



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

lsum = ~~2~~ 15

rsum = 0

Sum
15

lsum = 11

rsum = 1

12





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

		Sum
$lsum = 15$	$rsum = 0$	15
$lsum = 11$	$rsum = 1$	12
$lsum = 8$	$rsum = 8$	16





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

$$lsum = 15$$

$$rsum = 0$$

Sum
15

$$lsum = 11$$

$$rsum = 1$$

12

$$lsum = 8$$

$$rsum = 8$$

16





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

$$lsum = 15$$

$$rsum = 0$$

Sum
15

$$lsum = 11$$

$$rsum = 1$$

12

$$lsum = 8$$

$$rsum = 8$$

16

$$lsum =$$





		Sum
$lsum = \cancel{0} 15$	$rsum = 0$	15

$lsum = 11$	$rsum = 1$	12
-------------	------------	----

$lsum = 8$	$rsum = 8$	<u>16</u>
------------	------------	-----------

$lsum = 6$	$rsum = 9$	15
------------	------------	----

$lsum = 0$	$rsum = 11$	11
------------	-------------	----





		Sum
$lsum = \cancel{15}$	$rsum = 0$	15

$lsum = 11$	$rsum = 1$	12
-------------	------------	----

$lsum = 8$	$rsum = 8$	<div>16</div>
------------	------------	---------------

$lsum = 6$	$rsum = 9$	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

fun



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
```

```
    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 , nsum = 0 , maxsum = 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, maxsum = 0, maxsum = 0
```

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



Maximum points you can obtain from cards

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

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

```
{  
    sum = 0, maxSum = 0, maxSum = 0
```

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



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;
```

```
    for (i = k-1 ; i >= 0 ; i--)
```

```
        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;
```

```
    //
```

```
    for (i = k-1 ; i >= 0 ; i--)
```

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





?

```
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 =
```





?

```
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];  
        }
```



Maximum points you can obtain from cards

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

Annotations: A bracket above the first four elements [6, 2, 3, 4] is crossed out with a large 'X'. The last element '1' is marked with a checkmark and an arrow pointing to it from the label 'n-1'.

```
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];
```



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

```
rindex = n - 1
```

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

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

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

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





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

```
minLen = n - 1
```

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

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

```
    sum = sum + nums[minLen] ;
```

```
    minLen = minLen - 1 ;
```

```
maxSum = max(maxSum,
```



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

Annotations: A bracket is drawn over the first four elements [6, 2, 3, 4]. The elements 3 and 4 are crossed out with 'X'. The elements 7 and 1 at the end are checked with '✓'. Arrows point to the 7 and 1 with the label 'n-1'.

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

```
{
    sum = 0, rsum = 0, maxSum = 0
```

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

```
        rindex = n-1
```

```
        for (i = k-1; i >= 0; i--)
```

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

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

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

```
            maxSum = max(maxSum, sum + rsum);
```

```
        }
```



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

Annotations: A bracket above the first four elements [6, 2, 3, 4] is crossed out with an 'X'. The last two elements [7, 1] are checked with a checkmark. Arrows point to the last element '1' with the label 'n-1'.

```

func (nums, k)
{

```

```

    sum = 0, rsum = 0, maxSum = 0

```

```

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

```

```

    rindex = n-1

```

```

    for (i = k-1; i >= 0; i--)
    {

```

```

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

```

```

    maxSum = max(maxSum, sum + rsum);

```

```

}

```





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

```
    maxSum = max(maxSum, lsum + rsum);  
}  
return maxSum;
```





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

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





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

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

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

```
        rinden = n-1
```

```
        for (i = k-1; i >= 0; i--)
```

```
        {  
            lsum = lsum - nums[i];  
            rsum = rsum + nums[rinden];  
            rinden = rinden - 1;
```

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

```
TC → O(2 × k)
```

```
SC → O(1)
```





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

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

```
        rindex = n-1  
    O(k) ← for (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 → $O(1)$

THANKS FOR WATCHING!

Don't forget to subscribe



TAKE U FORWARD

