

Today's Content

1. Maximum AND Pair.

2. Maximum AND Pair, Count Pairs.

Recap:

Check i^{th} Bit set in N : $(N \gg i) \& 1 == 1$: Set.

Set i^{th} Bit in N : $N = N | (1 \ll i)$

Flip i^{th} Bit in N : $N = N \wedge (1 \ll i)$

$$a \wedge a = 0$$

Q4: Given $arr[N]$, Return max & between any pair $arr[i]$ & $arr[j]$

Ex: $arr[] = \{ \overset{0}{27} \overset{1}{18} \overset{2}{20} \}$: ans =

Ex: $arr[] = \{ \overset{0}{21} \overset{1}{18} \overset{2}{24} \overset{3}{20} \overset{4}{16} \}$ ans =

#Ideas:

#Ideas:

Obs: 1. $\begin{array}{r} a \\ \& b \\ \hline \end{array}$

2. $\begin{array}{r} a : 0 \ 0 \ 1 \\ \& b : 0 \ 1 \ 0 \\ \hline 0 \quad 0 \ 0 \ 0 \end{array}$

2. $ele1$

$ele2$

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

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

Ex: $arr[] = \begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \{ 24 & 12 & 23 & 25 & 7 & 26 & 27 & 31 & 29 \} \end{array}$

	4	3	2	1	0
24:	1	1	0	0	0
12:	0	1	1	0	1
23:	1	0	1	1	1
25:	1	1	0	0	1
7:	0	0	1	1	1
26:	1	1	0	1	0
27:	1	1	0	1	1
31:	1	1	1	1	1
29:	1	1	0	1	1

Max pair

4 3 2 1 0

a :

b :

and :

cnt:

And;

Note:

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

↓

val: 3 2 1 0

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

cnt:

&val:

Max pair

4 3 2 1 0

a:

b:

and:

int maxpair (int arr[], int N) { TC: $O(N^2)$ SC: $O(1)$ }

Q4: Given $arr[N]$, Return min \wedge between any pair $arr[i] \wedge arr[j]$ $i \neq j$

arr[] = { 7, 4, 6, 9, 10 } ans = 1

En: $7^4 = 3$ $7^6 = 1$

$$4^6 = 2$$
$$g^1_1 = 3$$

Idea: Generate all pairs, calculate nm & get overall min.

ans = INT_MAX; TC: $O(N^2)$ SC: $O(1)$

$$i=0; i \in N; i \neq 0 \}$$
$$\vec{j} = (j_x, j_y, j_z)$$
$$ans = \min(ans, ar[i] \wedge ar[j]);$$

return ans;

Idea 2: Sort arr[] & calculate diff between adjacent elements & get overall min.

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

#sort

if $i == \text{last_index}$ stop

arr = { 4, 6, 7, 9, 10 }

2, 1, 14, 3

int minXor(vector<int> &a) { TC: $O(N \log N + N) = O(N \log N)$ }

`sort(ar.begin(), ar.end());` Inbuilt sorting

```
int ans = INT_MAX, N = arr.size();
```

```
for(int i=0; i<N-1; i++) {
```

$$ans = \min(ans, arr[i] \wedge arr[i+1]);$$

ratum est;

Statement? Min xor value will be among adjacent element in sorted arr

Why?

Say we have 2 no: a b

Assume $a < b$

	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
a :	0	1	1	0	0	1	
b :	0	1	1	0	1	0	

Say we have 3 no: a b c

Assume $a < b < c$

	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
a :	0	1	1	0	0	1	
b :	0	1	1	0	1	0	
c :	0	1	1	0	1	1		

In c 4^{th} Bit = 1

$a < b < c$ # a^7c is new min

	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
a :	0	1	1	0	0	1	
b :	0	1	1	0	1	0	
c :	0	1	1	1	0	1	

-2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

a^7b : 0 0 0 0 1 1 ...

b^7c : 0 0 0 1 1 1 ...

a^7c : 0 0 0 1 0 0 ...

In c 3^{rd} Bit = 1

$a < b < c$ # a^7c is new min

	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
a :	0	1	1	0	0	1	
b :	0	1	1	0	1	0	
c :	0	1	1	0	1	1		

-2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

a^7b : 0 0 0 0 1 1 ...

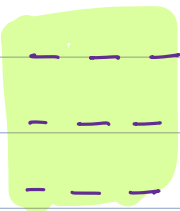
b^7c : 0 0 0 0 0 1 ...

a^7c : 0 0 0 0 1 0 ...

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

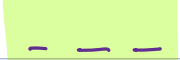
i

$a_0 :$



0

$a_1 :$



1

$c :$

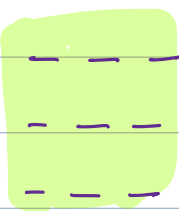


1

.

i

$a_0 :$



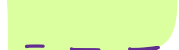
0

$a_1 :$



0

$c :$



1

-

i