算法描述

给定一个整数数组 nums ， 找到一个具有最大和的连续子数组（子数组最少包含一个元素），返回其最大和。

```
int maxSubArray(int *nums, int numsSize)
{
    int curSum, maxSum;
    curSum = maxSum = nums[0];
    for (int i = 1; i < numsSize; i++)
    {
        curSum = curSum + nums[i] > nums[i] ? curSum + nums[i] : nums[i];
        maxSum = maxSum > curSum ? maxSum : curSum;
    }
    return maxSum;
}
```

1. 初始化 `curSum` 和 `maxSum` 为第一个元素，其中 `curSum` 表示当前最理想的子数值和，`maxSum` 表示最大子数组的和。
2. 令 i=1，判断 `curSum + nums[i]` 是否大于 `nums[i]`，如果大于，则 `curSum` 加上 `nums[i]`，否则对`nums[i]`来说，加上`curSum`对往后数组和的贡献还不如它本身，此时抛弃前面计算的和，更新 `curSum` 为 `nums[i]`。
3. 更新 `maxSum` 为 `maxSum` 和 `curSum` 中的较大值。
4. 重复步骤 2 和 3，直到遍历完整个数组。

复杂度

- 时间复杂度：O(n)
- 空间复杂度：O(1)