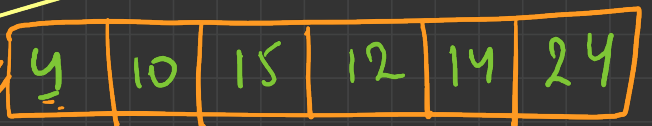
Prefix and Suffix

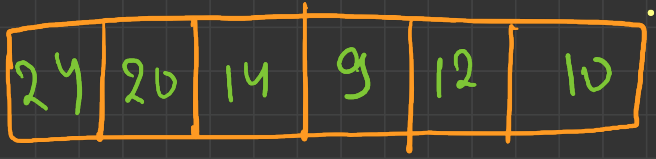
Prefix sum

Prefix

Suffix



→ Prefix Sum

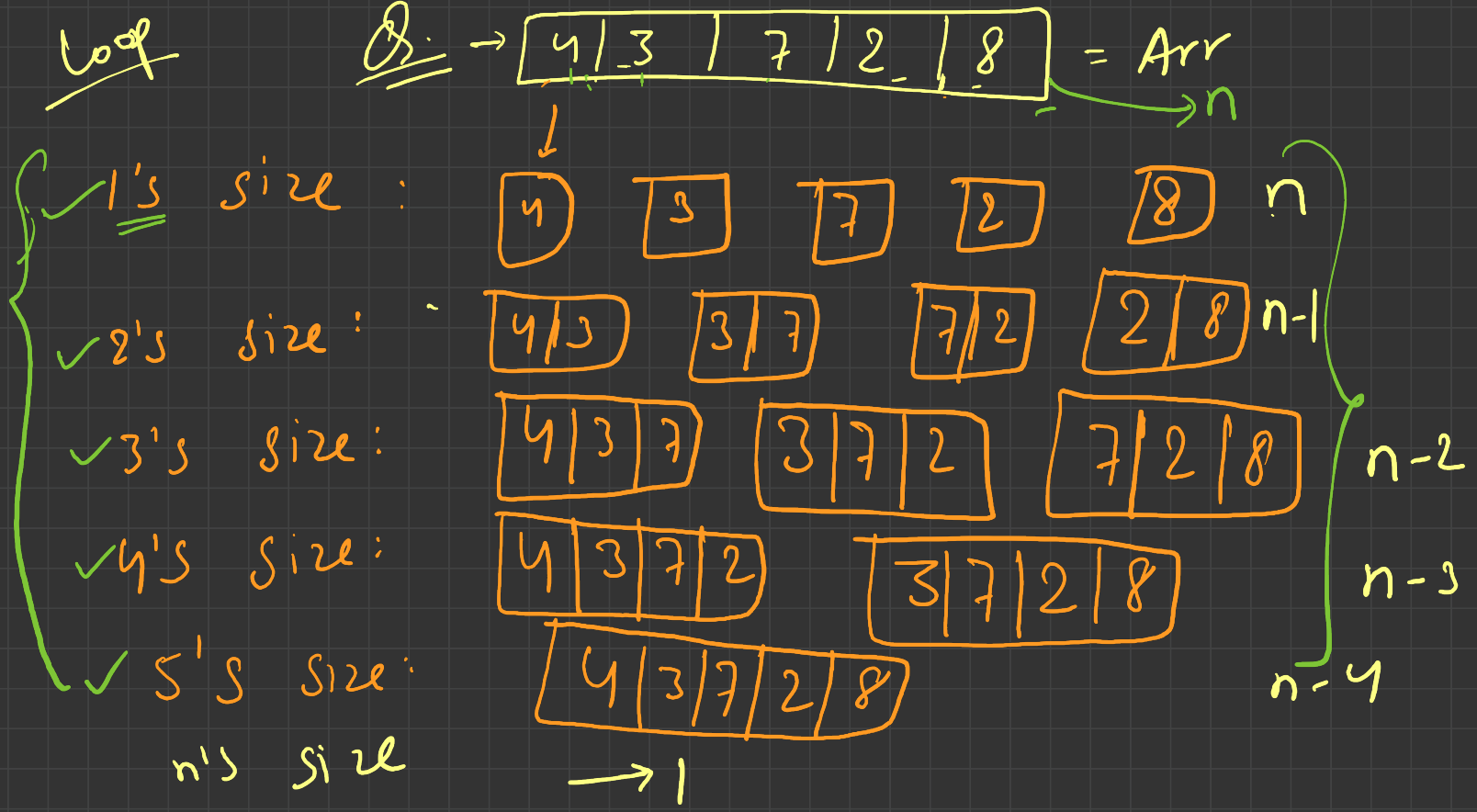


→ Suffix Sum

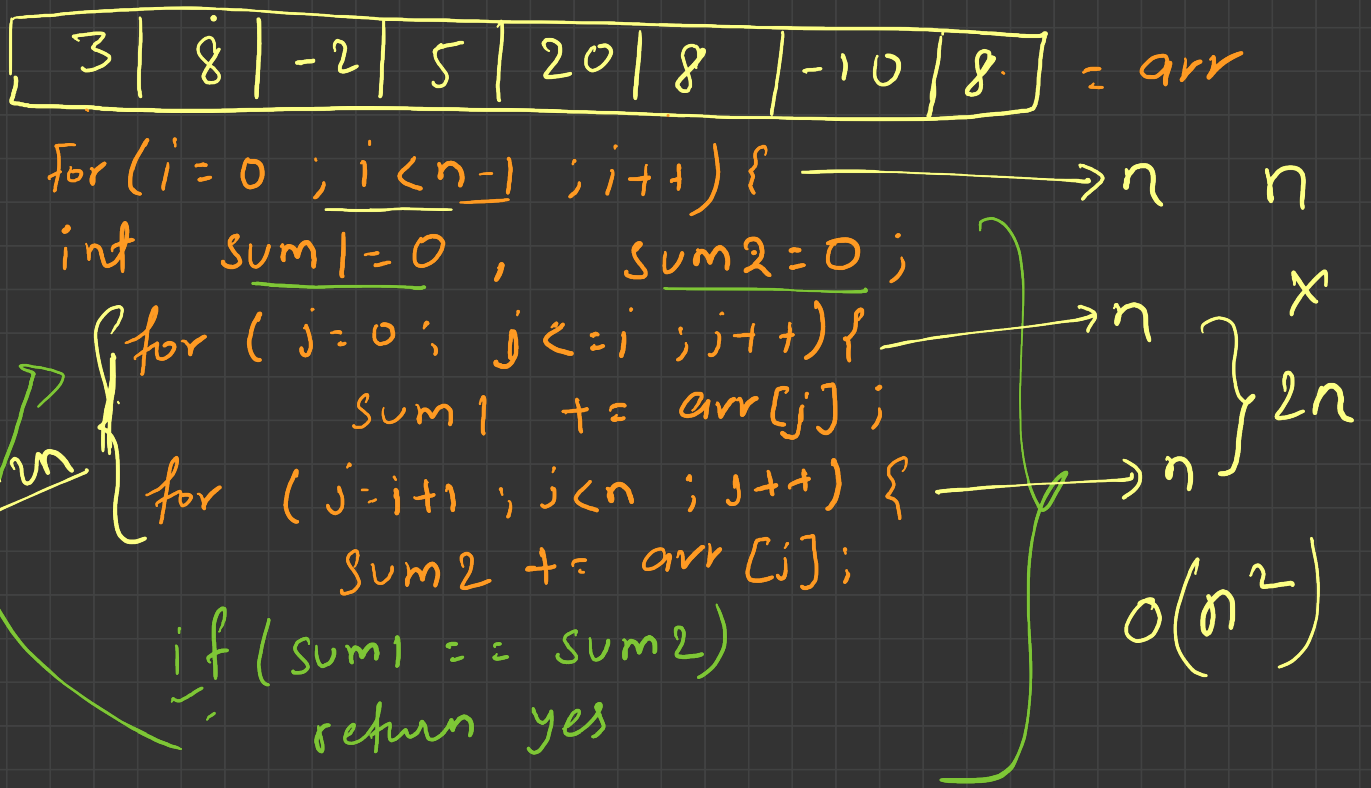
Prefix Sum

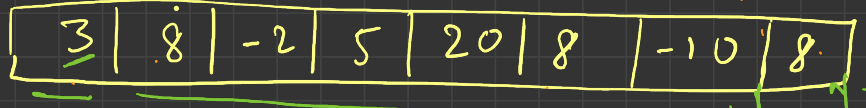
```
vector<int> prefix(n)
prefix[0] = arr[0];
for(i=1; i<n; i++){
prefix[i] = prefix[i-1] + arr[i];
}
```

H.W Suffix Sum Code



Divide array in 2 Subarray with equal sum





Total sum = 0;
for(i=0; i<n; i++) { Totalsum += arr[i]; }

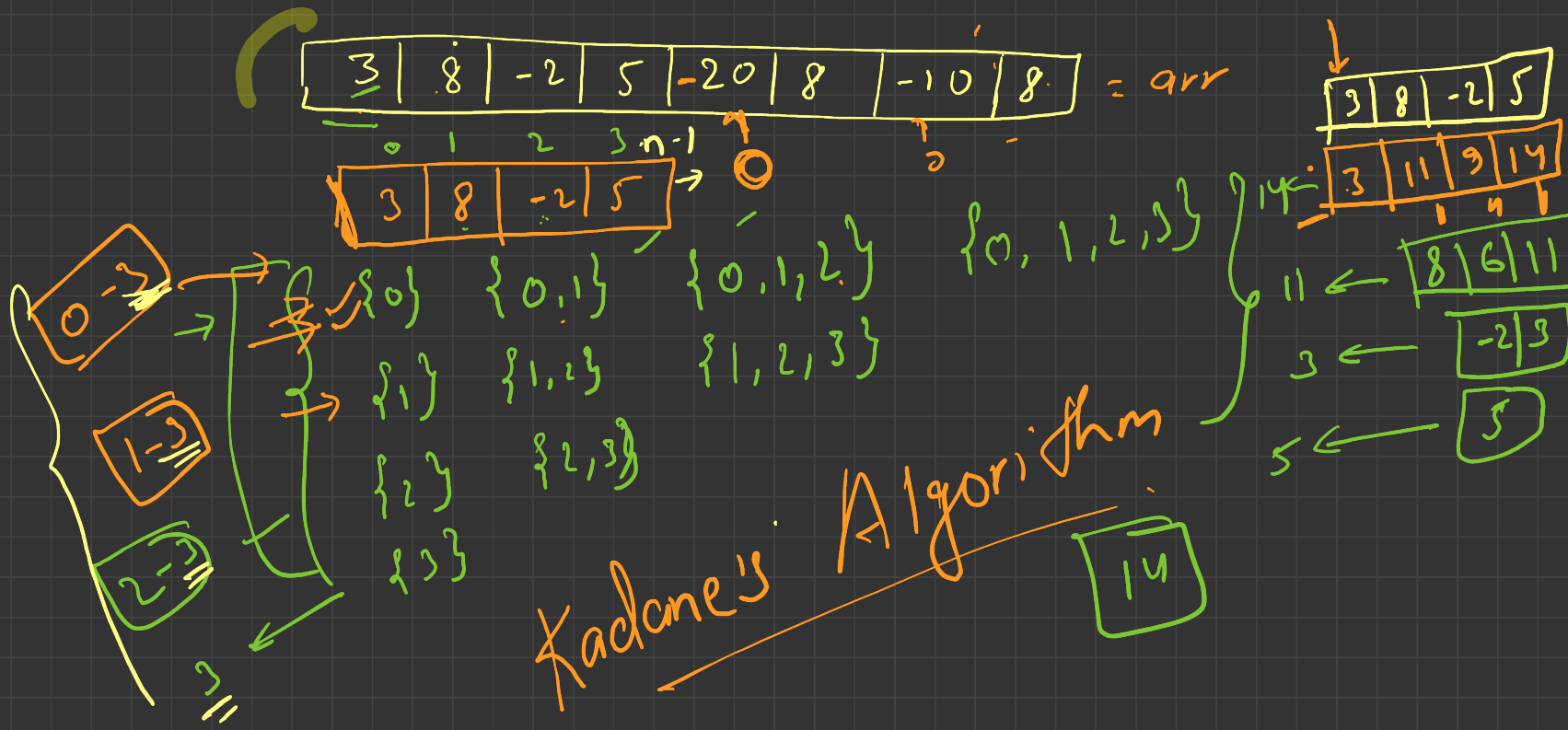
int prefix = 0;
for (i=0; i<n-1; i++)
prefix += arr[i];
ans = totalsum - prefix;
if (ans == prefix)
return 1
}
return 0;

n + n + n

O(n)

Prefix == Total - prefix

Largest Sum Contiguous Subarray



0-3
1-3
2-3
3-3

```
maxi = INT_MIN
for( i=0; i<n; i++) {
```

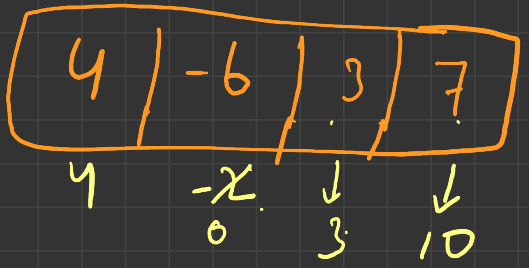
→ n

n × n

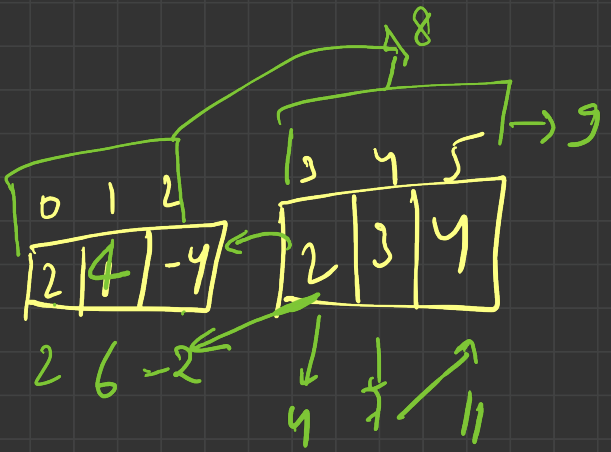
$O(n^2)$

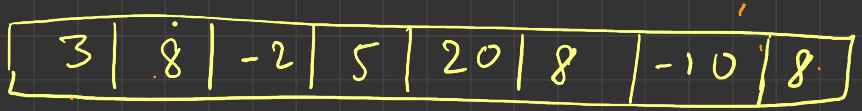
```
    prefix = 0
    For(j=i; j<n; j++) {
        prefix += arr[j]
        maxi = max(maxi, prefix);
    }
    return maxi;
```

→ n



maxi : INT_MIN 4 10



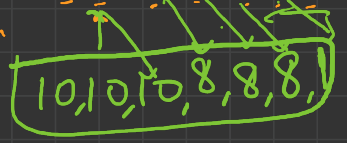



```
maxi : INT_MIN   prefix : 0;
for (i : 0; i < n; i++) {  ⟶  O(N)
    prefix += arr[i]
    maxi = max (maxi, prefix)
     if (prefix < 0) {
         prefix = 0;
```

Maximum Difference between Two Elements

ans = INT_MIN

arr = {2, 3, 10, 6, 4, 8, 1}




```
for (i=0; i<n-1; i++) {
    for (j=i+1; j<n; j++) {
        ans = max(ans, arr[j] - arr[i])
    }
}
```

$O(N^2)$

Maximum Difference between Two Elements

arr = {2, 3, 10, 6, 4, 8, 1} ⟶ Suffix

Suffix => 8

8 - 1 = 7