Today's Content:

→ Calculate Sum of xor of all pairs

→ Max and pair

→ Array Doubts: —
  - → Flip
  - → Max 0's sum

Sunday → (10AM - 12PM) → max

18) Given N Elements, calculate {sum of xr of all pairs (i, j)}?

+Ve

// ar[3] = { 3 1 5 }

→ {3,3} {3,1} {3,5}
→ {1,3} {1,1} {1,5}
→ {5,3} {5,1} {5,5}

Idea1: ar[5] = { 3 5 6 8 2 }

→ Calculate sum of xr of all pairs

→ O(N²)

```
sum = 0
for(i=0; i<N; i++){
    for(j=0; j<N; j++){
        sum = sum + ar[i]^ar[j]
```

Idea2:

3^3	3^5	3^6	3^8	3^2
5^3	5^5	5^6	5^8	5^2
6^3	6^5	6^6	6^8	6^2
8^3	8^5	8^6	8^8	8^2
2^3	2^5	2^6	2^8	2^2

Idea2: Either iterate in upper or lower Δ get sum of xr of all pairs.

sum

return 2*sum

TC: O(N²)

SC: O(1)

Idea3:

Contribution?

	2³	2²	2¹	2⁰
10 :	1	0	1	0
9 :	1	0	0	1
7 :	0	1	1	1
26		2	2	2

⇒ 2³ + 0 + 2¹ + 0

2³ + 0 + 0 + 2⁰

0 + 2² + 2¹ + 2⁰

2*2³ + 1*2² + 2*2¹ + 2*2⁰ = 26

2 : 0 0 1 0 , 3 : 0 0 1 1 , 5 : 0 1 0 1 , 6 : 0 1 1 0 , 8 : 1 0 0 0

	2^3	2^2	2^1	2^0
3^5 :	0	1	1	0
3^6 :	0	1	0	1
3^8 :	1	0	1	1
3^2 :	0	0	0	1
5^6 :	0	0	1	1
5^8 :	1	1	0	1
5^2 :	0	1	1	1
6^8 :	1	1	1	0
6^2 :	0	1	0	0
8^2 :	1	0	1	0
	2^3	2^2	2^1	2^0

Sum

How many set bits at each bit pos → 4 6 6 6

$S = 32 + 24 + 12 + 6 =$

return $2S = 148$

Idea3: At every bit pos iterate on all pairs & get number of set bits

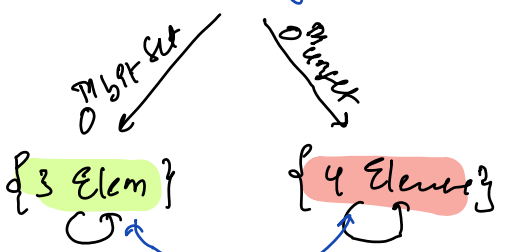
$31 * [O(N^2)] = 31N^2$

a : _ _ _ _ 0/1/ diff — 0^{th} : 1

b : _ _ _ _ 1/0/ diff — 0^{th} : 1

a^b : _ _ _ _ 1 — 0^{th} : 0

Idea: Given arr[7] :



Q : In how many nr pairs 0^{th} bit is set : ?

Idea: If we take an element from set & other from unset their nr of 0^{th} bit = 1

Total pairs = (3 * 4) = 12

2 : 0010, 3 : 0011, 5 : 0101, 6 : 0110, 8 : 1000

In how many elem : $\begin{cases} 0^{th} \text{ bit set} : 2 \\ 0^{th} \text{ bit unset} : 3 \end{cases}$ → pair 0^{th} bit = 6 — Cont — $6 * 2^0$

In how many elem : $\begin{cases} 1^{st} \text{ bit set} : 3 \\ 1^{st} \text{ bit unset} : 2 \end{cases}$ → pair 1^{st} bit = 6 — Cont — $6 * 2^1$

In how many elem : $\begin{cases} 2^{nd} \text{ bit set} : 2 \\ 2^{nd} \text{ bit unset} : 3 \end{cases}$ → pair 2^{nd} bit = 6 — Cont — $6 * 2^2$

In how many elem : $\begin{cases} 3^{rd} \text{ bit set} : 1 \\ 3^{rd} \text{ bit unset} : 4 \end{cases}$ → pair 3^{rd} bit = 4 — Cont — $4 * 2^3$

$S = 74$

return 25

Pseudo code

```
// Given ar[N] Elements? sum of xor of all pairs

int sum = 0;

for(i = 0; i < 30; i = i+1){

    // for every bit pos i, get no: of set & unset in array
    int c = 0
    for(j = 0; j < N; j = j+1){
        if(checkBit(ar[j], i){ c = c+1}
    }

    // In how xor pairs i^th bit set → (c)*(N-c)
    sum = sum + (c)*(N-c)*2^i (k < 1)
}

return 2*sum
```

N →
i^th bit set : c
i^th bit unset : N-c

TC: $31 * N \rightarrow O(N)$

SC: $O(1)$

28) Given N Array Elements, choose 2 indices (i, j) such that

i != j && get max { Ar[i] & Ar[j] } [bitwise and]

// get pair with max bitwise and b

a & a = a

VIMP

Ex1: ar[3] = { 27, 18, 20 } (indices 0, 1, 2)

(27, 18) = 18 (27 20) = 16 (18 20) = 16

27 : 1 1 0 1 1

18 : 1 0 0 1 0

20 : 1 0 1 0 0

Ex2: ar[4] = { 21, 18, 24, 17, 16 } → ans = 17

21 : 1 0 1 0 1

18 : 1 0 0 1 0

24 : 1 1 0 0 0

17 : 1 0 0 0 1

21, 17 = 17

21, 18 = 16

Ideal: get all pairs bitwise &, get max of them : O(N^2)

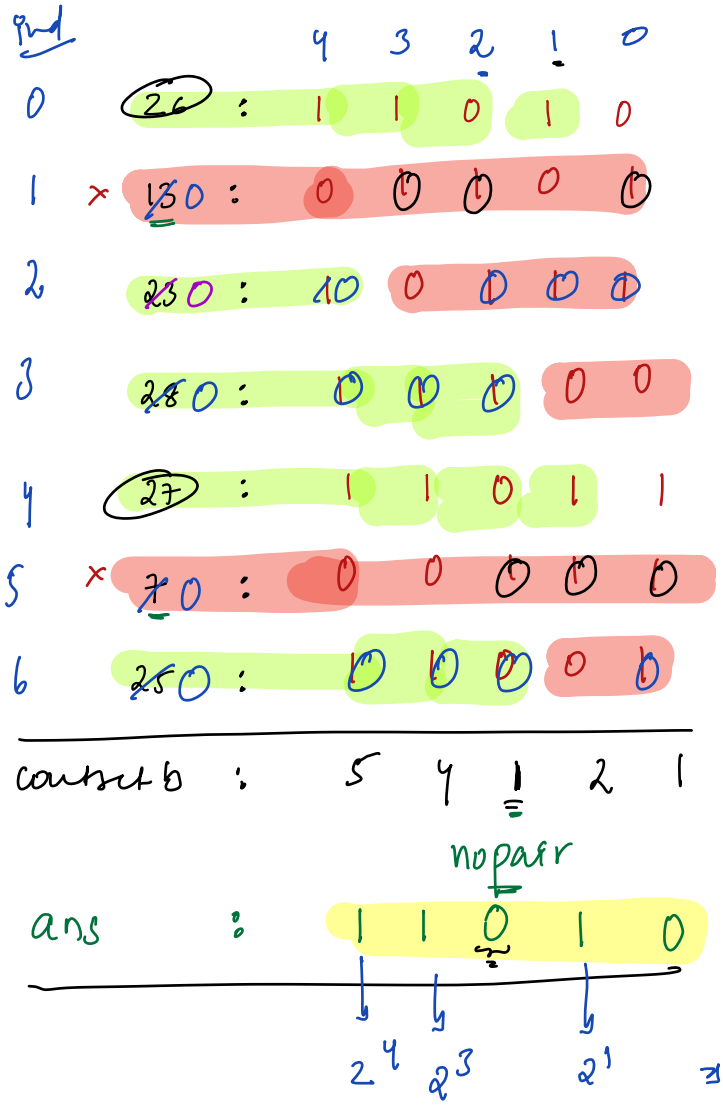
Idea2: And Between 1st max & 2nd max or 1 Elem = Max

1 0 0 0 0 (i = 2^4)

0 1 1 1 1 (2^0 + 2^1 + 2^2 + 2^3 = 15)

max And : give preference to Left most bit

Eg3: ar[7] = { 26, 13, 23, 28, 27, 7, 25 }



After 4th bit : ar[1] = 0, ar[5] = 0

After 3rd bit : ar[2] = 0

After 1st bit : ar[3] = 0, ar[6] = 0

// Iterate in array, non zero elements will form your pair.

Pseudo code :

```
ans = 0;

for(i = 30; i >= 0; i = i-1){

    // get no of element whose i^th bit set
       set c = 0
       for(j = 0; j < N; j = j+1){
           if(checkBit(arr[j], i){ c = c+1}
       }

    // if(c >= 2){ // We can pick a pair whose
                      bit with and value, (i^th bit set)
        ans = ans + (2^i)
        // All elements whose i^th bit unset = 0
          for(j = 0; j < N; j = j+1){
              if(!checkBit(arr[j], i)){arr[j]=0}
          }
    }
}

// return ans;
```

TC: $31 * [N + 1 + N] \rightarrow O(N)$ SC: $O(1)$

// google: a, b, c, d such that a & b & c & d is max

// a, b such that a & b is max

// calculate how many pairs (a, b) a & b is max : ?

→ get no: of non-zero element in arr[], after all updates = x

→ pairs = $^{x}C_{2}$

Flip Array:

(Explain)

→ Maximum Absolute diff

→ Max distance

→ Max chunks to make Sorted

Easiest

Sundays: 10AM - 12PM

↳ 2 hrs:

(Detailed Idea)

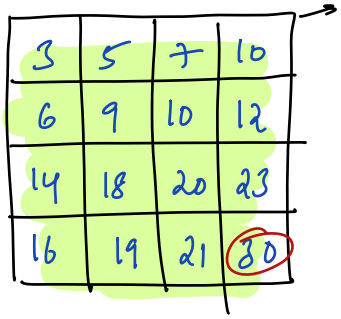
1) Flip ✓ (Sundays)

2) Max Subrect Sum, row-wise sorted & Colum wise sorted

3) Max Sum Square Submatrix

108) Max Submatrix Sum, Row-wise & Column wise Sorted

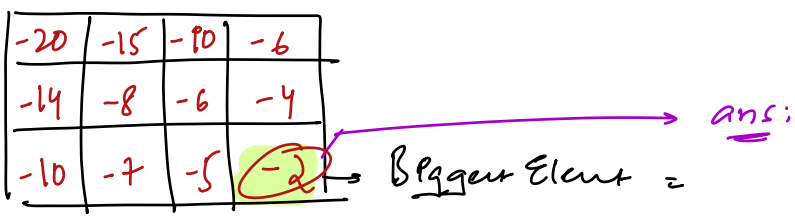
Ex:



3	5	7	10
6	9	10	12
14	18	20	23
16	19	21	30

→ If all Elements are +ve
= Sum of all matrix

Ex2:

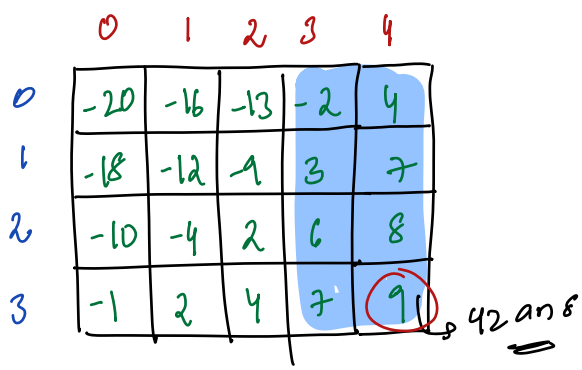


-20	-15	-10	-6
-14	-8	-6	-4
-10	-7	-5	-2

→ ans:

→ Biggest Elem =

Ex3:



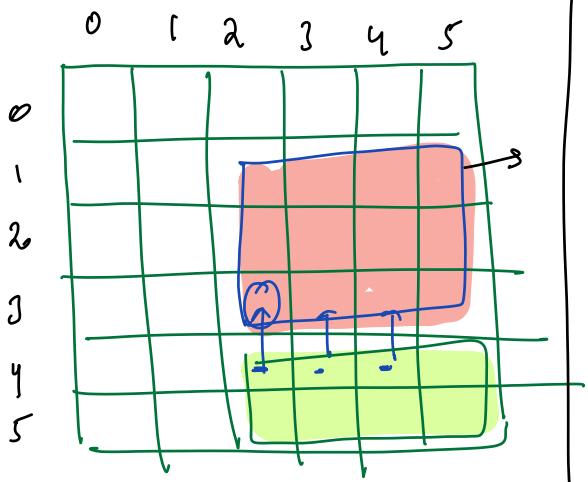
	0	1	2	3	4
0	-20	-16	-13	-2	4
1	-18	-12	-9	3	7
2	-10	-4	2	6	8
3	-1	2	4	7	9

→ 42 ans

// Largest Element in your matrix it should be : Bottom right

obs: Max Submatrix Sum it should end at BR

→ mat[N][M] → [N-1, M-1]



For TL at every cell, BR = [N-1, M-1]

(a_1, b_1)

// Construct pf[][]

```
a1 = 0; a1 < N; a1++){
    b1 = 0; b1 < M; b1++){
        TL = (a1, b1)
        BR = (N-1, M-1)
        Calculate pf's sum using
        pf[][]
    }
}
```

TC: $O(N^2 M)$

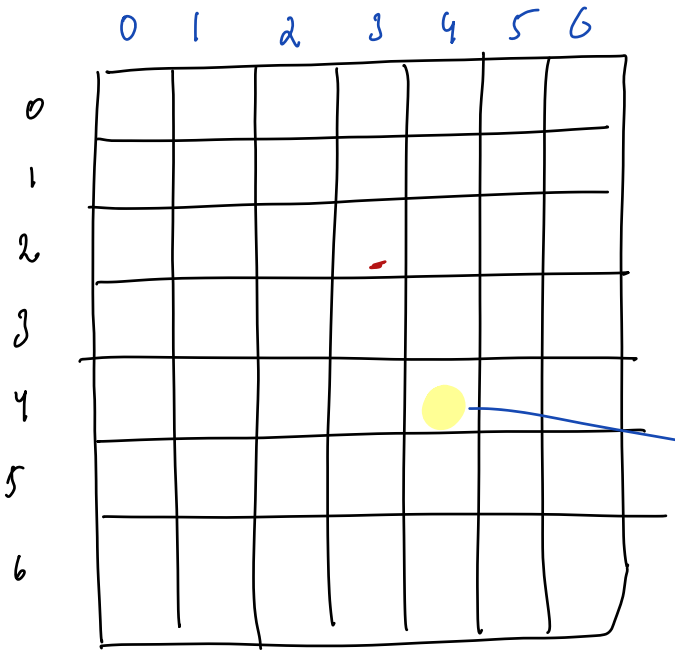
SC: $O(1)$ → in same matrix to get pf[][]

// Max Sum Square Submatrix

↳ given mat[N][N], find max submatrix sum → (Submatrix size B×B)

↳ {for every submatrix of size (B×B) Can we simply get sum & get overall max}

Ex:



mat[i][j]

{B = 4} ⇒ [4][4]

TL BR

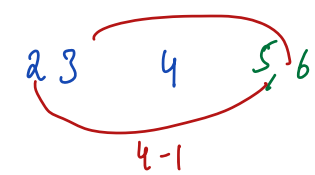
TL 4, 3

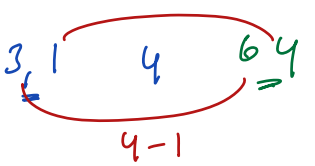
BR 7, 6

{outside matrix} don't do sum

Ex: TL B BR : Sum of matrix

0,0 4 3,3





// Construct Pf[][]

// Instead B×B

TL = (a_1, b_1)

BR = $(a_1 + B-1), (b_1 + B-1)$

→ get submat sum

// Pseudocode :

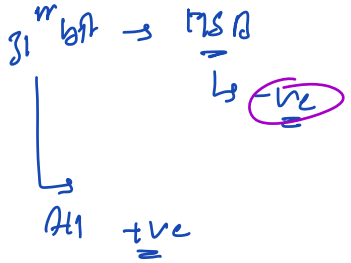
1) Construct Pf[][]

2) TL can be found any where B is given input

```
a1 = 0; a1 < N; a1++){
    b1 = 0; b1 < N; b1++){
        TL = (a1, b1)
        BR = (a1 + B-1, b1 + B-1)
        // Can we check if BR in matrix
        if( a1 + B-1 < N && b1 + B-1 < N){
            // get sum  TL -> BR
        }
    }
}
```

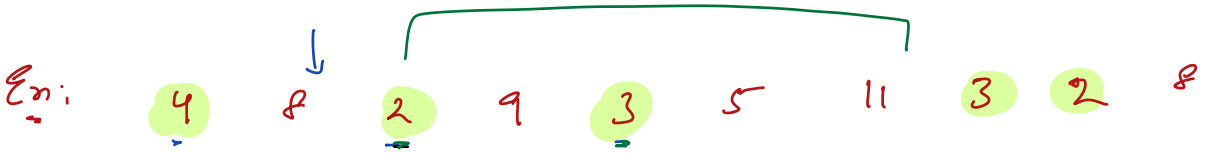
// → Subarray OR in 1 shot

→ 2D kadane's shot



// $|ar[i] - ar[j]| + |i - j|$

→ Min Swaps Required to bring all Ele $= k$ $\quad k = 4$



[0 4] : 2 swaps

[1 5] : 3 swaps

[2 6] : 3 swaps

for every window how many elements are $= 4$

Revision:

- Every week: Sunday: 1hr on that week notes