

Resolved paper

Famous

Page No.

Desktop...

$$1 \leq n \leq 26$$

ex 14 n_2

10

2] 

8] $\vdash ? \frac{A}{B} ? T ! A$

ex 2 4 128

1) π no leading zero

2 | 128 182 821 281 31 = 6

3) $W_8 \rightarrow 2^9 \cdot 7$ (T)

$n \rightarrow n!$

$(10^9)!$

TLE

Back rule - solve

bool solve(int n) {

// step 1 - shuffle (next-permutation)

// next-permutation begins end()

Time complexity: $n!$ \rightarrow $(n \cdot n!)$

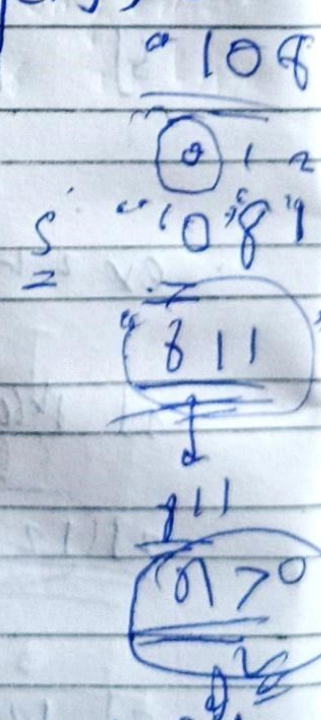
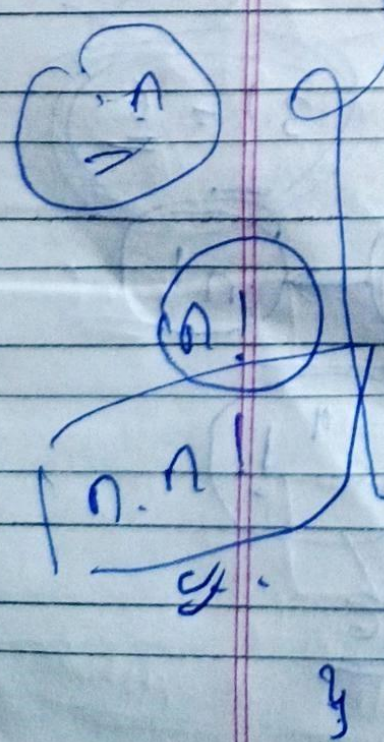
string s2 do_string(n);
// shuffle

do {
// leading zero
if (s[0] == '0')
continue

int n = stoi(s);
if (n > 0) {
// power of 2?

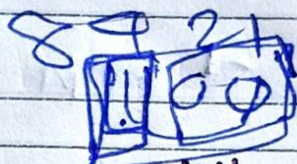
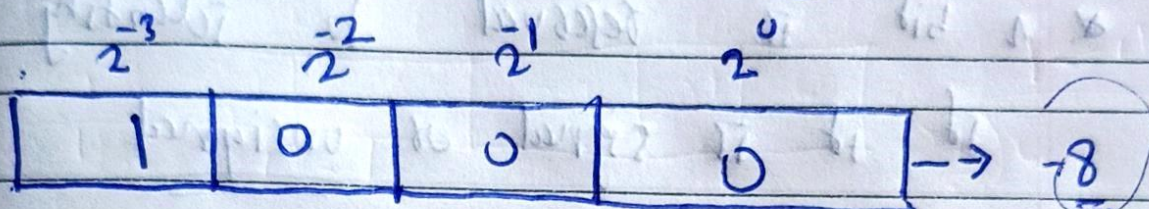
return (n & (n-1)) ? false : true;

while (next-permutation(s.begin(), s.end()))
return false;



Shuffle - 5n! x

3-1 Overflow in Signed & Unsigned Numbers.

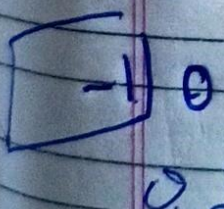
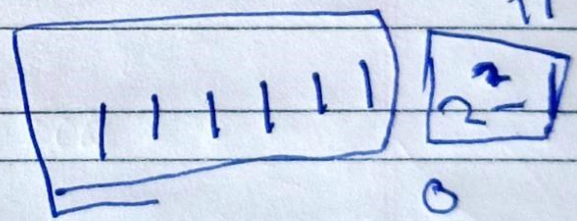
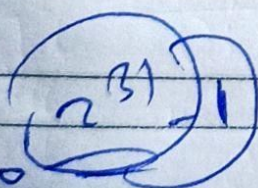
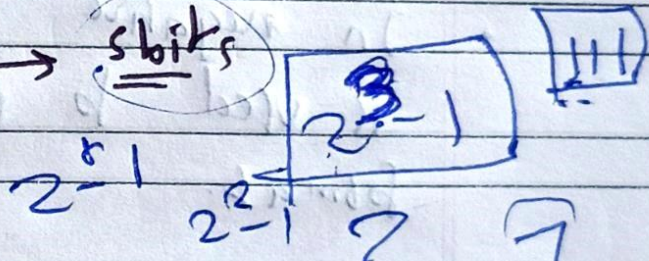


1 0 0 0 → 4bit

+ 1 0 0 0 → 4bit

1 0 0 0 → 5bits

int → 32 bit



100 2^0

2^1

128 - 127

~~Binary~~ Overflow

Integer

$$2^{-31} \text{ to } 2^{+31} - 1$$

Famous

Page No.:

Date:

* 1 bit is reserved to identify

if it is signed or unsigned

• why -1 in $2^{31} - 1$?

-1 is used becoz 0 is also a possible number that is 1 is included.

• In negative number 2's complement is used to represent numbers in binary format.

Revised Power of 2

3 No shuffle

$$2^n$$

$$2^6 = 64$$

n

46

Sorted	46	=	2	46	=	44
--------	----	---	---	----	---	----

B

2^1	1	11	=	X_2	126
-------	---	----	---	-------	-----

2^1	100(2)	=	X_2	121
-------	--------	---	-------	-----

2^2	100(4)	=	X_2	126
-------	--------	---	-------	-----

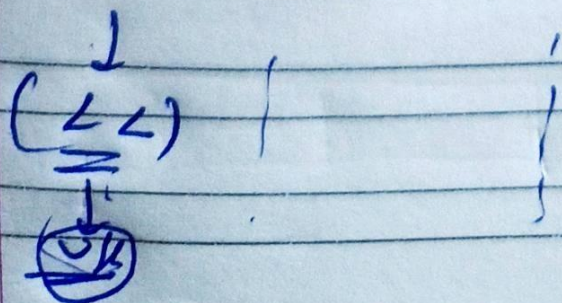
2^3	1000(8)	=	X_2	126
-------	---------	---	-------	-----

2^4	10000(16)	=	2	1269
-------	-----------	---	---	------

2^5	100000(32)	=	2	126 X
-------	------------	---	---	-------

2^6	1000000(64)	=	2	126 X
-------	-------------	---	---	-------

2^7	10000000(128)	=	2	126 X
-------	---------------	---	---	-------



Bitwise op

Left shift & op
(<<)

* Syntax (left op) (right op)
First op << Second op

* Ex:-

1 byte char var = 3

binary = 0000 0011 var << 1

↓ left shift by 1 pos
0 0 0 0 0 1 1

↓ Trailing pos
0 0 0 0 0 1 1 0 = 6

LL operators

LL is equivalent to multiplication by
2 right operand

Ex: var = 3;

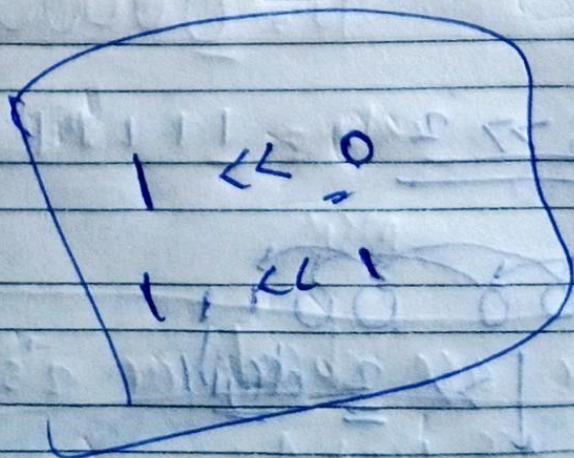
var LL 1

O/p: 6 [3x2]

Ex: var LL 4

3 LL 4

O/p: 48 [3x2⁴]
3x16



1x2⁰ = 1x1 ①

1x2¹ = 1x2 ②