1749. Maximum Absolute Sum of Any sybarray

1749. Maximum Absolute Sum of Any Subarray

→ Given array return Maximum absolute subarray sum.

Input : nums = [1, -3, 2, 3, -4]

output = 5

The subarr [2,3] has max abs sum = 5.

Brute force : find all subarray sum with abs value then return max of them.

Better Approach : Kadane's Algorithm

Input : nums = [1, -3, 2, 3, -4]

subarrays sum = [1], [-3], [2], [3], [-4]

[-2], [0], [3], [-1], [5], [2], [5]

[1], [4] ⇒ max = 5

Ex:- Input nums = [2, -5, 1, -4, 3, -2], output = 8

Subarray with max abs sum = [-5, 1, -4]

[2, -5, 1, -4, 3, -2]

min subarray sum = -8 } abs max(8,3) = 8

max subarray sum = 3 }

Dry Run :-

ind	curr_sum	Max_Sum
0	1	1
1	(-3, -3+1) = -2	(1, -2) = 1
2	(2, 2+(-2)) = 2	(2, 1) = 2
3	(2, 2+3) = 5	(5, 2) = 5
4	(5, 5+(-4)) = 5	(5, 5) = 5

max_sum

Now we have

max_subarray sum = 5

min_subarray sum = (-8) = 8

max(5, 8) = 8 Output

→ we'll use kadane's Algo for finding minimum & maximum Subarray sum.

→ the we return max abs value of them.

Code :-

```
int maxAbsSum(vector<int>& nums){
    int n = nums.size();
    // max subarray sum
    max_sum = nums[0];
    curr_sum = nums[0];
    for(int i=1; i<n; i++){
        curr_sum = max(nums[i], curr_sum
                   + nums[i]);
        max_sum = max(curr_s, max_s);
    }
    // min subarray sum
    curr_sum = nums[0];
    min_sum = nums[0];
    for(int i=1; i<n; i++){
        curr_sum = min(nums[i], curr_
                   + nums[i]);
        min_s = min(curr_s, min_s);
    }
    return max(abs(min_s), max_s);
}
```

[1, -3, 2, 3, -4]

ind	curr_sum	min sum
0	1	1
1	(-3, -3+1) = -3	(1, -3) = -3
2	(2, 2+(-3)) = -1	(-1, -3) = -3
3	(-1, 3) = 1 ,	(-1, -3) = -3
4	(-1, -1+-4) = -5	(-3, -3+5) = -8

min_sum

Time Complexity = O(n)

Space Complexity = O(1)