**SLIDING WINDOW ALGORITHM PROBLEM :**

This is problem solving technique that is design to transform to nested loop into a single loop

**Sliding window algorithm problem :**

1 . The problem in based on array list , string type

2 . it will ask you to sub ranges in the array to give the longest shortest or target value of the string .

3 . it concept is mainly based on ideas like the longest sequence or shortest sequence of the string the satisfies the technique .

package day25;

import java.util.\*;

public class task2 {

public static int maxsum(int[] arr, int k) {

int wsum = 0;

int msum = Integer.***MIN\_VALUE***;

System.***out***.println(msum);

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

wsum += arr[i];

}

msum=wsum;

for(int i = k; i < arr.length; i++) {

wsum = wsum - arr[i - k] + arr[i];

msum = Math.*max*(msum, wsum);

}

return msum;

}

public static void main(String[] args) {

Scanner in = new Scanner(System.***in***);

int[] arr = Arrays.*stream*(in.nextLine().split(" ")).mapToInt(x -> Integer.*parseInt*(x)).toArray();

int n = arr.length;

int k = in.nextInt();

System.***out***.println(*maxsum*(arr, k));

}

}

2 9 31 -4 21 7

3

48

**BIT MANIPULATION :**

Data type :

A data type allocate memory in the ram to store the data this storage has a data representation only used In bits 0 & 1 . for instance a data type that can store 3 bits has the combination of , 0 0 0 , 0 0 1 , 0 1 0 , 0 1 1 , -4 -> 1 0 0 , -3 -> 1 0 1 , -2 -> 1 1 0 , -1 -> 1 1 1 since the same 3 bits are used for storing both positive and negative data we store the data with the concept of dividing the combination exactly in two parts beg at 0 , beg at 1 . the 1 with start are negative . the combination is determine the **2 ^ (no of pos).**

BINARY CONVERSION :

import java.util.Scanner;

public class task3 {

public static void main(String[] args) {

Scanner s = new Scanner(System.***in***);

int n = s.nextInt();

String res = "";

while(n>=1)

{

int x = n%2;

n = n/2;

res = x + res;

}

System.***out***.println(res);

System.***out***.println(Integer.*toBinaryString*(13)); 🡪 inbuild class

}

}

13

1101

1101

**toBinaryString() 🡪** implements the power of conversion of the data based on the number of bits that represents the data in that number system .

System.***out***.println(Integer.*toOctalString*(1345));

System.***out***.println(Integer.*toHexString*(5463));

System.***out***.println(Integer.*parseInt*(str,2));

System.***out***.println(Integer.*parseInt*(str,8));

-3 -> 1 0 1 representation

Use the 2 compliment to get the positive 3 .

& , && operation

7 & 8 🡪 0 1 1 1

1 1 1 1

------------

0 1 1 1 🡪 7 so answer is 7 .

**LEFT SHIFT :**

Num \* base ^ shift 🡪 base – 2

**RIGHT SHIFT :**

Num \* base ^ - shift 🡪 base – 2

**Negate :** ~5 -> - ( num+1) 🡪 the – only add the last .

**Turn on – turn of :**

Use & by 0

10101001

00000000

-------------

00000000

**No change use & by 1 .**

10101001

1

-------------

10101001

**MASK :** ~(1 << i)