Sliding Window Maximum

nums = [1,3,-1,-3,5,3,6,7]   K = 3
(↑ window size)

1	3	-1	-3	5	3	6	7

∴ [3,3,5,5,6,7] —— ANSWER
→ maximum of all subarrays of size K.

Brute Force Approach

```
for(i=0 to n)
    for(j=1 to i+K)     } O(n)
        max
```

Optimal

- We will use deque (will store only viable answers)
- First element of current window will be maximum of window.

```
curr window
    currel >= dw.          } smaller
             dw.pop()
```

Pseudocode

① Analyze 1st window

```
for(i=0 to n){
while(dw.size()>0 && nums[dw.back()] <=
                               nums[i])
        dw.pop_back();
        dw.push_back(i)
    }
```

② SWM for other winds

```
for(i=k for ){
  res.pb(nums[dav.front()]) — ans
  while(dav.size() && dav.front <= i-k)
        dav.pop_front()
  while(dav.size()>0 && nums[dav.back()] <=
                          dav nums[i])
        dav.pop_back()
  }
  dav.push_back(i)
}
res.pb(nums[dav.front()]);
```

T.C = O(n) — linear

S.C = O(n)