

Kadane's algo: Finds maximum subarray sum in $O(n)$ complexity.

4	3	-2	6	-12	7	-1	6
0	1	2	3	4	5	6	7

currentMax

4, {4}

7, {4, 3}

5, {4, 3, -2}

11, {4, 3, -2, 6}

1, {4, 3, -2, 6, -12}

7, {7}

maxSoFar / bestSum

4 - {4}

7 - {4, 3}

7 - {4, 3}

11 - {4, 3, -2, 6}

11 - {4, 3, -2, 6}

11 - {4, 3, -2, 6}

11 - {4, 3, -2, 6}

11 - {4, 3, -2, 6} x

Q. Should i become part of current subarray?

$6 + 6 = 12$

Or

Q. Should i start a new subarray?

6

Solution:

```
int maxSubArray(vector<int>& nums) {
```

```
    int currentSum = nums[0], totalSum = nums[0];
```

```
    for(int i=1; i<nums.size(); i++) {
```

```
        //Current max sum is either the current element OR current element + Previous Maximum subarray)
```

```
        currentSum = max(nums[i], currentSum+nums[i]);
```

```
        //If the current maximum array sum is greater than the global total. Update it
```

```
        totalSum = max(totalSum, currentSum);
```

```
    }
```

```
    return totalSum;
```

```
}
```