Arrays

Introduction

Arrays: Array Advance Game

Array Advance Game

You are given an array of non-negative integers. For example:

[3,3,1,0,2,0,1]

Each number represents the maximum you can advance in the array.

Array Advance Game

You are given an array of non-negative integers. For example:

[3,3,1,0,2,0,1]

Each number represents the maximum you can advance in the array.

Question:

Is it possible to advance from the start of the array to the last element?

Array Advance Game: Example

A = [3,3,1,0,2,0,1]

Array Advance Game: Example

A = [3,3,1,0,2,0,1]

1. From A[0] move 1 position forward.

Array Advance Game: Example

A = [3,3,1,0,2,0,1]

1. From A[0] move 1 position forward.
2. From A[1] move 3 positions forward.

Array Advance Game: Example

A = [3,3,1,0,2,0,1]

1. From A[0] move 1 position forward.
2. From A[1] move 3 positions forward.
3. From A[4] move 2 positions forward.

Array Advance Game: Example

A = [3,3,1,0,2,0,1]

1. From A[0] move 1 position forward.
2. From A[1] move 3 positions forward.
3. From A[4] move 2 positions forward.
4. Success.

Array Advance Game: Example (Unwinnable)

A = [3,2,0,0,2,0,1]

Array Advance Game: Example (Unwinnable)

A = [3,2,0,0,2,0,1]

[3,2,0,0,2,0,1]

Array Advance Game: Example (Unwinnable)

A = [3,2,0,0,2,0,1]

[3,2,0,0,2,0,1]

Array Advance Game: Example (Unwinnable)

A = [3,2,0,0,2,0,1]

[3,2,0,0,2,0,1]
[3,2,0,0,2,0,1]

Array Advance Game: Example (Unwinnable)

A = [3,2,0,0,2,0,1]

[3,2,0,0,2,0,1]
[3,2,0,0,2,0,1]

Array Advance Game: Example (Approach 1)

A = [2,4,1,1,0,2,3]

Use “greedy” strategy. Advance as much as possible for each number. (Note this will not work).

Array Advance Game: Example (Approach 1)

A = [2,4,1,1,0,2,3]

Idea: Use “greedy” strategy. Advance as much as possible for each number.

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

Array Advance Game: Example (Approach 1)

A = [2,4,1,1,0,2,3]

The greedy approach does not work in this instance. Note:

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

Array Advance Game: Example (Approach 2)

- Iterate through each entry in array.

- Track furthest we can reach from entry (A[i] + i)

- If for some “i” before the end is the furthest that we can reach, we can’t reach the last index. Otherwise, the end is reached.

- i : index processed
- Furthest possible to advance from “i”: A[i] + i

Array Advance Game: Example (Approach 2)

[3,3,1,0,2,0,1]

furthest_reached = 0

Array Advance Game: Example (Approach 2)

[3,3,1,0,2,0,1]

i = 0

furthest_reached = max(furthest_reached, A[0] + 0) = 3

Array Advance Game: Example (Approach 2)

[3,3,1,0,2,0,1]

i = 1
furthest_reached = max(furthest_reached, A[1] + 1) = 4

Array Advance Game: Example (Approach 2)

[3,3,1,0,2,0,1]

i = 2
furthest_reached = max(furthest_reached, A[2] + 2) = 4

Array Advance Game: Example (Approach 2)

[3,3,1,0,2,0,1]

i = 3
furthest_reached = max(furthest_reached, A[3] + 3) = 4

Array Advance Game: Example (Approach 2)

[3,3,1,0,2,0,1]

i = 4

furthest_reached = max(furthest_reached, A[4] + 4) = 6

Array Advance Game: Example (Approach 2)

[3,3,1,0,2,0,1]

i = 5

furthest_reached = max(furthest_reached, A[5] + 5) = 6

Array Advance Game: Example (Approach 2)

[3,2,0,0,2,0,1]

Array Advance Game: Example (Approach 2)

[3,2,0,0,2,0,1]

i = 0

furthest_reached = max(furthest_reached, A[0] + 0) = 3

Array Advance Game: Example (Approach 2)

[3,2,0,0,2,0,1]

i = 1

furthest_reached = max(furthest_reached, A[1] + 1) = 3

Array Advance Game: Example (Approach 2)

[3,2,0,0,2,0,1]

i = 2

furthest_reached = max(furthest_reached, A[2] + 2) = 3

Array Advance Game: Example (Approach 2)

[3,2,0,0,2,0,1]

i = 3

furthest_reached = max(furthest_reached, A[3] + 3) = 3

Array Advance Game: Example (Approach 2)

[3,2,0,0,2,0,1]

i = 4

furthest_reached = max(furthest_reached, A[4] + 4) = 3

i > furthest_reached -- i.e. end is not reachable.

Arrays: Arbitrary-Precision Increment

Arbitrary Precision Increment

Given:

An array of non-negative digits that represent a decimal integer.

Problem:

Add one to the integer. Assume the solution still works even if implemented in a language with finite-precision arithmetic.

Arbitrary Precision Increment

A =

1	4	9

This array represents the digit 149.
Adding 1 to 149 gives us the updated array:

A =

1	5	0

Arbitrary Precision Increment:

A =

1	4	9

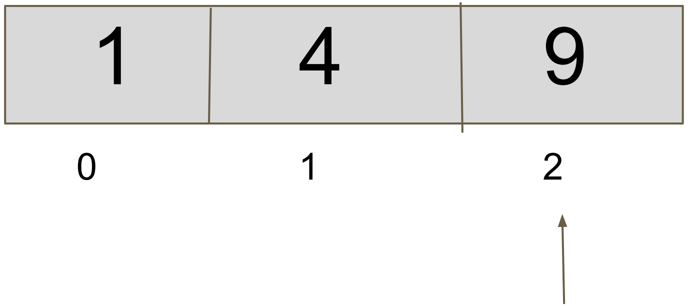
Algorithm:

- Add 1 to rightmost digit.
- Propagate carry throughout array.

This will be similar to the “standard grade school” approach.

Arbitrary Precision Increment: Example 1

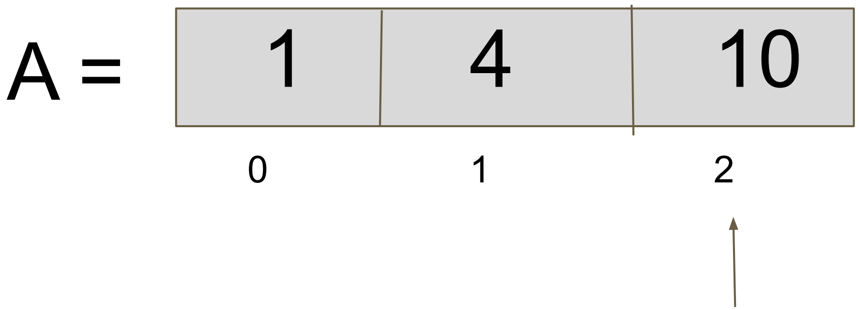
A =



1	4	9
0	1	2

Add 1 to rightmost digit.

Arbitrary Precision Increment: Example 1

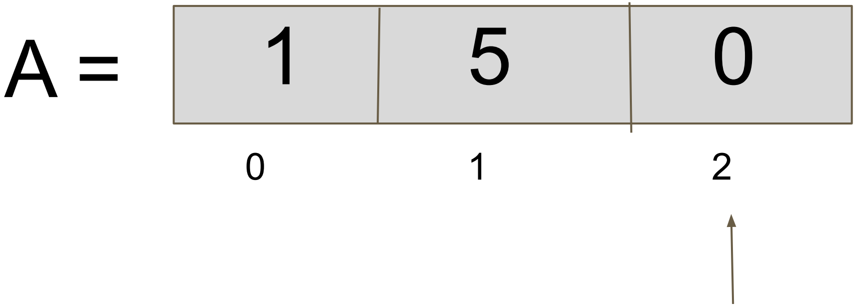


A =

1	4	10
0	1	2

If sum yields a 10, replace with 0 and add a carry of 1 over to the left.

Arbitrary Precision Increment: Example 1

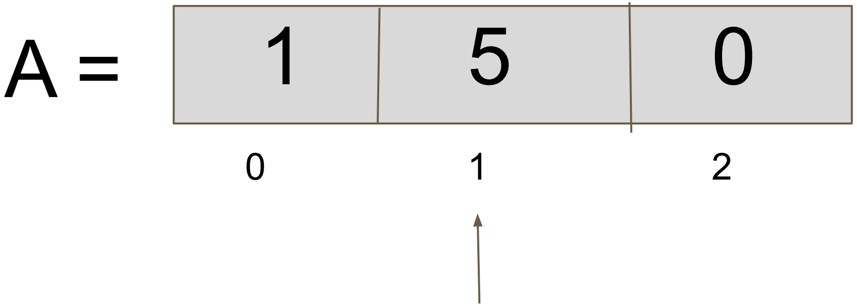


A =

1	5	0
0	1	2

If sum yields a 10, replace with 0 and add a carry of 1 over to the left.

Arbitrary Precision Increment: Example 1

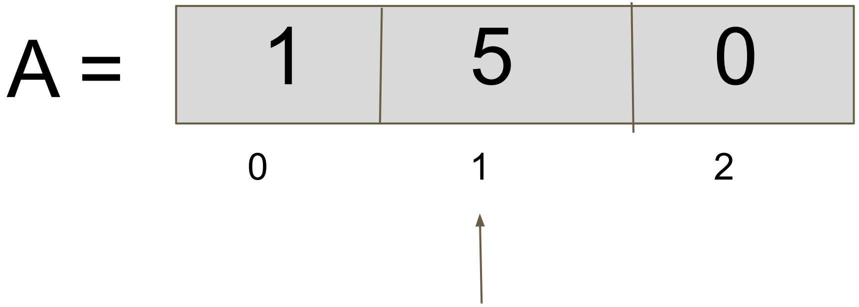


A =

1	5	0
0	1	2

Keep progressing from back to front until we don’t encounter any 10s (i.e. we don’t require further carries.).

Arbitrary Precision Increment: Example 1

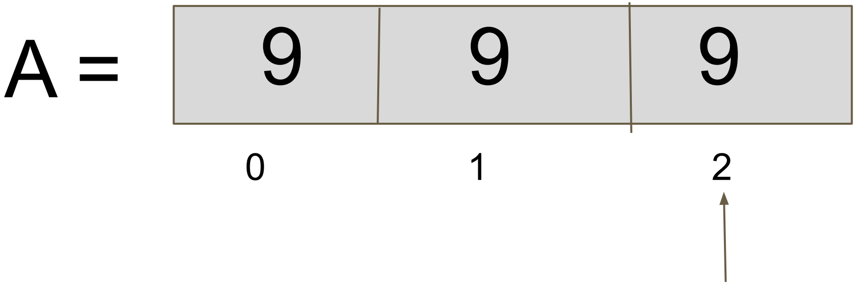


A =

1	5	0
0	1	2

Since there is no 10, we are finished progressing through the array.

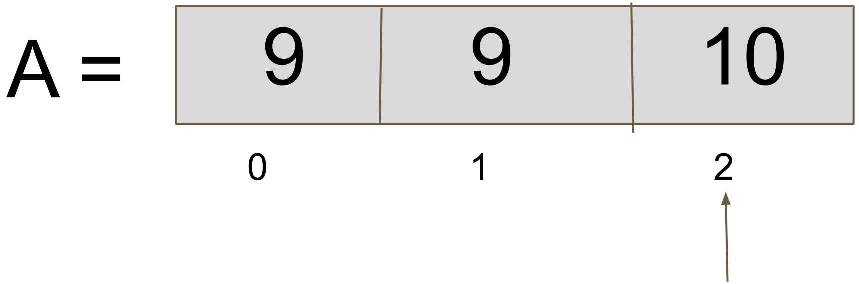
Arbitrary Precision Increment: Example 2



A =

0	1	2
9	9	9

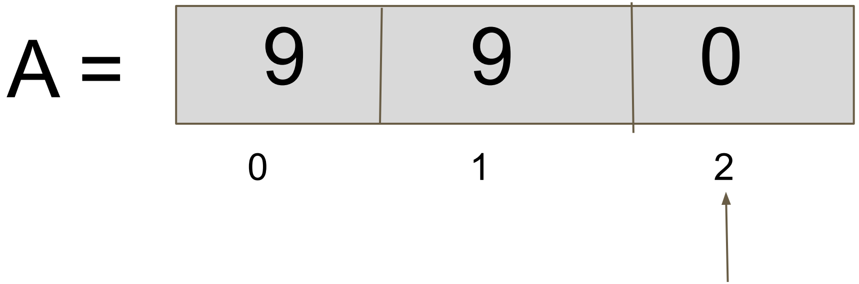
Arbitrary Precision Increment: Example 2



A =

9	9	10
0	1	2

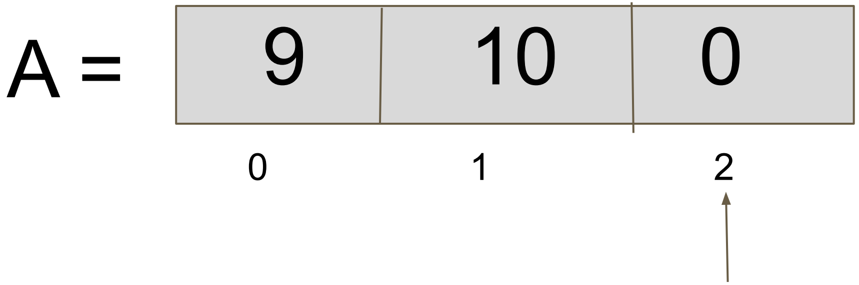
Arbitrary Precision Increment: Example 2



A =

9	9	0
0	1	2

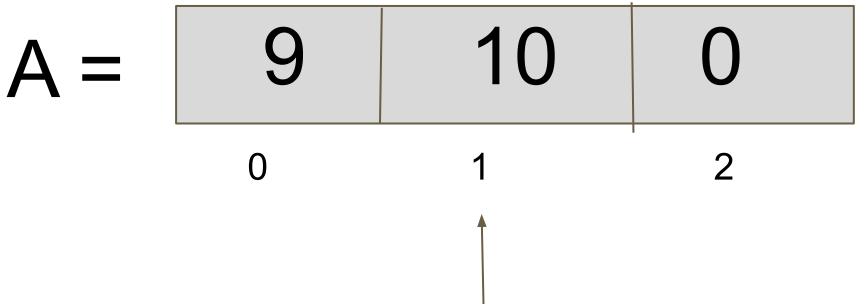
Arbitrary Precision Increment: Example 2



A =

9	10	0
0	1	2

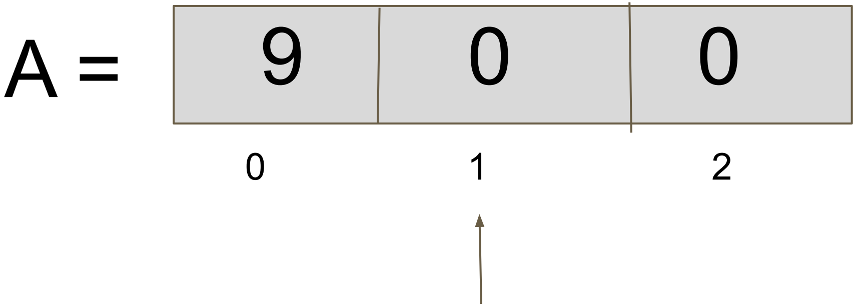
Arbitrary Precision Increment: Example 2



A =

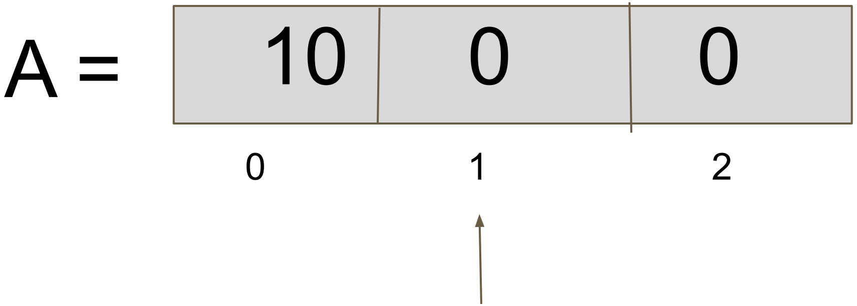
9	10	0
0	1	2

Arbitrary Precision Increment: Example 2

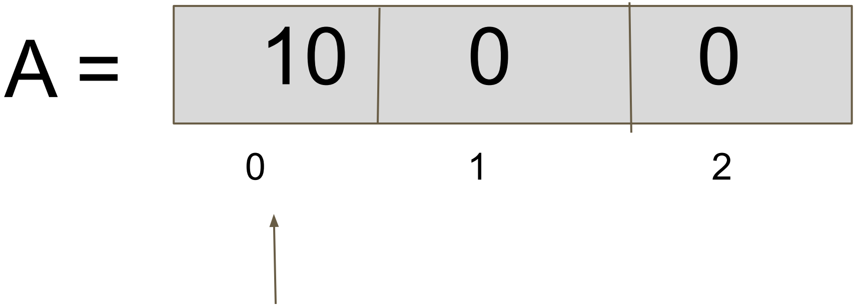


A =	9	0	0
	0	1	2

Arbitrary Precision Increment: Example 2

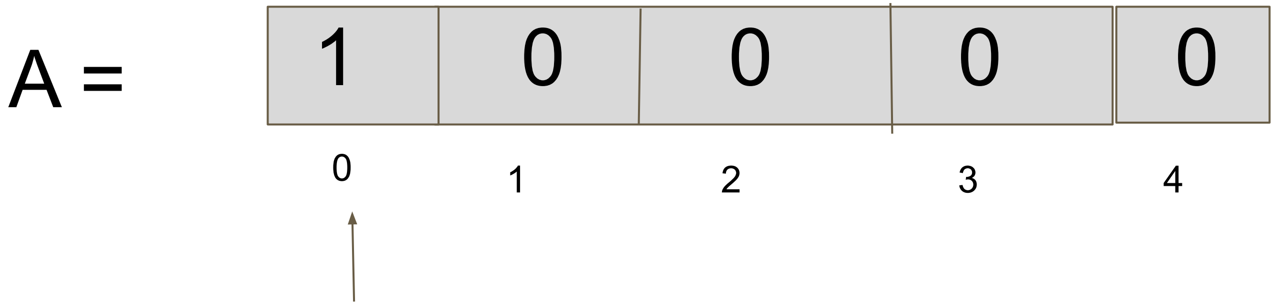


Arbitrary Precision Increment: Example 2



A =	10	0	0
	0	1	2

Arbitrary Precision Increment: Example 2

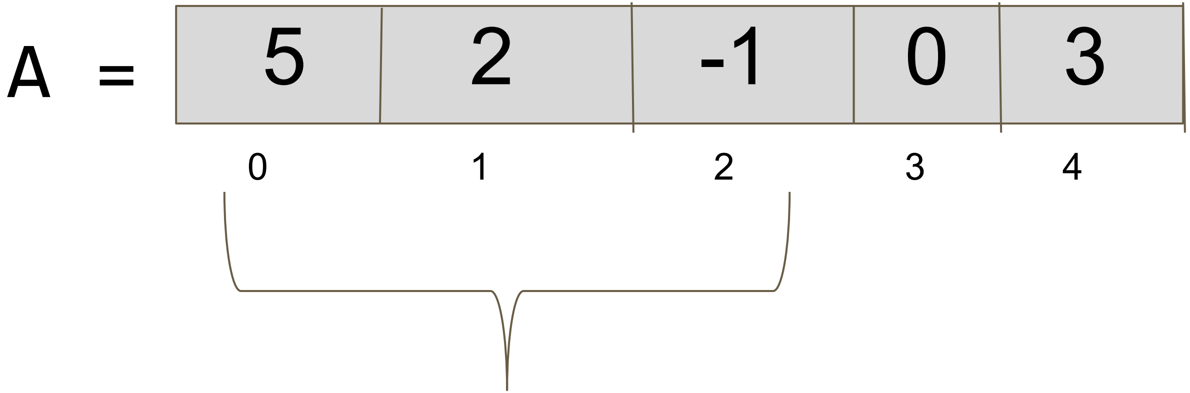


A =

0	1	2	3	4
1	0	0	0	0

Arrays: Window Sliding Technique

Window Sliding Technique



A =

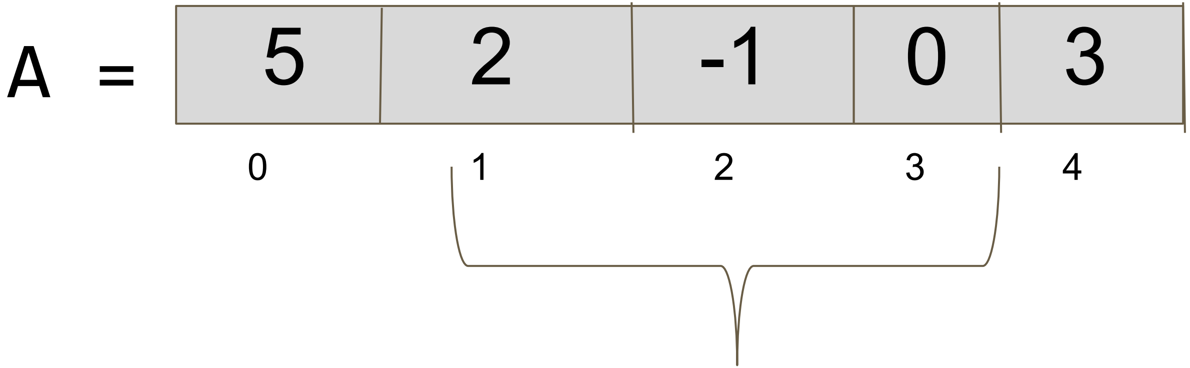
0	1	2	3	4
5	2	-1	0	3

k = 3

```
window_sum = sum(A[0:k])
```

In this case, “window_sum” is 6. Since 5 + 2 + -1 = 6

Window Sliding Technique



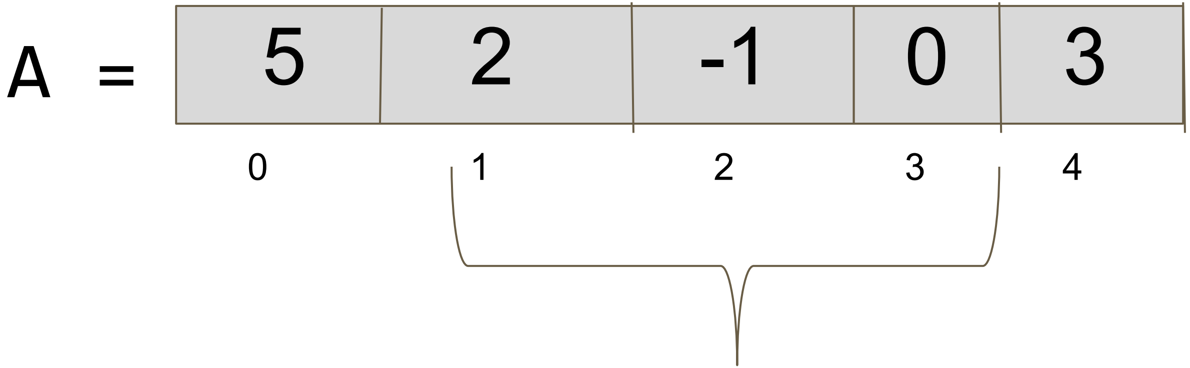
A =	5	2	-1	0	3
	0	1	2	3	4

k = 3

```
window_sum += A[i] - A[i-k]
```

Now process rest of array from “k” onward.

Window Sliding Technique



A =

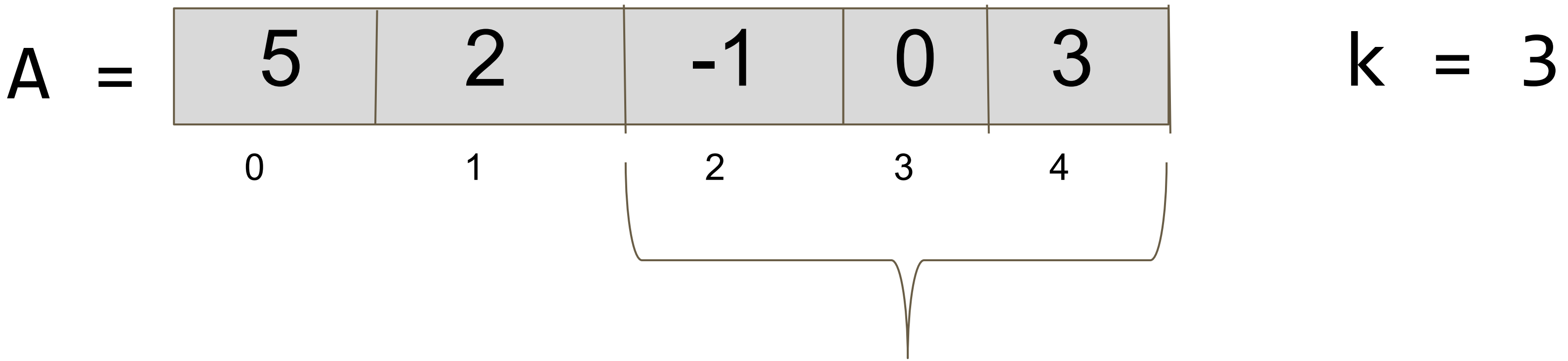
5	2	-1	0	3
0	1	2	3	4

k = 3

```
window_sum += A[i] - A[i-k]
6          +=   0  -  5     =  1
```

Which is 2 + -1 + 0 = 1

Window Sliding Technique



A =

5	2	-1	0	3
0	1	2	3	4

k = 3

```
window_sum += A[i] - A[i-k]
1          +=   3  -   2      =  2
```

Which is -1 + 0 + 3 = 2