JS:

function max\_sub\_array\_of\_size\_k(k, arr) {

let maxSum = 0,

windowSum = 0,

windowStart = 0;

for (window\_end = 0; window\_end < arr.length; window\_end++) {

windowSum += arr[window\_end]; // add the next element

// slide the window, we don't need to slide if we've not hit the required window size of 'k'

if (window\_end >= k - 1) {

maxSum = Math.max(maxSum, windowSum);

windowSum -= arr[windowStart]; // subtract the element going out

windowStart += 1; // slide the window ahead

}

}

return maxSum;

}

console.log(`Maximum sum of a subarray of size K: ${max\_sub\_array\_of\_size\_k(3, [2, 1, 5, 1, 3, 2])}`);

console.log(`Maximum sum of a subarray of size K: ${max\_sub\_array\_of\_size\_k(2, [2, 3, 4, 1, 5])}`);

C++

using namespace std;

#include <iostream>

#include <vector>

class MaxSumSubArrayOfSizeK {

public:

static int findMaxSumSubArray(int k, const vector<int>& arr) {

int windowSum = 0, maxSum = 0;

int windowStart = 0;

for (int windowEnd = 0; windowEnd < arr.size(); windowEnd++) {

windowSum += arr[windowEnd]; // add the next element

// slide the window, we don't need to slide if we've not hit the required window size of 'k'

if (windowEnd >= k - 1) {

maxSum = max(maxSum, windowSum);

windowSum -= arr[windowStart]; // subtract the element going out

windowStart++; // slide the window ahead

}

}

return maxSum;

}

};

int main(int argc, char\* argv[]) {

cout << "Maximum sum of a subarray of size K: "

<< MaxSumSubArrayOfSizeK::findMaxSumSubArray(3, vector<int>{2, 1, 5, 1, 3, 2}) << endl;

cout << "Maximum sum of a subarray of size K: "

<< MaxSumSubArrayOfSizeK::findMaxSumSubArray(2, vector<int>{2, 3, 4, 1, 5}) << endl;

}

Python:

def max\_sub\_array\_of\_size\_k(k, arr):

max\_sum , window\_sum = 0, 0

window\_start = 0

for window\_end in range(len(arr)):

window\_sum += arr[window\_end] # add the next element

# slide the window, we don't need to slide if we've not hit the required window size of 'k'

if window\_end >= k-1:

max\_sum = max(max\_sum, window\_sum)

window\_sum -= arr[window\_start] # subtract the element going out

window\_start += 1 # slide the window ahead

return max\_sum

def main():

print("Maximum sum of a subarray of size K: " + str(max\_sub\_array\_of\_size\_k(3, [2, 1, 5, 1, 3, 2])))

print("Maximum sum of a subarray of size K: " + str(max\_sub\_array\_of\_size\_k(2, [2, 3, 4, 1, 5])))

main()

JAVA:

class MaxSumSubArrayOfSizeK {

public static int findMaxSumSubArray(int k, int[] arr) {

int windowSum = 0, maxSum = 0;

int windowStart = 0;

for (int windowEnd = 0; windowEnd < arr.length; windowEnd++) {

windowSum += arr[windowEnd]; // add the next element

// slide the window, we don't need to slide if we've not hit the required window size of 'k'

if (windowEnd >= k - 1) {

maxSum = Math.max(maxSum, windowSum);

windowSum -= arr[windowStart]; // subtract the element going out

windowStart++; // slide the window ahead

}

}

return maxSum;

}

public static void main(String[] args) {

System.out.println("Maximum sum of a subarray of size K: "

+ MaxSumSubArrayOfSizeK.findMaxSumSubArray(3, new int[] { 2, 1, 5, 1, 3, 2 }));

System.out.println("Maximum sum of a subarray of size K: "

+ MaxSumSubArrayOfSizeK.findMaxSumSubArray(2, new int[] { 2, 3, 4, 1, 5 }));

}

}