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

Maximum points you can obtain from cards

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

sum

lsum = 15 rsum = 0

Maximum points you can obtain from cards

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

lsum = 15 rsum = 0 sum 15

lsum = 1

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

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

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

			sum
lsum = ~~0~~ 15	rsum = 0		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

			Sum
lsum = ~~0~~ 15	rsum = 0		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

		Sum
lsum = ~~0~~ 15	rsum = 0	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 )
{
    int 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 )
{
    lsum = 0 , rsum = 0, maxSum = 0
    for ( i = 0 -> k-1) lsum = lsum + nums[i];
        maxSum = lsum;
```

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

    rindex = n-1
for ( i = k-1 ; i >= 0 ; i-- )
{
    lsum = lsum - nums[i] ;
    rsum = rsum + nums[rindex] ;
    rinde
```

```
for ( i = 0 ; i < k ; i++ ) { 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 ;
```

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

n-1

```
fun ( nums , k )
{
    lsum = 0 , rsum = 0, mansum = 0
    for ( i = 0 → k-1 ) lsum = lsum + nums[i] ;
    mansum = lsum ;

    rindex = n-1
    for ( i = k-1 ; i >= 0 ; i-- )
    {
        lsum = lsum - nums[i] ;
        rsum = rsum + nums[rindex] ;
        rindex = rindex - 1 ;

        mansum = max(mansum, lsum + rsum) ;
    }
```

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

n-1

```
fun ( nums , k )
{
    lsum = 0 , rsum = 0, mansum = 0
O(K) ← fun ( i = 0 → k-1) lsum = lsum + nums[i] ;
    mansum = lsum;

    rindex = n-1
O(K) ← fun ( i = k-1 ; i >= 0 ; i--)
    {
        lsum = lsum - nums[i] ;
        rsum = rsum + nums[rindex] ;
        rindex = rindex - 1 ;

        mansum = max(mansum, lsum + rsum) ;
    }
```

```
        rsum = rsum + nums[rinden];
        rinden = rinden - 1;

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

TC → O(2×k)
SC → O(1)

```
func ( nums , k )
{
    lsum = 0 , rsum = 0 , mansum = 0
O(K) ← for ( i = 0 → k-1 ) lsum = lsum + nums[i] ;
    mansum = lsum ;

    rindex = n-1
O(K) ← for ( i = k-1 ; i >= 0 ; i-- )
    {
        lsum = lsum - nums[i] ;
        rsum = rsum + nums[rindex] ;
        rindex = rindex - 1 ;

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

TC → O(2×K)
SC → O(1)