Subarrays

↳ It is continuous part of an array

↳ It should be in forward direction

Example

arr = [20, 4, 9, 100]

Subarrays

[20]
[20, 4]
[20, 4, 9]
[20, 4, 9, 100]
[4]
[4, 9]
[4, 9, 100]
[9]
[9, 100]
[100

Is it a subarray?

(i.) [4, 100] → No

(ii.) [20, 9, 100] → No

(iii.) [20, 4, 9] → Yes

Q. Find all the possible subarrays in

Q: Find all the possible subarrays in given array

↳ [40, 9, 6, 1, 0]

Algorithm:-

```
   0  1  2  3
[20, 4, 9, 100]
```

20 (start:0, end:0)

20 4 (start:0, end:1)

20 4 9 (start:0, end:2)

20 4 9 100 (start:0, end:3)

4 (start:1, end:1)

4 9 (start:1, end:2)

4 9 100 (start:1, end:3)

9 (start:2, end:2)

9 100 (start:2, end:3)

100 (start:3, end:3)

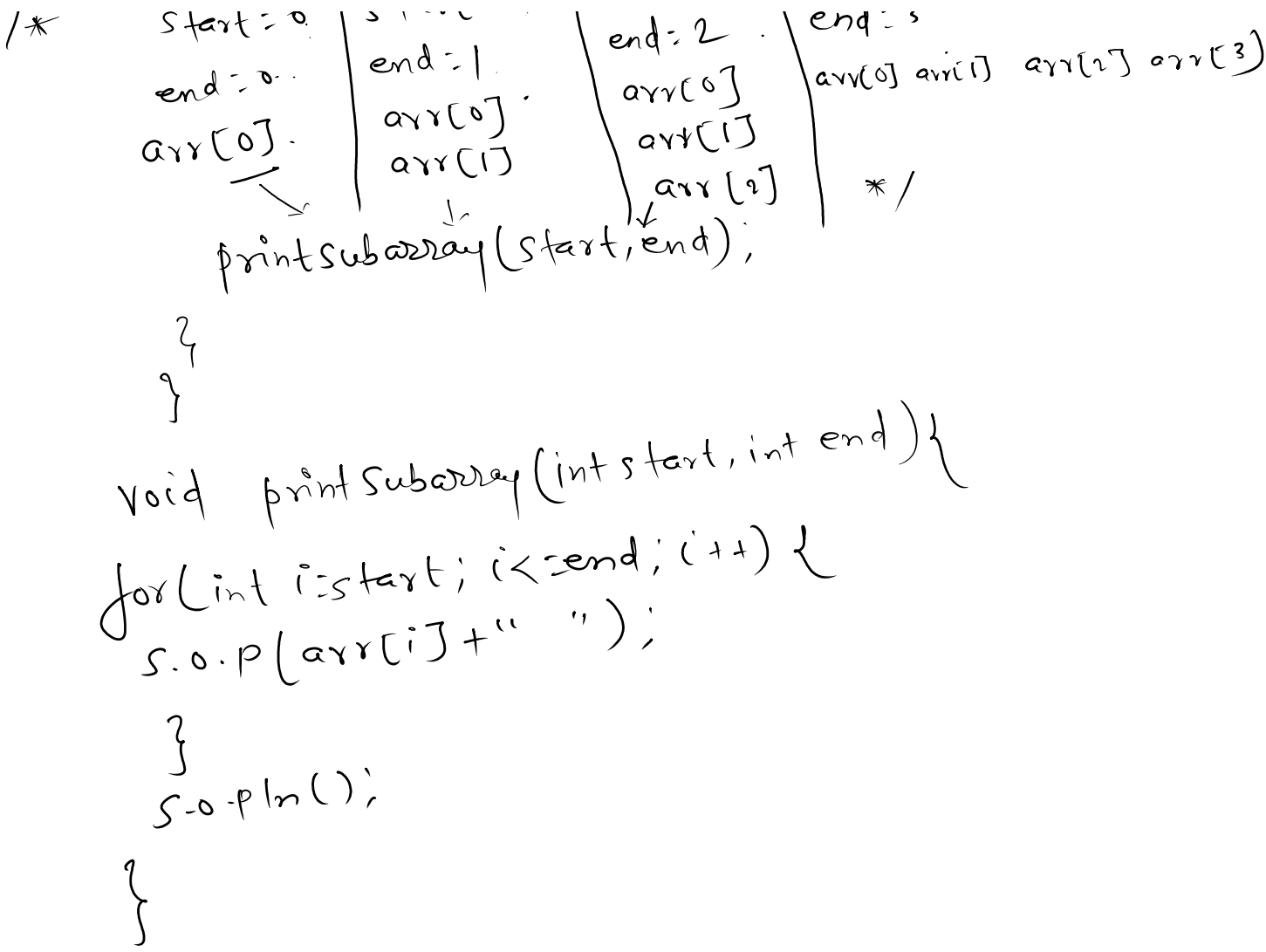
start:0, end:0 to 3

Start:1, end:1 to 3

start:2, end:2 to 3

start:3, end:3 to 3.

```
for(int start=0; start<n; start++){
    for(int end=start; end<n; end++){
/*    start:0 | start:0 | start:0 | start:0
      end:0   | end:1   | end:2   | end:3
                                    arr[0] arr[1] arr[2] arr[3]
```




```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class sh
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int arr[] = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = sc.nextInt();
        }

        for(int start =0;start<n;start++){
            for(int end = start; end<n;end++){
                printsubarryay(arr,start,end);
            }
        }
    }
     static void printsubarryay(int arr[], int start, int end){
        for(int i=start;i<=end;i++){
            System.out.print(arr[i]+" ");
        }
        System.out.println();
    }
}
```

Sum Equals zero

4 -> n

arr = [-1, 1, 2, 3]

Subarrays of given array

Sum		Sum
-1	-1	2 -> 2
0	-1 1	2 3 -> 5
2	-1 1 2	3 -> 3
5	-1 1 2 3	
1	1	
3	1 2	
6	1 2 3	

```
p.s.v.m(String[] args) {
    for(int start=0; start<n; start++) {
        for(int end=start; end<n; end++) {
            sum=0;
```

```
        int sum=0;
    for(int i=start; i<=end; i++){
        sum+=arr[i];
      }
      if(sum==0){
       s.o.pln("true");
        return;
      }
    }
  }
 }
 s.o.pln("false");
}
}
```

Max Subarray 2

n -> 5

arr = [-1, 2, 3, -2, 1]

Subarrays

-1	2	3 -2
-1 2	2 3	3 -2 1
-1 2 3	2 3 -2	-2
-1 2 3 -2	2 3 -2 1	-2 1
-1 2 3 -2 1	3	1

Time Complexity by finding all subarrays then find maxSum

↳ $O(n^3)$ (Brute Force Approach)

Kadane's Algorithm

↳ O(n)

It is used where we need to find maximum sum subarrays or maximum product subarrays

Example 1:

arr = [2, 4, 6, 1]

2, 4, 6 → 12

[2, 4, 6, 1] → 13

2
2 4
2 4 6
✓ 2 4 6 1

Example 2 arr = [10, 50, 2, 1]

[10, 50, 2, 1] → 63

Example 3. arr = [10, 50, -2, 1] max → min. value

[10, 50] = 60 ✓

[10, 50, -2, 1] = 59

sum: 0
sum: 10 ← max: 10
sum: 60 ← max = 60
sum: 58 ← max = 60
sum: 59 ← max = 60

Example 4. arr = [90, -20, 30, 80, 7]

[30, 80, 7] = 117

[90, -20, 30, 80, 7] = 187

sum: 0
sum = 90 + (-20)
= 70 + 30
= 100 + 80
= 180 + 7 = 187.

Example 5

arr = [40, -60, 15, 10, 25]

[15, 10, 25] → 50.

sum: 0
sum = 0 + 40 = 40 ← max
sum = 40 + (-60) = -20. max = 40

Sum = -20+15 = -5 , max=40
Sum = -5+10 = 5 , max=40
Sum = 5+25 = 30, max=40

sum<0
sum=0

Sum = 0+15 = 15 , max=40
Sum = 15+10=25, max=40
sum = 25+25 =50, max=50 ✓

Code:-

```
int sum=0; int maxSum=Integer.MIN_VALUE;
for(int i=0; i<n; i++){
  sum+=arr[i];
  if(sum>maxSum){
    maxSum=sum;
  }
  if(sum<0){
    sum=0;
  }
}
```

Example:-

sum=0

arr=[10, -5, -15, 20, 30]

arr = [10, -5, -15, 20, 30]

sum = 10, max = 10

sum = 10 + (-5) = 5, max = 10

sum = 5 - 15 = -10, max = 10

sum < 0, sum = 0

sum = 0 + 20 = 20, max = 20

sum = 20 + 30 = 50, max = 50