Agenda

1) Distinct numbers in window

2) No. of distinct 2D points

3) Class object as key

Q.1 Given an array, calculate no. of distinct elements in every subarray of size K.

	0	1	2	3	4	5	6	7
A :	2	4	3	8	3	9	4	9

K = 4

0 to 3 → 4

1 to 4 → 3

2 to 5 → 3

3 to 6 → 4

4 to 7 → 3

Idea : using sliding window with hashset.

		s			e			
	0	1	2	3	4	5	6	7
A :	2	4	3	8	3	9	4	9

ans → 0 to 3
(last window ans)

→ from ans remove impact of A[s-1]

→ Add impact of A[e]

i) calculate ans for first window → 0 to k-1.

ii) Then apply sliding window tech. on rest of windows.

$$\begin{array}{lcccccccc} & & & & s & & & e & \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A: & 2 & 4 & 3 & \underline{8} & \underline{3} & \underline{9} & \underline{4} & 9 \end{array}$$

$k = 4$

s	e	removing	Adding	hashset	ans
0	3	–	–	$\boxed{2\ 4\ 3\ 8}$	4
1	4	A[0]	A[4]	$\boxed{4\ 3\ 8}$	3
2	5	A[1]	A[5]	$\boxed{9\ 3\ 8}$	3
3	6	A[2]	A[6]	$\boxed{9\ 4\ 8}$	3 ✗

HashSet ✗

let's try hashmap.

	0	1	2	3	4	5	6	7
A :	2	4	3	8	3	9	4	9

$k = 4$

s	e	removing	Adding	hash map	ans
0	3	-	-	$2 \to 1$ $4 \to 1$ $3 \to 1$ $8 \to 1$	4
1	4	A[0]	A[4]	$4 \to 1$ $3 \to 2$ $8 \to 1$	3
2	5	A[1]	A[5]	$3 \to 2$ $8 \to 1$ $9 \to 1$	3
3	6	A[2]	A[6]	$3 \to 1$ $8 \to 1$ $9 \to 1$ $4 \to 1$	4
4	7	A[3]	A[7]	$3 \to 1$ $9 \to 2$ $4 \to 1$	3

```
void  solve ( int [] A, int k ) {
    // calculate the ans. of 1st window
    HashMap < Integer, Integer > map = new HashMap<>();
    for (int i=0; i<k; i++) {
        if (map.containsKey (A[i]) == false) {
                map.put (A[i], 1);
        }
        else {
                int temp = map.get(A[i]);
                temp++;
                map.put(A[i], temp);
        }
    }
    SOPln(map.size());
    int s= 1, e= k;
    while (e < A.length)   {
            // remove the impact of A[s-1]
            int f= map.get (A[s-1]);
            f--;
            map.put (A[s-1], f);
            if ( map.get (A[s-1])= = 0 ) {
                    map.remove(A[s-1]);
            }
        // add the impact of A[e]
        if (map.containsKey(A[e]) == false) {
                map.put(A[e], 1);
        }
        else {
                int temp = map.get(A[e]);
                temp++;
                map.put(A[e], temp);
        }

        SOPln(map.size());

        s++; e++;
    }
}
```

total subarrays of k len = $n-k+1$

(for → k)

(while → n-k)

total itr : $k + n - k = n$

TC : $O(n)$

SC : $O(n)$ { map can't contain more than k element in it }

Dry run

```
void  solve ( int [] A, int  K ) {
    // calculate  the  ans.  of  1st  window
    HashMap < Integer, Integer > map = new HashMap<>();
    for (int  i=0;  i<k;  i++) {
        if  (map.containsKey (A[i]) == false) {
                map.put (A[i], 1);
        }
        else  {
                int  temp = map.get (A[i]);
                temp++;
                map.put (A[i], temp);
        }
    }
    SOPln (map.size());
    int  s= 1,  e= k;
    while  (e < A.length)   {
        // remove  the  impact  of  A[s-1]
        int  f =  map.get (A[s-1]);
        f--;
        map.put (A[s-1], f);
        if ( map.get (A[s-1]) == 0 ) {
              map.remove (A[s-1]);
        }

        // add  the  impact  of  A[e]
        if (map.containsKey (A[e]) == false) {
              map.put (A[e], 1);
        }
        else  {
              int  temp = map.get (A[e]);
              temp++;
              map.put (A[e], temp);
        }

        SOPln (map.size());
        s++;  e++;
    }
}
```

K = 4

$$A = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 2 & \underline{4} & \underline{2} & \underline{6} & \underline{7} \end{array}$$

s	e	A[s-1]	A[e]
1	4	2	2
2	5	3	6
3	6	2	7
4	7		

```
2 → 1
4 → 1
6 → 1
7 → 1
```

ans: 3 3 3 4

Q-2 Given a 2D array denoting points on a 2D plane. Return total no. of distinct points in the array.

```
A = { {5,6},
      {2,8},
      {-1,-1},
      {2,-3},
      {2,8},
      {7,7},
      {2,8},
      {2,-3}
    };
```

distinct points: 5

(5,6) (2,8) (-1,-1)
(2,-3) (7,7)

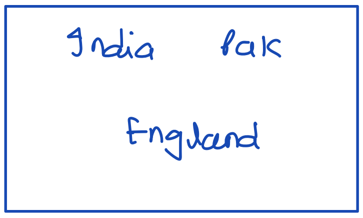
```
A = { {5,2},
      {1,-1},
      {-1,-1},
      {-1, 1},
      {-1,-1}
    }
```

distinct points:

(5,2) (1,-1) (-1,-1)
(-1,1)

one point == another point when both x and y are same.

```
HashSet <String> hs = new HashSet<>();

hs.add ("India");

hs.add ("Pak");

hs.add ("India");

hs.add ("England");
```

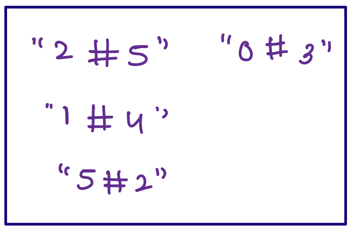


idea
```
{ String str = x + "#" + y;
  hs.add (str);
```

A :

	0	1
0	2	5
1	1	4
2	5	2
3	2	5
4	0	3

i	x	y	str
0	2	5	"2#5"
1	1	4	"1#4"
2	5	2	"5#2"
3	2	5	"2#5" ✗
4	0	3	"0#3"



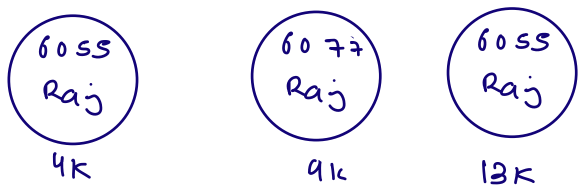
hs

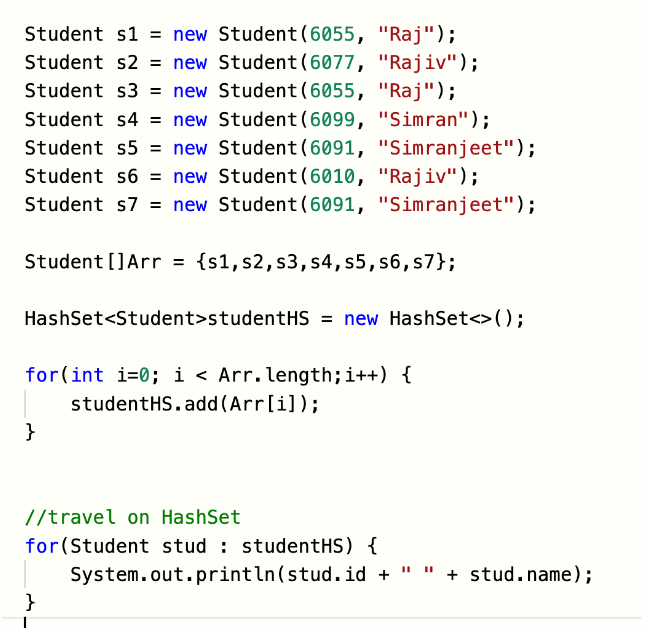
ans = hs.size()

Object as Key in Hashing

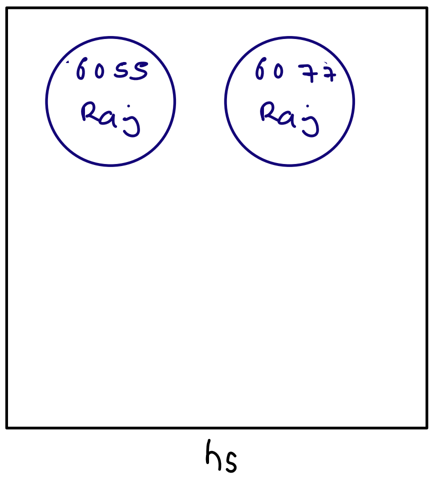
i) Every distinct key has a hashcode, which is used in implementation of hashmap | hashcode.

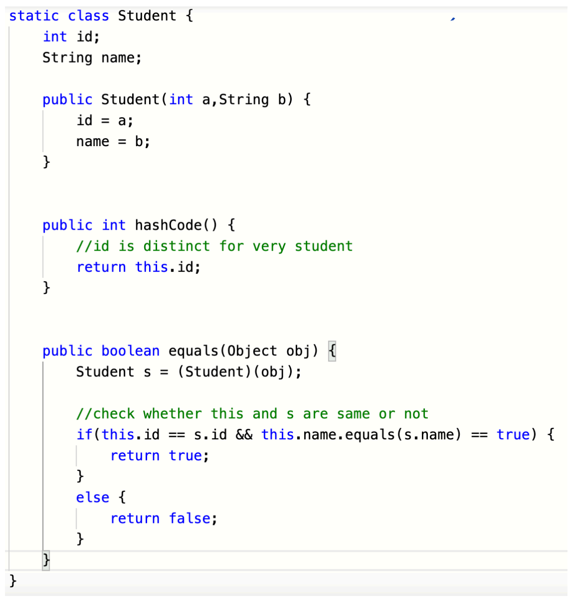
int





```
Student s1 = new Student(6055, "Raj");
Student s2 = new Student(6077, "Rajiv");
Student s3 = new Student(6055, "Raj");
Student s4 = new Student(6099, "Simran");
Student s5 = new Student(6091, "Simranjeet");
Student s6 = new Student(6010, "Rajiv");
Student s7 = new Student(6091, "Simranjeet");

Student[]Arr = {s1,s2,s3,s4,s5,s6,s7};

HashSet<Student>studentHS = new HashSet<>();

for(int i=0; i < Arr.length;i++) {
    studentHS.add(Arr[i]);
}


//travel on HashSet
for(Student stud : studentHS) {
    System.out.println(stud.id + " " + stud.name);
}
```





```
static class Student {
    int id;
    String name;

    public Student(int a,String b) {
        id = a;
        name = b;
    }


    public int hashCode() {
        //id is distinct for very student
        return this.id;
    }


    public boolean equals(Object obj) {
        Student s = (Student)(obj);

        //check whether this and s are same or not
        if(this.id == s.id && this.name.equals(s.name) == true) {
            return true;
        }
        else {
            return false;
        }
    }
}
```

Doubts

		0	1	2	3	4	5	6	7
A :		3	11	-4	1	-2	5	6	2
sum :	0	3	14	10	11	9	14	20	22

$\uparrow$

sum = 6

sum-K , sum = 20

sum - K = 20 - 6 = 14

HashMap → prefix sum vs first index

		0	1	2	3	4	5	6	7
A[ ]		2	4	7	1	-5	4	2	7
sum:	0	2	6	13	14	9			

K = 9

sum - K = 0

0 → -1	14 → 3
2 → 0	
6 → 1	
13 → 2	

s = ~~-1~~ map.get(sum-K)+1

e = ~~-1~~ i

```
int [] solve (int [] A, int k) {
    HashMap <Integer, Integer> map = new HashMap<>();
    map.put (0, -1);
    int sum = 0;
    int sp = -1, ep = -1;
    for (int i = 0; i < A.length; i++) {

        sum += A[i];

        if(map.containsKey (sum - k) == true) {
                sp = map.get (sum - k) + 1;
                ep = i;
                break;
        }
        // put your impact in map
        if (map.containsKey(sum) == false){
                map.put (sum, i);
        }
    }
    // create subarray from sp to ep and return that
    (loop required from sp to ep)
}
```