Leetcode – 1423

Maximum points you can obtain from cards

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

Now by seeing constraint if we use n^2 type of code then (10^5 *10^5)/10^8 still 100 seconds and more and in cp questions should be in 1 second

We can take cards from only beginning or from end :

What shall we do now:

1. Suppose start k elements give maximum sum to us
2. Now make a loop from start to end to check elements like :
3. Like as for (i=k-1;i>=0;i--) and then remove elements from left and add from right and check it will give maximum or not if it is then goahead else check all possibilities

```
// code
class Solution {
    public int maxScore(int[] cardPoints, int k) {
        int leftsum=0,rightsum=0;
        int maxsum=Integer.MIN_VALUE;
        int n=cardPoints.length;

        for (int i=0;i<k;i++)
        {
         leftsum+=cardPoints[i];
        }
        maxsum=leftsum;
        int rightindex=n-1;
        for (int i=k-1;i>=0;i--)
        {
         leftsum=leftsum-cardPoints[i];
         rightsum=rightsum+cardPoints[rightindex];
         rightindex--;
```

```
        maxsum=Math.max(maxsum,leftsum+rightsum);
       }
return maxsum;


    }
}
```