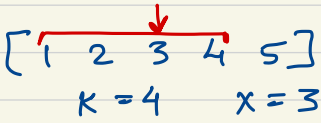
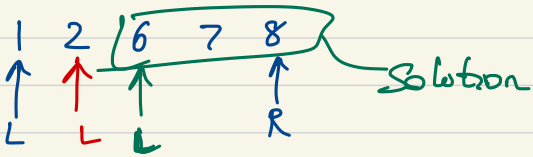
K closest elements



o/p: [1 2 3 4]

K = 4 x = 6



distL = |6 − 1| = 5 more ∴ L++
distR = |6 − 8| = 2

distL = |6 − 2| = 4 more : L++
distR = 2

Brute Force:

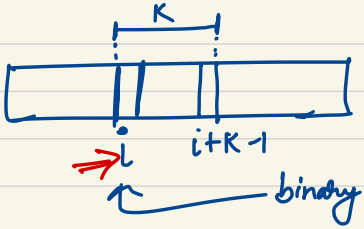
→ Sort by |val − x| then pick K elements
T.C : O(nlgn)

PQ:

→ Use Priority Queue
O(nlgK) : Comparator

T.C. : O(n)

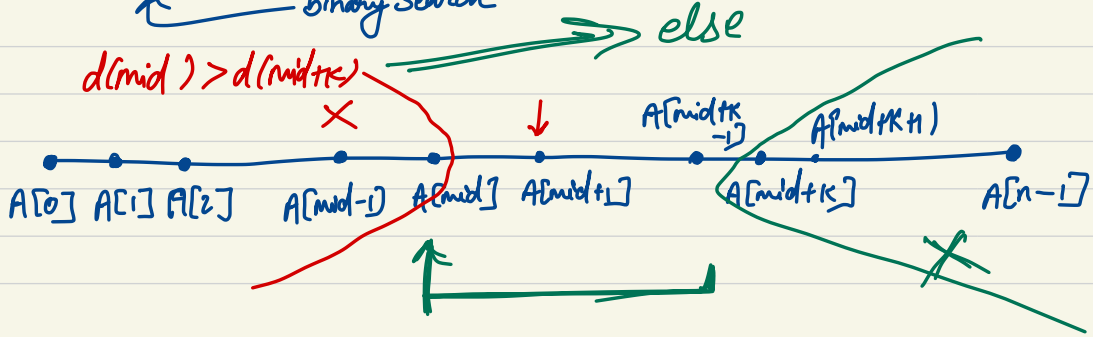
```
while (R - L + 1 > K)
    distL ← |x - A[L]|
    distR ← |x - A[R]|
    if distL > distR
        L++
    else
        R--
```

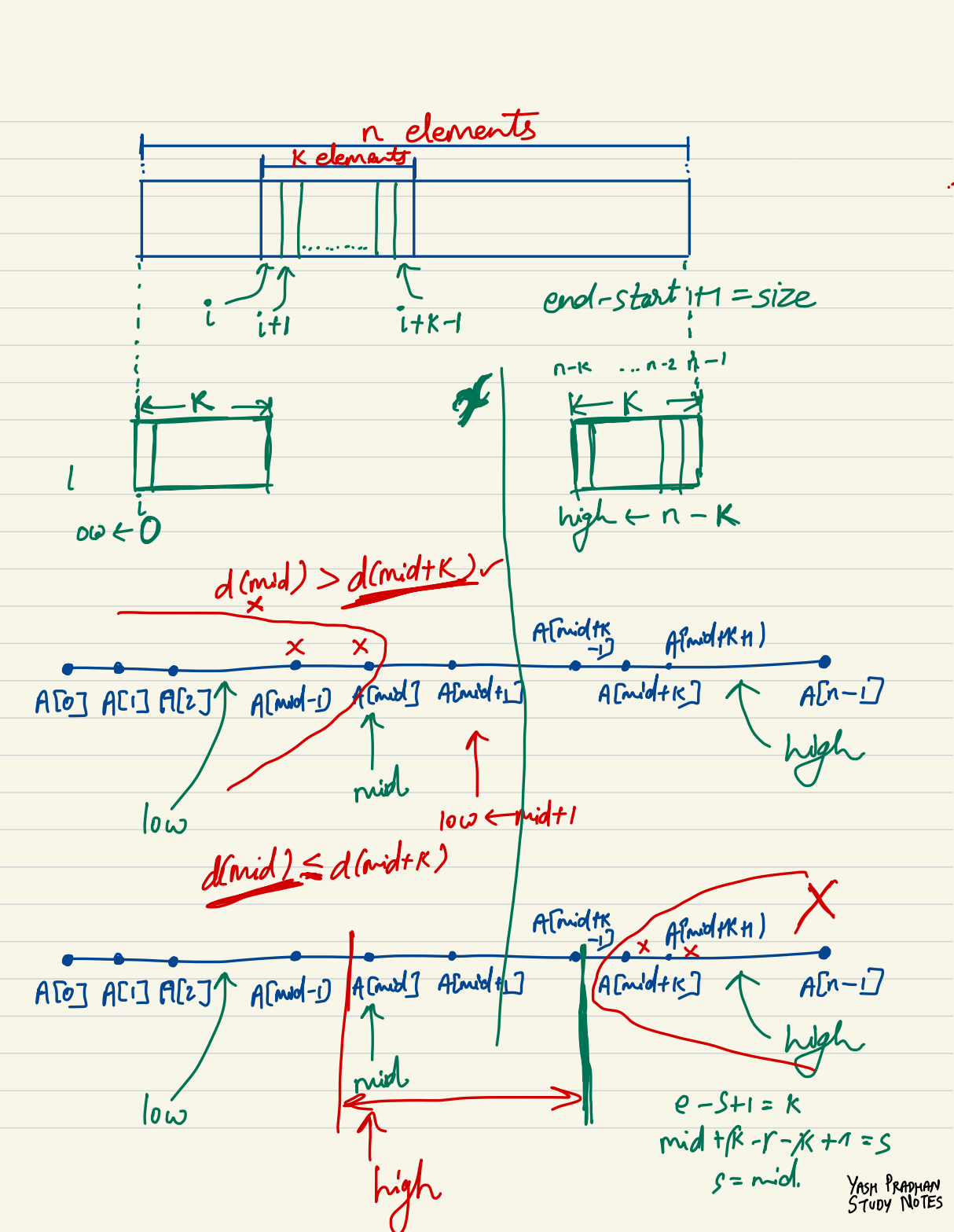


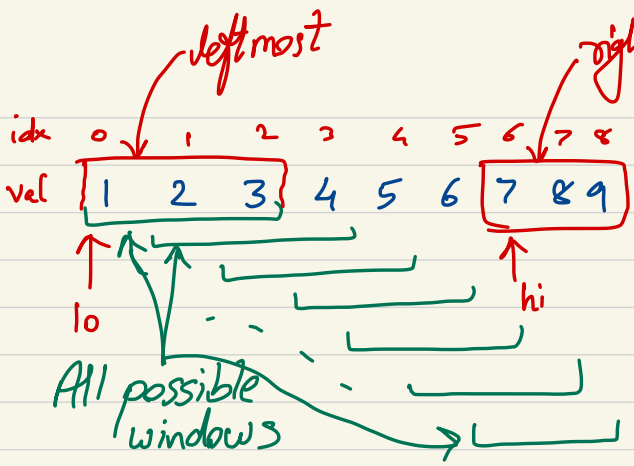
lower bound
i = 0

upper bound
i = n − K

Goal:
low at beginning of desired window




n elements
K elements
i
i+1
i+K-1
end-start i+1 = size
K
K
low ← 0
high ← n - K
d(mid) > d(mid+K)
A[0] A[1] A[2]
A[mid-1]
A[mid]
A[mid+1]
A[mid+K-1]
A[mid+K+1)
A[mid+K]
A[n-1]
low
mid
high
low ← mid+1
d[mid] ≤ d(mid+K)
e - s + 1 = K
mid + K - r - K + 1 = s
s = mid.
Yash Pradhan
Study Notes



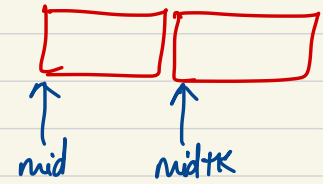
$k = 3 \quad n = 9$

$x = 5$

$lo \leftarrow 0$

$hi \leftarrow 9 - 3 = 6$

2 adjacent candidate windows.



while low ≤ high

mid ← (low + high)/2

if d(mid) > d(mid+k)

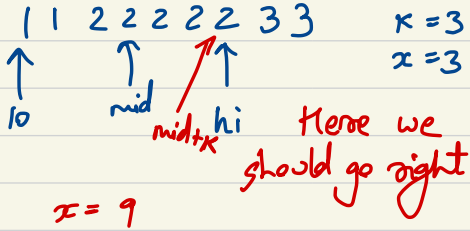
// mid is far away

lo ← mid+1

else

// mid+k far away.

hi ← mid



$x = 1 \quad x = 4 \quad x = 6 \quad x = 9$

value / index	3 (mid)	7 (mid+k)
	d(mid)	d(mid+k)
x = 1	-2 ✓✓	6
x = 4	1 ✓✓	3
x = 6	3	1 ✓✓
x = 9	6	-2 ✓✓

✓✓ indicates closer.

$d(mid) = x - arr[mid]$

$d(mid+k) = arr[mid+k] - x.$