# Sliding window

**Ques**

Max of every k size window

k = 4

s

10, 2, 1, 5, 3, 8, 4, 7, 9

e


k = 4

k == e - s + 1

10, 5, 8, 8, 8, 9

k = n/2

## Ques Sum of every k sized subarray

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

$K = 3$

13, 8, 9, 16, 15, 19, 20

0, 1, 2, 3, 4

1, 2, 3, 4, 5

0 + 1 + 2

1 + 2 + 3

2 + 3 + 4

(1) First window completely analysed

(2) sliding → s++ e++

(3) Ans $e - s + 1 == k$ → complete

(4) Stop when $e == a.length$.

```
s=0, e=0; sum=0;
while (e < a.length) {
    sum += a[e]; // work
    if (e-s+1 < k) {
        e++; // complete window
    }
    else if (e-s+1 == k) {
        // answer
        syso(sum);

        // remove previous computation
        sum = sum - a[s];

        // slide window
        s++;
        e++;
    }
}
```

Time $\Rightarrow$ $O(n)$

**Ques** First negative of every window $k=3$

5 4 -1 3 7 9 12 -5, -2, 8 -6

-1 -1 -1 0 0 -5 -5 -5 -2

| -1 | -5 | -2 | -6 |
|---|---|---|---|

---

**Ques** $p = abcd$

e b g c a d b c a e

s

e

$map$

| | |
|---|---|
| a | 1 |
| b | 1 |
| c | 1 |
| d | 1 |

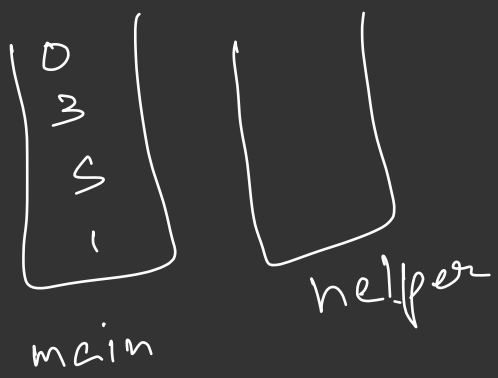
| | |
|---|---|
| a | 1 |
| d | 1 |
| b | 1 |
| c | 1 |

# Ques stack supporting min operations

push
pop
min

push(1) —
pop() —
push(3) —
push(0) — = 0
min() — O(1)
pop()
min() = 1

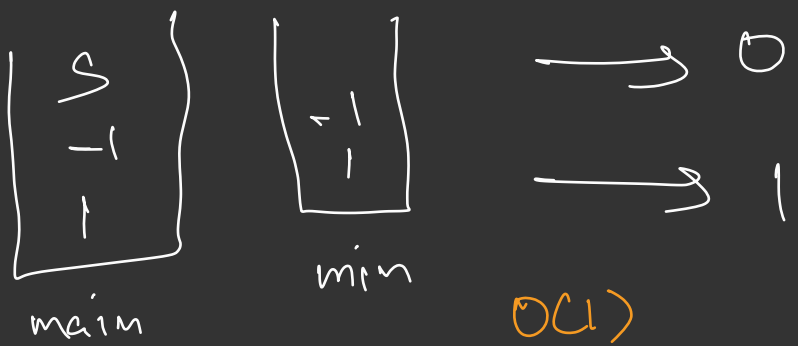
(1)




min = 0

O(n)

(2)




O(1)

S

4

6

1

pop

2

| |
|---|
| -4 |
| 6 |
| -5 |
| 5 |

min = ~~5~~ ~~4~~ ~~4~~ ~~4~~ 2

---

Ques target sum Quadruple

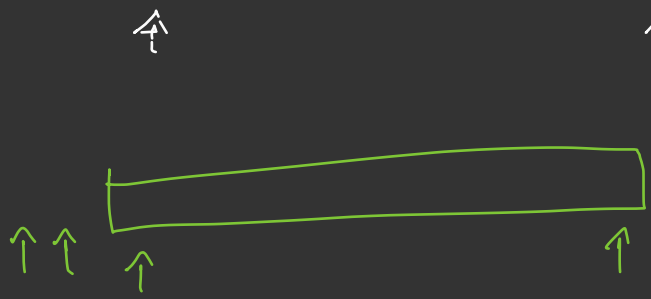
| 1 | 13 | 8 | 3 | 9 | 6 | 4 | 11 |
|---|---|---|---|---|---|---|---|

k = 24

```
for(  ) {
  for      {
    for    {
      for  {
      }
    }
  }
}
```

$O(n^4)$

① Sort $n \log n$



$O(n \times n \times n) + O(n \log n)$

$O(n^3)$

pair $\Rightarrow O(n \log n)$

triplet $\Rightarrow O(n^2)$

Quadruple $\Rightarrow O(n^3)$