FINDING MAX SUM OF CONTIGUOUS SUBARRAY

Let's say we've been given an array of length 5,

-1 -1 4 5 6

Here all the contiguous subarrays are,

-1	0th index	sum = -1
-1 -1	0th,1st index	sum = -2
-1 -1 4	0th,1st,2nd index	sum = 2
-1 -1 4 5	0th,1st,2nd,3rd index	sum = 7
-1 -1 4 5 6	0th,1st,2nd,3rd,4th index	sum = 13
-1	1th index	sum = -1
-1 4	1th,2nd index	sum = 3
-1 4 5	1th,2nd,3rd index	sum = 8
-1 4 5 6	1th,2nd,3rd,4th index	sum = 14
4	2nd index	sum = 4
4 5	2nd,3rd index	sum = 9
4 5 6	2nd,3rd,4th index	sum = 15
5	3rd index	sum = 5
5 6	3rd,4th index	sum = 11
6	4th index	sum = 6

Here, the subarray with maximum sum is 15, which is this subarray

4 5 6

But how are we going to find this?

BRUTE FORCE

For the brute force method, we need to traverse the array twice, for every subarray. So for this, time needed is O(n^2).

```
int ans =0;
for(int i=0;i<n;i++){
        int sum =0;
    for(int j=i;j<n;j++){
            sum+=arr[j];
        }
        sum = max(ans,sum);
}
```

We need a better approach!!

KADANE'S ALGORITHM

We need a sum variable and maxi variable.
Sum initialized to 0, and maxi initialized to the 0th element of the array.

```
int sum = 0, maxi = arr[0];
```

Then we'll traverse through the array once and keep updating maxi and sum.

```
for(int i=0;i<n;i++){
        int sum +=arr[i];
  maxi = max(sum,maxi);
}
```

And another thing, if the sum becomes negative, we'll set this to zero. Because a negative sum will reduce the sum.

```
int sum = 0, maxi = arr[0];
 for(int i=0;i<n;i++){
          int sum +=arr[i];
          maxi = max(sum,maxi);
        if(sum<0){
          sum =0;
          }
}
```

This approach costs O(n) time and will give us the maximum sum of the contiguous subarrays.

Note: This approach won't work for negative arrays(an array with all negative values).