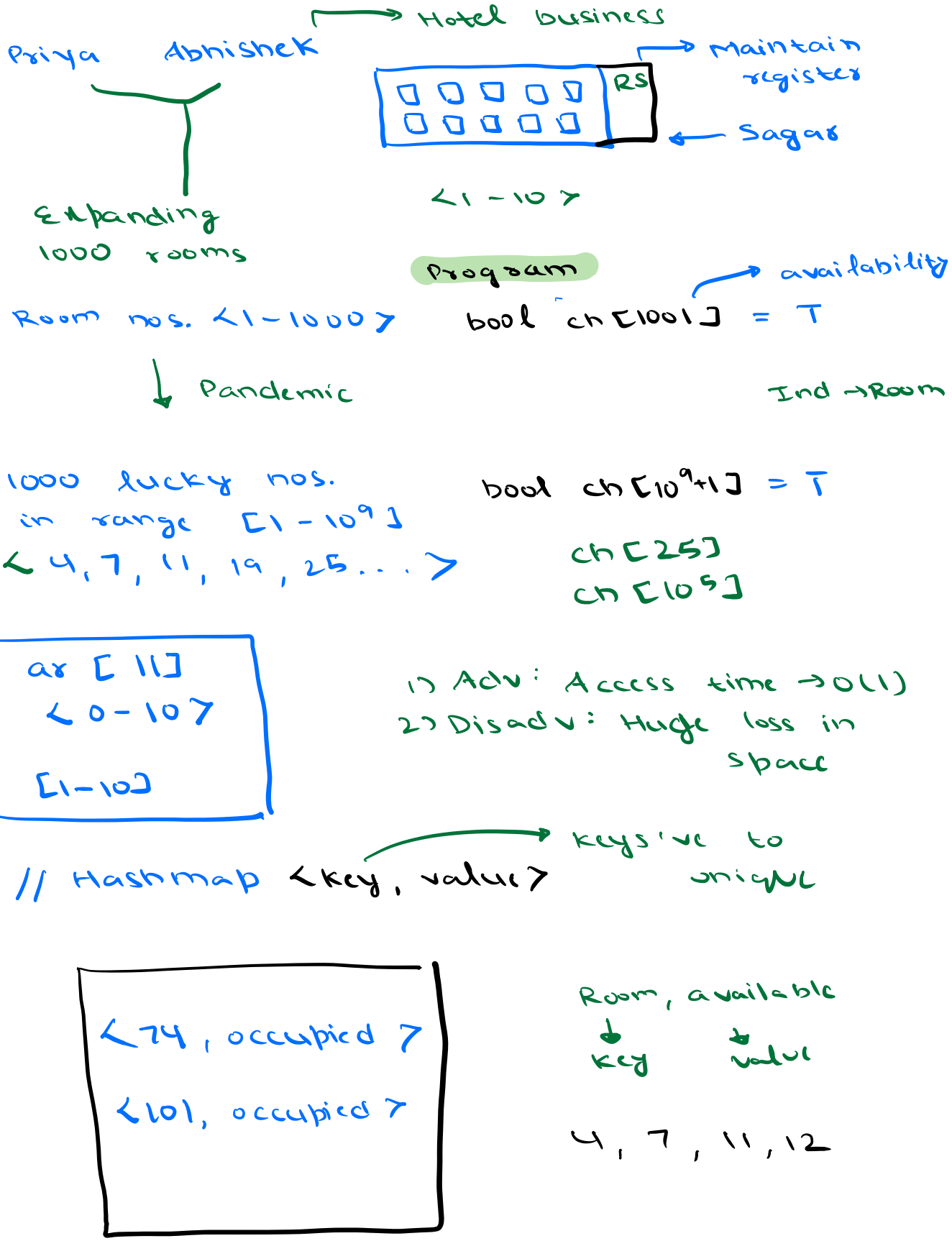

Hotel business
Priya
Abhishek
Expanding
1000 rooms
RS
Maintain register
Sagar
<1 - 10>
Program
availability
Room nos. <1-1000>
bool ch[1001] = T
Pandemic
Ind → Room
1000 lucky nos.
in range [1 - 10^9]
<4, 7, 11, 19, 25 . . . >
bool ch[10^9+1] = T
ch[25]
ch[10^5]
ar[11]
<0-10>
[1-10]
1) Adv: Access time → O(1)
2) Disadv: Huge loss in space
// Hashmap <key, value>
keys've to unique
<74, occupied>
<101, occupied>
Room, available
key
value
4, 7, 11, 12

// If we want to store only keys →

Hashset <key>

↳ key → unique

1) Store population of every country

country & population

unique

Hashmap <key, value>

pop, country

country → key

population → value

String

Long

Hashmap <String, Long>

2) No. of states in each country

Hashmap <key, value>

country → key

no. of state → value

Hashmap <String, Int>

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

Hashmap <key, value>

country → key

all states → value

Hashmap <String, List <String>>

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

Hashmap <key, value>

key → Country (String)

value → Population of each state (Hashmap <State, pop> / String, Long)

Hashmap < String, Hashmap < String, Long >>

// value - can be anything

// key - <String, int, long, float, ... >

Any primitive data type

→ a list can't be kept as a key

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

JS	C#
map	
set	

Hashmap <K, V>

Hashset <K>

Arr [ ]

Ind → room

Hashmap <key, value>

Worst case → O(n)

insert <key, value>
search <key>
remove <key>
update <key, new value>

size() → no. of keys in hashmap

All operations → O(1) (avg) → for a single operation

Even if you're searching a key in 10^4 K/V pairs → O(1)

If we insert N key/value pairs → SC : O(N)

Hashset <key>

insert (key)
search (key)
remove (key)

update (key) → remove (key) → insert (new key)

<india, china, pak>

For a single operation → O(1)

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>

Query = 4

2 : 3
8 : 3
3 : 2
5 : 0

Hashmap <K, V>

array elements — freq

int — int

Hashmap <int, int> hm

Idea: For every query, iterate & get count

TC: O(QN)
SC: O(1)

Hm

<2, X 2 >
<6, 1>
<3, X 2 >
<8, X X 3 >
<10, 1>

new entry

hm[2]

if key present ? hm[key] : 0

// Pseudocode

```
Hashmap <int, int>  hm;

for (i=0 ; i<N ; i++ ){                    // N
    if (arr[i] is in hm){                  // search arr[i] in hm
        // update freq by +1
        hm[arr[i]] ++
    }
    else {
        hm.insert ( <arr[i], 1>)
            // insert <arr[i], 1>  in hm
    }
}

for (i= 0 ; i < Q ; i++) {                 // Q
    // x
    if ( x is in hm) {
        print (hm[x])
    }
    else { print (0) }
}
```

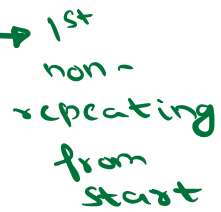
Insert array elems in hm

TC: O(Q+N)

SC: O(N)

2. Find the first non-repeating element.

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



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

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

Idea:

1. Insert all elements in hm
2. Iterate on hm and get 1st key with freq = 1

(Note - in hashmap, insertion order is not maintained; when we print hashmap we'll get any order)

Idea:

1. Insert all elements in hm → O(N)
2. Iterate on array and get elem with freq = 1 → O(N)

TC: O(N+N) = O(N)

SC: O(N)

```
for (key in hm.getkeys())
    print(key, hm[key])
```

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

Ex 1 ar[5] : <3, 5, 6, 5, 4>

+1 +1 +1 +1 → (4)

Ex 2 ar[7] : <6, 3, 7, 3, 8, 6, 9>

+1 +1 +1 +1 +1 → (5)

Hashset < key > (key → element, int)

Hashset <int> hs;

hs

6
3
7
8
9

hs.size =5

Note - Even if we insert same key twice in hashset, it will only store key once / unique occurrence.

```
for (i = 0; i < N; i++) {
    hs.insert(ar[i])
}
print(hs.size())
```

TC : O(N)

SC : O(N)

4. Given N array elements, check (bool) if all elements are distinct or not.

arr[7] = < 1, 2, 3, 4, 5, 6, 7 >

1. Insert all elements in hashset hs
2. If (hs.size == N) <
   // all distinct
   return true
   >
   else < // all not distinct
   return false
   >

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
pf[]	2	4	5	2	6	9	10	8	5	7

l — r

Idea 1: go to every subarray, find sum, compare with 0

TC: O(N^3)

Sum[l r] = pf[r] - pf[l-1] = 0

⇒ pf[r] = pf[l-1]

Idea 2 TC: O(N^2)

Idea 3 : If we've a repetition in pf[], then there is subarray with sum = 0

1. Construct pf[] → n
2. hashset<int> hs
   Insert all pf[] elements in hs → n

```
if (hs.size() < N || 0 in hs)
    return true
else
    return false
```

TC: O(n)

SC: O(n)

arr[]	2	0	3
pf[]	2	2	5

✓

hs → 2

arr[] = -1 4 -3 2

pf[] = -1 3 0 2

(code fails)

if pf is 0, subarray sum = 0

a	x[]	0	2	3	3
1	pf[]	0	2	5	8

(code fails)

Doubts

	0	1	2	3	4	5	6	7	8	9
arr[]:	2	2	1	-3	4	3	1	-2	-3	2
pf[]	2	4	5	2	6	9	10	8	5	7

sum: 2 4 5 2

```
pf[0] = arr[0]
for (i = 1; i < N; i++) {
    pf[i] = pf[i-1] + arr[i]
}
```

OR

```
sum = 0
for (i = 0; i < N; i++) {
    sum = sum + arr[i]
}
```

pf sum[] space saved

* Get marks of student

HashMap < Int / Long, HashMap <String, int >>

Adm No. → Subject, Marks

* We can't keep student name as key bcoz it is not unique

* get apurva's marks hm[80]['history'] → adm no. (Apurva)