Стратегия скользящего окна для алгоритмов решения

Метод скользящего окна — алгоритм [трансформации](https://wiki.loginom.ru/articles/data-transformation.html), позволяющий сформировать из членов [временного ряда](https://wiki.loginom.ru/articles/time-series.html) набор данных, который может служить [обучающим множеством](https://wiki.loginom.ru/articles/training-set.html) для построения модели [прогнозирования](https://wiki.loginom.ru/articles/forecasting.html).

Под окном в данном случае понимается временной интервал, содержащий набор значений, которые используются для формирования [обучающего примера](https://wiki.loginom.ru/articles/training-sample.html). В процессе работы алгоритма окно смещается по временной последовательности на единицу наблюдения, и каждое положение окна образует один пример.

Например, если еженедельно поступают данные о продажах в течение 50 недель и установлено окно в 5 недель, то в первом примере используются данные с 1 по 5 неделю, а целевым значением будут данные за 6-ю неделю. Во втором случае используются данные со 2 по 6 неделю, а в качестве целевого значения берутся данные за 7-ю и т.д.

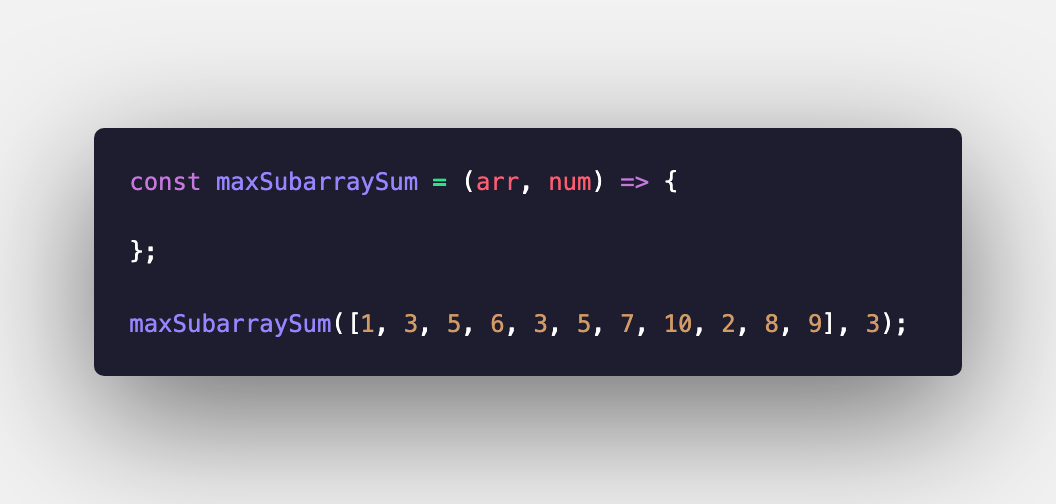
Обработка данных методом [скользящего окна](https://wiki.loginom.ru/articles/windowing-method.html) применяется при предварительной обработке данных в задачах прогнозирования, когда на вход анализатора (например, нейронной сети) требуется подавать значения нескольких смежных отсчетов исходного набора данных. Термин скользящее окно отражает сущность обработки — выделяется некоторый непрерывный отрезок данных, называемый окном, а окно, в свою очередь, перемещается, "скользит" по всему исходному набору данных.

В результате будет получен набор данных, где в одном поле будет содержаться значение, соответствующее текущему отсчету (оно будет иметь то же имя, что и в исходном наборе), а слева и справа от него будут расположены поля со значениями, смещенными от текущего отсчета в прошлое и в будущее соответственно.

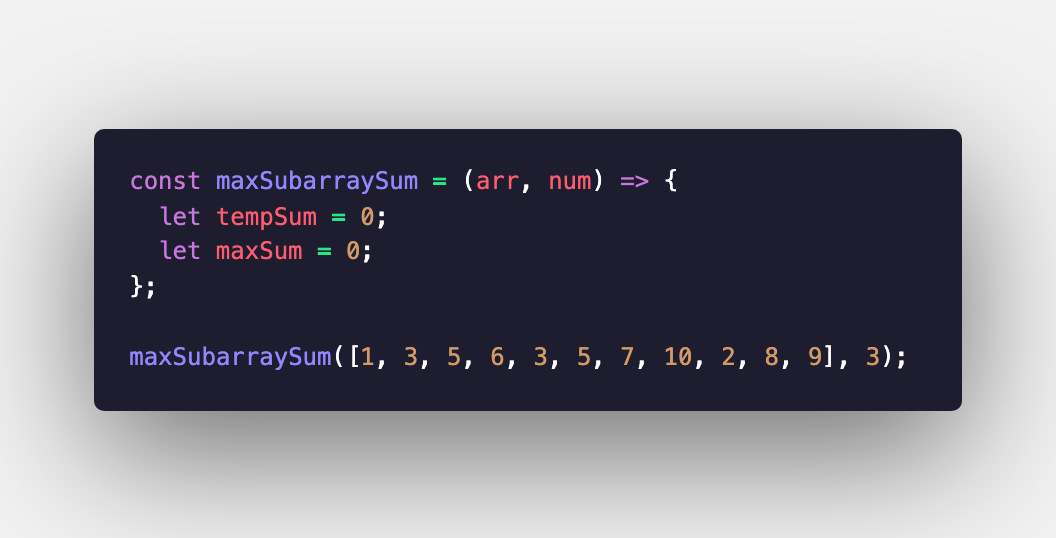
Стратегия включает в себя создание окна или подмножества массива и его перемещение вверх или вниз в зависимости от записанного вами условия. Это полезный способ отслеживать небольшой набор данных, который вы затем будете сравнивать со следующим набором данных при перемещении по массиву.

Алгоритм, называется maxSubarraySum. Цель алгоритма - найти наибольшую возможную сумму, которую можно составить из последовательных чисел в массиве. Это довольно простой пример алгоритма скользящего окна. В этом руководстве я буду использовать JavaScript.

Как всегда, первое, что мы хотим сделать, это объявить функцию и передать аргументы. Эта функция принимает два аргумента: несортированный массив положительных чисел и целое число, которое сообщает нам длину подмассива, который мы должны проверять, чтобы найти сумму. В этом случае мы ищем подмассив длиной три.



Что мы хотим сделать дальше, так это объявить две переменные, которые мы будем использовать в качестве временной переменной, и переменную для хранения текущей максимальной суммы, которую мы нашли на данный момент. Мы хотим установить их оба равными нулю, потому что параметры алгоритма указывают, что все числа будут положительными.



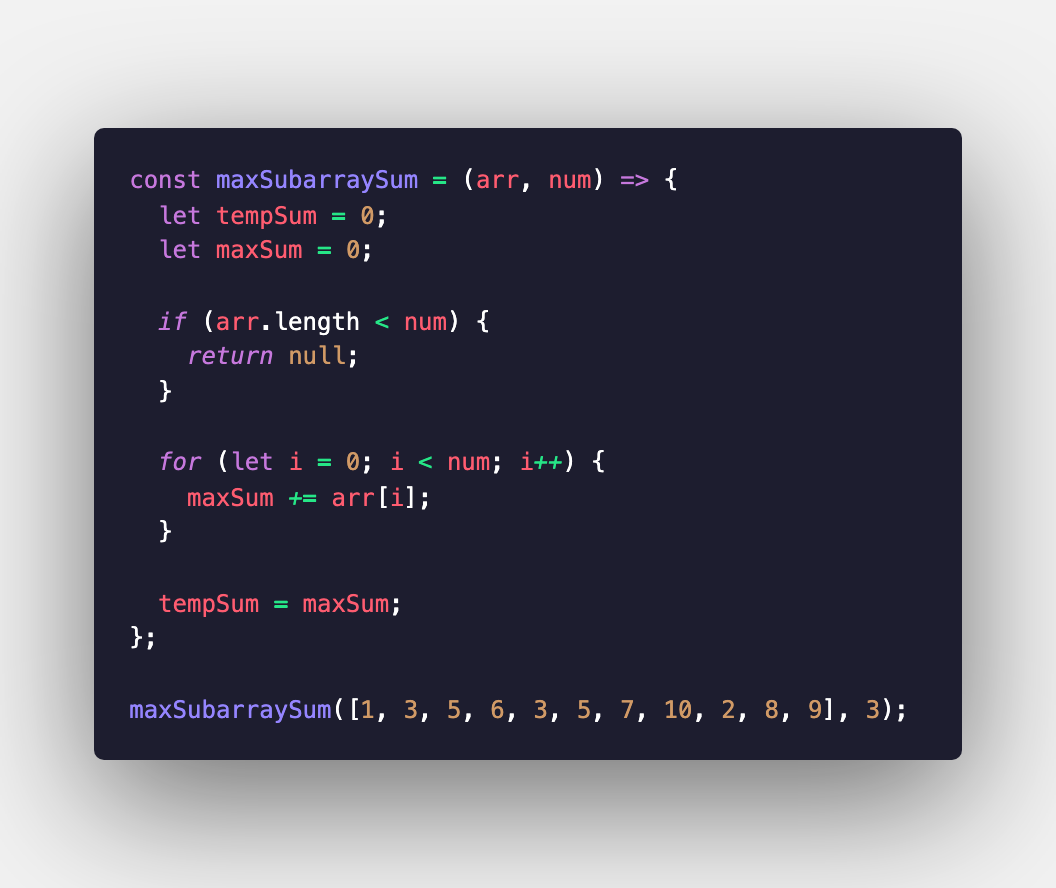
Один крайний случай можно обрабатывать сразу. Если длина массива меньше, чем количество элементов, которые мы должны проверять, мы можем вернуть null.



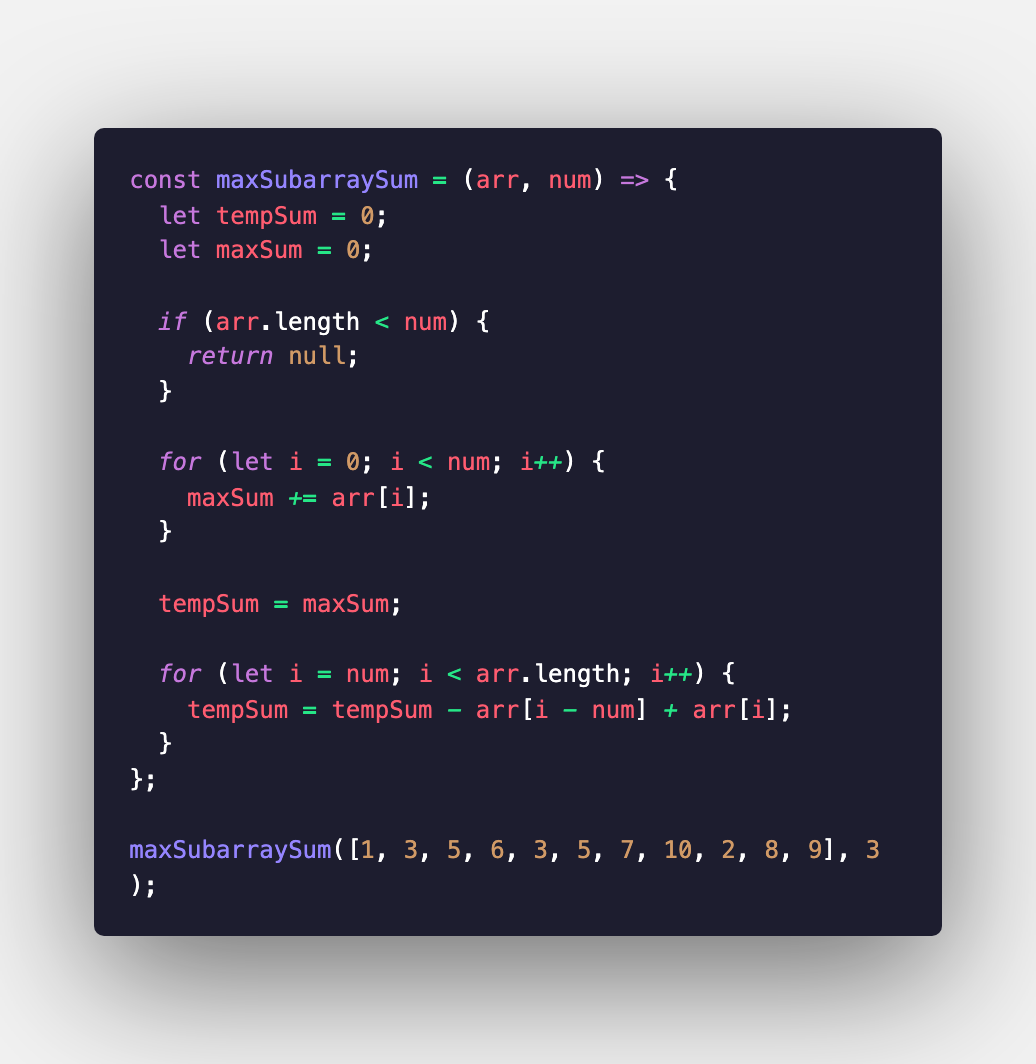
Что нам нужно сделать дальше, так это перебрать массив и добавить числа к текущей общей сумме. Если мы начинаем **i** с нуля и увеличиваем его только до длины num, мы фактически добавляем общее количество чисел подмассива правильной длины.



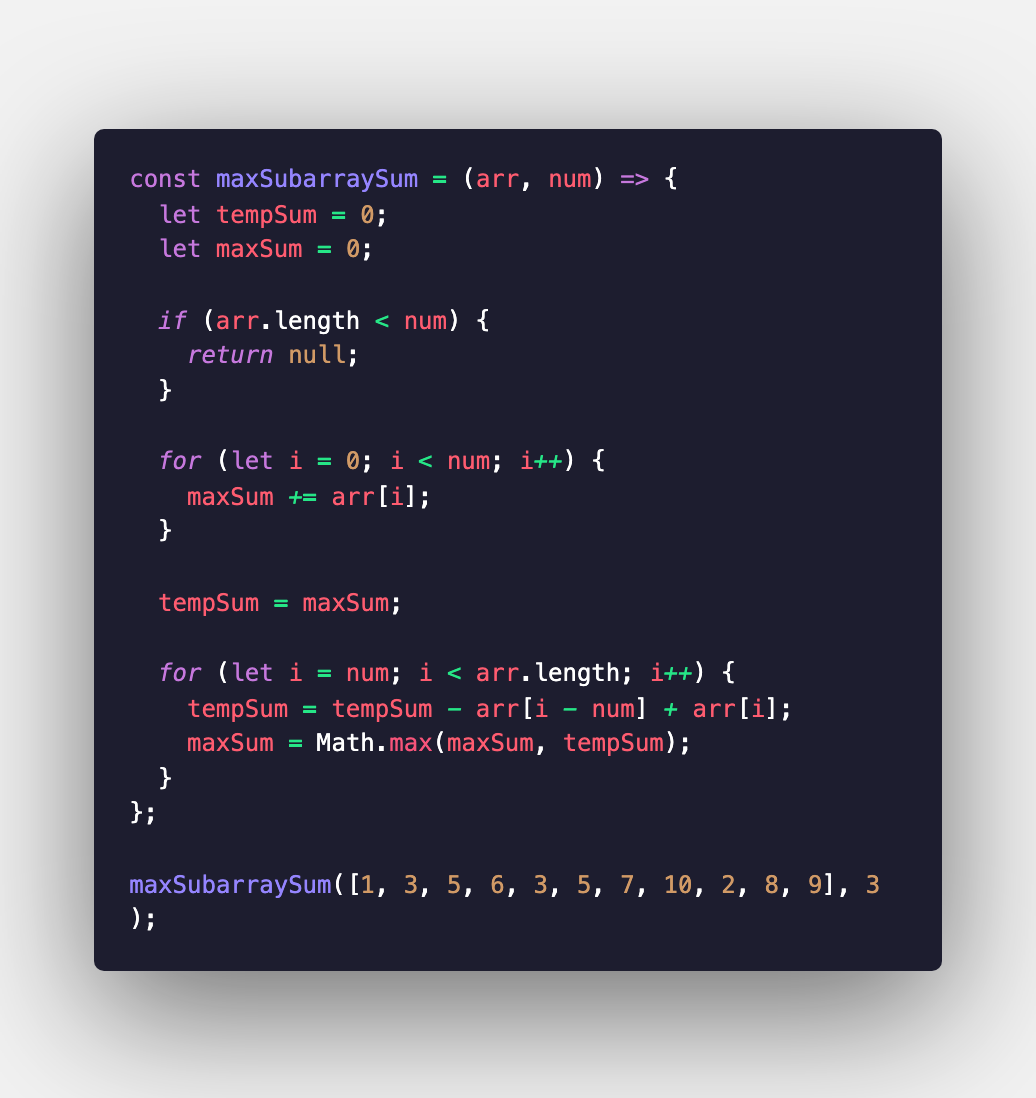
Как только мы найдем текущую самую высокую сумму, мы хотим установить как tempSum, так и maxSum равными этому числу.



В следующем цикле мы фактически создадим скользящее окно. На этот раз мы начинаем итерацию **i** с длины, которую мы ранее достигли в первом цикле (num). По мере того, как мы перебираем длину всего массива, мы постоянно устанавливаем tempSum равным сумме нового подмассива, который создается путем вычитания первого числа из предыдущего подмассива и добавления числа, которое находится в индексе после конца. предыдущего подмассива. Каждый раз, когда подмассив увеличивается вперед, мы вычитаем тот, который перемещается из подмассива, и добавляем тот, который находится перед ним, - создавая скользящее окно.



Однако способ, которым это написано до сих пор, перезаписывает tempSum при каждой итерации. Мы никогда не сможем узнать фактическую максимально возможную сумму. Что нам нужно сделать, так это взять максимальное число между tempSum и maxSum и использовать его для обновления maxSum. Мы можем сделать это очень легко с помощью Math.max ().



На каждой итерации этого цикла maxSum будет сравниваться с временной переменной tempSum. Если новое число больше, maxSum будет установлено равным большему числу. В конце этого цикла maxSum будет равняться максимально возможной сумме любого subArray заданной длины. Все, что нам нужно сделать, это вернуть maxSum. Убедитесь, что вы делаете это вне цикла.

