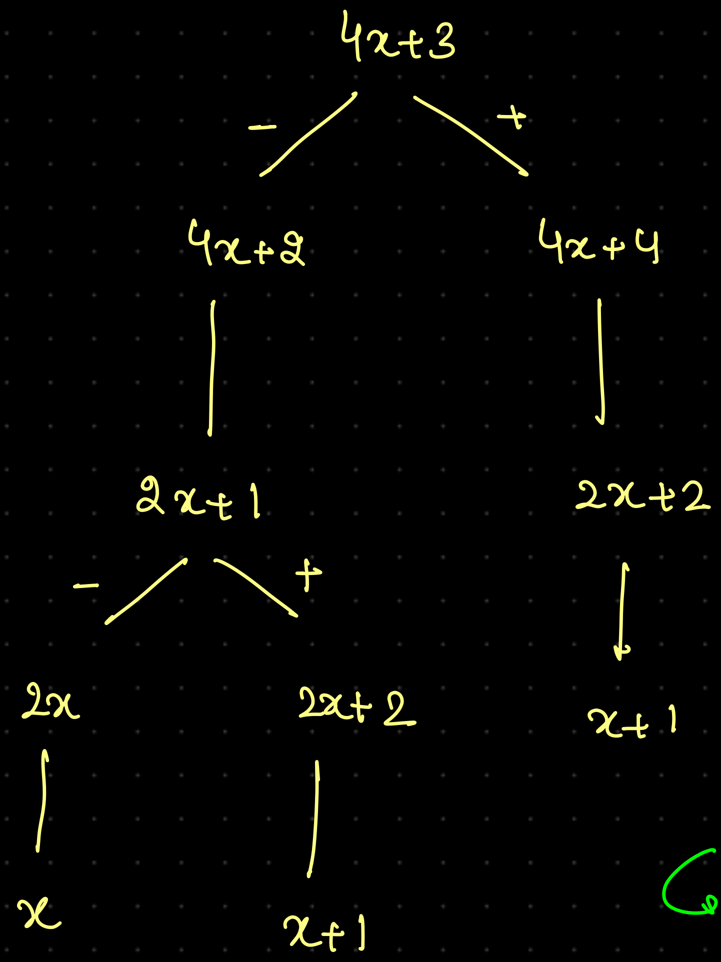
Conclusion from

Both → $\boxed{4x+1}$ → $-$ve is Beneficial

$125 = 4 * 31 + 1$


125
124
126
62
63
31
62
64
30
32
31
32
15
16
16
17
16
8
8
7
8
4
4
6
8
2
2
1
1
1


4x+1
4x
4x+2
2x
2x+1
x =15
↳ k steps.
2x
2x+2
x
↳ k steps
x+1
↳ (k+1) steps



Case I → suppose if $x+1$ is advantageous.

−ve from $4x+3$ is beneficial ?] ×

+ve from $4x+3$ is beneficial ?] → Part 1

case-II - Suppose if x is advantageous

−ve from $4x+3$ is beneficial ?

+ve from $4x+3$ is beneficial ?] Part 2

Both are equal!

from part (1) and (2)

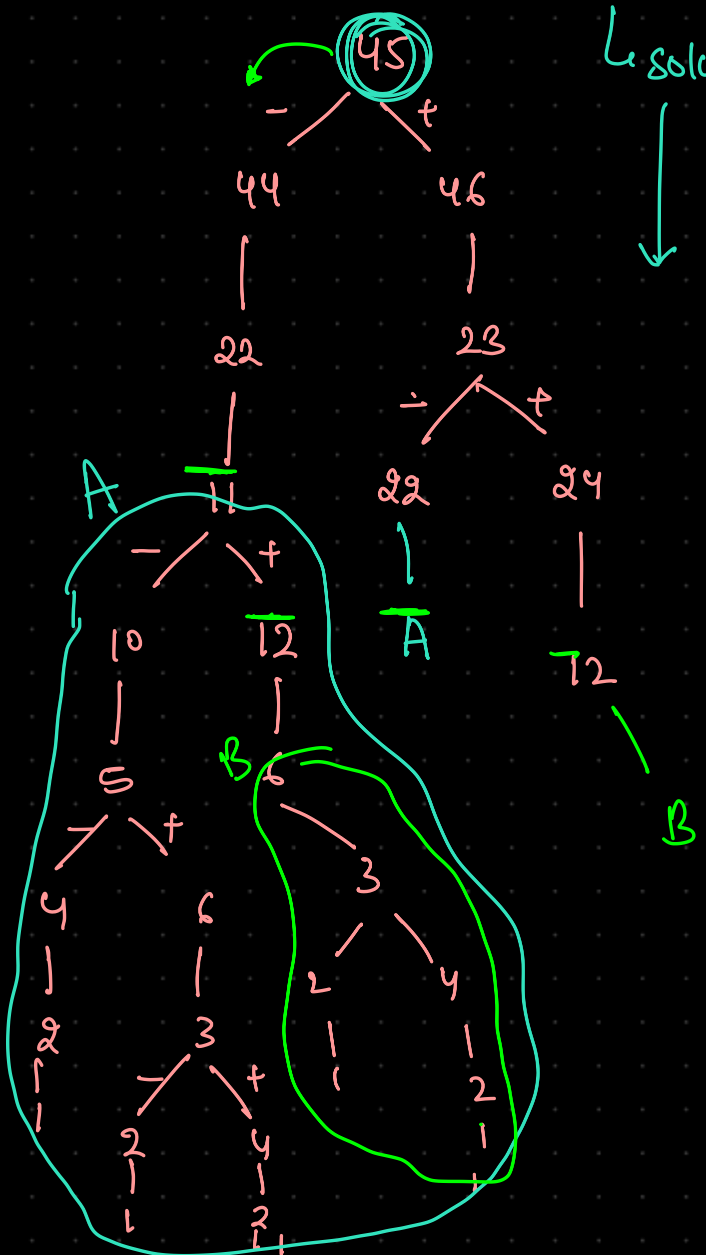
if no. is of $4x+3$ type then +ve is advantageous.

$x+1$ is advantageous $(x+1) \xrightarrow{\text{to}} 1$] less steps required.

n= 45

4*11+1

4x+1



Bit → tools →

Maths

↳ solution(optimise)

Small Range Effect

n==∞

↳ 7

4x+3

∞+1

(+)

↳ −∞

n=3

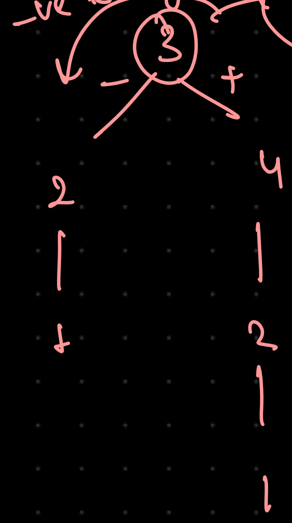
4*0+3

4x+3

Exceptional case.

−ve is beneficial

4x+3



only for 3

45 → 4*11+1

(−)

44

22

11 → 4*2+3

(+)

12

6

3 → 4*0+3

(−) → special case

2

1

8 Steps.

```java
public static int reduceNto1(long n) {
    int count = 0;
    while(n != 1) {
        if(n % 2 == 0) {
            // even
            n = n / 2;
        } else if(n == 3) {
            // special case
            n = n - 1;
        } else if(n % 4 == 1) {
            // odd 1 type
            n = n - 1;
        } else if(n % 4 == 3) {
            // odd 2 type
            n = n + 1;
        }
        count++;
    }
    return count;
}
```

% operator take more time then bits operator.

4x+1 type → ______ 01

binary Representation

if (______01 & 0000011 == 000000 1)

it is of type 4x+1 3 1

if [(num & 3) == 1]

4x+3 type → ______ 11

binary Representation

if (______11 & 000011 == 000000 11)

it is of type 4x+3 3 3

if ((num & 3) == 3)

XOR of sum of all pairs :→

array → a b c d e

XOR sum of all pairs →

$a+a$	$\cancel{b+a}$	$\cancel{c+a}$	$\cancel{d+a}$	$\cancel{e+a}$
$\cancel{a+b}$	$b+b$	$\cancel{c+b}$	$\cancel{d+b}$	$\cancel{e+b}$
$\cancel{a+c}$	$\cancel{b+c}$	$c+c$	$\cancel{d+c}$	$\cancel{e+c}$
$\cancel{a+d}$	$\cancel{b+d}$	$\cancel{c+d}$	$d+d$	$\cancel{e+d}$
$\cancel{a+e}$	$\cancel{b+e}$	$\cancel{c+e}$	$\cancel{d+e}$	$e+e$

XOR of sum of all pairs - → $O(n^2)$

Allowed → $O(n)$

$$XOR \rightarrow \underbrace{2a \wedge 2b \wedge 2c \wedge 2d}\wedge 2e$$

$$\rightarrow \underbrace{2*(a \wedge b \wedge c \wedge d \wedge e)}$$

Pepcoder and Bits: →

we have a number N, count of +ve integer which are less than N and have same no. of bit is ON.

N = 12 → 1100

no. which have 2-ON bit

Smaller than 12

1 →	0001
2 →	0010
3 →	0011
4 →	0100
5 →	0101
6 →	0110
7 →	0111
8 →	1000
9 →	1001
10 →	1010
11 →	1011

→ count = 5.

→ Nos are strictly less than N.

→ Same no. of sets bits are present.

eg → 1 0 0 1 0 1 1 1

no. of set bits → 5

23 → no. of ways

1 0 0 1 0 1 1 1

→ OFF 0 $\underbrace{- - - - - - -}$

21

5 bits out of 7 bits

$^7C_5 = \frac{7 \times 6 \times 5!}{5!\, 2!} = 21$

total no. of ways such that 5 bits are ON out of 7 → $^7C_5$

2

→ ON

I 2 1 0 0 1 0 1 1 1

Required ON bits 4

off 0 0 1 0 1 1 1

2

off 0 1 0 1 1 1

2 1 0 1 1 1

1

→ OFF 0 $\underbrace{- - - -}$

total no. of ways such that 4 bits are ON out of 4 → $^4C_4 = 1$

1

→ ON → 1 0 1 1 1

II

1

off 0 1 1 1

Required bit

1 1 1 0

→ OFF → 0 - -

total no. of ways such that 3 bits are ON out of 2 bit $\cancel{^2C_3} = 0$

1

→ ON → 1 1 1 → 1

Required set bits → 2

1 1 0 0

OFF 0 _ _ _ 2 → $^3C_2$

$\frac{3!}{2!\,1!}$

ON 1 1 0 0

3 Required set bit 1

1 0 0 ← 2

OFF 0 _ _ → $^2C_1$

ON 1 0 0

1 Required Set bit = 0

0 0

Res = count − 1

1 → 0 0 0 1
2 → 0 0 1 0
3 → 0 0 1 1
4 → 0 1 0 0
5 → 0 1 0 1
6 → 0 1 1 0
7 → 0 1 1 1
8 → 1 0 0 0
9 → 1 0 0 1
10 → 1 0 1 0
11 → 1 0 1 1
12 → 1 1 0 0

count

Res = count − 1

function(num, required set bits, left most set bit index)

↳ Need for first time then decrement

1, 1, 0

ON | C_1 = OFF = ${}^{1}C_{2}$ ==>

11, 2, 1

ON ↑ C_1 = OFF = ${}^{2}C_{3}$ ==0

111, 3, 2

↑ C_1 = 0

0111, 3, 3

ON ↑ C_1 = OFF = ${}^{4}C_{4}$ = 1

10111, 4, 4

4 3 2 1 0

↑ C_1 = 0

010111, 4, 5

↑ C_1 → 0

0010111, 4, 6

ON ↑ C_1 = OFF → ${}^{7}C_{5}$

$\frac{7 \times 6 \times 5!}{5!\, 2!} = 21$

${}^{7}C$

0 0 0 0 1 0 0 1 0 1 1 1, 5, 7

7 6 5 4 3 2 1 0

7

23

n = lbi - 1

r = rsb

Start from left most bit

↳ 63th bit

= 63

Result = 23-1 = 22

How to find ^nC_r without finding factorial →

$$\underbrace{{}^nC_r \times \frac{n-r}{r+1} = {}^nC_{r+1}}$$

$$^nC_r \times F = {}^nC_{r+1}$$

$$F = \frac{n-r}{r+1}$$

Pascal's triangle.

1

1 1

1 2 1

1 3 3 1

1	4	6	4	1

$^nC_0$ $\quad$ $^nC_1 \times F$ $\quad$ $^nC_2 \times F$ - - - - - $\quad$ ^nC_r

1 × F

1 × F

Factor = $\frac{i-j}{j+1}$

$= \frac{n-r}{r+1}$

Pascal's triangle, Space → O(n)

↳ Solve without space

$\underbrace{\text{Arrays \& Strings.}}$

```java
// n -> number, srb -> required set bits, lbi -> leftmost bit index => 63
private static long pepcoderAndBits_rec(long n, int rsb, int lbi) {
    if(rsb == 0) return 0L;   1L
    long count = 0L;
    // check left most index bit
    long bm = (1L << lbi);
    if((n & bm) == 0) {
        // leftmost bit index's bit is OFF
        count = pepcoderAndBits_rec(n, rsb, lbi - 1);
    } else {
        // bit is ON
        // find count if lbi is considered as OFF
        count = ncr(lbi, rsb);
        // find count if lbi is ON
        count += pepcoderAndBits_rec(n, rsb - 1, lbi - 1);
    }
    return count;
}
```

↳ final Result = count − 1 = 6 − 1 = 5

if rsb = 0
↳ one no. of ways.

1

00, 0, 1

ON

01

count = $^{2}C_{1} = 2$

100, 1, 2

ON

3

count = 3

$^{3}C_{2} = 3$

1100, 2, 3

3 2 1 0

6

Abbreviation-1 using bits:

pep

\- - -

0 0 0 → p e p → pep

0 0 1 → p e 1 — pe1

0 1 0 → p 1 p — p1p

0 1 1 → p 2 — p2

1 0 0 → 1 ep — 1ep

1 0 1 → 1 e 1 — 1e1

1 1 0 → 2p — 2p

1 1 1 → 3 — $\boxed{3}$