★ Sum of array Except Self :

eg: [4, 3, 2, 10]

op: [3+2+10, 4+2+10, 4+3+10, 4+3+2]

=> [15, 16, 17, 9]

① Find total sum = 4+3+2+10 = 19

② go to every element and total − arr[i]

[4, 3, 2, 10]
0 1 2 3
↑ ↑ ↑ ↑

(4+3+2+10) − 4
=> 3+2+10
=> 15

19 − 3 = 16

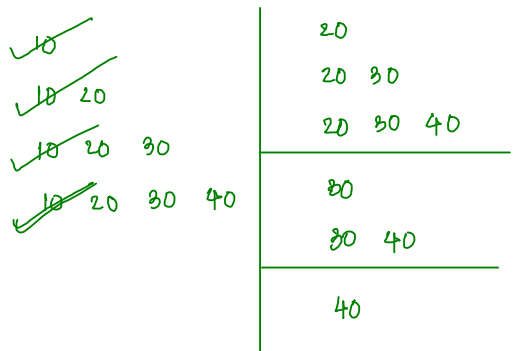
19 − 2 = 17

19 − 10 = 9

res = ~~[]~~
~~[15]~~
~~[15, 16]~~
~~[15, 16, 17]~~
[15, 16, 17, 9]

★ Generate subarrays : (any continuous segment of the arr)

Eg: [10, 20, 30, 40]
0 1 2 3



Q: How to generate all subarrays starting with arr[0] = 10 ?

```
let subarr = "";
for(let i=0; i<n; i++) {
    subarr += arr[i] + " ";
    cl(subarr);
}
```

subarr = "" "10 " "10 20 " "10 20 30 "

[10, 20, 30, 40] "10 20 30 40 "
↑ ↑ ↑ ↑

```
for (let start = 0; start < n; start++) {
  let subarr = "";
  for (let end = start; end < n; end++) {
    subarr += arr[end] + " ";
    console.log(start, end, subarr);
  }
}
```

[10, 20, 30, 40]

s e

Op:
0 0 10
0 1 10 20
0 2 10 20 30
0 3 10 20 30 40
1 1 20
1 2 20 30
1 3 20 30 40
2 2 30
2 3 30 40
3 3 40

① start = 0 ~~<10 20 30 40>~~
→ subarr = ~~<10>~~ ~~<10 20>~~ ~~<10 20 30>~~
→ end = start = 0
→ end = 1
→ end = 2
→ end = 3
→ end = 4 X

② start = 1 ~~<20 30 40>~~ ~~<20 30>~~
→ subarr = ~~<20>~~
→ end = start = 1
→ end = 2
→ end = 3
→ end = 4 X

③ start = 2 ~~<30 40>~~
→ subarr = ~~<30>~~
→ end = start = 2
→ end = 3
→ end = 4 X

④ start = 3
→ subarr = ~~<40>~~
→ end = 5 ⇒ 3
→ end = 4 X

⑤ start = 4 X

★ Subarray with zero sum:

Eg: [3, 4, -7, 3, 1, 3]
0 1 2 3 4 5

Op: subarray from 0 to 2
subary from 1 to 3
subary from 2 to 5

3 4 -7 3 1 3
7 -3 -4 4 4
0 0 -3 7
3 1 0
4 4
7

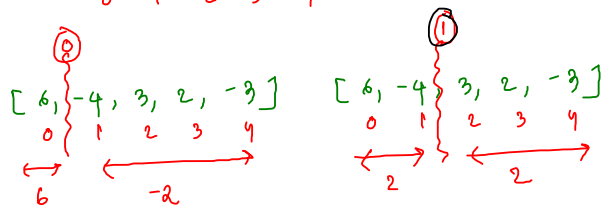
we know generating subarays,
here we need generating subary sum.

```
for(let start =0; start <n; start++) {
    let sum=0;
    for(let end =start; end <n; end++) {
        sum += ar[end];
        cl(sum);
    }
}
```

★ Find split point : n=5

eg: [6, -4, 3, 2, -3] op: 1

0 1 2 3 4




```
for(split = 0; split < n-1; split++) {
    const lsum = sum(ar, 0, split);
    const rsum = sum(ar, split+1, n-1);
    if (lsum == rsum) {
        return split;
    }
}
return -1;
```

★ I dont need this
★ I can use prefixsum as my left sum.

≠ optimal :

① lets run prefix sum, and observe

```
sum=0;
for (let split = 0; split < N-1; split++){
    sum += a1[split];
    cl(sum);
}
```

acting like your left sum

Sum = 0

[6, -4, 3, 2, -3]

0 1 2 3

6 2 5 7

① sum += 6 → 6

② sum += (-4) → 2

③ sum += 3 → 5

④ sum += 2 → 7

[6, -4, 3, 2, -3]

6

[6, -4, 3, 2, -3]

2

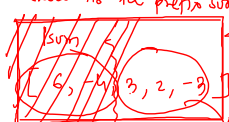
[6, -4, 3, 2, -3]

5

[6, -4, 3, 2, -3]

7

② Now lsum we know its the prefix sum, how to get the rsum?



rsum = total - lsum

⇒ (6-4+3+2-3)

− (6-4)

⇒ (3+2-3) rsum

suffix sum ⇒ totalsum − prefix sum

prefix = a+b+c

c+d+e+f suffix

total = a+b+c+d+e+f

★ Geometric triplets :

Q: what is G.P ?

Eg: 2, 4, 8, 16, 32, ------

$4/2 = 2$ $8/4 = 2$ $16/8 = 2$ $32/16 = 2 = R$

Eg: 3, 9, 27, 81, 243, ---

3 3 3 3 = R

a, b, c, d, e, ...

$\Rightarrow b/a = c/b = d/c = e/d = R$

0 1 2 3 4 5

(1, 2, 6, 10, 18, 54)

2, 6, 10

2, 6, 18

2, 6, 54

6, 18, 54 So on

① generate triplets,

(1, 2, 6) $\longrightarrow$ $2/1 = 6/1 \Rightarrow X$

(1, 2, 10) $\longrightarrow$ $2/1 = 10/2 \Rightarrow X$

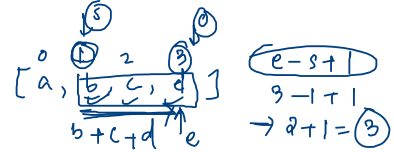
② check (a, b, c) are in G.P

b/a ~~$=$~~ c/b

$\Rightarrow b * b = a * c$

$\Rightarrow b^2 = ac$

★ Find subarrays with given sum and given length :



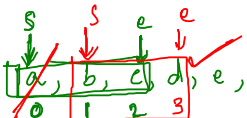
eg: [2, 2, 1, 3, 2]

d = 4 ⇒ sum = birthday

m = 2 ⇒ len = birth month

op: 2 [2, 2]

and [1, 3]



len = 3 ✓

sum = a+b+c

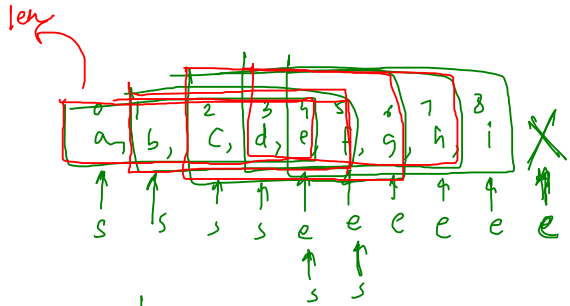
sum = a+b+c − a + d

= b+c+d

sum -= ar[s];
s++;
e++;
sum += ar[e];

```
for (let start = 0; start < n; start++) {
    let sum = 0;
    for (let end = start; end < n; end++) {
        sum += ar[end];
        if (sum == d && end-start+1 == m) {
            cnt++;
        }
    }
}
```

→ Optimise using two pointer (HW)



len = 5

Sum = a+b+c+d+e

$\Rightarrow (a+b+c+d+e) - a + f$

$\Rightarrow (b+c+d+e+f) - b + g$

$\rightarrow c+d+e+f+g$

★ Sliding window Technique

(helpful to focus on the given length of subarrays instead of generating all subarrays and then picking the given length)