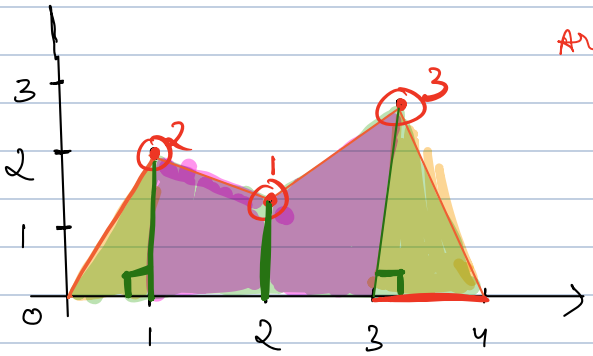
Contest discussion

Class starts at 9:05

Q

| 2 | 1 | 3 |
|---|---|---|




Area $\Delta 1 = \frac{1}{2} \times 1 \times 2 = 1$

Area $\Delta 2 = \frac{1}{2} \times 1 \times 3 = \frac{3}{2}$

for ( 1 ⟶ n-2. )

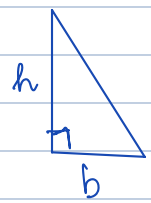
trapezium 1 $= \frac{1}{2}(2+1) \times 1 = \frac{3}{2}$

trapezium 2 $= \frac{1}{2} \times (1+3) \times 1 = 2$

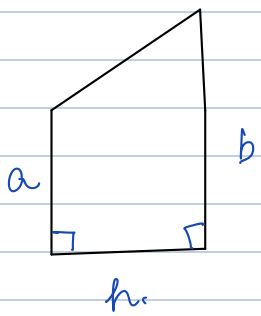
$1 + \frac{3}{2} + \frac{3}{2} + 2 = 6$.

1) Area of right $\angle \Delta = \frac{1}{2} \times b \times h$.

2) Area of a trapezium $= \frac{1}{2} \times (a+b) \times h$.



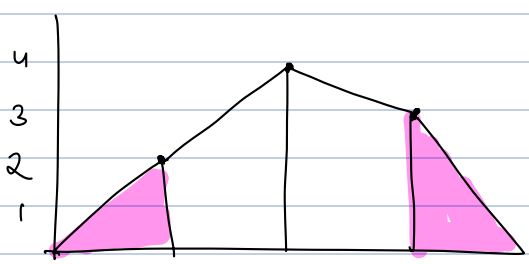

$= \frac{1}{2} \times b \times h$

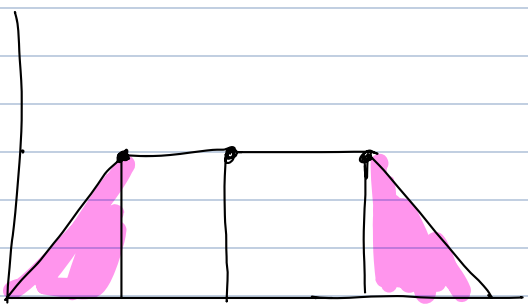



$\frac{1}{2} \times (a+b) \times h$.

TC → O(N)

2 4 3



Q. Candies

A = | 0 | 0 | 0 | 0 |
1 2 3 4

B = [ [1 2] ✓
[4 4] ✓
✓ [1 3] ]

⇒ 1 1 0 0
1 1 0 1
2 2 1 1 ✓

C = [ 1 2 3 ]
↓ ↓ ↓
O/P. 4 2 0

Brute force → Distributing |M| × |A|

m * n.

→ queries |C| * |A|

C * n.

TC → (m*n) + (C*n)

Optimisation → Prefix Sum.

A = | 0 | 0 | 0 | 0 |
1 2 3 4

B = [ [l r] ✓
[4 4] ✓
[1 3] ]

put +1 at l and put -1 at (r+1) if exists

r+1 ↓

1 0 -1 0 →
1 0 -1 1
2 0 -1 0
prefix 2 2 1 1

1 2 3 4
1 -1 1
2
↓ ↓ ↓ ↓
prefix. 2 2 1 1

C = [1 2 3]

either exactly equal to 1 or just greater than 1.

1 1 2 2.

0, 4

0 0 0 0

```
lowerbound (int arr[], int low, high, element) {
    while (low < high) {
        mid = low + (high - low)/2;
        if (arr[mid] < element) {
            low = mid+1;
        }
        else {
            high = mid;
        }
    return low
}
```

TC - O(N) + (N log N)

q * log N.

1 1 4 6 8

q = (2)

Q. A → 1 3 4 4 2 2

make all elements of the arry or a multiple of 4.

eg A = 1 3 4 4 2 2

4

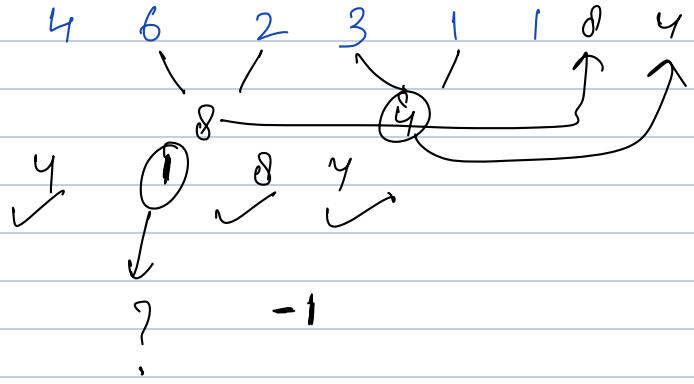
⇒ 4 4 2 2 4

4

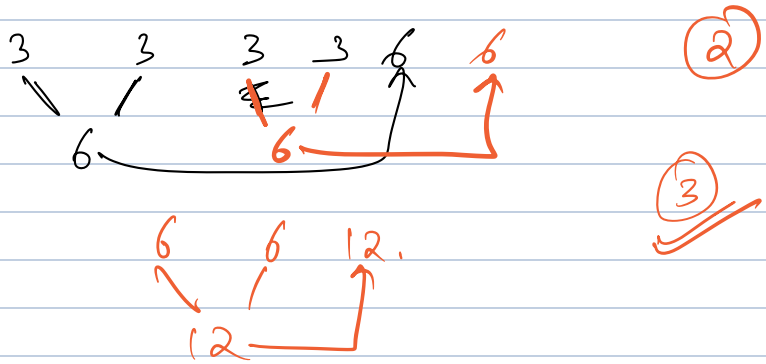
4 4 4 4.

return total no of steps req to do this
if impossible then return -1; only +ve nos.

ex.



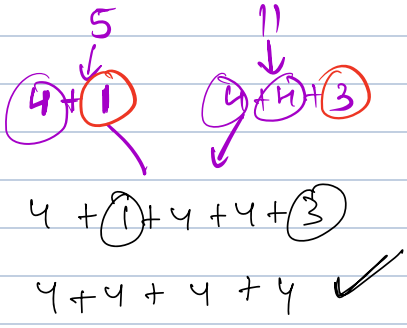

⇒ ex




1.) Calc mod 4 for each element.

arr[i] = arr[i] % 4;

0
1
2
3



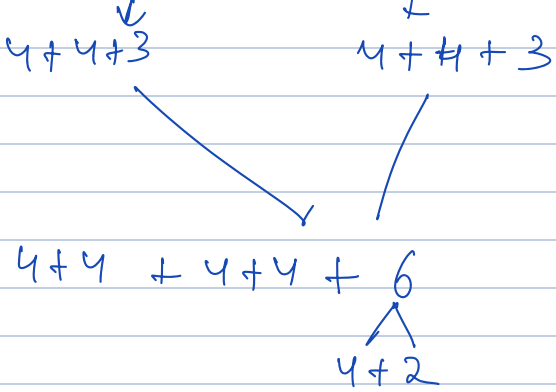

2.) Sum the elemnt [mod 1 & mod 3]
3 2

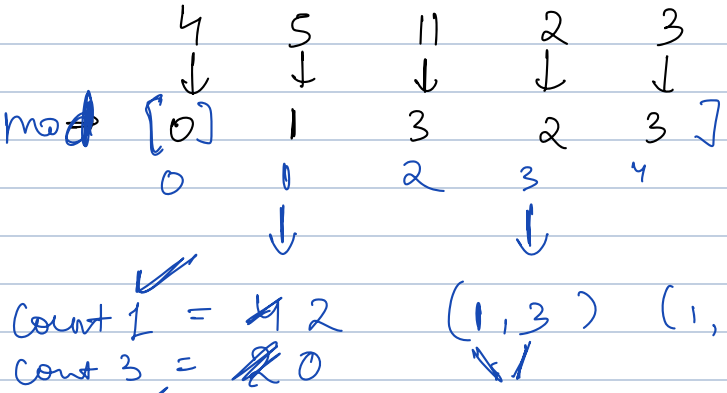
count = count + 2. a mod 2 ✓

3.) Sum elemens [mod 3 & mod 3] and
[mod 1 & mod 1]
b mod 2 ✓

count = count + pair.

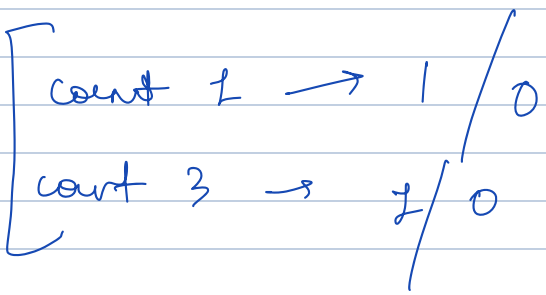
Count mod[2] += a+b.

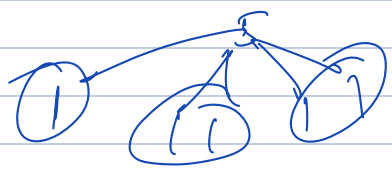


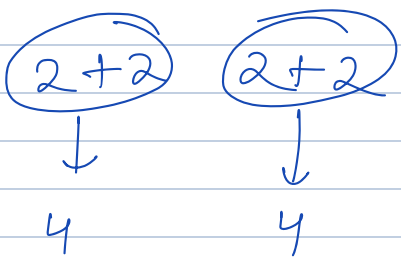

1 + 1 → %4 = 2.

count 2 ++;



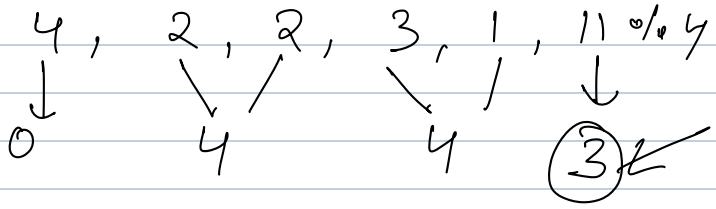

count 2. <



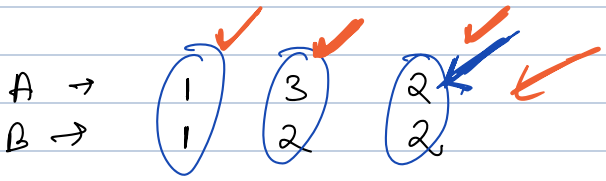

4.). Lala the count 2 is even., array will be div by 4

" " odd, retun -1;

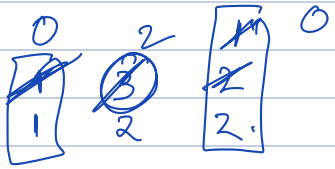
count 1 or count 3 or cont 2 ! = 0
retun -1;




Q. Little Penny & Tax Saving.




ex.

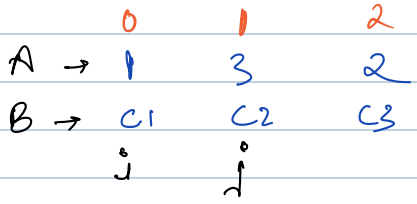



d1 Tax = 1 + 2 + 2 = 5
d2 Tax = 1 + 2 + 2 = 5
d3 = 1 + 2 + 2 = 5
d4. 2 + 2 = 4
d5 = 2
d6 = 2

= 23.

(1)

Observation → select 1 acct & empty it completely first.




lets have 2 acct, see which one to choose.

acct 1

i

A[i] * (B[i] + B[j]) + A[j] * B[j]

acct 2.

j

A[j] * (B[j] + B[i]) + A[i] * B[i]

A[i] * B[i] + A[i] * B[j] + A[j] * B[j]

A[j] * B[j] + A[j] * B[i] + A[i] * B[i]

A[i] * B[j] < A[j] * B[i]

Sorted.

A: 2 1 3

B: 2 1 2

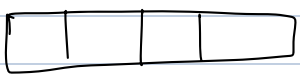
2 + 1 + 2

2 × 2 + 1 × (2+1) + (2+1+3) × 2.

= 19.

```
for(i=0; i<n; i++) {
    sum += A[i]
    tax = tax + (B[i] * sum).
}
```

O(N)



indexes array

TC → O(N log N) + O(N)

= O(N log N).

SC → O(N).

Q5. Array A. Find $B^{th}$ smallest element from the sum of all the triplets in A.

A → | 2 | 4 | 3 | 2 |

B = 3

All triplets -

2 4 3 → 9

2 4 2 → 8

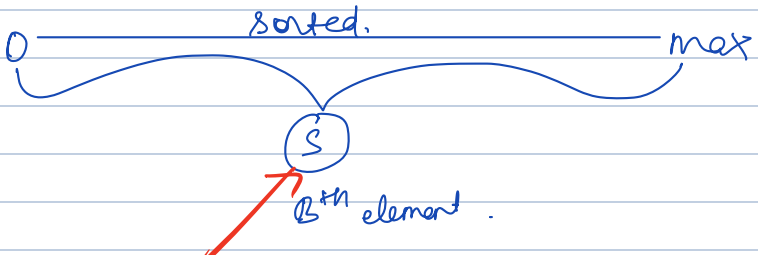
2 3 2 → 7

4 3 2 → 9

7 8 9 9

1st 2nd 3rd

$O(n^3) + n\log n$




mid element
if it has exact (B-1) elements less than itself

count > B.
B+5 < mid.

$\leq$ B-1 element.

1.) Sort the array
2.) Use 2 pointer approach to find no of triplet sum less than value mid.

A → | 2 | 4 | 3 | 2 |

B = 3

All triplets -
2 4 3 → 9
2 4 2 → 8
2 3 2 → 7
4 3 2 → 9

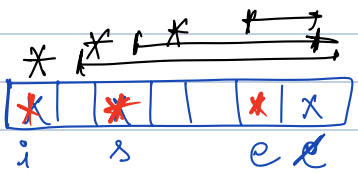
— 7 8 9 9
1st 2nd 3rd

sorted. 2 2 3 4
s e

mid = 7

sum = arr[i] + arr[s] + arr[e]

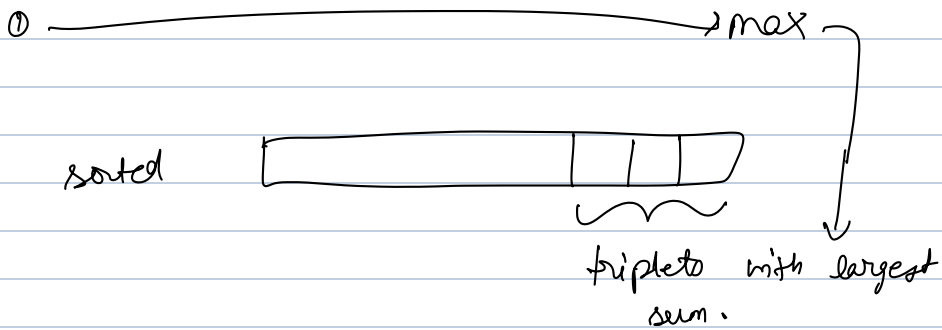
```
if(sum > mid.) {
    e--;
}
else {
```




sum < ~~to~~ mid.
Count = count + e - s.

$O(N^2)$

```
func(          ) {
for(int i = 0; i < n; i++) {
    int s = i+1, e = n;
    while (s < e) {
        if (A[i] + A[s] + A[e] < mid {
            count += e - s;
        }
        else
            e--;
    }
return count
```

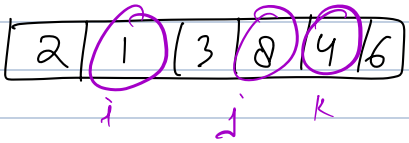


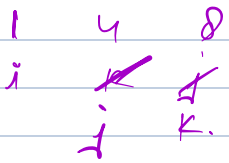

low = 0 , high = A[n-1] + A[n-2] + A[n-3]

```
while (low <= high) {
    int mid = low + (high - low)/2;
// count no of triplets sum less than mid.
    int count = func(mid, A);

    if (count == B-1)
        return mid;

    else if (count >= B)
        high = mid - 1;
    else {
        ans = mid
        low = mid + 1;
    }
return ans;
```


2
1
3
8
4
6
i
j
k


1
4
8
i
k
j
k