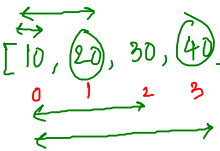
★ Generate Subarray : [any continuous segment of array]

eg : [10, 20, 30, 40]



10	20
10 20	20 30
10 20 30	20 30 40
10 20 30 40	30
	30 40
	40

20, 40 ⇒ is not a subarray because they are not continuous

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

★ using running stream.

```
let subar = "";
for(let i = 0; i<n; i++) {
    subar = subar + arr[i] + " ";
    console.log(subar);
}
```

~~""~~

~~"10"~~ ⟶ cl("10")

~~"10 20"~~ ⟶ cl("10 20")

~~"10 20 30"~~ ⟶ cl("10 20 30")

"10 20 30 40" ⟶ cl("10 20 30 40")

Q: Now generate all subarrays using previous slide logic.

```
805  for (let start = 0; start < n; start++) {
806    let subarr = "";
807    for (let end = start; end < n; end++) {
808      subarr += arr[end] + " ";
809      console.log(start, end, subarr);
810    }
811  }
```

① start = 0

→ subarr = " "

→ end = start = 0, subarr += arr[0] + " " = "10 "

→ end = 1, subarr += arr[1] + " " = "10 20 "

→ end = 2, subarr += arr[2] + " " = "10 20 30 "

→ end = 3, subarr += arr[3] + " " = "10 20 30 40 "

② start = 1

→ subarr = " "

→ end = start = 1, subarr += arr[1] + " "
= "20 "

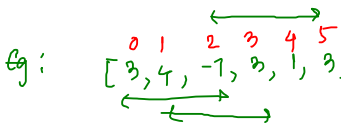
→ end = 2, subarr += arr[2] + " "
= "20 30 "

→ end = 3, subarr += arr[3] + " "
= "20 30 40 "

③ start = 2 { 30 }

④ start = 3 { 40 }

* Zero Sum Subarrays :



→ Instead of running string for printing subarray to get sum of each subarray lets use running sum.

```
function zeroSubarray(arr) {
  //Write your code here
  const n = arr.length;
  let isFound = false; // I did not find any subarr with 0 sum

  for (let start = 0; start < n; start++) {
    let sum = 0;
    for (let end = start; end < n; end++) {
      sum += arr[end];
      // at every point sum => sum of arr[start....end]
      if (sum == 0) {
        console.log(`Subarray found from Index ${start} to ${end}`);
        isFound = true;
      }
    }
  }

  if (isFound == false) {
    console.log(-1);
  }
}
```

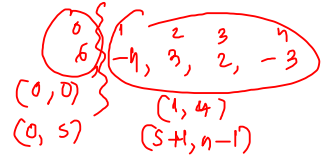
★ Find split point :

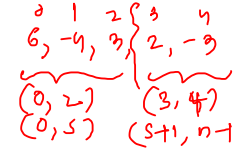
eg: [6, -4, 3, 2, -3]

op: 1

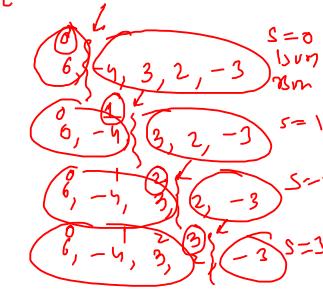
eg: [100, 10, 20, 30, 40]

op: 0






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



efficient way :

[a, b, c, d, e]
 0 1 2 3 4

① s=0, 0→0 ↷ 1, 1→4 ↷ 4 } ⑤

② s=1, 0→1 ↷ 2, 2→4 ↷ 3 } ⑤

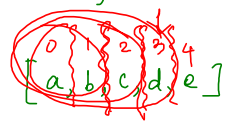
③ s=2, 0→2 ↷ 3, 3→4 ↷ 2 } ⑤

④ s=3, 0→3 ↷ 4, 4→4 ↷ 1 } ⑤

★ total = 5+5+5+5 = 20
= (n-1) * n Iterations
= 4 * 5 = 20 Iterations

① totalsum ⇒ a+b+c+d+e

② runningsum



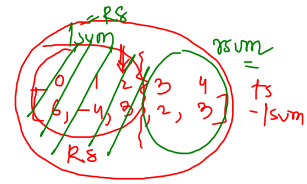
★ Rs will give you the leftsum

① s = 0
Rs += arr[s]
Rs = a
rsum = total - Rs
= a+b+c+d+e - a
= b+c+d+e

② s = 1
Rs += arr[s]
Rs = a+b
rsum = a+b+c+d+e - (a+b)
= c+d+e

③ s = 2
Rs += arr[s]
Rs = a+b+c
rsum = a+b+c+d+e - (a+b+c)
= d+e

④ s = 3
Rs += arr[s]
Rs = a+b+c+d
rsum = e
(a+b+c+d+e) - (a+b+c+d)



* Geometric triplets :

q: what is G.P ?

eg: 2, 4, 8, 16, 32, 64, . . .

$4/2 = 2$ 2 2 2 2

a, b, c, d, e

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

eg: 3, 9, 27, 81, 243

3 3 3 3

0 1 2 3 4 5

[1, 2, 6, 10, 18, 54]

1, 2, 6 $\Rightarrow$ $2/1 == 6/2$ ✓

2, 6, 18 $\Rightarrow$ $6/2 == 18/6$ $(3 == 3)$

op: print all such triplets

a, b, c $\longrightarrow$ G.P arr[i], arr[j], arr[k]

$b/a == c/b$

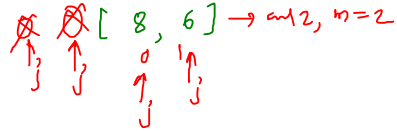
$\Rightarrow \boxed{b^2 = ac}$

$\frac{arr[j]}{arr[i]}$ $\times$ $\frac{arr[k]}{arr[j]}$

arr[j] * arr[j] == arr[i] * arr[k]

★ Array Subtraction :

Eg: [1, 2, 3, 4] → arr1, n = 4



[8, 6] → arr2, m = 2

res = [] [3] [3,4] [3,4,1] [3,1,1,1]

c = 0 → 0

diff = (arr1[i] + carry) - arr2[j]

= (4 + 0) - 6

= -2

diff < 0 ⇒ c = -1, diff = -2 + 10 = 8

```
function findSubtraction(a, n, b, m) {
  //Write your code here
  const res = [];
  let carry = 0;
  let i = n - 1;
  let j = m - 1;

  while (i >= 0 || j >= 0) {
    // let diff = (arr[i] + carry) - arr[j];
    let diff = a[i] + carry;
    if (j >= 0) diff -= b[j];

    if (diff < 0) {
      carry = -1;
      diff += 10;
    } else { carry = 0; }
    res.push(diff);
    i--;
    j--;
  }
}
```

diff = arr1[1] + carry

= 2 + (-1)

= 1

c = 0

diff = arr1[?] + carry

= 1 + 0

= 0

c = 0

diff = (arr1[i] + carry) - arr2[j]

= (3 + (-1)) - 8 = (3 - 1) - 8

= 2 - 8 = -6

diff < 0, c = -1, diff = -6 + 10 = 4

Eg: [1, 2, 3, 4] → arr1

[5, 6, 7, 8] → arr2

* num(arr1) < num(arr2)

1. swap
2. subtract → same
3. multiply -1

① arr1 → [5, 6, 7, 8]

arr2 → [1, 2, 3, 4]

```
  5  6  7  8
- 1  2  3  4
-------------
  4  4  4  4
```

⇒ [4, 4, 4, 4]

⇒ [-4, 4, 4, 4]

② multiply -1,

⇒ -4444

Eg:

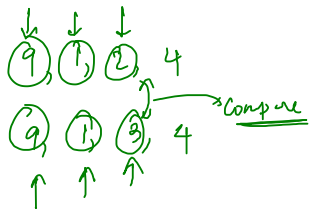
```
      0  1  2
   [1, 2, 3]  → arr1
[1, 2, 3, 4]  → arr2
 0  1  2  3
```

arr1 → [1, 2, 3, 4]

arr2 → [1, 2, 3]

```
(-)  1  1  1  1
```

Q: How to check num(arr1) < num(arr2)?



1. find 1st non equal elements

```
if (arr1[i] < arr2[i])
    return true
else
    return false
```

* len(arr1) > len(arr2)

return false

len(arr1) < len(arr2)

return true

9 < 3

⇒ num(a) < num(b)

9 1 3 4 → a

9 1 2 4 → b

⇒ 3 < 2 → false

⇒ num(a) > num(b)

9 9 9 9 → n=4

9 9 9 9 → m=4