



Maximum points you can obtain from cards

arr = [6 2 3 4 7 2 1 7 1] K = 4





Maximum points you can obtain from cards

arr = [6, 2, 3, 4, 7, 2, 1, 7, 1] K = 4





Maximum points you can obtain from cards

arr = [6 2 3 4 7 2 1 7 1] K = 4





Maximum points you can obtain from cards

arr = [6 2 3 4 7 2 1 7 1] K = 4





Maximum points you can obtain from cards

arr = [6 2 3 4 7 2 1 7 1] K=4





Maximum points you can obtain from cards

arr = [6 2 3 4 7 2 1 7 1] K=4





Maximum points you can obtain from cards

arr = [6 2 3 4 7 2 1 7 1] K = 4





Maximum points you can obtain from cards

arr = [6 2 3 4 7 2 1 7 1] K=4  
~~x~~







Maximum points you can obtain from cards

arr = [6 2 3 4 7 2 1 7 1] K=4

lsum = ~~0~~ 15

rsum = 0

sum





Maximum points you can obtain from cards

arr = [6 2 3 4 7 2 1 7 1] K=4

lsum = ~~0~~ 15

rsum = 0

sum  
15

lsum = 1



arr = [6 2 3 4 7 2 1 7 1] K=4

lsum = 15

rsum = 0

sum  
15

lsum = 11

rsum = 1

12

lsum = 8

rsum = 8

16



arr = [6 2 3 4 7 2 1 7 1] K=4

lsum = ~~15~~

rsum = 0

sum  
15

lsum = 11

rsum = 1

12

lsum = 8

rsum = 8

16

lsum = 6

rsum = 9

15



arr = [6 2 3 4 7 2 1 7 1] K = 4

lsum = <del>15</del>	rsum = 0	Sum 15
lsum = 11	rsum = 1	12
lsum = 8	rsum = 8	16
lsum = 6	rsum = 9	15
lsum = 0	rsum = 11	11





Maximum points you can obtain from cards

arr = [6 2 3 4 7 2 1 7 1] K=4

lsum = ~~0~~ 15

rsum = 0

sum

15

lsum = 11

rsum = 1

12

lsum = 8

rsum = 8

16

lsum = 6

rsum = 9

15

lsum = 0

rsum = 11

11





Maximum points you can obtain from cards

arr = [6 2 3 4 7 2 1 7 1] K = 4

```
func (nums, k)
```

```
{  
    sum = 0
```

```
    for (i = 0 → k - 1) { sum = sum + nums[i];
```





Maximum points you can obtain from cards

arr = [6 2 3 4 7 2 1 7 1] K = 4

```
func (nums, k)
```

```
{  
    sum = 0, rsum = 0, maxsum = 0  
    for (i = 0 → k-1) {sum = sum + nums[i];  
        maxsum = sum;
```







```
fun ( i = 0 ; i < n ; i++ ) { sum = sum + nums[i] ;  
    maxSum = (sum ;
```

```
    minden = n-1
```

```
fun ( i = k-1 ; i >= 0 ; i-- )  
{
```

```
    lsum = lsum - nums[i] ;
```

```
    rsum = rsum + nums[minden] ;
```

```
    minde
```





```
fun ( i = 0 ; i < n ; i++ ) { sum = sum + nums[i] ;  
    maxSum = (sum ;
```

```
    minSum = minSum ;
```

```
fun ( i = k-1 ; i >= 0 ; i-- )  
{
```

```
    lsum = lsum - nums[i] ;
```

```
    rsum = rsum + nums[rindex] ;
```

```
    rindex = rindex - 1 ;
```



arr = [6 2 3 4 7 2 1 7 1] K = 4

Handwritten annotations: A bracket above the first four elements (6, 2, 3, 4) with 'X' marks below them. Two arrows labeled 'n-1' point to the last two elements (7, 1) with checkmarks below them.

```
func (nums, k)
```

```
{  
    lsum = 0, rsum = 0, maxsum = 0
```

```
    for (i = 0 → k-1) { lsum = lsum + nums[i];  
        maxsum = lsum;
```

```
        rindex = n-1
```

```
    for (i = k-1; i ≥ 0; i--)
```

```
    { lsum = lsum - nums[i];
```

```
        rsum = rsum + nums[rindex];
```

```
        rindex = rindex - 1;
```

```
        maxsum = max(maxsum, lsum + rsum);
```

```
    }
```





```
lsum = rsum - nums[i];
```

```
rsum = rsum + nums[rindex];
```

```
rindex = rindex - 1;
```

```
maxsum = max(maxsum, lsum + rsum);
```

```
}
```

```
return maxsum;
```



arr = [6 2 3 4 7 2 1 7 1] K = 4

Handwritten annotations: A bracket above the first four elements (6, 2, 3, 4) with 'X' marks below them. An arrow points from 'n-1' to the last element (1), with a checkmark below it.

```

func (nums, k)
{

```

lsum = 0, rsum = 0, maxsum = 0

$O(K) \leftarrow \text{for } i = 0 \rightarrow k-1$  lsum = lsum + nums[i];  
maxsum = lsum;

rindex = n-1

$O(K) \leftarrow \text{for } i = k-1 ; i \geq 0 ; i--$

lsum = lsum - nums[i];

rsum = rsum + nums[rindex];

rindex = rindex - 1;

maxsum = max(maxsum, lsum + rsum);

}





```
lsum = rsum + nums[rindex];  
rindex = rindex - 1;
```

```
    maxSum = max(maxSum, lsum + rsum);  
}  
return maxSum;  
}  
  
TC  $\rightarrow O(2 \times K)$   
SC  $\rightarrow O(1)$ 
```





```
func (nums, k)
```

```
{  
    sum = 0, rsum = 0, maxsum = 0  
    o(k) ← fun (i = 0 → k-1) {sum = sum + nums[i];  
        maxsum = sum;
```

```
        rindex = n-1  
    o(k) ← fun (i = k-1 ; i ≥ 0 ; i--)
```

```
{  
    sum = sum - nums[i];  
    rsum = rsum + nums[rindex];  
    rindex = rindex - 1;
```

```
        maxsum = max(maxsum, sum + rsum);  
    }  
    return maxsum;  
}
```

TC →  $O(2 \times k)$

SC →  $O(1)$

