Advanced Bitmasking Problems

→ Med - Hard problems

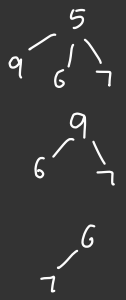
Q Given an array, you find out sum of XOR of all pairs in the array.

arr[] = { 5 9 6 7 }

sum = 0

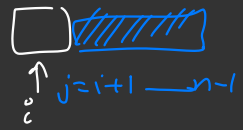
⇒ 5 ^ 9 = 12
⇒ 5 ^ 6 = 3
⇒ 5 ^ 7 = 2
⇒ 9 ^ 6 = 15
⇒ 9 ^ 7 = 14
⇒ 6 ^ 7 = 1

47




```
1001
0110
----
1111
```

```
110
111
---
001
```



Brute Force

```
Sum = 0
for( i=0 ____ i<=n-2) {
    for(j=i+1 ____ j<=n-1) {
        sum += (a[i]^a[j]);
```

$O(N^2)$ time

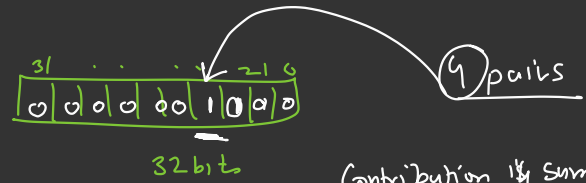
$O(1)$ space

Optimised Approach

Hint-1

$1 \wedge 0 = 1$

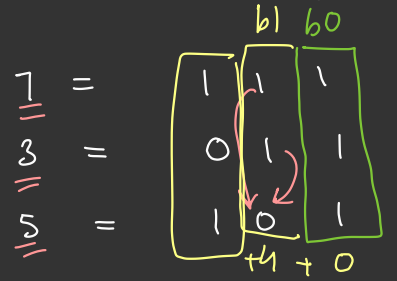
Hint-2 Can you count how many pairs will contribute a 1 to ith bit position



Contribution by sum = 8×4

$= 32$

arr = { 7, 3, 5 }



BF Logic

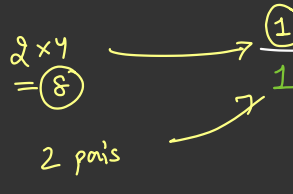
7 ^ 3 = 100 = 4
7 ^ 5 = 010 = 2
3 ^ 5 = 110 = 6
12

7^5 + 3^5 , 7^3 + 3^5

how many pairs will contribute = 12

4 + 4 + 0

cnt = 2×1 | 2 times | 3×0 = 0

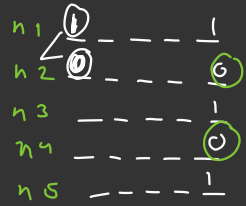


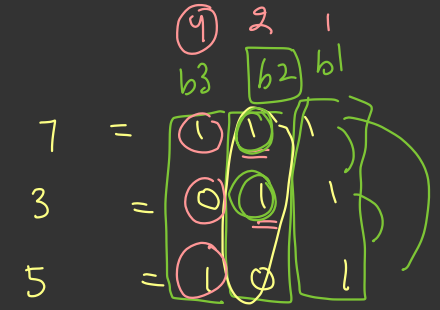
cnt 1's * cnt 0's

1 ^ 0 = 1
3 ways 2 ways
= 6 ways

3 no's { 1 1 1 } 2 no's { 0 0 }

a ^ b = 1





$2 \times 1 \times 4 = 8 + 4 = 12$

$2 \times 1 \times 2 = 4$

0, 0, 0

0

$7 \wedge 3 = 0$ $+ 3 \wedge 5 + 7 \wedge 5 = 0$

$= 7 \wedge 3 + 3 \wedge 5 + 7 \wedge 5$

$= 1 \wedge 3 + 3 \wedge 5 + 7 \wedge 5$

$1, 0 \quad 1, 0$

$= 10 + 10$

$= 2 \times 2$

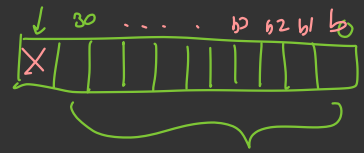
$= 4$

8 4 2 1

1

11

2




```
Sum = 0
for ( i=0 ; i<= 31 , i++ ) {
    A = 0, B = 0 // cnt 1's
    for ( x : arr ) {
        if ( x & (1<<i) > 0 ) {
            A ++ ,
        }
        else {
            B ++ ,
        }
    }
    Contribution = A * B * (1<<i) ;
    Sum = Sum + Contribution ;
}
```

set I^th bit

mask

O(N)

Time

iterate on bits of many nos

ith bit

3 _ _ _ 1

5 _ _ 1 _ 1

7 _ _ 1 _ 1

↑ 2 1

ith bit

```
CheckBit(no , i) {
    return (no & (1<<i)) > 0 ;
}
```

	i=2	b1	b0	A0	B0
3	0	1	1	3	0
5	1	0	1	2	1
7	1	1	1	2	1

3 & 4 = 0

5 & 4 = 0

7 & 4 = 0

1<<2 = 4 2 1 / 1 0 0

1<<0 = 1

3 & 1 > 0

5 & 1 > 0

7 & 1 > 0

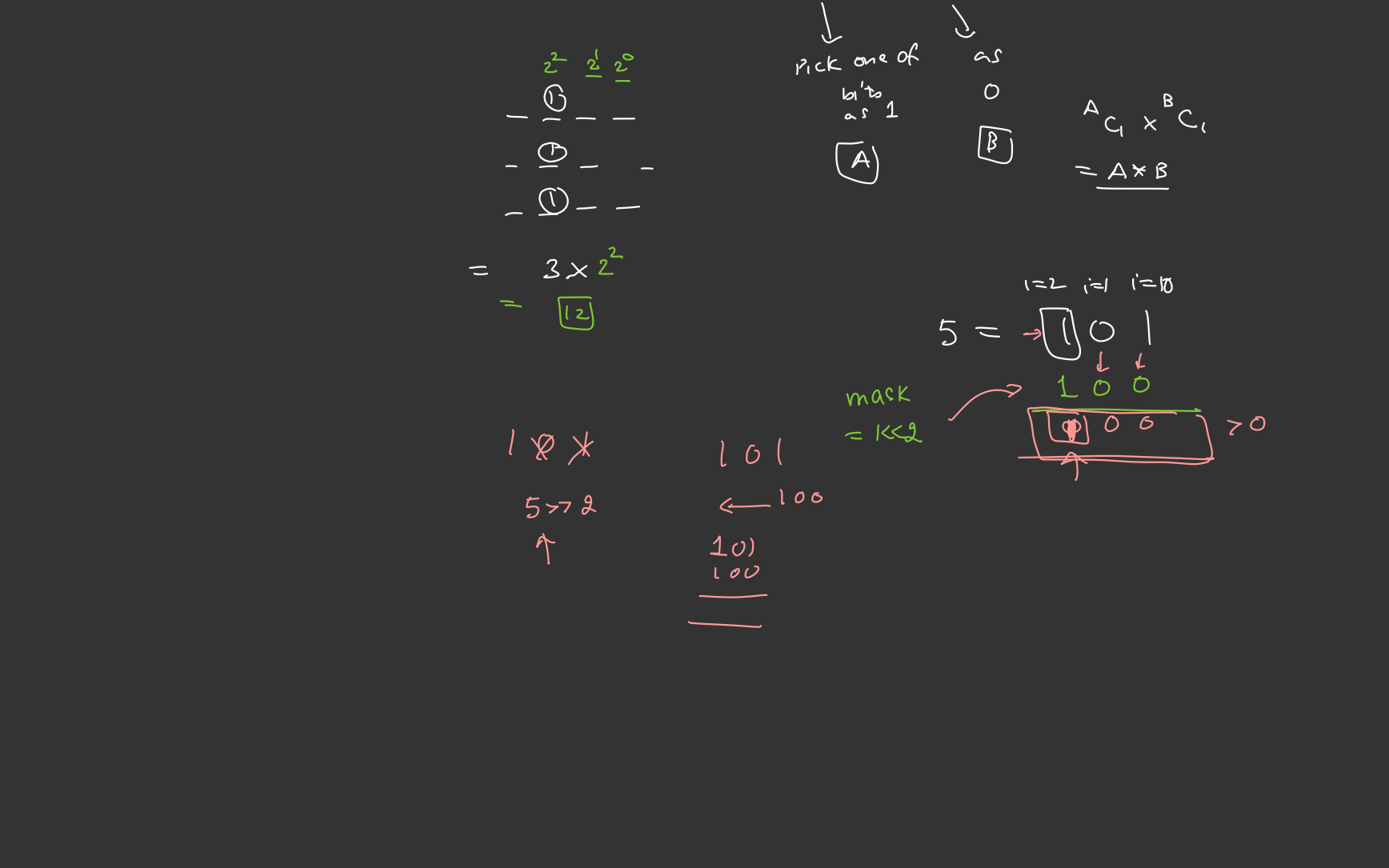
3 × 0 × (2^0) = 0

2 × 1 × (2^1) = 4

2 × 1 × 2^2 = 8

12

1 & 1 | no + | no + | no + | 0 & 1 | 0 & 0


2^2 2^1 2^0
1
1
1
$= 3 \times 2^2$
$= 12$
Pick one of bits as 1
A
as 0
B
$^{A}C_1 \times {}^{B}C_1$
$= A \times B$
i=2 i=1 i=0
5 =
1 0 1
1 0 0
mask
= 1<<2
0 0 0
>0
1 0 1
5>>2
1 0 1
← 100
101
100

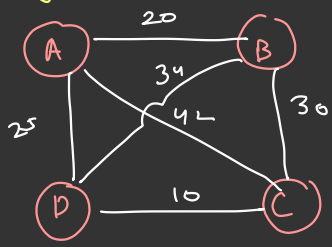
Additional Challenge →

Travelling Salesman Problem

→ N cities, given distance b/w all pair of cities

Visualise

Network of nodes ↓ graph



Input

Adj Matrix

	A	B	C	D
A	0	20	42	25
B	20	0	30	34
C	42	30	0	10
D	25	34	10	0

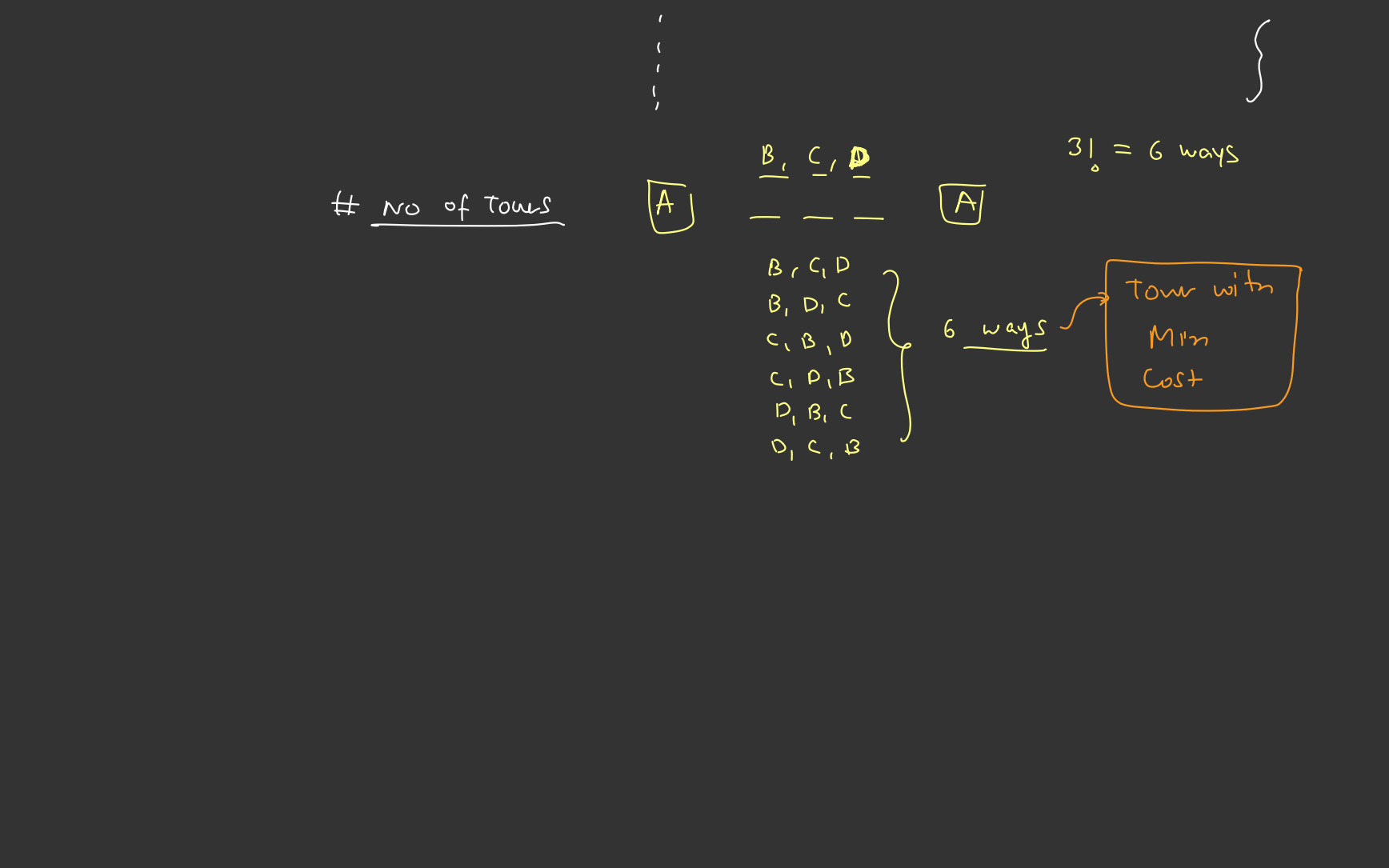
Make a cyclic tour starting from A and come back to A by travelling all cities in min distance

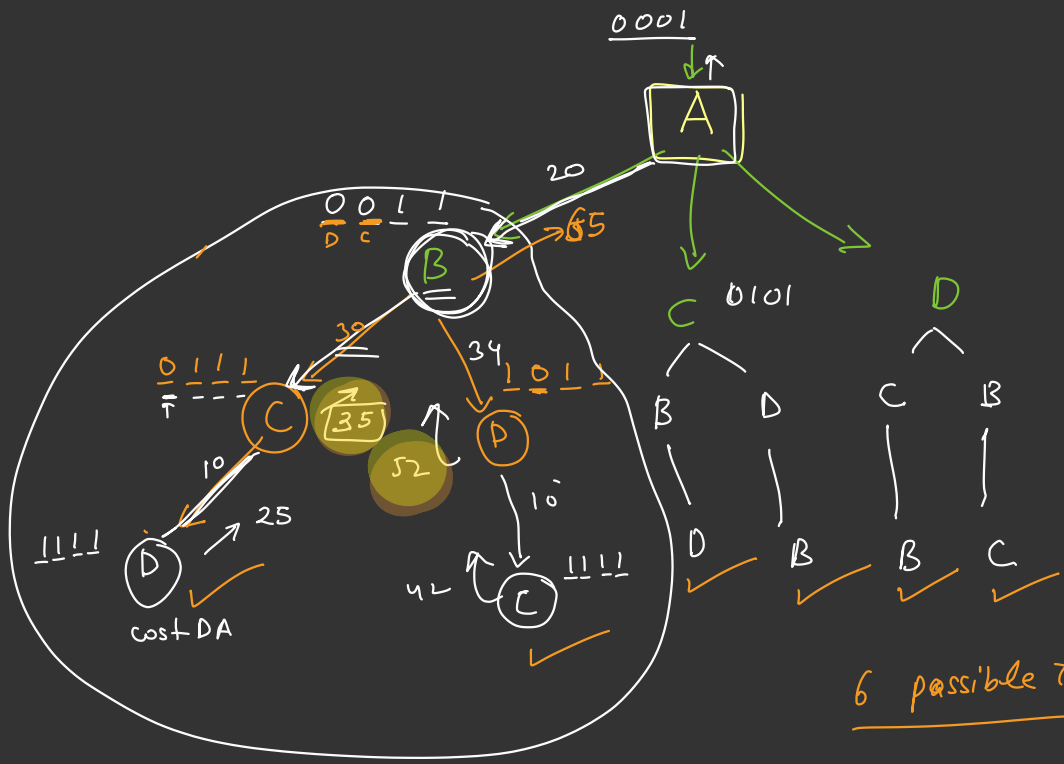
Tour → A → B → C → D → A = 20 + 30 + 10 + 25 = 85 ✓

A → D → B → C → A = 25 + 34 + 30 + 42 = 131 ✓

A → C → D → B → A = 42 + 10 + 34 + 20 = 106 ✓

min


No of Tours
A
B, C, D
— — —
A
3! = 6 ways
B, C, D
B, D, C
C, B, D
C, D, B
D, B, C
D, C, B
6 ways
Tour with
Min
Cost

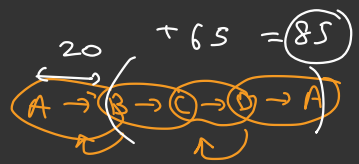


visited

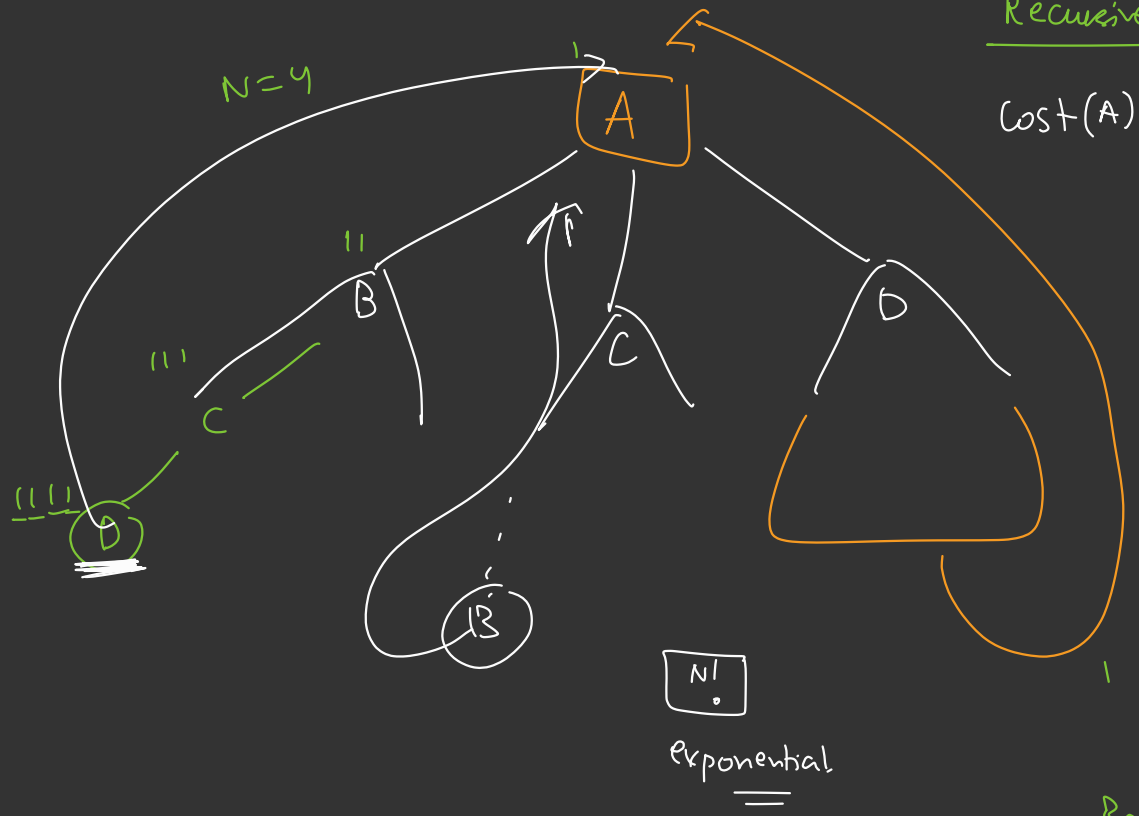
0	0	0	1
D	C	B	A

6 possible Tour

	A	B	C	D
A	0	20	42	25
B	20	0	30	34
C	42	30	0	10
D	25	34	10	0



A → B → D → C → A



Exponential

Recursive Case

$$\text{Cost}(A) = \min\left(\text{Cost from all univisited cities}\right) + \text{Cost To Reach that city}$$

$$= \min\begin{cases} \text{Cost}(B) + A\text{–}B \\ \text{Cost}(C) + A\text{–}C \\ \text{Cost}(D) + A\text{–}D \end{cases}$$

Base Case → if all cities are visited

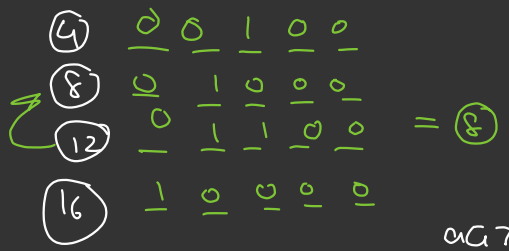
return ~~cost~~ mat[current][A]

Break. 10 Min (10:30)

Challenge → Max AND Pair

n elements, find a pair of elements that generates a maximum AND value.

{ 4, 8, 12, 16 }



8 & 12 = 8

Brute Force

Two loops

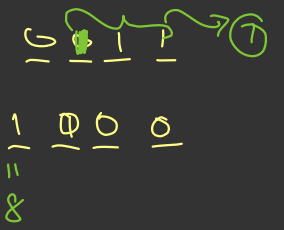
$O(N^2)$

arr[i] & arr[j] (j > i)

optimisation

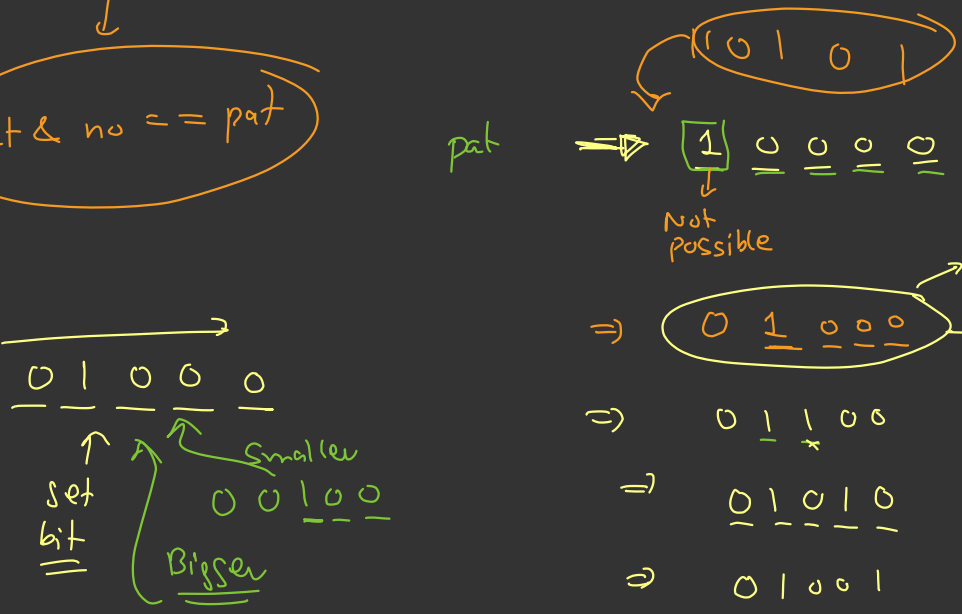
Max AND → More sets on the left side bit

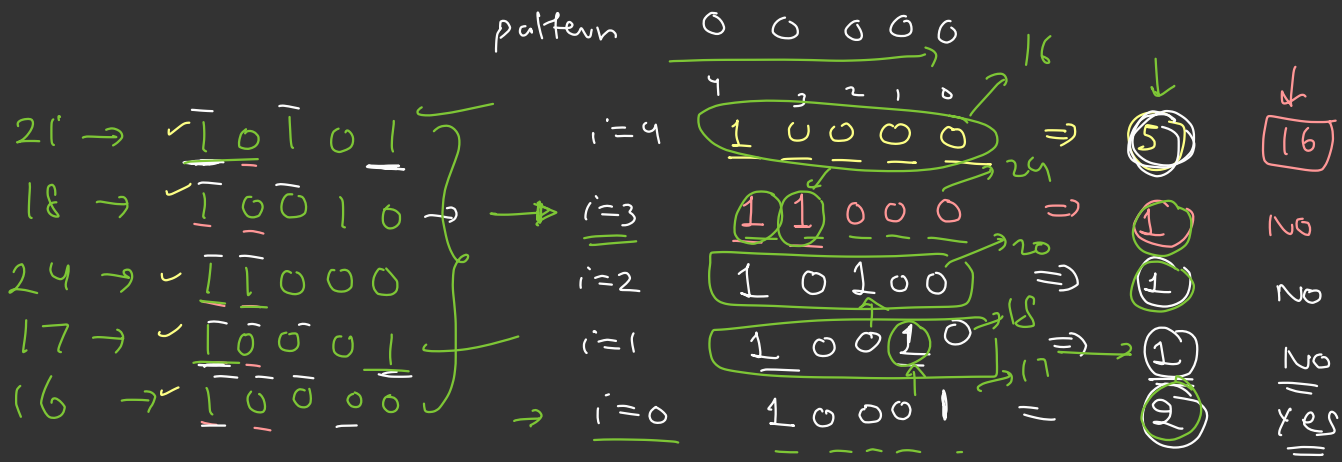
hint



Condition

pat & no == pat






```java
int pattern = 0;
for(int i=31; i>=0; i--){

    //count the no of numbers having given pattern
    int count = countNumbers(arr, pattern | (1<<i));

    if(count>=2){
        //actually update the pattern by setting ith bit
        pattern = (pattern|(1<<i));
    }
}
System.out.println(pattern);
```

pattern | 1<<i

assign

Set Bits in atleast 2 no's at given i

i=4 , i=3 , i=2 , i=1

00000 + 10000 = 10000

10000 + 1000

= 10000 + 00001 = 10001

pattern