Todays Content

1. Maximum AND Pair.
2. Maximum AND Pair, Count Pairs.

Recap:

Check i^{th} Bit set in N : `(N >> i) & 1 == 1 : Set.`

Set i^{th} Bit in N : `N = N | (1 << i)`

Flip i^{th} Bit in N : `N = N ^ (1 << i)`

$a \wedge a = 0$

Q4: Given ar[N], Return max & between any pair (i,j), $i \neq j$

Ex: ar[] = { 27 18 20 } : ans = 18

(0 1 2)

ar[0] & ar[1] = 18

ar[0] & ar[2] = 16

ar[1] & ar[2] = 16

$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

27: 1 1 0 1 1

18: 1 0 0 1 0

20: 1 0 1 0 0

Ex: ar[] = { 21 18 24 20 16 } ans = 20

(0 1 2 3 4)

ar[0] & ar[3] = 20

#Ideal: Generate all pairs & take & & calculate overall max.

#Given ar[] & N

```
int ans = INT_MIN;
for(int i=0; i<N; i++){ TC: O(N^2) SC: O(1)
    for(int j=i+1; j<N; j++){
        ans = max(ans, ar[i] & ar[j])
    }
}
return ans;
```

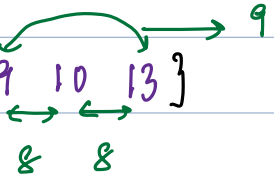
#Obs: The max ar[] value need not give maximum and value.

#Idea2: Sort & Take between adjacent elements * logic wont work

Ex: ar[] = { 10 13 9 }

Sort

ar[] = { 9 10 13 }



Obs: 1. a: 1 & b: 1 = 1

2. a : 0 0 1 & b : 0 1 0 = 0 0 0

2. ele1 = 2^4 2^3 2^2 2^1 2^0 = 1 0 0 0 0

ele2 = 2^4 2^3 2^2 2^1 2^0 = 0 1 1 1 1

3. Since we need max AND, we will set bit from left → right

	0	1	2	3	4	5	6	7	8
ex: ar[9] = {	24	12	23	25	7	26	27	31	27 }

→ b & y 2 1

4 3 2 1 0

	4	3	2	1	0
0 24:	1	1	0	0	0
0 12:	0	1	1	0	0
0 23:	1	0	1	1	1
0 25:	1	1	0	0	1
0 7:	0	0	1	1	1
0 26:	1	1	0	1	0
27:	1	1	0	1	1
31:	1	1	1	1	1
27:	1	1	0	1	1
cnt:	7	6	1	4	3
And:	1	1	0	1	1 = 27

Max pair

	y	3	2	1	0
a :	1	1	1	0	0	1	1
b :	1	1	0	1	0	1	1
and :	1	1	0	1	1

27 & 31 = 1 1 0 1 1

27 & 27 = 1 1 0 1 1

27 & 31 = 1 1 0 1 1

} 3 pairs with max and.

Note:

Pseudo code:

```
and = 0;
Iterate i: [30..0]: # i bit position
    # iterate on arr(), calculate
      no: of elements with i^th bit set = c;
    if( c >= 2) { # if i^th bit and value = 1.
        and = and | (1<<i); # Set i^th bit in and;
        iterate in arr(), if for an element
        i^th bit is unset, discard = 0.
    }
return and.
```

i^th

a 1

b 1

and 1

$$^{N}C_{2} = \frac{(N)(N-1)}{2}$$

# of Q: return no: of pairs with their bitwise & maximum.

Iterate in arr() & calculate no: of non zero ele = n

return $^{n}C_{2}$ # Among n ele take any 2 & form pair:

Eg: arr[ ] = { 10 14 1 6 11 10 8 6 } ans:

	3	2	1	0
val:				
10:	1	0	1	0
14:	1	1	1	0
1:	0	0	0	1
6:	0	1	1	0
11:	1	0	1	1
10:	1	0	1	0
8:	1	0	0	0
6:	0	1	1	0

Max pair

	4	3	2	1	0
a:					
b:					
and:					

int maxpair(int arr[], int N){ TC: O( ) = O( ) SC: O( )

Q4: Given ar[N], Return min ^ between any pair (i, j) $i \neq j$

	0	1	2	3	4	
ar[] = {	7	4	6	9	10 }	ans = 1

Ex: $7 \wedge 4 = 3$ $9 \wedge 10 = 3$

$4 \wedge 6 = 2$

$7 \wedge 6 = 1$

#Idea1: Generate all pairs, Calculate ^ & return overall min.

#Given ar[] & N

```
int ans = INT_MAX
for(int i=0; i<N; i++){ TC: O(N^2) SC: O(1)
    for(int j=i+1; j<N; j++){
        ans = min(ans, ar[i]^ar[j])
    }
}
return ans;
```

#Idea2: Sort ar[] & Take ^ among adjacent ele & return overall min.

	0	1	2	3	4	
ar[] = {	7	4	6	9	10 }	

i → i → i → i → i : if i == last index stop

	0	1	2	3	4	
ar[] = {	4 ↔	6 ↔	7 ↔	9 ↔	10 }	ans = 1
	2	1	14	3		

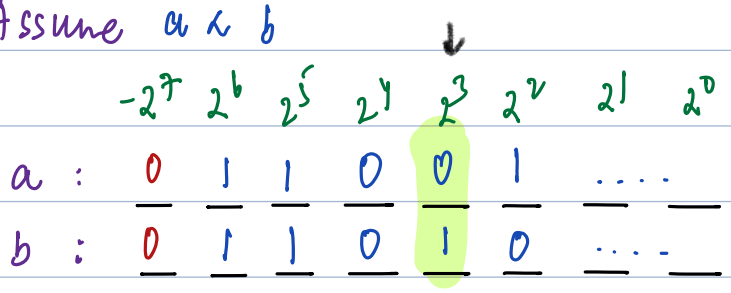
```
int minXor(vector<int> &ar){ TC: O(NlogN + N) = O(NlogN)
    sort(ar.begin(), ar.end());   // Sorting
    int ans = INT_MAX;            // i == N-1 = stop
    for(int i=0; i<N-1; i++){
        ans = min(ans, ar[i]^ar[i+1]);
    }
    return ans;
}
```

Statement? Min xor will be between adjacent elements in sorted array.

Why?

Say we have 2 +ve ele: a b

Assume a < b

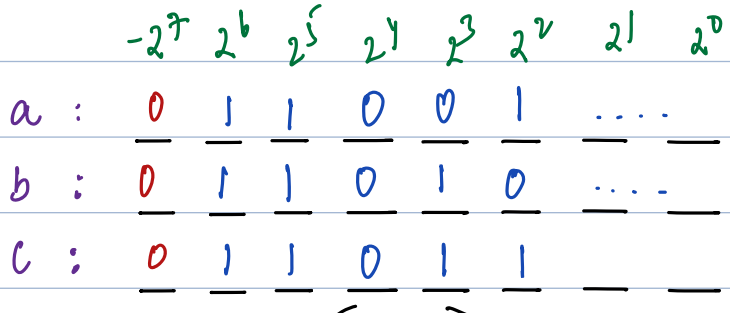


11001

11010

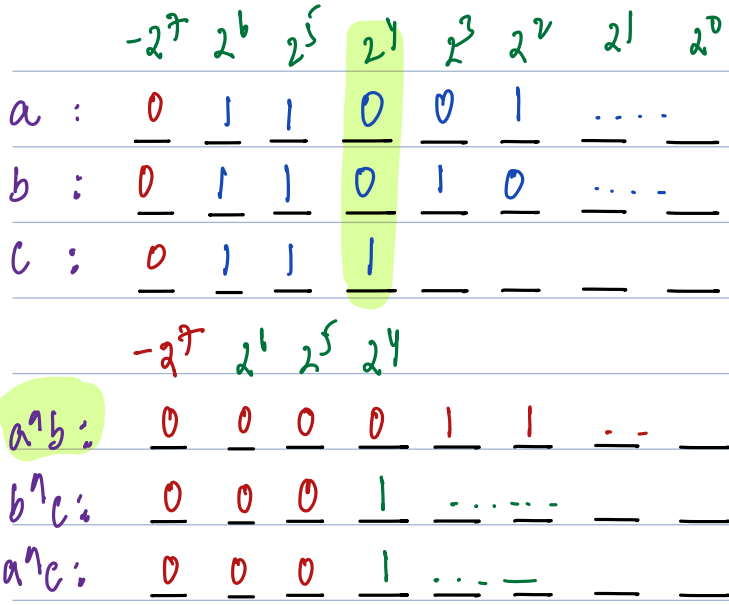
Say we have 3 +ve ele: a b c

Assume



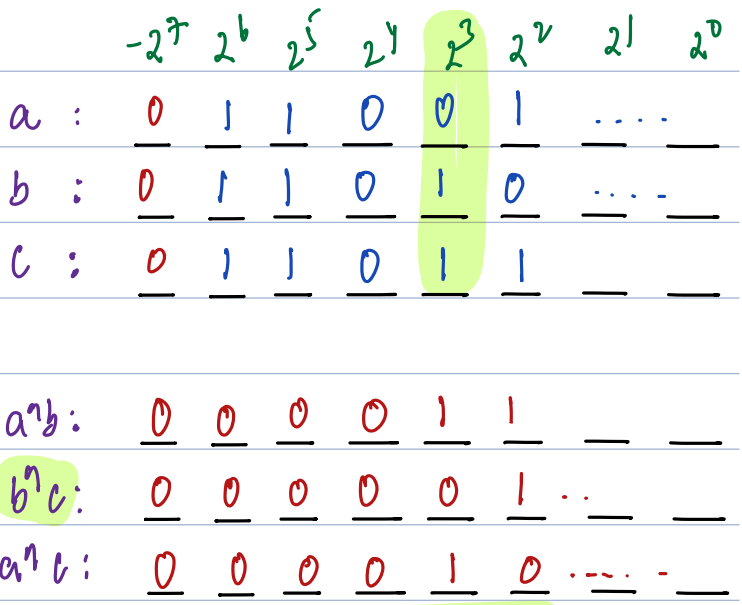
At 4th Bit Diff

a < b < c # a^c should not be min



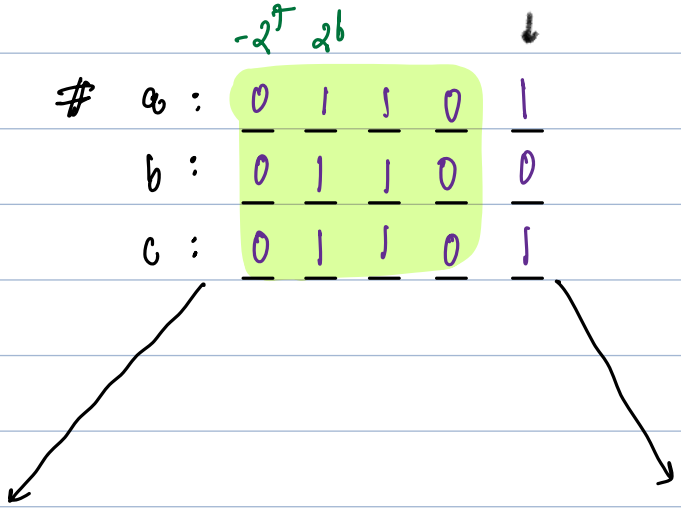
At 3rd Bit Diff

a < b < c # a^c should not be min



#Conc: In this way we can prove that min lies among adjacent elements.

1st Bit where a b c are not same



	-2^7	2^6			
a :	0	1	1	0	0
b :	0	1	1	0	1
c :	0	1	1	0	1
a^b :	0	0	0	0	1
b^c :	0	0	0	0	0
a^c :	0	0	0	0	1

	-2^7	2^6			
a :	0	1	1	0	0
b :	0	1	1	0	0
c :	0	1	1	0	1
a^b :	0	0	0	0	0
b^c :	0	0	0	0	1
a^c :	0	0	0	0	1