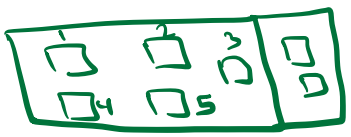
1. Hashmap Intro
2. Query: Frequency of element
3. First Non-Repeating Element
4. No. of distincts elements
5. Subarray with sum = 0

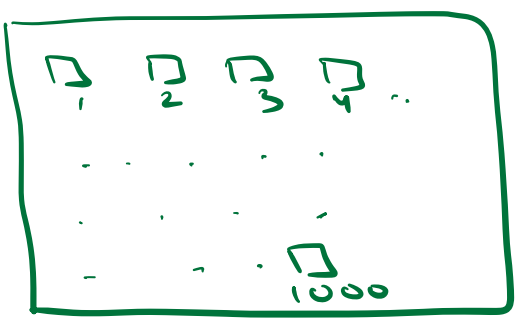
Hashing Intro

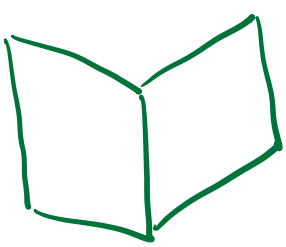
Ridhi & Joshua



Occupied → Register

	Room No	Occupied
Tushar →	1	~~No~~ Yes
	2	No
Amrit →	3	~~No~~ Yes
	4	No
	5	No





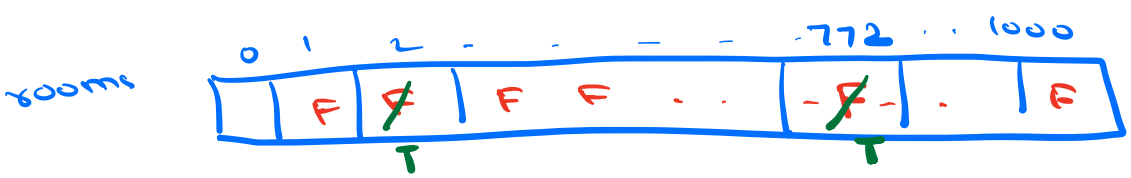
Room, Occupied

↓

1 - 1000

bool rooms[1001]

ind → 0 — 1000



Index represent room info

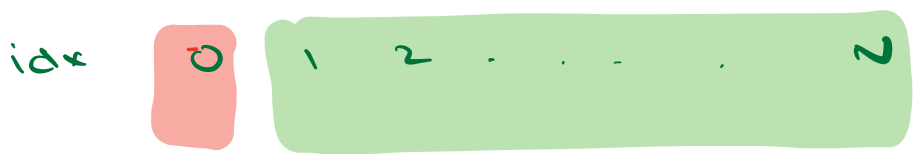
rooms [772]

Not occupied → F

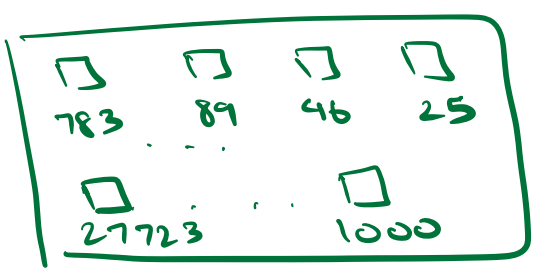
Occupied → T

Riddhi & Joshua 1 → N rooms

bool rooms [N+1]



Numerologist →



N = 1000 rooms

Random lucky nos. from [1 - 10^9]

bool rooms

bool rooms [1001] ✗

idx 0 1 2 3 1000

bool rooms [10^9 + 1]

idx 0 1 2 . . . . . 10^9

Status of room 72345

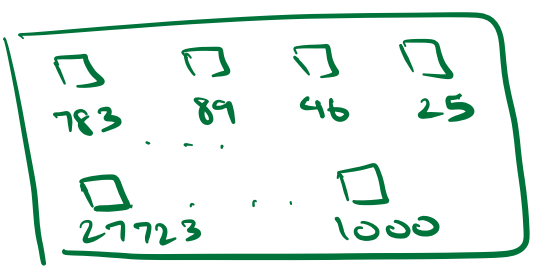
rooms [72345]

Valid idx → rooms = 1000

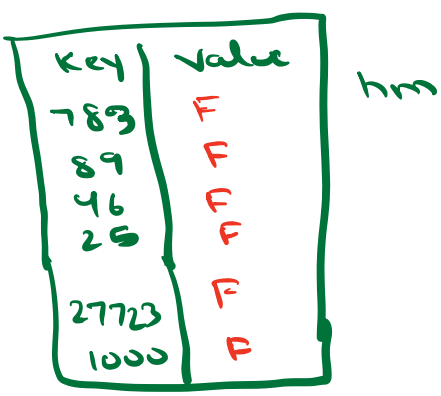
In an array of 10^9 size, access 1000 idx

Issue → space wastage

Sol : Hashmap → <key, value>



[1 - 10^9]



hm [46]

① Search in HM

↓

TC : O(1) SC : O(N)

② Key → unique

③ Value → can be anything

1) Store population of every country

population, country

Hashmap < key, value >

↓ country ↓ population

Hashmap < String, Long >

2) No. of states in each country

Info → No. of state, country

country, no. of state

Hashmap < String, Int >

3) For every country, we want to store all state names

Country, state name

Key value

Hashmap < String, List <String> >

↓

Dynamic List

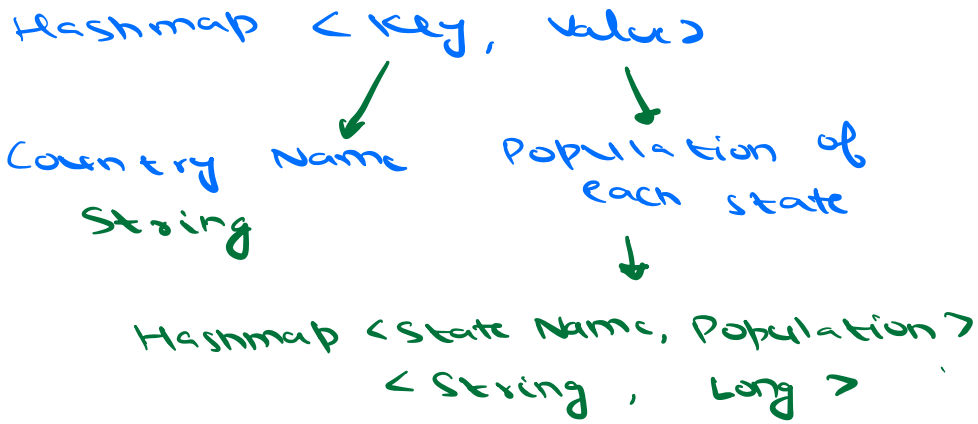
↓

Vector → C++

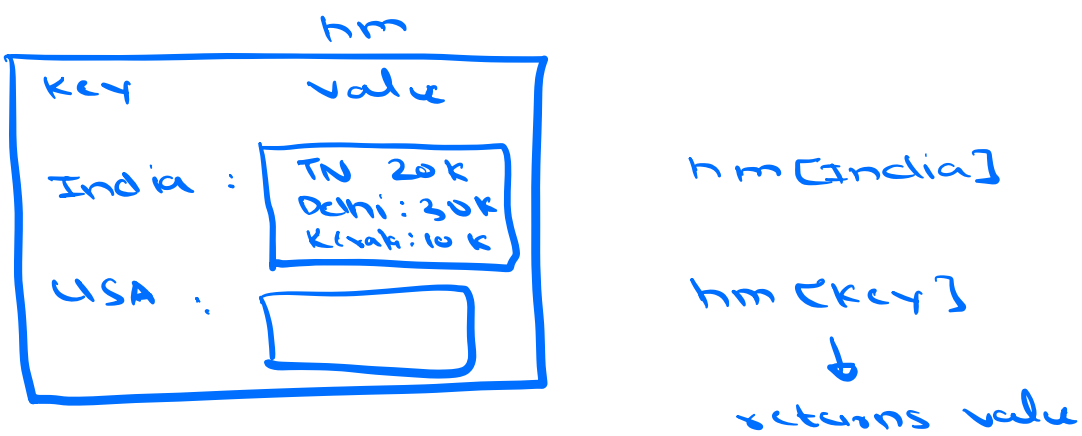
ArrayList → Java

Key	value
India	[UP, TN, Delhi]
USA	[LA, —]

4) For every country, store population of each state.



Hashmap <String, Hashmap <String, Long>>



hm[India][TN] → 20K

- Value can be anything
- Key → primitive datatype
  int / long / float / double / String

Hashset

Sometimes we want to store keys and we do not associate any values with them, we use Hashset

```
Hashset < key >
```

- Key must be unique
- Search in HS TC : O(1)

Array of strings

HashMap and Hashset Functionalities

TC of all operations : O(1)

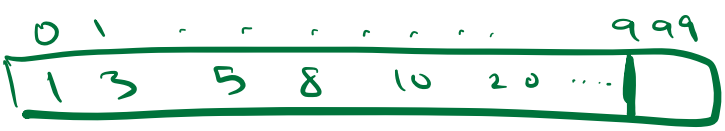
HashMap	Hashset
Insert (key, value) new key value pair is inserted. If key exists in HM, no change	Insert (key) inserts a new key. If key already there, no change
Size → returns no. of keys	Size → no. of keys
Delete (key) → delete key - value pair for given key	Delete (key) → delete key

search (key) → search for key

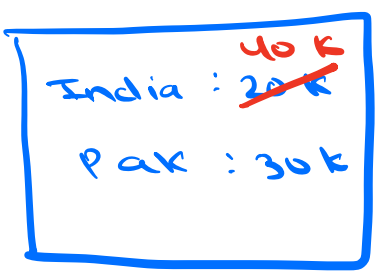
update (key, value) → previous value overridden with new val

search (key) → search for key

update ✗



HM



insert (India, 20 K)

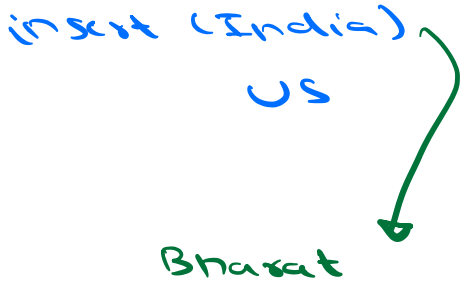
(Pak, 30 K)

Update (India, 40 K)

HS

insert (India)

US



Bharat

updation = delete old info,
insert new info

Hashing library name in diff. languages

	Java	C++	Python
Hashmap	Hashmap	unordered_map	dictionary
Hashset	Hashset	unordered_set	set

	JS	C#
Hashmap	map	dictionary
Hashset	set	Hashset

10:31

1. Find Frequency of numbers
Given N array elements & Q queries. For each query, find frequency of given element in that query.

N = 11 arr[11] = <2, 6, 3, 8, 2, 8, 2, 3, 8, 10, 6>

Q: 4
2 : 3
8 : 3
3 : 2
5 : 0

Idea 1 : For each query, iterate and find frequency of element.
TC: O(QN) SC: O(1)

Elem, freq

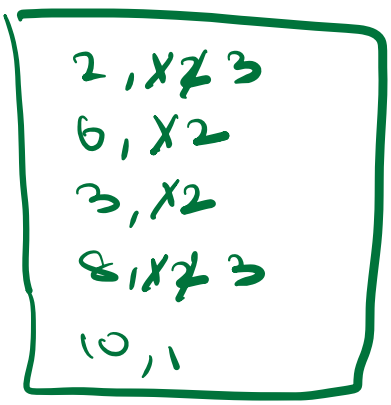
Idea 2: Iterate array once, store info (elem, freq) in hashmap

N = 11 arr[11] = <2, 6, 3, 8, 2, 8, 2, 3, 8, 10, 6>
(0 1 2 3 4 5 6 7 8 9)

Hashmap <key, value>
↓ ↓
elem freq

Hashmap <int, int> hm

2, 3
6, 2
3, 2
8, 3
10, 1



① Insert (elem, 1)

② If elem is already in HM

hm[elem]++

Q: 4
2: 3
8: 3
3: 2
5: 0

For any query, search for key (elem

YES ↓ NO

return hm[key]

return 0

```
void frequency(A[], Q[]) {

    Hashmap<int, int> hm
    q = Q.length
    n = A.length

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

        if (hm.search(A[i]) == true) {
            hm[A[i]]++
        }
        else {
            hm.insert(A[i], 1)
        }
    }

TC: O(N+Q)
SC: O(N)

    for (i=0 ; i<q ; i++) {          → Q
            // Q[i]

        if (hm.search(Q[i]) == true)
                print(hm[Q[i]])

        else
                print(0)
    }

}
```

2. Find the first non-repeating element. → from starting

Ex 1 ar[6] = < 1 2 3 1 2 5 >

ans = 3

Ex 2 ar[8] = < 4 3 3 2 5 6 4 5 > ans = 2

Ex 3 ar[7] = < 2 6 8 4 7 2 9 > ans = 6

first unique elem from start

Idea 1 : Use hashmap to store freq of each elem

key, value
↓ ↓
elem freq

ar[6] = < 1 2 3 1 2 5 >

HM

1, X 2
2, X 2
3, 1
5, 1

Insertion order

1
2
3
5

Print HM

3, —
5, —
2, —
1, —

HM does not maintain insertion order

↓

can't iterate HM and pick 1st key/elem with freq 1

Idea 2 : Store info in HM
Iterate over the array

↓

for each element

↓

check freq

↓ >1 → i++

↓ =1 → ans

arr[6] = < 1 2 3 1 2 5 >

HM

1,	2
2,	2
3,	1
5,	1

```
void firstNonRepeating(A[]) {
    Hashmap<int, int> hm
    int n = A.length
    for (i=0 ; i<n ; i++) {
        if (hm.search(A[i]) == true) {
            hm[A[i]]++
        }
        else {
            hm.insert(A[i], 1)
        }
    }
    for (i=0 ; i<n ; i++) {
        if (hm[A[i]] == 1) {
            print(A[i]) break
        }
    }
}
```

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

```
    if (i == n)
        print(-1)
}
```

3. Given N array elements, find no. of distinct elements.

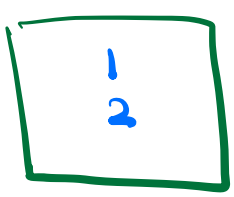
Ex 1 ar[5] : <3, 5, 6, 5, 4> ans = 4
(+1 +1 +1 +1)

ar[3] = <3, 3, 3> ans = 1
(+1)

ar[5] = <1, 2, 1, 2, 2> ans = 2
(+1 +1)

Idea → Insert element in Hashset
ans = hashset.size()

ar[5] = <1, 2, 1, 2, 2>



Note : In hashset, if a single key is inserted multiple times, its occurrence remains once.

```
void countDistinct(A[]) {
    Hashset<int> hs
    for (i=0 ; i<n ; i++) {
        hs.insert(A[i])
    }
    print(hs.size())
}
```

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

5. Given N array elements, check (bool) if there exists a subarray with sum = 0.

	0	1	2	3	4	5	6	7	8	9
arr[]:	2	2	1	-3	4	3	1	-2	-3	2

ans = True

	0	1	2
arr[] :	1	2	3

ans = False

Idea 1 : Check every subarray's sum

TC: $O(N^3)$ 2 loops → go to every subarray
3rd loop → iterate

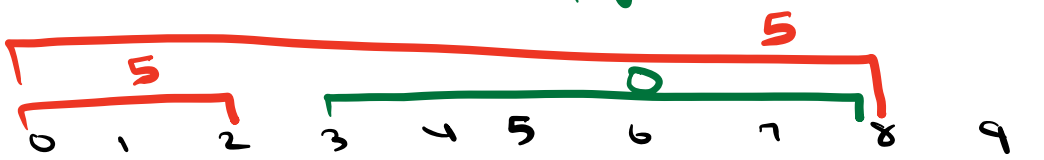
TC: $O(N^2)$ pfsum bf[] + subarray (s→e)
SC: O(N)
bf: N^2, subarray: N^2

TC: $O(N^2)$ Carry forward
SC: O(1)

Optimization → subarray with sum = 0

↓

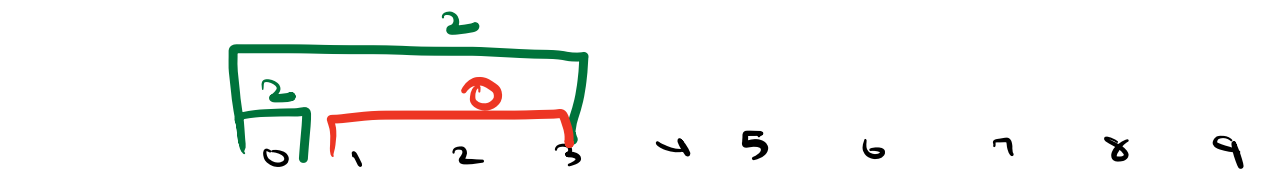
pf[ ]



Obs : Duplicates in pf[ ]

↓

subarray with sum = 0 ✓



① sum(s − e) → 0

↓

pf[e] − pf[s−1] = 0

⇒ pf[e] = pf[s−1]

② sum(0 − e) → 0

↓

pf[e] = 0

ar	1	-1	2
bf	1	0	2

① Look for duplicates in bf[]
② If any bf[ ] = 0

```
// create bf[N]                                   TC: O(N)
hashset <int> hs                                  SC: O(N)
for (i = 0; i < n; i++) {
    if (bf[i] == 0)
        return true
    hs.insert(bf[i])
}
```

sum (0 → i)

if (hs.size() < n) ← array size
    return true
else
    return false

	0	1	2	3	4	5	6	7	8	9
ar[ ] :	2	2	1	-3	4	3	1	-2	-3	2
bf[ ] :	2	4	5	2	6	9	10	8	5	7
	—	—	—	X	—	—	—	—	X	—

ar ↓ 10 bf ↓ 10

HS: 2, 4, 5, 6, 9, 10, 8, 7 → 8

Hint :

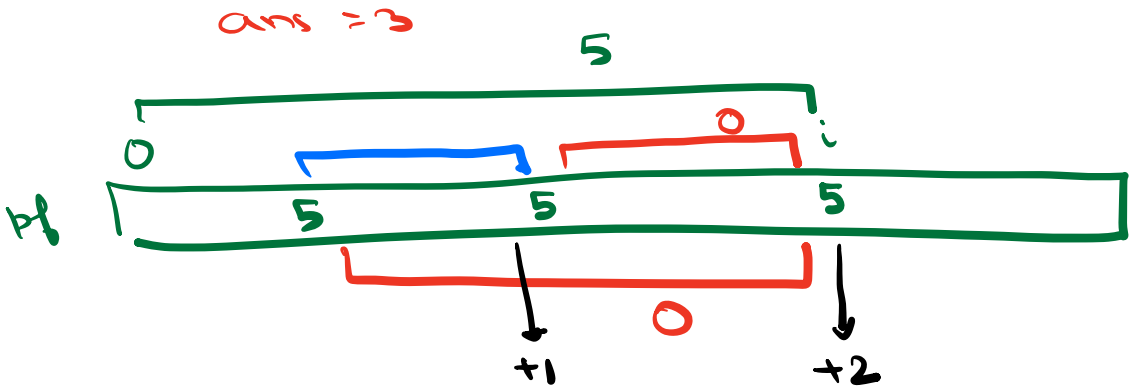
Count subarrays with sum = 0

$A = [\underset{0}{1}, \underset{1}{-1}, \underset{2}{-2}, \underset{3}{2}]$

$0 \to 1$

$2 \to 3$

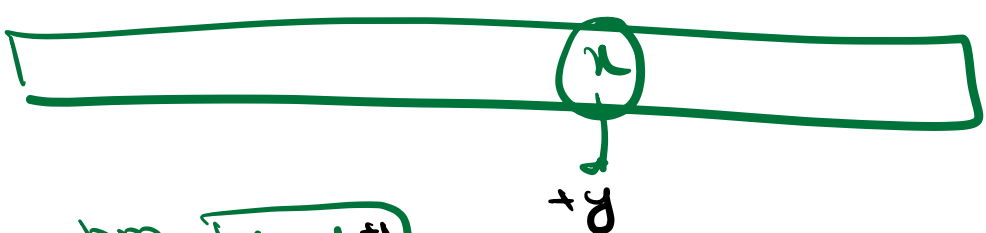
$0 \to 3$



How many subars have sum = 0 ending at ith idx ?

For any pf[ ], cnt of duplicates before it

HM < pref, freq >



hm [x, y+1]

0

ar	1	−1	2	−2
bf	1	0	2	0
cnt		+1		+1

---

unordered → HM

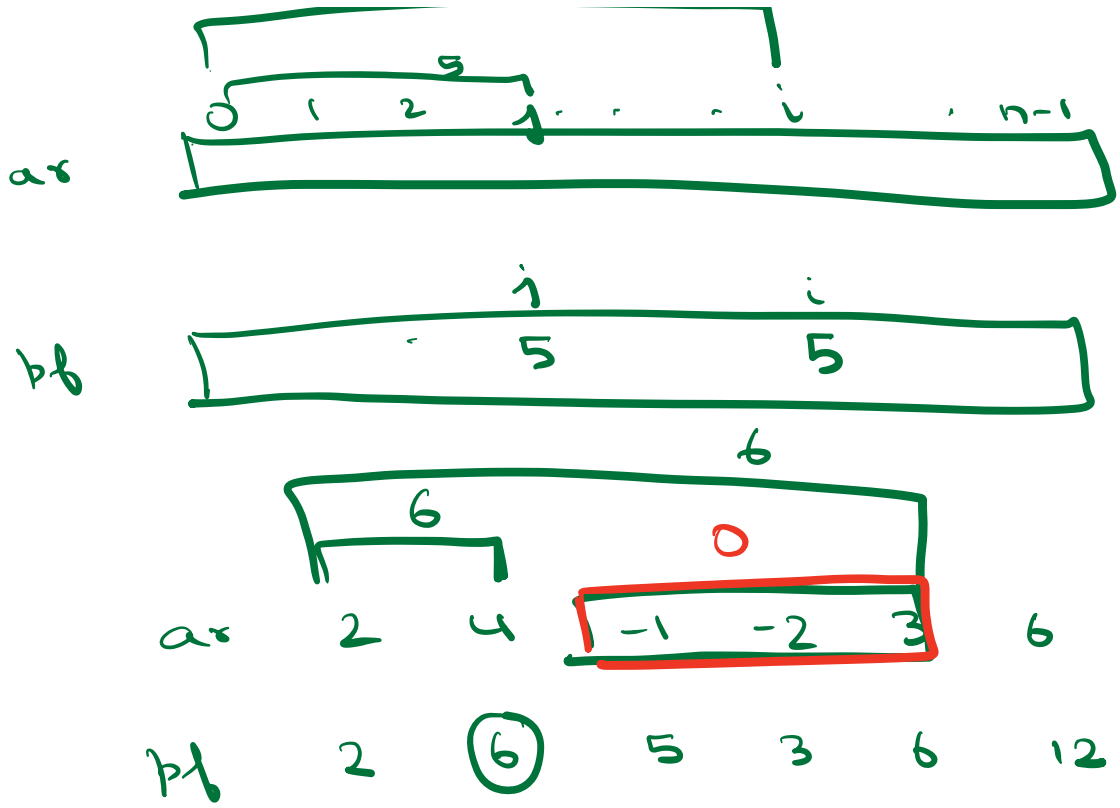
ordered → [ ]

① Any subarray (Idn)

② All subarray

Cntry → state→pop

5


arr
0 1 2 3 i n-1
bf
j i
5 5
6
6
0
arr 2 4 -1 -2 3 6
bf 2 6 5 3 6 12