Sliding Window

Sliding Window

- ✔ Used for finding subarrays inside an array or Substring in String
- ✔ Can only apply when you have deal with consecutive elements
- ✔ Subset of dynamic programming
- ✔ One of the common asked interview because of its efficiency:
  - Time Complexity : O(n)
  - Space Complexity: O(1)

Problem Statement

Given an array of integers and a number k.
Return the highest sum of any k consecutive elements in the array.

Input Array :

1	5	2	3	7	1

Target (k) : 3

Sliding Window

1	5	2	3	7	1

- Sliding Window Technique is a method for finding subarrays in an array that satisfy given conditions.
- We do this via maintaining a subset of items as our window and resize and move that window within the larger list until we find a solution.

Sliding Window

1	5	2	3	7	1

3

Brute Force

Input Array :

1	5	2	3	7	1

8

Target (k) : 3

Max: 8

Sub Array: [1,5,2]

Brute Force

Input Array :

1	5	2	3	7	1

10

Target (k) : 3

Max: ~~8~~ 10

Sub Array: [5,2,3]

Brute Force

Input Array :

1	5	2	3	7	1

12

Target (k) : 3

Max: ~~10~~ 12

Sub Array: [2,3,7]

Brute Force

Input Array :

1	5	2	3	7	1

11

Target (k) : 3

Max: 12

Sub Array: [2,3,7]

Brute Force

$O[N^2]$

Straightforward approach to solving a problem

- ✔ Loop through each element (outside)
- ✔ Loop through the next set of k elements (inside)
- ✔ Add the values
- ✔ Check if the sum is greater than previous max
- ✔ If yes, make that as max
- ✔ Return the max of all

Sliding Window

- ✔ Used for finding subarrays inside an array or Substring in String
- ✔ Can only apply when you have deal with consecutive elements
- ✔ Subset of dynamic programming
- ✔ One of the common asked interview because of its efficiency:
  - Time Complexity : O(n)
  - Space Complexity: O(1)

Working Behaviour

Given an array of integers and a number k.
Return the highest sum of any k consecutive elements in the array.

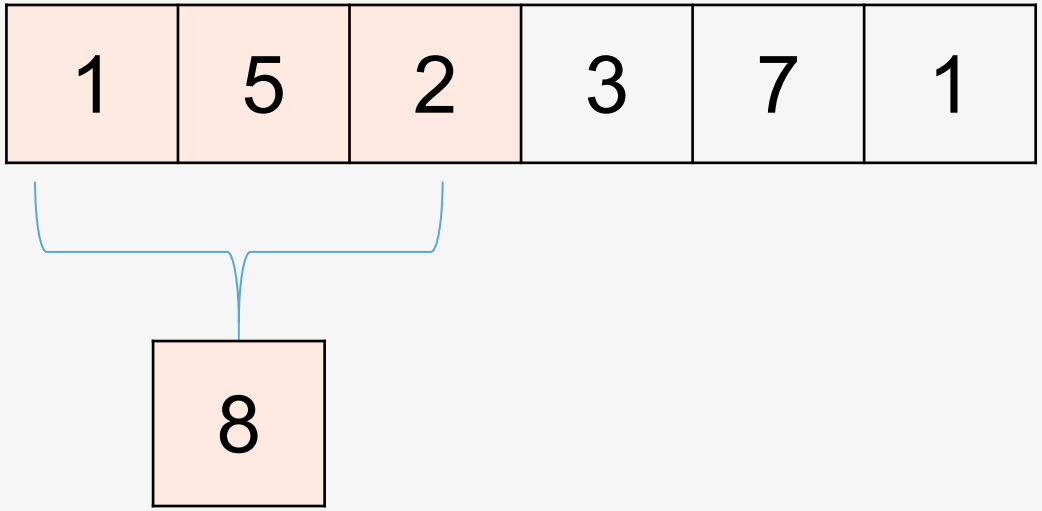
Input Array :

1	5	2	3	7	1

Target (k) : 3

Sliding Window

Input Array :



Target (k) : 3

Max: 8

Sub Array: [1,5,2]

Sliding Window

Input Array :

1	5	2	3	7	1

[8] – 1 + 3 = 10

Target (k) : 3

Max: 10

Sub Array: [5,2,3]

Sliding Window

Input Array :

1	5	2	3	7	1

[10] – 5 + 7 = 12

Target (k) : 3

Max: 12

Sub Array: [2,3,7]

Sliding Window

Input Array :

1	5	2	3	7	1

[12] – 2 + 1 = 11

Target (k) : 3

Max: 12

Sub Array: [2,3,7]