Kadane's Algorithm
(Maximum Subarray Sum)

Subarray :-

It is the continuous part of array.

1	2	3	4	5

$\boxed{15}$ {
1,2,3,4,5
12,23,34,45
123,234,345
1234,2345
12345
↓

All possible subarrays

Array = n then the Number of Subarrays = $\frac{n*(n+1)}{2}$

$\frac{5*(5+1)}{2} = \frac{5*6}{2}$

$\frac{30}{2} = 15$

continous parts

Start	End
0	0,1,2,3,4
1	1,2,3,4
2	2,3,4
⋮	⋮

For start and End

```
for(st=0, st<n; st++){ ← for Start
    for(end=st; end<n; end++){ ← for end
        print (st to end)
    }
```

Maximus Subarray Sum:

Brute Force Approach

{3,-4,5,4,-1,7,-8}

```
for(st=0; st<n; st++){
    arrsum=0
    for(end=st; end<n; end++){
        CS+ = arr[end]
        maxSum = max(CS, MS);
    }
return MS;
```

More Optimized Approach

b=big
s=small

Kadane's Algorithm

{3, -4, 5, 4, 1, 7, -8}

Removed ---→ $3-4=-1$ ✗

```
for(i=0; i<n; i++){
    currSum+= arr[i]
    maxSum=max(C.S, M.S)
    if(CS<0){
        CS=0
    }
}
```

Intution

$\overset{b}{+ve} + \overset{b}{+ve} = +$

$\overset{s}{-ve} + \overset{b}{+ve} = +$

$\overset{s}{+ve} + \overset{b}{-ve} = -ve$

$\overset{s}{+ve} + \overset{b}{-ve} = -ve$

↓ It will give negative outcome, so this algorithm replaces it with 0 to remove negative outcome.