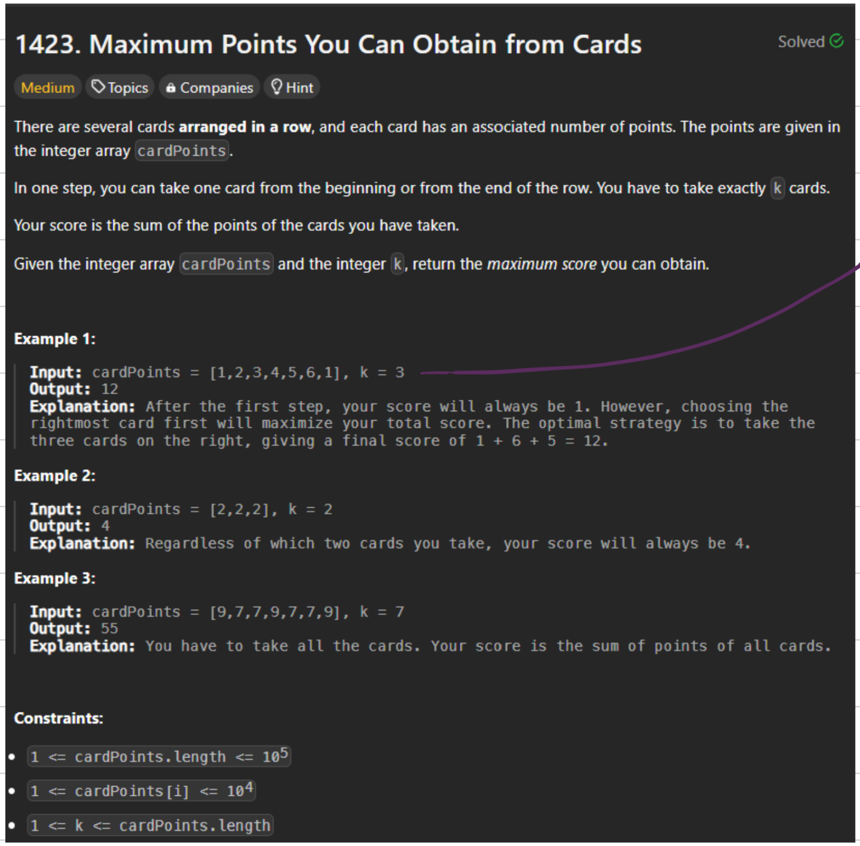
Max Points you can obtain from Cards



1423. Maximum Points You Can Obtain from Cards

Solved

Medium Topics Companies Hint

There are several cards **arranged in a row**, and each card has an associated number of points. The points are given in the integer array `cardPoints`.

In one step, you can take one card from the beginning or from the end of the row. You have to take exactly `k` cards.

Your score is the sum of the points of the cards you have taken.

Given the integer array `cardPoints` and the integer `k`, return the *maximum score* you can obtain.

Example 1:

```
Input: cardPoints = [1,2,3,4,5,6,1], k = 3
Output: 12
Explanation: After the first step, your score will always be 1. However, choosing the
rightmost card first will maximize your total score. The optimal strategy is to take the
three cards on the right, giving a final score of 1 + 6 + 5 = 12.
```

Example 2:

```
Input: cardPoints = [2,2,2], k = 2
Output: 4
Explanation: Regardless of which two cards you take, your score will always be 4.
```

Example 3:

```
Input: cardPoints = [9,7,7,9,7,7,9], k = 7
Output: 55
Explanation: You have to take all the cards. Your score is the sum of points of all cards.
```

Constraints:

- `1 <= cardPoints.length <=` 10^5
- `1 <= cardPoints[i] <=` 10^4
- `1 <= k <= cardPoints.length`

problem Statement is :

~~You can pick~~

→ arr = [1, 2, 3, 4, 5, 6, 1] , k = 3

if k = 3, then these are the following ways to pick elements

[1, 2, 3, 4, 5, 6, 1] → Sum = 6

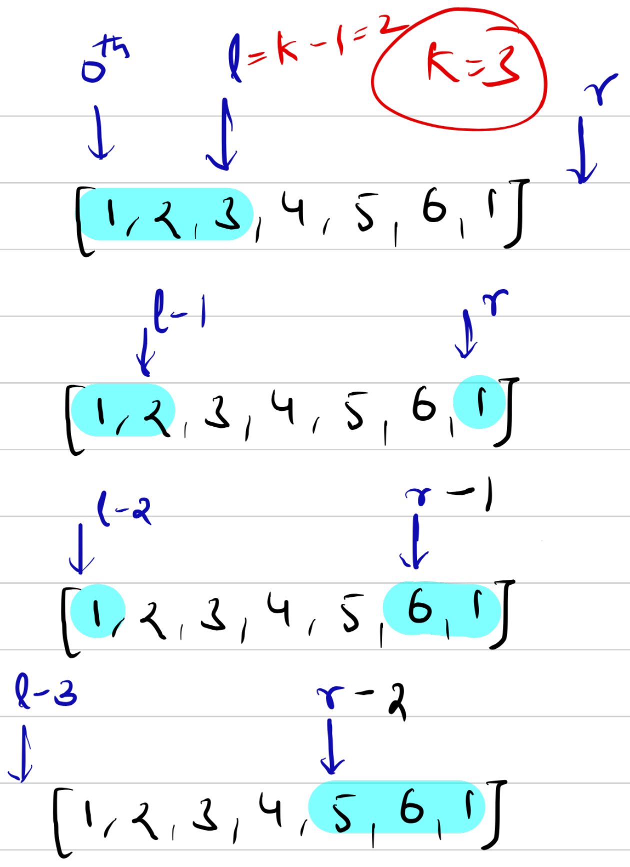
[1, 2, 3, 4, 5, 6, 1] → Sum = 4

[1, 2, 3, 4, 5, 6, 1] → Sum = 8

[1, 2, 3, 4, 5, 6, 1] → Sum = 12

↑ max ans

Either you can pick from front or back (in Sequence)
You Can't pick from middle



* If we observe, we can handle these 4 cases with 2 movable pointers & Slide them accordingly
  (l, r)

* Initially, l is at $(k-1)$th position
  r is at (n)th position

* Sum up from 0th index to l

* Now decrement l and subtract value of l

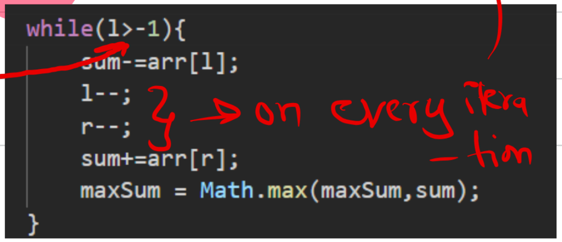
* Move right pointer to left and add value of r to the Sum

$$\text{arr[7]} : \{1, 2, \overset{\overset{l}{\downarrow}}{3}, 4, 5, 6, 1\}^{\overset{r}{\downarrow}} , \; k = 3$$

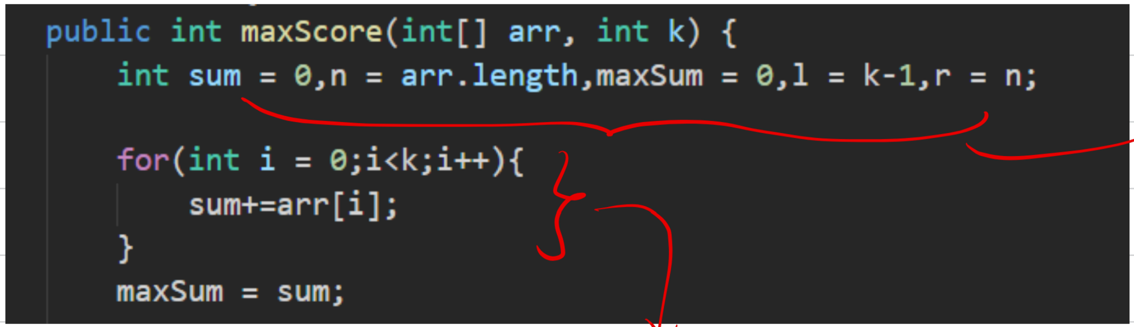
lsum + rsum

	leftSum	rightSum	maxSum	
$\{\underbrace{\boxed{1, 2, \overset{l}{3}}}_{\text{sum}} 4, 5, 6, 1\}^{r}$	6	0	$\cancel{-\infty}$ 6	(l--, r--)
$\{1, \overset{l}{2}, 3, 4, 5, 6, \overset{r}{1}\}$	$6 - \underset{\uparrow \text{ previous(l)}}{\overset{3}{\cancel{arr}[l]}} = 3$,	0+1	6	(l--, r--)
$\{\overset{l}{1}, 2, 3, 4, 5, \overset{r}{6}, 1\}$	3 - 2 = 1,	1+6 = 7	$\cancel{7}$ 9	(l--, r--)
$^{l}\{1, 2, 3, 4, \overset{r}{5}, 6, 1\}$	1 - 1 = 0,	7 + 5 = 12	$\cancel{9}$ 12	

* When l hits -1th index return maxSum



```
while(l>-1){
    sum-=arr[l];
    l--;
    r--;
    sum+=arr[r];
    maxSum = Math.max(maxSum,sum);
}
```



```
public int maxScore(int[] arr, int k) {
    int sum = 0,n = arr.length,maxSum = 0,l = k-1,r = n;

    for(int i = 0;i<k;i++){
        sum+=arr[i];
    }
    maxSum = sum;
```

Initializing all Variables, as Said in 2nd Slide

l is at (k-1)th position

r is at nth position

Sum of 0th to lth position Values