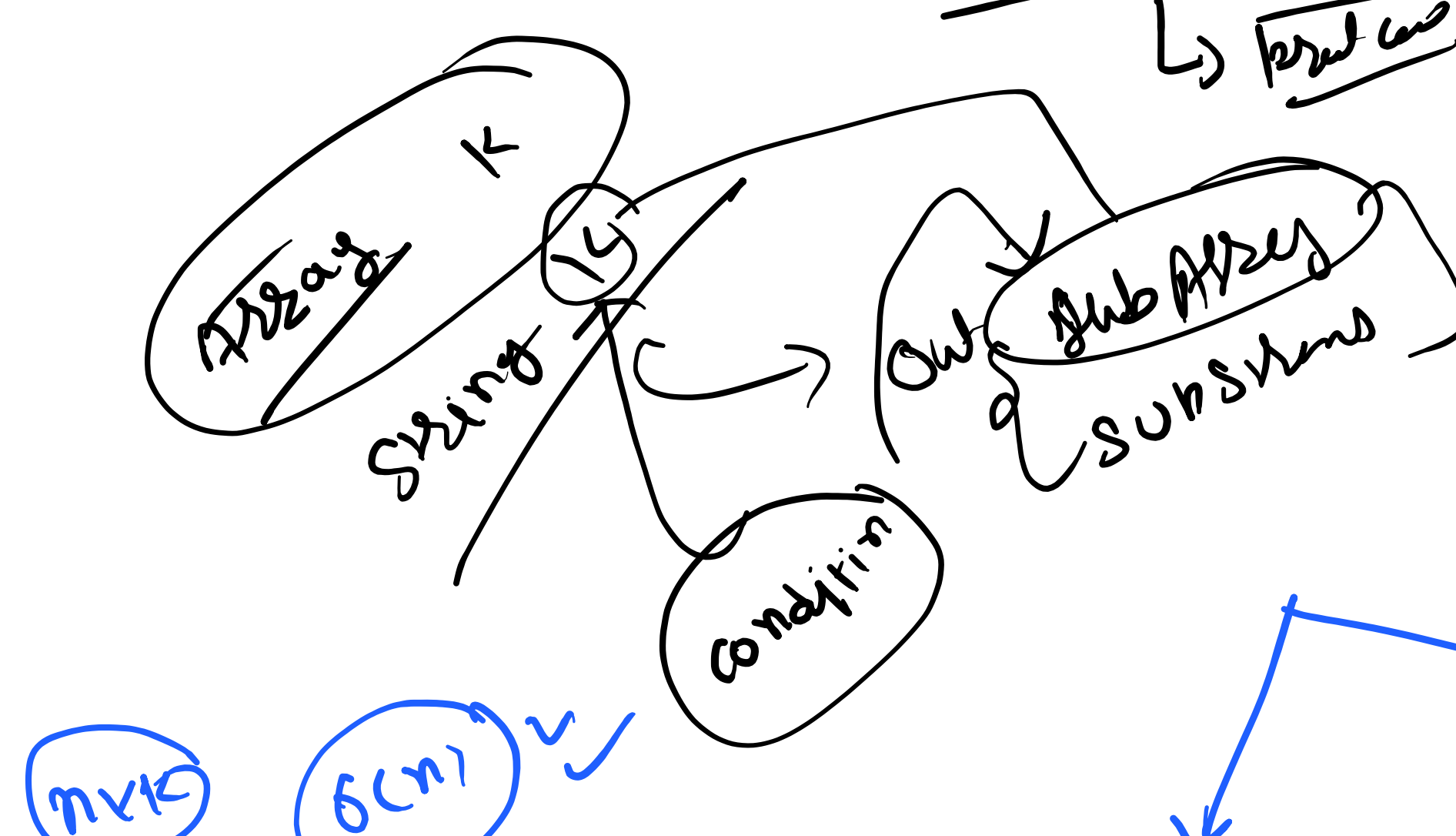


Sliding window → Product

① How to initialize prod. sliding



① 1st window arr
② grow
③ shrink
④ ans update

Fixed

K = 3
arr = [2, 3, 1, 4, 3, 7, 9, 2, 3]
int ans = 0, sum = 0
for (i = 0; i < n; i++) {
 sum = sum + arr[i];
 for (j = i; j < i + k; j++) {
 sum = sum + arr[j];
 ans = max(sum, ans);
 }
}

2 3 1 4 3 7 9 2 3
Sum = 6 + 4 - 2 = 8 + 3 - 3 = 8 + 7 - 1 = 14 + 9 - 4 = 19 + 2 = 21
ans = 8
8, 14, 19

2 3 1 → 6
3 1 4 → 8
1 4 3 → 8
4 3 7 → 14
3 7 9 → 19
7 9 2 → 18
9 2 3 → 14

1 + 2 + 3 + 1 + 2 = 9 ✓

1 2 3
1 2 3
4 2 3

0 - 0 = 1
1 - 0 = 2

si
[1 2 3 4 2]
ei

K = 10 ✓

int si = 0, ei = 0, p = 1
while (ei < n) {
 p = p * arr[ei];
 while (p >= k) {
 p = p / arr[si];
 si++;
 }
 ei++;
}

ans = ans * (ei - si + 1)
window size

```
public static int Product_Less_Than_K(int[] arr, int k) {
    int si = 0, ei = 0, p = 1, ans = 0;
    while (ei < arr.length) {
        // grow
        p = p * arr[ei];
        // shrink
        while (p >= k) {
            p = p / arr[si];
            si++;
        }
        // ans update
        ans += (ei - si + 1);
        ei++;
    }
    return ans;
}
```

0 > 0
1 > 0

1

si si si si
[1 2 3]
0 1 2
ei

[10, -2, -1, 3, -1, -3, 5, 3, 6, 7] K = 4

i - k = 4 - 4 = 0 S = 4 = 1

10 - 2 - 1 - 3

10 ✓
6 - 4 = 2
9 - 4 = 5
8 - 4 = 4

10 ✓
[-2, -1, 3, -1] 3 ✓
[-1, 3, -1, -3] 3
[3, -1, -3, 5] 5
[-1, 3, 5, 3] 5
[-3, 5, 3, 6] 6
[5, 3, 6, 7] 7

n - k + 1 = ✓

10 - 2 - 1 - 3 - 1 - 3 - 5 - 3 - 6 - 7
0 1 2 3 4 5 6 7 8 9

```
public static int[] Window_Maximum(int[] arr, int k) {
    int n = arr.length;
    int[] max_val = new int[n - k + 1];
    List<Integer> ll = new ArrayList<>();
    // 1st window ka maximum
    for (int i = 0; i < k; i++) {
        while (!ll.isEmpty() && ll.get(ll.size() - 1) < arr[i]) {
            ll.remove(ll.size() - 1);
        }
        ll.add(i);
    }
    int j = 0;
    max_val[j++] = arr[ll.get(0)];
}
```

[10, -2, -1, 3, -1, -3, 5, 3, 6, 7]
0 1 2 3 4 5 6 7 8 9

0 1 2 3

```
public static int[] Window_Maximum(int[] arr, int k) {
    int n = arr.length;
    int[] max_val = new int[n - k + 1];
    List<Integer> ll = new ArrayList<>();
    // 1st window ka maximum
    for (int i = 0; i < k; i++) {
        while (!ll.isEmpty() && arr[ll.get(ll.size() - 1)] < arr[i]) {
            ll.remove(ll.size() - 1);
        }
        ll.add(i);
    }
    int j = 0;
    max_val[j++] = arr[ll.get(0)];
    for (int i = k; i < arr.length; i++) {
        // grow
        while (!ll.isEmpty() && arr[ll.get(ll.size() - 1)] < arr[i]) {
            ll.remove(ll.size() - 1);
        }
        ll.add(i);
        // shrink
        if (ll.get(0) == i - k) {
            ll.remove(0);
        }
        // ans update
        max_val[j++] = arr[ll.get(0)];
    }
    return max_val;
}
```

10 3 3 5
0 1 2 3 4 5 6

[10, -2, -1, 3, -1, -3, 5, 3, 6, 7]
0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9