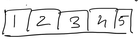
maximum subarray sum / Kadane's algo



1, 2, 3, 4, 5
12, 23, 34, 45
123, 234, 345
1234, 2345
12345

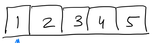
} all possible & valid subarrays

∴ total no of subarrays = $\frac{n*(n+1)}{2}$

∴ $\frac{5 \times (5+1)}{2} = 15$

```
code to just print all the
      possible subarrays
for (int st=0; st<n ; st++){
   for (int end=st; end<n; end++){
      for (int i=st; i<=end ;i++){
         cout << arr[i];
      }
      cout << " ";
   }
   cout <<endl;
}
```

two pointer : start & end (st to n-1)



start	end
0	0, 1, 2, 3, 4
1	2, 3, 4
2	3, 4
3	4

for loop to find subarray ka start → `for (st=0; st<n; st++){`

for loop to find subarray ka end → `for (end=st; end<n; end++){`

maximum subarray SUM

(1) Brute force approach

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

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

time complexity : $O(n^2)$

Rough work to understand

st=0

3 -4 5 4 -1 7 -8

cs = 3

3 - 4 = -1

-1 + 5 = 4

4 - 1 = 3

st=1

3 -4 5 4 -1 7 -8

cs = -4

-4 + 5 = 1

1 + 4 = 5

(2) Kadane's algorithm

intuition

(+ve) + (+ve) → +
(-ve) + (+VE) → +
(+ve) + (-VE) → −
(-ve) + (-VE) → −

reset to 0

apply just a single loop and keep adding the elements and store in curr sum and if it becomes negative then reset to 0

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

Rough work to understand

currSum = 0 maxSum = -∞

3 -4 5 4 -1 7 -8

st=0

currSum = 3 maxSum = 3

3 -4 5 4 -1 7 -8

st=1

currSum = 3 - 4 = -1 < 0

maxSum = 3 ms > cs

currSum resets to 0

3 -4 5 4 -1 7 -8

st=2

currSum = 0 + 5 = 5 ∴ maxSum = 5

forget purane values and continue from st=2 only

3 - 4 ← 4 -1 7 - 8

currSum = 9 maxSum = 9

3 - 4 ← 4 -1 7 - 8

currSum = 8 maxSum = 9

3 - 4 ← 4 -1 7 - 8

currSum = 15 maxSum = 15

3 - 4 ← 4 -1 7 - 8

currSum = 7 maxSum = 15

∴ max Sum = 15