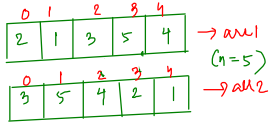
★ Adding two Arrays :

Eg :




```
for (i = n-1 → 0) {
    res[i] = arr1[i] + arr2[i]
}
```

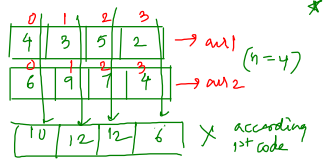
res[n-1] = arr[n-1] + arr2[n-1]

= arr[4] + arr2[4]

= 4 + 1

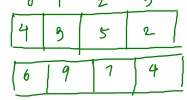
res[4] = 5

Eg :



★ we do not iterate on arrays, we are actually iterating on some indices.

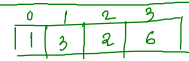
```
let carry = 0;
for (let i = n-1; i>=0; i--) {
    const sum = arr1[i] + arr2[i] + carry;
    res[i] = sum % 10;
    carry = sum/10;
}
```



① sum = 2+4+0
= 6
6%10 = (4) → res
6/10 = (0) → Carr

② Sum = 5+7+0
= 12
12%10 = (2) → res
12/10 = (1) → c

res :

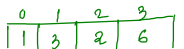


③ sum = 3+9+1
= 13
13%10 = (3) → res
13/10 = (1) → carry

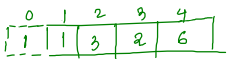
④ Sum = 4+6+1
= 11
11%10 = (1) → res
11/10 = (1) → carry

If there carry left after the process,

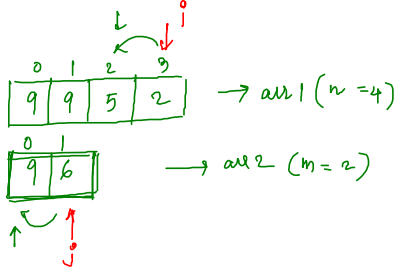
```
if (carry == 1) {
    add (1) to the start of the array
}
```



res =



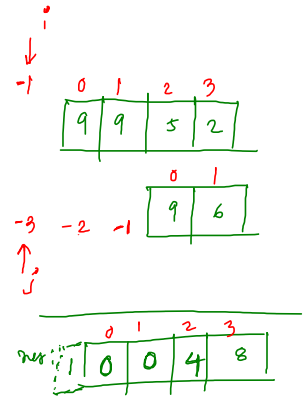
eg:




```
i = n-1   carry = 0
j = m-1
while ( i >= 0 ) {
  let sum;
```

```
if (j>=0) sum = arr1[i] + arr2[j] + carry;
else sum = arr1[i] + carry;

  res[i] = sum % 10;
  carry = sum/10;
  i--;
  j--;
}
```

9 9 5 2
+ 9 6



(1) Sum = 9+6+0 = 8
8 % 10 = 8 → rem
8/10 = 0 → carry

(2) sum = 5+9+0 = 14
14 % 10 = 4 → rem
14/10 = 1 → carry

(3) Sum = 9+1 = 10
10 % 10 = 0 → rem
10/10 = 1 → C

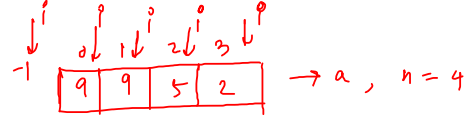
(4) Sum = 9+1 = 10
10 % 10 = 0 → rem
10/10 = 1 → carry

```
// 1. initialize / create empty result array
const res = [];
for (let i = 0; i < n; i++) {
  res.push(0);
}

// 2. start adding from last
let i = n - 1;
let j = m - 1;
let carry = 0;

while (i >= 0) {
  let sum = j >= 0 ? a[i] + b[j] + carry : a[i] + carry;
  res[i] = sum % 10;
  carry = sum / 10;

  i--;
  j--;
}

// 3. If there is carry left, add 1 to the beginning of array
res.unshift(1);
```

C = True

C = False

if (carry == 1)



1. res = [] [0] [0,0] [0,0,0] [0,0,0,0]

0 0 4 8

2. i = 4-1 = 3, j = 2-1 = 1, c = 0

① sum = 9+6+0 = 8
res[3] = 8 % 10 = 8
c = 8/10 = 0

res [0,0,4,8]

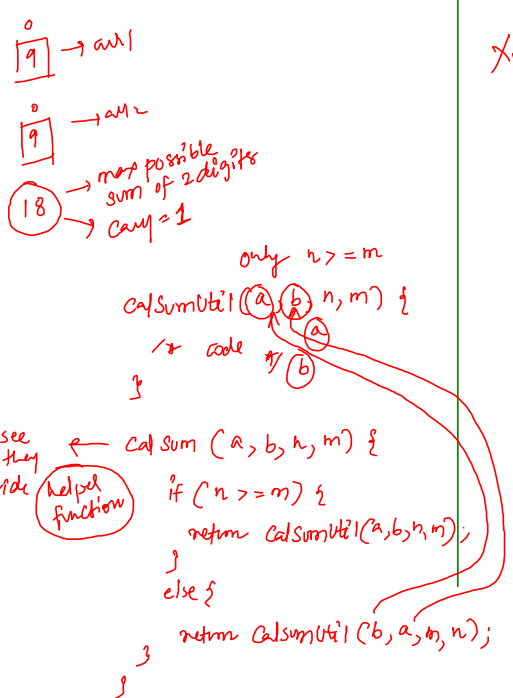
② sum = 5+9+0 = 14
res[2] = 14 % 10 = 4
c = 14/10 = 1

③ sum = 9+1 = 10
res[1] = 10 % 10 = 0
c = 10/10 = 1

④ sum = 9+1 = 10
res[0] = 10 % 10 = 0
c = 10/10 = 1

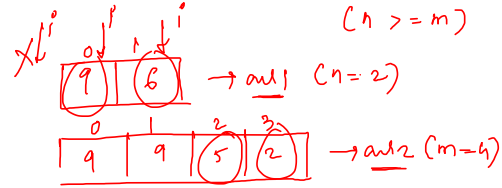
res [1,0,0,4,8]

* carry can never be > 1 (it is always 0 or 1)



* our previous code assumes arr1 >= arr2 (n >= m)

len len



res = [0, 0]

= [1, 4, 8] ✓

a = [9, 6] n = 2

b = [9, 9, 5, 2] m = 4

9:20 pm - 9:35 pm

BREAK

```
arr = [ 1, 1, 2, 3 ]
      0  1  2  3
```

arr.unshift(20) → adds an ele to beginning of array

```
[ 20, 1, 1, 2, 3 ]
  0   1  2  3  4
```

arr.unshift(4)

```
[ 4, 20, 1, 1, 2, 3 ]
  0   1   2  3  4  5
```

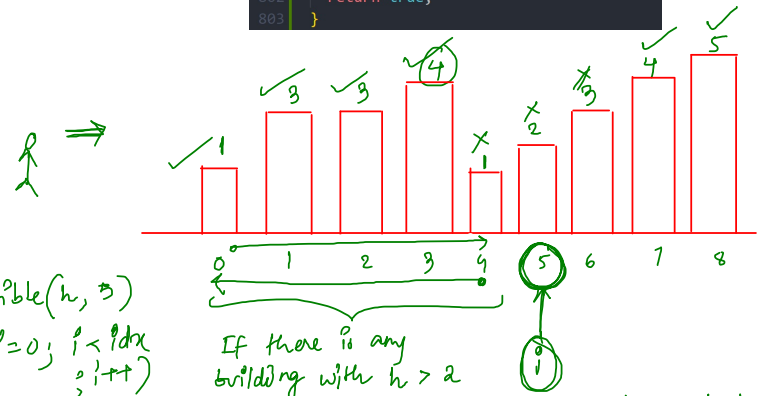
★ Buildings :

```
function isVisible(heights, idx) {
  for(let i = idx - 1; i >= 0; i--) {
    if(heights[i] > heights[idx]) {
      return false;
    }
  }
  return true;
}
```

0 1 2 3 4 5 6 7 8

ip: [1, 3, 3, 4, 1, 2, 3, 4, 5]

op: 6



isVisible(h, s)

for(i=0; i<idx; i++)

If there is any building with h > 2 then I cannot see

★ go to every element,

if (see) cnt++;

how to decide?

(how can you say the building is hidden or not)

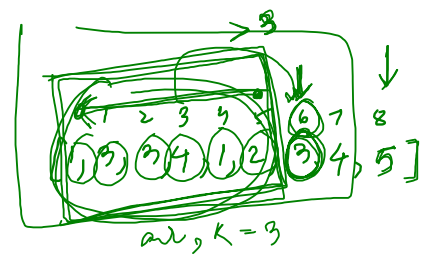
0 1 2 3 4 5 6 7 8

[1, 3, 3, 4, 1, 2, 3, 4, 5]

✓ ✓ ✓ ✓ × × × ✓ ✓

cnt = 0 1 2 3 4 5 6

```javascript
function isVisible(heights, idx) {
  for (let i = idx - 1; i >= 0; i--) {
    if (heights[i] > heights[idx]) {
      return false;
    }
  }
  return true;
}

function countVisibleRoofs(heights) {
  // Write your code here
  const n = heights.length;
  let cnt = 0;
  for (let i = 0; i < n; i++) {
    if (isVisible(heights, i)) {
      cnt++;
    }
  }

  return cnt;
}
```



i = 7
4 > 5 (2) 3 > 4
i = 6 3 > 5 (1) 3 > 4
i = 5 2 > 4 (0) 1 > 4
i = 4 1 > 5
i = 3 4 > 5

⇒ given an array check whether is there any element greater than some K

(heights, idx) → (h, 6)

(1) i = 3
h[3] > h[6]
(4) > (3) → I can't see 6th ? idx building
blocked

(1) i = 5
h[5] > h[6] (2 > 3)

(2) i = 4
h[4] > h[6] (1 > 3)

Improve :

```
for(going to every ele) {
    isVisible() ⟶ (iterating on left part)
}
```

```
i = 0,  isVisible() → 0
i = 1,  isVisible() → 1
i = 2,  isVisible() → 2
i = 3,  3
i = 4,  4
  .
  .
  .
i = n-1,  n-1
```

$0 + 1 + 2 + 3 + 4 + \cdots + n-1$

$\Rightarrow \frac{n(n-1)}{2} \Rightarrow \frac{n^2 - n}{2}$

$\simeq n^2$ instructions / iterations

running maximum :

If tallest > Current
⇒ you cannot see

* If tallest building itself cannot block me then no other building on left will block me.

* Using RM, there is no need of calculating max again & again

Eg :

0	1	2	3	4	5
1	4	2	3	6	5

runningMax = $-\infty$ → This will tell you the max^e on left part

$1 > -\infty$	✓
$4 > 1$	✓
$6 > 4$	✓

cnt = 0 ✗ ✗ 3

① $-\infty > 1$ → no ✓

② $1 > 4$ → no ✓

③ $4 > 3$ → yes ✗

④ $4 > 3$ → yes ✗

⑤ $4 > 6$ → NO ✓

⑥ $6 > 5$ → yes ✗

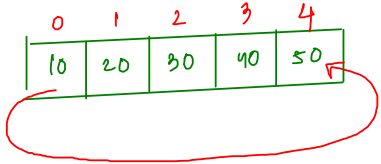
★ running stream Technique

(max/min/sum/prod)

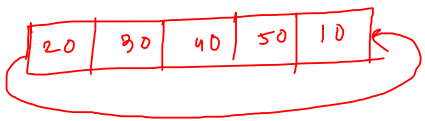
```
function countVisibleRoofs(heights) {
  // Write your code here
  const n = heights.length;
  let cnt = 0;
  let runningMax = -Infinity;
  for (let i = 0; i < n; i++) {
    if (runningMax <= heights[i]) {
      cnt++;
      runningMax = heights[i];
    }
  }

  return cnt;
}
```

* Rotate Array :

eg :

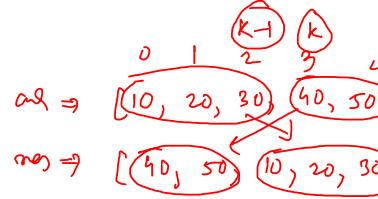


k = 3

① 

② 

③ 



① (k → n-1) (push to res) (3 → 4)

② (0 → k-1) (push to res) (0 → 2)

res = [ ] ; * k = k % n

```
for(let i = k; i < n; i++) {
    res.push(arr[i]);
}
for(let i = 0; i < k; i++) {
    res.push(arr[i]);
}
```

res = [ ]
= [10]
= [40, 50]
= [40, 50, 10]
= [40, 50, 10, 20]
= [40, 50, 10, 20, 30]

* what if $k >= n$?

$k = 100;$

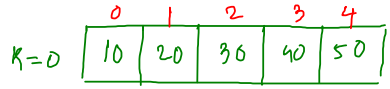
$n = 5$ [10, 20, 30, 40, 50]

for (k → n-1)

⇒ (100 → 4) ✗

for (0 → k-1)

⇒ (0 → 99) ✓



n = 5

K=1 → 20 30 40 50 10

K=2 → 30 40 50 10 20

K=3 → 40 50 10 20 30

K=4 → 50 10 20 30 40

K=5 → 10 20 30 40 50

K=6 → 20 30 40 50 10

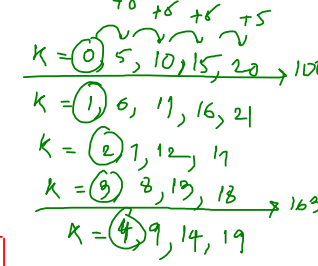
K=7 → 30 40 50 10 20

K=8 → 40 50 10 20 30

K=9 → 50 10 20 30 40

K=10 → 10 20 30 40 50

K=11 → 20 30 40 50 10



K = 100

K = K % n

= 100 % 5

= 0

K = 103

= 103 % 5

= 3

Improve :

1st code was taking (n) iteration, we have to do inplace (in original array)

arr = [10, 20, 30, 40, 50] n = 5, k = 3

① reverse the entire array

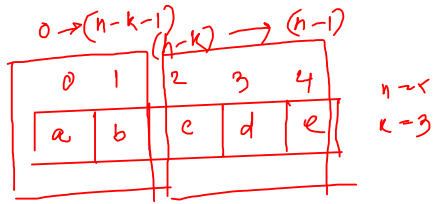
[50, 40, 30, 20, 10]

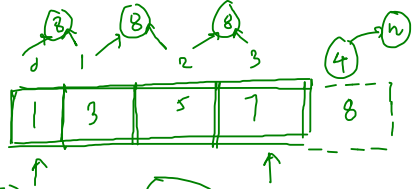
② reverse 1st n-k elements (5-3 = 2)

[40, 50, 30, 20, 10]

③ reverse last k elements (3)

[40, 50, 10, 20, 30]





i=0 1<8 and 8<3

i=1 3<8 and 8<5

i=2 5<8 and 8<7

return i+1;

8

i>=4

3>=4

5>=4

i>=0

4

```
if (b <= a[0]) return 0;
if (b > a[n-1]) return n;
for (i=0; i<n-1; i++) {
    if (b <= a[i+1]) {
        return i+1;
    }
}
```

```
if (a[i] >= b) {
    return i;
}
```