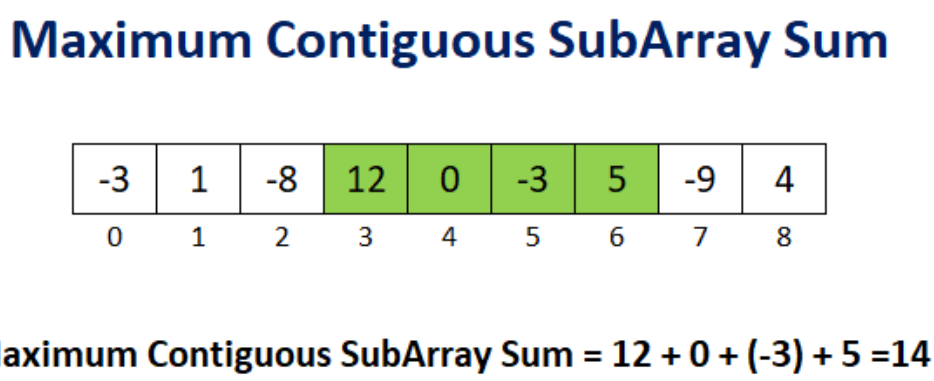
Kadane's Algorithm



Used to find the maximum sum of continuous/contiguous array.

For example, let's look at the "Best Time to Buy and Sell Stock" question of LeetCode.

Best Time To Buy and Sell Stock

You are given an array `prices` where `prices[i]` is the price of a given stock on the `ith` day.

You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return `0`.

Ex:

```
Input: prices = [7,1,5,3,6,4]

Output: 5
```

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

The easiest method is to use brute force for solving it.

```
c=0
n=len(prices)
for i in range(n-1):
   for j in range(i+1,n):
      c=max(c,prices[j]-prices[i])
return c
```

Time Complexity: O(n^2)
Space Complexity: O(1)

This fails at boundary/edge cases and therefore we need a faster algorithm.
Another method for solving this is to use Kadane's Algorithm which is used to find the maximum sum of continuous/contiguous array.
Kadane solves the same problem in linear time with constant space.

Algorithm :
maxCur=0
maxSoFar=INT_MIN(based on constraints)
Loop for each element of array

A. **maxCur=maxCur+a[i]**
B. **If maxSoFar < maxCur)**
    a. **maxSoFar = maxCur**
C. **If maxCur<0**
    a. **maxCur=0**

Return maxSoFar

In Kadane, we keep track of the current maximum value using the variable maxCur and also the track of overall maximum using the maxSoFar variable.

Solution for Best Time To Buy and Sell Stock:

```
n=len(prices)
maxSoFar=0
maxCur=0
for i in range(1,n):
   maxCur+=prices[i]-prices[i-1]
   maxCur=max(0,maxCur)
   maxSoFar=max(maxSoFar,maxCur)
return maxSoFar
```

Time Complexity: O(n)
Space Complexity: O(1)

NOTE: I was unable to solve this question in one go. I tried brute force but it failed at boundary cases and then I took the help of discussions where I realized that this was a straight Kadane implementation.

Day 1 Brute Force

Day 1 Kadane