Sliding window

Example

Problem

Given an array, find the average of all contiguous subarrays of size ‘K’ in it.

Input

Array: [1, 3, 2, 6, -1, 4, 1, 8, 2], K=5

Output

[2.2, 2.8, 2.4, 3.6, 2.8]

Solution

Here, we are asked to find the average of all contiguous subarrays of size ‘5’ in the given array. Let’s solve this:

1. For the first 5 numbers (subarray from index 0-4), the average is: $(1+3+2+6-1)/5 => 2.2$
2. The average of next 5 numbers (subarray from index 1-5) is: $(3+2+6-1+4)/5=>2.8$
3. For the next 5 numbers (subarray from index 2-6), the average is: $(2+6-1+4+1)/5 => 2.4$

Here is the final output containing the averages of all contiguous subarrays of size 5:

Output: [2.2, 2.8, 2.4, 3.6, 2.8]

Brute-force algorithm

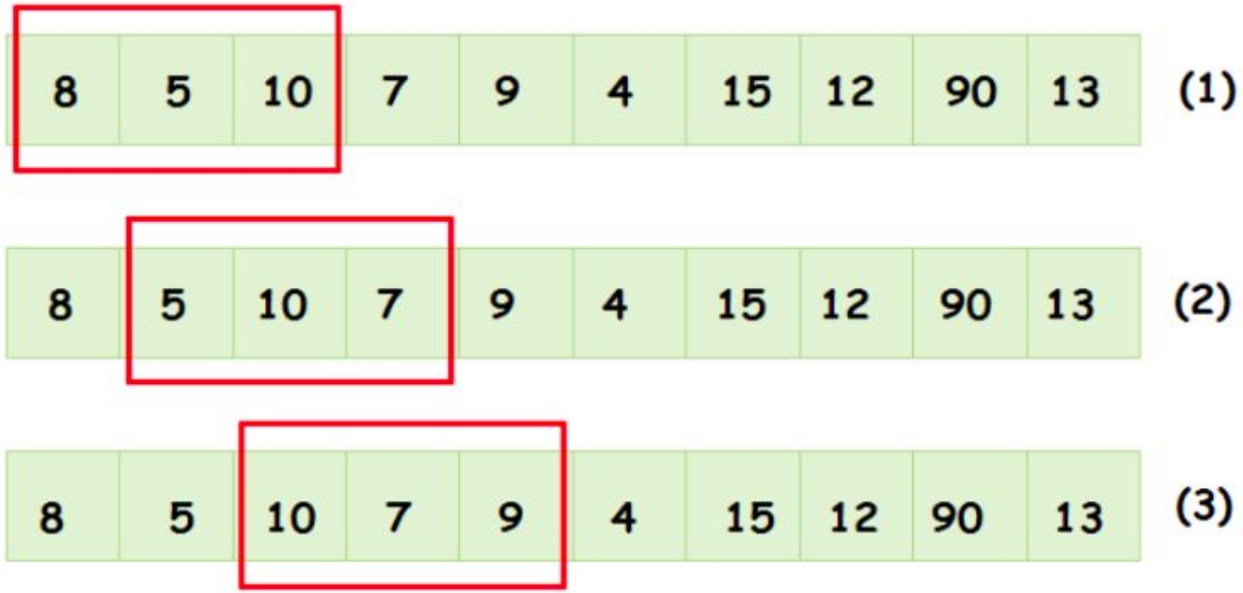
```java
public static double[] findAverages(int K, int[] arr) {
  double[] result = new double[arr.length - K + 1];
  for (int i = 0; i <= arr.length - K; i++) {
    // find sum of next 'K' elements
    double sum = 0;
    for (int j = i; j < i + K; j++)
      sum += arr[j];
    result[i] = sum / K; // calculate average
  }

  return result;
}
```

Time complexity

the time complexity of the above algorithm will be O(N*K) where ‘N’ is the number of elements in the input array.

Sliding window



How do you identify them?

1. Minimum value
2. Maximum value
3. Longest value
4. Shortest value
5. *K*-sized value

How do you identify them?

1. One of the biggest clues that one can use a sliding window is the word **contiguous**. Remember: in the context of programming, contiguous means that the elements are sequentially placed next to each other.

2. The problem will involve a data structure that is ordered and iterable.

3. Some common data structures one will be using a sliding window on are **strings, arrays,** and even **linked lists.**

4. There is an apparent naive or brute force solution that runs in $O(N^2)$, $O(2^N)$ or some other large time complexity.

Time and space complexity

The amazing thing about sliding window problems is that most of the time they can be solved in O(N) time and O(1) space complexity.

Resources

Articles:
https://github.com/liyin2015/Algorithms-and-Coding-Interviews
https://medium.com/outco/how-to-solve-sliding-window-problems-28d67601a66
https://stackoverflow.com/questions/8269916/what-is-sliding-window-algorithm-examples
https://levelup.gitconnected.com/an-introduction-to-sliding-window-algorithms-5533c4fe1cc7
https://leetcode.com/problems/find-all-anagrams-in-a-string/discuss/92007/sliding-window-algorithm-template-to-solve-all-the-leetcode-substring-search-problem
https://leetcode.com/discuss/general-discussion/657507/sliding-window-for-beginners-problems-template-sample-solutions/
https://medium.com/leetcode-patterns/leetcode-pattern-2-sliding-windows-for-strings-e19af105316b
https://medium.com/algorithms-and-leetcode/magic-solution-to-leetcode-problems-sliding-window-algorithm-891e3d60bf89

Videos:
https://www.youtube.com/watch?v=MK-NZ4hN7rs&list=PLVmRRBrc2pRDWhSLD5fYW247krxGZx-vQ&index=4&t=1595s&ab_channel=TheSimpleEngineer

Tasks:
https://leetcode.com/tag/sliding-window/
https://leetcode.com/list/x17aw7vm/