Kadane's Algorithm

0 1 2 3 4 5 6 7 8

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

n = 9

(-2) = -2

(-2, 1) = -1

(-2, 1, -3) = -4

(-2, 1, -3, 4)

(-2, 1, -3, 4, -1)

(-2, 1, -3, 4, -1, 2)

(-2, 1, -3, 4, -1, 2, 1)

-2, 1, -3, 4, -1, 2, 1, -5

-2, 1, -3, 4, -1, 2, 1, -5, 4

(1)

(1, -3)

(1, -3, 4)

(1, -3, 4, -1)

and so on

Approach 1

maxi = Default -ve constant

for(i=0 to n) {

for(j = i to n) {

for(k = i to j) {

possible subarrays

Sum += arr(k);

maxi = max(Sum, maxi);

}

}

}

return maxi

}

Nested Loops

$O(n^3)$

↳ Logic

Approach 2

$O(n^2)$

maxSubarray{

for(i=0 to n) {

Sum = 0, maxi = Default -ve constant

for(j = i to n) {

sum += arr[j];

maxi = max(sum, maxi)

}

}

return maxi

}

Approach 3 (Application of DP)

→ Kadane's Algorithm

↓

O(n)

n = 5

nums = [5,4,-1,7,8]

0 1 2 3 4

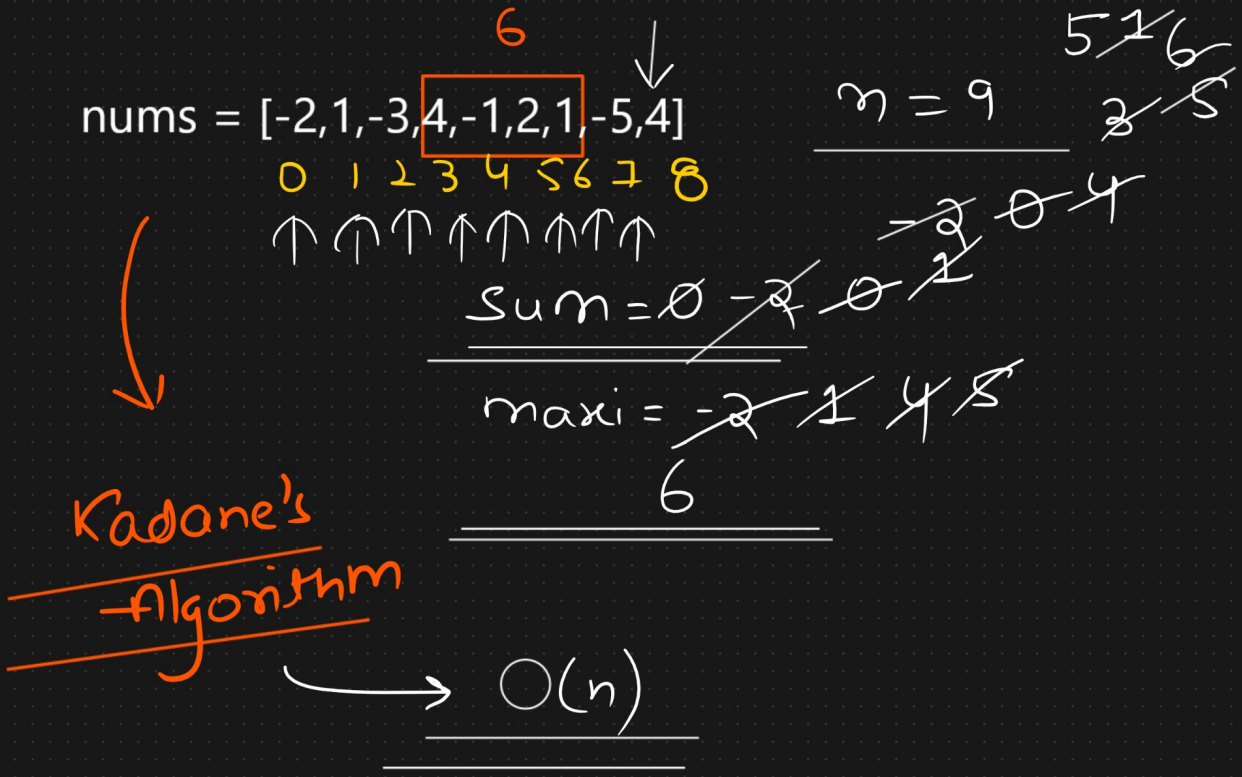
23

sum = 0; 5 9 8 16 23

maxi = 5; arr(0);

```
for(i=0 to n){
    sum += arr(i);
    if(sum > maxi){
        maxi = sum;
    }

    if(sum < 0){
        sum = 0;
    }
}
return maxi;
```


6
nums = [-2,1,-3,4,-1,2,1,-5,4]
0 1 2 3 4 5 6 7 8
n = 9
Sum = 0 -2 0 1
maxi = -2 1 4 5
6
Kadane's Algorithm
O(n)