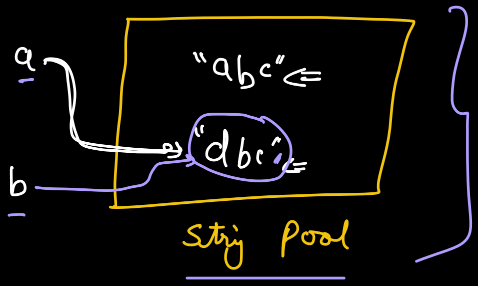
String a = "abc";

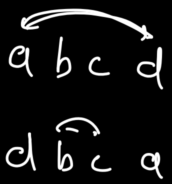
a[0] = 'd' ;

a = "dbc";

Strg b = "dbc";

Strg c = "abc";





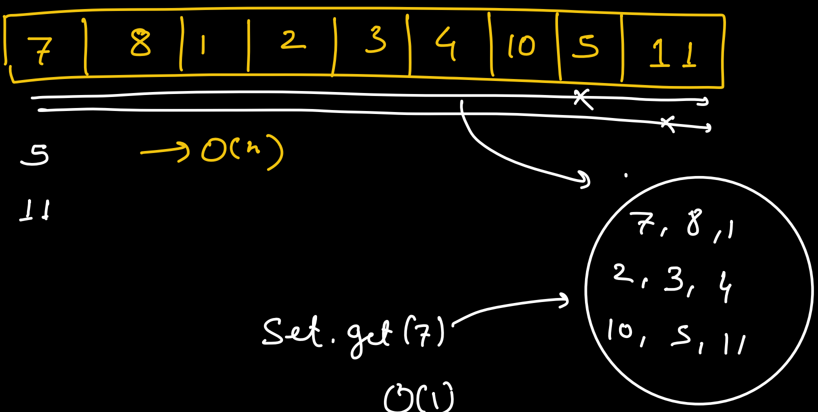
abcd
dbca
dcba

String Buildr sb = new Strg Buildr ( );

sb.charAt (i)

sb. setCharAt (i, 'c')

sb. toString ( ) ⟶ Strg



Set

Search / find => O(1)

[Hash Code

Self balancing BST) Advance

Java / C# — Hash Set

C++ — unordered_set

Python / Ruby / JS — set

C — X

Set
- → get / find (x)
- → add ~~(x)~~ / insert (x)
- → size ()

} O(1)

A : [1, 2, 1, 2, 1, 2, 3, 3, 4, 4]

1, 2
3, 4

Set only stores unique elements

Q Given an array. Count no. of distinct elements in the array.

A: [7, 3, 2, 1, 3, 7, 0] ⇒ 5

```
Set s = { }

for (i = 0; i < N; i++){
        Set.add( A[i]);
}

ret Set.size();
```

TC : O(N)

SC : O(N)
(Extra)

Q Given an array & Q queries.

In every query return the freq of an element.

A : 7, 2, 7, 1, 2, 8, 1, 0, 8

Key ⟷ Value

⇒ 7 ⟹ 2 ⇒ O(N)

1 ⟹ 2 ⇒ O(N)

0 ⟹ 1 ⇒ O(N)

7 ⟹ 2 ⋮

⋮

O(QN)

Q Given an array of strings.

Q queries having a string in each query.

Return sum of ascii values of char of string.

N → Size of array

M → Max possible size of string

Q>>N

⇒ A: abc, bcd, a, c, e

Key Value

⇒ bcd ⇒ 297 ⇒ O(M)

a ⇒ 97 ⇒ O(M)

e ⇒ 101 ⇒ O(M)

⇒ bcd ⇒ 297 ⇒ O(M)

abc ⇒ 294

⋮

bcd ⇒

⋮

bcd ⇒

O(QM)

Map

Java	C++	Python	Ruby/Js
HashMap	unordered_map	dict	map

C ⟶ C++

Map < Key , value >

↑ ↑

Q Store Countries - Capitals

Map < String , String >

Q Country - Population

Map < String , Long >]

↑

Q Country → all states of the country.

Map < String , Array [String] >

Q Given , Country name & a state name | Return Population of that state of that Country.

Map < String , Map < String , LONG > >

↑ Country Name ↑ State Name

Q Store marks scored by students in a subject.

Map < String , Double >

~~Name~~

↳ Roll No.

Keys in Hash Map are always unique

Map

- → get / find (K) //Returns value associated with K
- → add / Put (K, V) // add <K,V> to map
- → size ( ) // Returns size
- → update (K, V) // update value of K
- → isPresent (K) / ContainsKey (K) / KeyExists (K) // Check if K is present as Key
- → Remove (K) / delete (K) //Remove Key K along with its value.

O(1)

Amazon Q Given an array of size N.
Find the first non-repeating element in the array.

A : 8, 2, 8, 3, 1, 2, 6, 5

⇒ 3

Quiz A : 1, 2, 3, 1, 2, 5

A[i]	freq(A[i])
1	2
2	2
3	1
5	1

Non - Repeating ⇒ (freq = 1)

First

⇒ Build a hash Map to store the freq of every element.

⇒ Iterate over the array & return the first element with freq = 1.

```
HashMap < Int ,  Int >  map;
           ↑      ↑
          A[i]   freq(A[i])
```

A : 1, 2, 3, 1, 2, 5

A(i)	freq(A(i))
<1	2>
<2	2>
<3	1>
<5	1>

```
for (i=0; i<N; i++) {
   // if A[i] is present in map
      if (map.containsKey(A[i])) {
            map.put(a[i], map.get(A[i]) +1);
           // map[A[i]] ++;

   // if A[i] is not present in map
      else {
             map.put ( A[i], 1 );
   }

// Find first non-repeating element
  for(i=0; i<N; i++){
     if (map.get(A[i]) = = 1) {
               ret A[i];
     }
}
```

TC : O(N)
SC : O(N)
[Building the freq map]

TC: O(N)

TC : O(N)
SC : O(N)
(Extra)

Amazon
MS
Google

Q. Given an array of size N.
Return true if there exists a subarray with sum = 0.

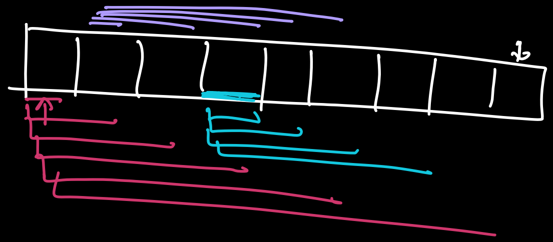
	0	1	2	3	4	5	6	7	8
A:	2,	2,	1,	-3,	4,	3,	1,	-2,	-3

True

No of subarrays in an array of size N

$N + (N-1) + (N-2) + \dots 1$

$\frac{N(N+1)}{2} \approx O(N^2)$



```
for (i=0; i<N; i++){
        for (j=i; j<N; j++){
            sum = 0;
            for (K=i; K<=j; K++){
                sum += a[K];
            }
            if (sum == 0)
                ret true;
        }
}
ret False.
```

sum = PS[j] - PS[i-1]

TC: $O(N^3)$

use PS

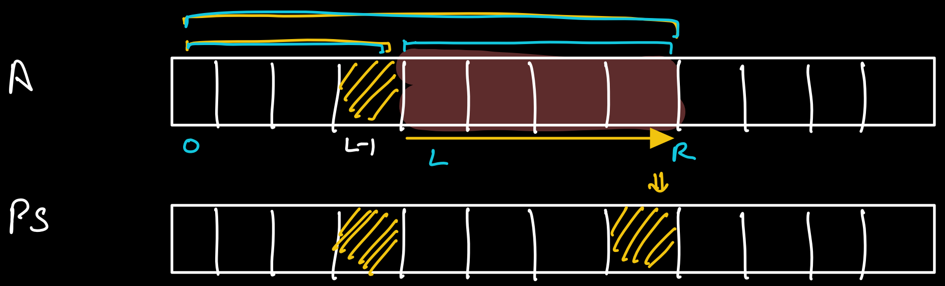
$O(N^2)$

$$\underline{\text{Sum}(L, R)} = \underline{PS[\overset{\checkmark}{R}]} - \underline{PS[L-1]}$$

if $\text{Sum}(L, R) = 0$ (Assume)

$$0 = PS[R] - PS[L-1]$$

$$\Rightarrow \boxed{PS[R] = PS[L-1]}$$

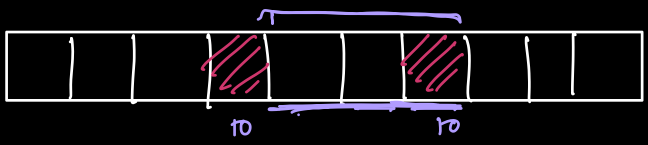


$$\text{Sum}(0-R) = \text{sum}(0, L-1) + \text{Sum}(L, R)$$

if $\text{sum}(L, R) = 0$

$$\text{Sum}(0-R) = \text{sum}(0, L-1)$$

$$\underline{PS[R]} = \underline{PS[L-1]}$$



	0	1	2	3	4	5	6	7	8
A :	2,	2,	1,	-3,	4,	3,	1,	-2,	-3
Ps :	2,	4,	5,	2,	6,	9,	10,	8,	5

Sum(3, 8) = Ps[8] - P[2]

If any two values repeat in PS array
ans is true

	0	1	2	3	4	5	6	7	8
A :	2,	2,	1,	-2,	4,	3,	1,	-2,	-3
Ps :	2,	4,	5,	2,	6,	9,	10,	8,	5

A : [0, 1, 2, 3]

Ps : 0, 1, 3, 6

2, 4
5

A : [3, -1, -2, 4]

Ps : 3, 2, 0, 4

A : [2, 1, 0, 8]

Ps : [2, 3, 3, 11]

$$\text{Sum}(L, R) \begin{cases} \underline{PS[R]} - \underline{PS[L-1]} & \text{if } \underline{L>0} \checkmark \\ \underline{PS[R]} & \text{if } L=0 \checkmark \end{cases}$$

$$\boxed{\begin{array}{l} TC : O(N) \\ SC : O(N) \end{array}}$$