2.Medium\04.Kadane's_algorithm.cpp

```cpp
/*
QUESTION:-
Given an integer array nums, find the subarray with the largest sum, and return its sum.

Example 1:

Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: The subarray [4,-1,2,1] has the largest sum 6.

Example 2:

Input: nums = [1]
Output: 1
Explanation: The subarray [1] has the largest sum 1.
*/

/*
APPROACH:-
-> Initialize two variables: maxSum and currentSum. Set both variables to the first element of the array.
-> Iterate through the array starting from the second element:
    Update currentSum by adding the current element to it.
    If currentSum becomes negative, reset it to 0. This step ensures that we consider only the subarrays with positive sums.
    Update maxSum by taking the maximum value between maxSum and currentSum. This keeps track of the maximum subarray sum encountered so far.
-> After the iteration, the maxSum variable will hold the largest sum of any subarray.
-> Return the maxSum as the result.
*/

// CODE:-
int maxSubArray(vector<int> &nums)
{
    int curr_sum = 0;
    int ans = INT_MIN;
    for (int i = 0; i < nums.size(); i++)
    {
        curr_sum += nums[i];
        ans = max(ans, curr_sum);
        if (curr_sum < 0)
            curr_sum = 0;
    }
    return ans;
}

// TIME COMPLEXITY = O(N)
// SPACE COMPLEXITY = O(0)
```