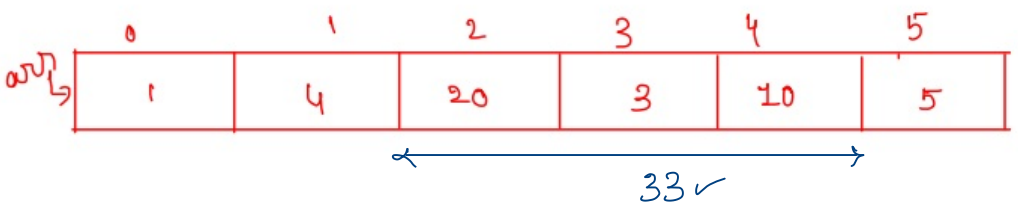
S2-Class2 [Sliding Window-2]

Q) Return True : If there exists a sub array whose sum is equal to given sum

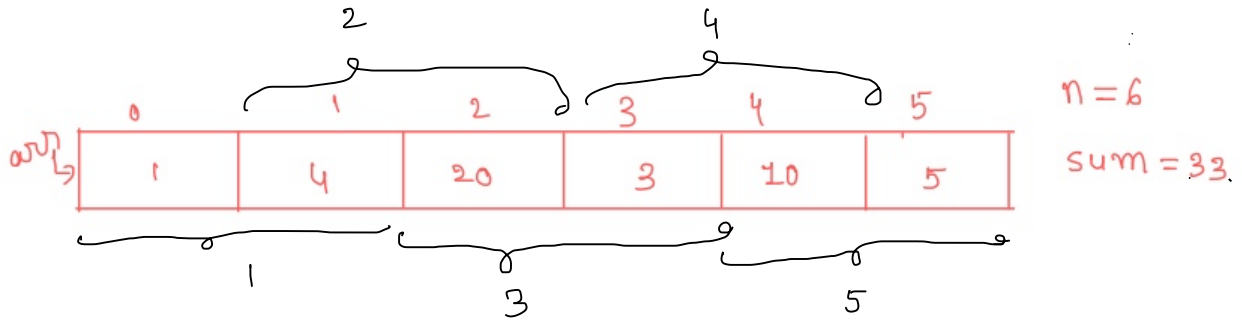
False : otherwise



n = 6

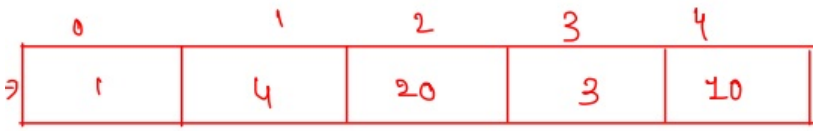
sum = 33

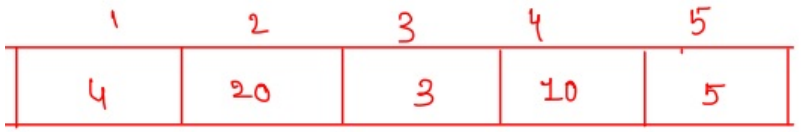
o/p :- T



$n = 6$

sum = 33.





$k = 5$ $L_5 = 2$ SA (n).

$k = 6$ $L_6 = 1$ SA (n)

$n + n + \cdots + n = 6 \cdot n$

General: $n \cdot n = O(n^2)$ ✓

BF:-

# of subarrays $= \frac{n(n+1)}{2}$

$n = 6 \Rightarrow \frac{6 \times 7}{2} = 21$

L_1 L_2 ... L_6

$|L_5| = 2$

✓✓ BF $\rightarrow O(n^3)$ ✓

$L_1 = 6$ SA comparison. (n)

---

* $L_2 = 5$ SA Loop 5 times, every time

$k = 2$ (n)

$L_3 = 4$ SA

$k = 3$ (n)

$L_4 = 3$ SA

$k = 4$ (n)

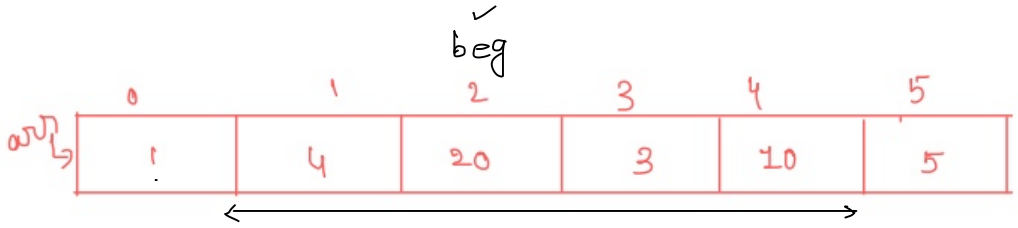
$O(n^3)$ ××

→ $O(n^2)$ × ××

↳ $O(n\log n)$ ✓

*$O(n)$ ✓

↳ ×



AIOBE

$n = 6$

$sum = 33$ (given)

- sub-array
- Sum = 33 (T)

↳ res = 0 + 1 = 1 + 4 = 5

5 + 20 = 25

25 + 3 = 28

1 + 4 + 20 + 3 (28) + 10 = 38

repeat

38 − 1 = 37

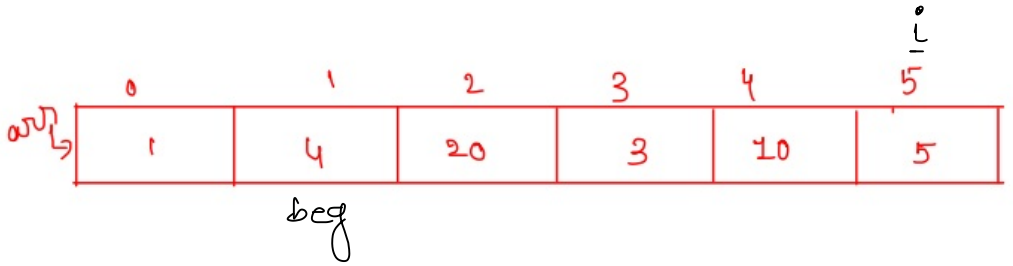
37 − 4 = 33

res ✓

res V/S sum

c_1	c_2	c_3
res == sum	res < sum	res > sum
stop (T)	res = res + a[i]	repeat (res > sum) res = res − arr[beg] beg ++

```
while(beg<n && res>sum)
{
    res=res-arr[beg];
    beg++;
}
```



n = 6

sum = 33

res = 0

1 5

25

28

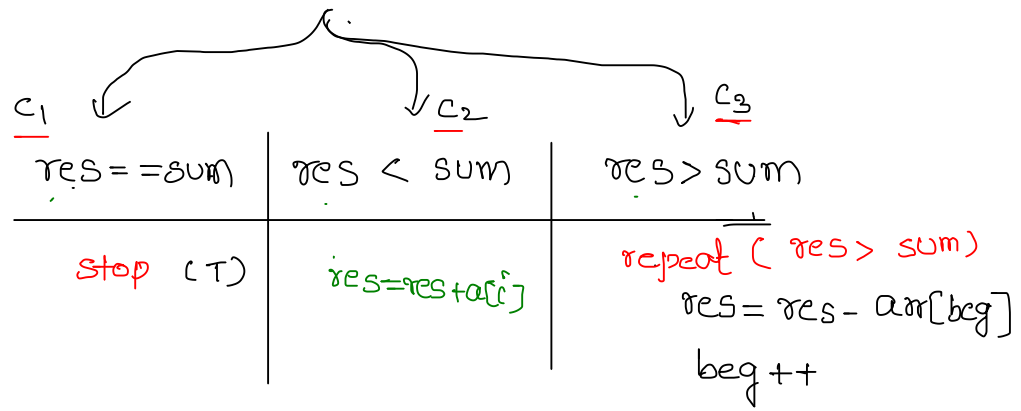
38

38 < 38x

37 33

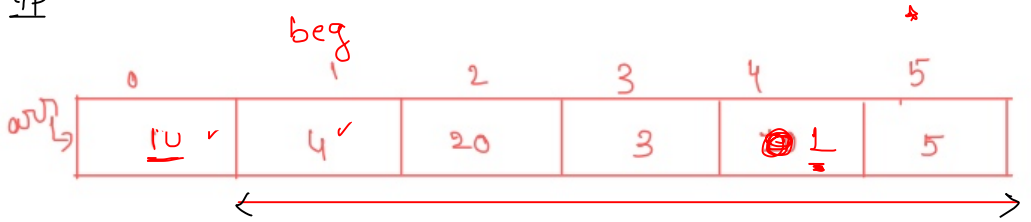
```
boolean fun(int arr[], int n, int sum)
{
        int res=0, beg=0;
        for(int i=0;i<n;i++)
        {
            C2  if(res<sum)
                {
                        res=res+arr[i];
                }
            C1  if(res==sum)
                        retrun true;
            C3  while(beg<n && res>sum)
                {
                        res=res-arr[beg];
                        beg++;
                }
        }
        retrun false;
}
```

res V/S sum




```
while(beg<n && res>sum)
{
        res=res-arr[beg];
        beg++;
}
```

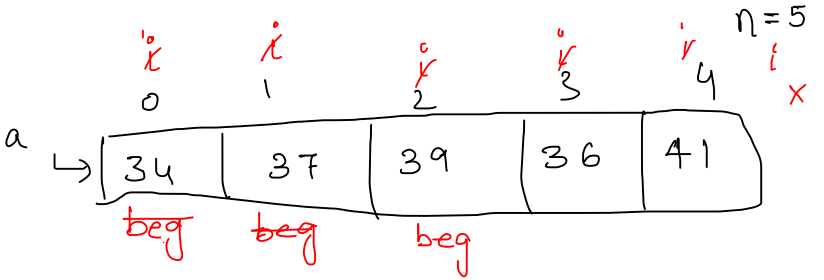
i/p



n = 6

sum = 33

o/p




```
boolean fun(int arr[], int n, int sum)
{
        int res=0, beg=0;
        for(int i=0;i<n;i++)
        {
            -> if(res<sum)
               {
                    res=res+arr[i];
               }
            if(res==sum)
                    retrun true;
            while(beg<n && res>sum)
               {
                    res=res-arr[beg];
                    beg++;
               }
        }
        retrun false;
}
```

10 + 4 + 20 = 34 - 10 = 24

24 + 3 = 27 + 1 = 28 + 5

= 33

res = 0 ~~34~~ 0 ~~37~~ 0

```java
import java.util.Scanner;
class Main
{
public static void main(String args[]){
        Scanner sc =  new Scanner(System.in);
        int tc = sc.nextInt();

        for(int i = 0;i<tc;i++){
            int n = sc.nextInt();
            long[] arr = new long[n];
            long k = sc.nextLong();
            for(int j=0;j<n;j++){
                arr[j]= sc.nextLong();
            }
            boolean result = isSum(arr,n,k);
            if(result){
                System.out.println("Yes");
            }
            else{
                System.out.println("No");
            }
        }
    }
public static boolean isSum(long arr[], int n, long sum)
{
        long res=0; int beg=0;
        for(int i=0;i<n;i++)
        {
                if(res<sum)
                {
                        res=res+arr[i];
                }
                if(res==sum)
                        return true;
                while(beg<n && res>sum)
                {
                        res=res-arr[beg];
                        beg++;
                }
        }
        return false;
}
}
```

→ O(n²) ✗ Analysis

```java
public static boolean isSum(long arr[], int n, long sum)
{
        long res=0; int beg=0;
        for(int i=0;i<n;i++)
        {
                if(res<sum)
                {
                        res=res+arr[i];
                }
                if(res==sum)
                        return true;
                while(beg<n && res>sum)
                {
                        res=res-arr[beg];
                        beg++;
                }
        }
        return false;
}
}
```

25 * C

i = 0 → j : 5 times

i = 1 → j = 5 . . .

i = 5 → j = 5

$\Rightarrow O(n^2)$

---

i = 0 → n times

i = 1 → n times

i = 2 → n times

i = n → n times

j loop;

$n + n + n + \ldots + n = n * n = n^2 \Rightarrow O(n^2)$

---

i = 0 → n

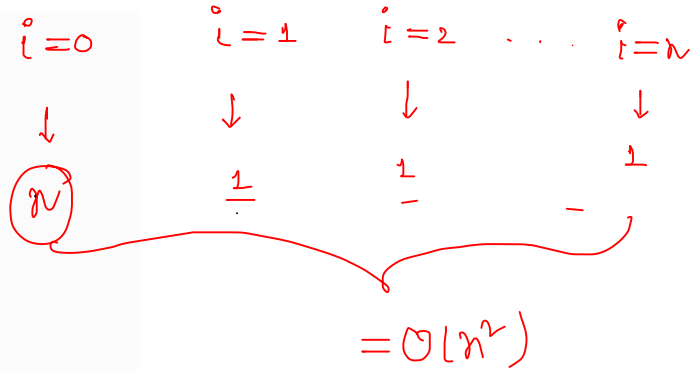
i = 1 → n − 1

i = 2 → n − 2

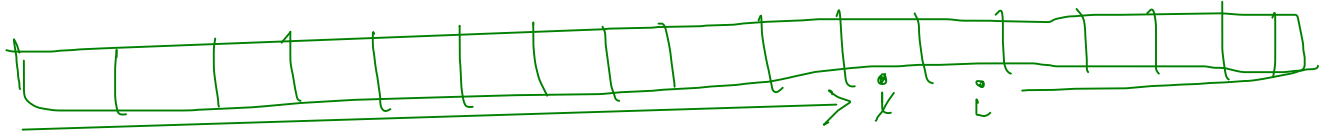
i = n → 1

$1 + 2 + 3 + \ldots + n - 1 + n = \frac{n(n+1)}{2} = O(n^2)$

```java
public static boolean isSum(long arr[], int n, long sum)
{
        long res=0; int beg=0;
        for(int i=0;i<n;i++)
        {
                if(res<sum)
                {
                        res=res+arr[i];
                }
                if(res==sum)
                        return true;
                while(beg<n && res>sum)
                {
                        res=res-arr[beg];
                        beg++;
                }
        }
        return false;
}
}
```



$$(1)\ n + (n-1)(1) = n+n-1 = 2n-1 = O(n)$$




```java
public static boolean isSum(long arr[], int n, long sum)
{
        long res=0; int beg=0;
        for(int i=0;i<n;i++)
        {
                if(res<sum)
                {
                        res=res+arr[i];
                }
                if(res==sum)
                        return true;
                while(beg<n && res>sum)
                {
                        res=res-arr[beg];
                        beg++;
                }
        }
        return false;
}
}
```

beg

i=0 → 1 → beg=1

i=1 → 1 → beg=2

i=2 → 1 → beg=3

i=3 → 1 → beg=4

10th: i=n → 1

= O(n)

---

i=0 → X → beg=0

i=1 → X → beg=0

i=2 → X → beg=0

i=n-1 → ✓ → n

O(n)

---

i=0 → X

i=1 → X

i=2 → X

i=n/2

prob name :- TF

every problem

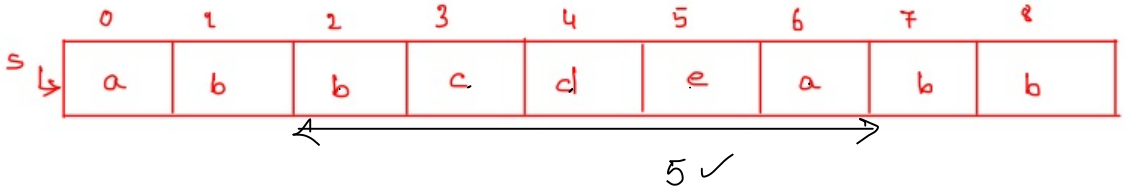
	Ap_1	Ap_2	Ap_3
	BF	sw (fixed)	sw (variable)
T.C	$O(n^3)$	$O(n^2)$	$O(n)$
S.C	$O(1)$	$O(1)$	$O(1)$

→ LBR

→ $O(\log n)$ } NOT possible.
$O(\sqrt{n})$

OJ BF

9) Find the size of largest sub-string which doesn't contains any repeated characters in given string



```
function longestUniqueSubsttr(s,n)
  {

    let hm be a hashmap/ object

    maximum_length = 0;


    start = 0;

    for(i= 0; i < n; i++)
    {

      if(hm.containsKey(s[i]))
      {
        start = Math.max(start, hm.get(s[i]) + 1);
      }
     hm.put(s[i], i);
      maximum_length = Math.max(maximum_length, i-start + 1);

    }
    return maximum_length;
  }
```