07/02/22

DSML Intermediate – DSA

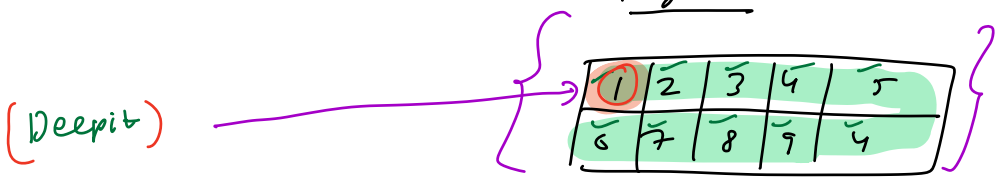
Hashing-1

Example

(10 rooms)

(Dhiman + Mohit) Hotel

Register



① Check avail

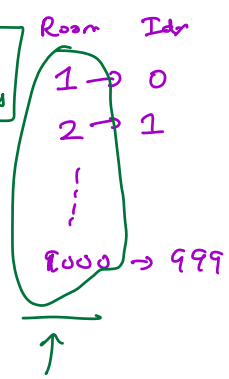
② Update avail

Tycoons

(1000 rooms)

How to store data for 1000 rooms?

available = Array with 1000 True values



① Check i room avl or not

available [i-1]

O(1)

② Update available [i-1] = False

O(1)

---

availability = [[T, 2], [F, 1], _ _ _ _ _]

availability[i-1][0] → T/F

availability[i-1][1] → #

avl — #people which can be accommodated

Dhiman + Mohit Losses

↳ Numerologist

1000 lucky room no.s $[1 - 10^8]$ $N = 10^8$

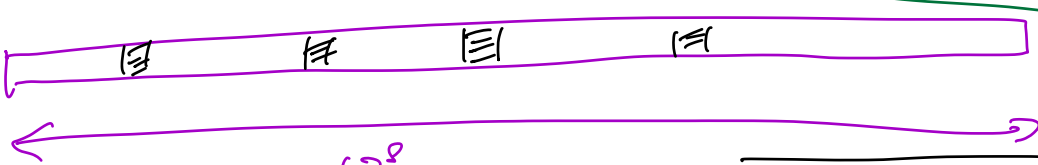
$\Rightarrow$ [2, 5, 10^8, 17, 9999, 22234, ---]

$\Rightarrow$ available = 10^8 size array size

avl[i-1]

$10^3 \rightarrow 10^8$ Disadv

$\Rightarrow$ $10^8 - 10^3$ Lot of space wastage.



10^8

Adv → It provides you O(1) lookup.

O(1) update

✓ 10^9 bytes $\simeq$ 1 GB

10^8 bytes $\simeq \frac{1}{10}$ = 0.1GB = 100 MB

1 byte = 8 bits

1 KB $\tilde{=}$ 10^3 bytes

1 MB $\tilde{=}$ 10^3 KB

1 GB $\simeq$ 10^3 MB

= 10^6 KB

= 10^9 bytes

✓ HashMap in general

✓ Dictionary in Python

(Inbuilt) implementation of HM

- No memory wastage
- Memory used = data stored
- O(1) operations.

1. Q) What is HM ? How it stores data? ✓

   Q) How it is working under the hood? How implemented? (internally)

2. Q) Time & Space complexity of operations

   Q) What are collisions etc.? What is chaining? →

3. Q) What is Hashset? What is Hash Table?

4. Q) Diff blw HashMap & HashSet ?

   A) Diff blw HashMap & Dictionary? → impl of HashMap ✓

5. Q) Is it ordered or not ordered ?

   Q) What is Hashing? → Technique

   Q) Is it mutable - dictionary? ✓

6. Q) When to use HashMap? When Hashset?

Impl of Hashing.

LinkedList

Binary search Trees.

Adv Sessions.

(1) (Q) What is HM ? How it stores data ?

Key, Value.

Key, Value pairs.

HashMap<Key, Value> availability → avl

Python specific syntax

look it up.

avl[30] = [True, 2]

avl[51] = [False, 5]

avl[10110] = [False, 3]

} Updatery

res = avl[15]

Lookup on the Key → associated value.

⇒ Distinct Keys

What can be a key or value?

Keys can be complex (LL)

LL nodes

Eg 1. Store population of every country.

HashMap<K,V> pop

↳ pop["India"]

Key → countryName (string)

val → population (large number)

Eg 2. No. of states in each country.

key → countryName (str)

num → number

Eg. 3. For every country store name of all states. = names

names["India"]

↳ val

key → countryName (str)

value → list of names (list of str)

Eg. 4. For every country, population of each state.

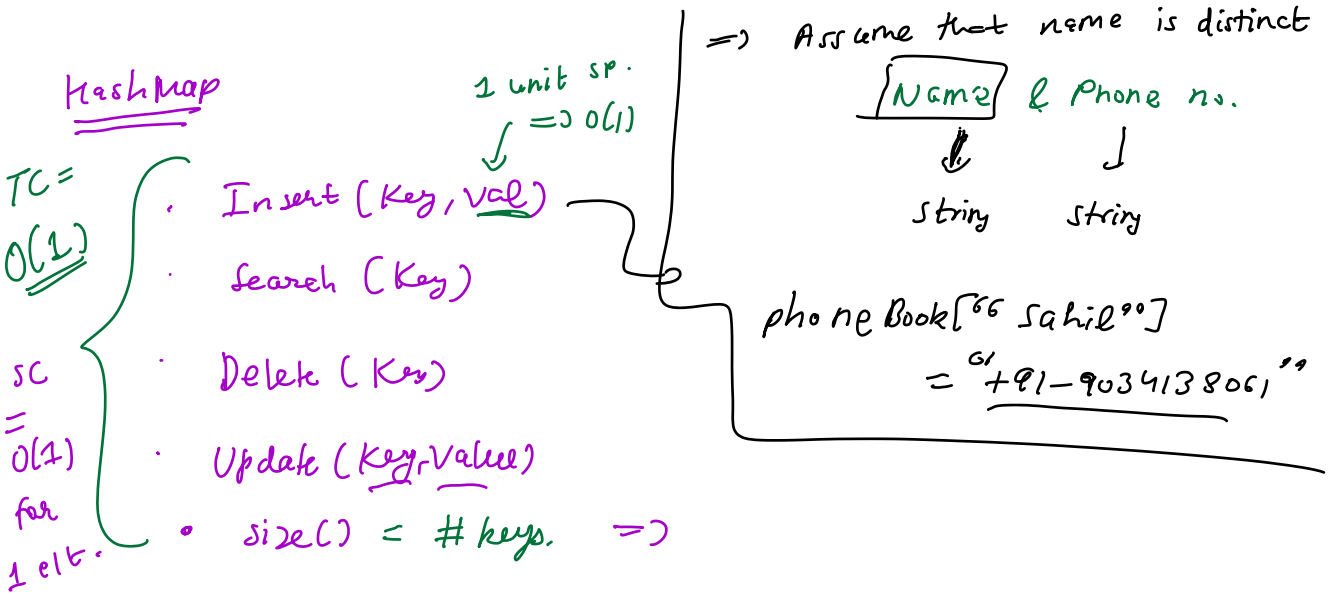
key = countryName

val → HashMap. (key = stateName, val = population.)

Nested dictionary

pop["India"]["Gujarat"] = 1M

(2) (Q) Time & Space complexity of operations



numHarpreet = phonebook["Harpreet"]

phonebook["Harpreet"] = "__" ✓

For N elements → TC: O(N)

SC: O(N) only for insert

SC: N search → O(1).

3. (a) What is HashSet? ~~What is HashTable?~~

4. (a) Diff b/w HashMap & HashSet? ✓

5. (a) Is it ordered or not ordered?

---

normal set

HashSet → only Key.

→ email Ids

~~items~~

→ room No. [1, 2, 100, 3, 4, 11$, 10$]

vs

HashMap

TC = O(1)

- ✓ Insert (Key, ~~value~~)
- ✓ Search (Key)
- ✓ Delete (Key)
- ~~Update (key, value)~~
- ✓ size() = # keys.

SC = O(1) for 1 elt.

Normally HashMap & HashSet are not ordered

In general → NOT sorted in any way.

HashMap <roomNo, available> ave;

ave[123] = True

ave[234] = True

ave[1191] = False.

⇝ print — ave

Dict

Try

(Try:)

roomNums = avl.keys()

↑

List

- keys()
- values()
- items()

---

⑥ (Q) When to use HashMap? When Hashtet?

=) Problems.

(Q1) Given N array elements,

Q queries. x ⇒ return the freq. of the elt. x in the array.

Brute →

O(N) time for each query.

x = 8

[2, 6, 3, 8, 2, 8, 2, 3, 8] →

Q → O(N·Q)

T = O(NQ)

SC = O(1)

HashMap<Key, Val> Freq.

↑ num ↑ count

✓ (1) Build Hashmap.

✓ (2) Answer the queries. (key)

present ← → not present.

$1 \le N \le 10^5$ (1)

freq = {} # HashMap.

[2, 6, 3, 8, 2, 8, 2, 3, 8] →

for num in nums:

search O(1) → if num not in freq:

Insert O(1) → freq[num] = 1

else:

Update O(1) → freq[num] += 1

3

2 : ~~1~~ ~~2~~ 3
6 : 1
3 : ~~1~~ ~~2~~
8 : ~~1~~ ~~2~~ 3

3 + 3 + 3 ... + 3 = 3N = O(N)

Preprocessing.

② Answer queries.

Q = [1, 2, 3, 7, 3, 4, 8]

ans = [0, 3, 2, 0, 2, 0, 3]

```
ans = []
for i in range(0, Q):        → 8
    if query[i] in freq:
        ans.append(freq[query[i]])
    else:
        ans.append(0)
```

freq[B]

Search } → O(1)
Access } O(1) →

$2 + 2 + \cdots + Q$ times $= 2 \cdot Q = O(Q)$

Total TC = $O(N+Q)$

Total SC = $O(N)$

Worst case = N keys + N values = 2N ⇒ O(N) space

Q2. Given N array elements, find the no. of distinct elements.

A1 $A = [6, 3, 7, 3, 8, 6, 9]$

freq-

6	: 2
3	: 2
7	: 1
8	: 1
9	: 1

No use to us ⇒ Simply we can use a Hashset

no. of distinct elts = no. of keys

ans = size(freq)

Tc. O(N), Sc. O(N)

set — Hashset = set in Python.

A-2

Using Hashset

```
num_distinct = set()  # empty set.
for num in nums:
    num_distinct.add(num)
return size(num_distinct)
```

Insert

size → syntax.

Python ↳ attach ref?

$A = [6, 3, 7, 3, 8, 6, 9]$

8, 6, 3, 7, 9 ⇒

Tc: O(N) Sc: O(N)

Q3. Given N array elements, find the first non-repeating element in the array.

[1, 2, 3, 1, 2, 5]

Find 1st elt that has a freq. of 1.

freq.

1	2
2	2
3	1
5	1

HashMap in general doesn't maintain order!

① Build the freq hashmap → O(N)

② Iterate over the array & find the first elt with freq. 1 → O(N)

if freq[x] == 1: → O(1)

nums = [4, 8, 8, 3, 4, 2, 3, 9]

4	2
8	2
3	2
2	1
9	1

② for num in nums:
    if freq[num] == 1:
        return num

Q4. Given N array elements, check if there exists a subarray with sum = 0. [True/False]

O(N) time

O(N) space

0 1 2 3 4 5 6 7 8

A: [2, 7, 1, -3, 4, 3, 1, -2, -3]

1 -3

3 -8

ans = True.

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

Brute force → $N^2 * N$

O(1) S O(N3) T

Carry Forward → O(N^2) T

O(1) S

0 1 2 3 4 5 6 7 8

A: [2, 7, 1, -3, 4, 3, 1, -2, -3] Observations??

ps: [2, 4, 5, 2, 6, 9, 10, 8, 5]

sum(i,j) = ps[j] - ps[i-1] if i != 0

ps[j] if i == 0

= 0

① i != 0 ps[j] - ps[i-1] = 0

⇒ ps[j] = ps[i-1]

⇒ If 2 elts. in ps (prefix sum) are same, we can say that there is a subarray with sum = 0.

Check the freq. of each elt. in HashMap.

freq[elt] > 1

0 1 2 3 4 5 6 7 8

A: [2, 7, 1, -3, 4, 3, 1, -2, -3]

PS: [2, 4, 5, 2, 6, 9, 10, 8, 5] N = 9

Obs1 If elts are repeating in PS, we have subarray with sum = zero.

free

2 : ✗ 2
4 : 1
5 : ✗ 2
6 : 1
9 : 1
10 : 1
8 : 1

keys. = 7

Obs2 If no. of elts in Hashset < N ⇐ True
⇒ There has to be a duplicate.

① PS: [1, 2, 3, 4]

1
2
3
4

Hashset sz = 4 = N

② PS: [1, 2, 1, 3]

1, 2, 3

sz = 3 < N = 4

Edge Case

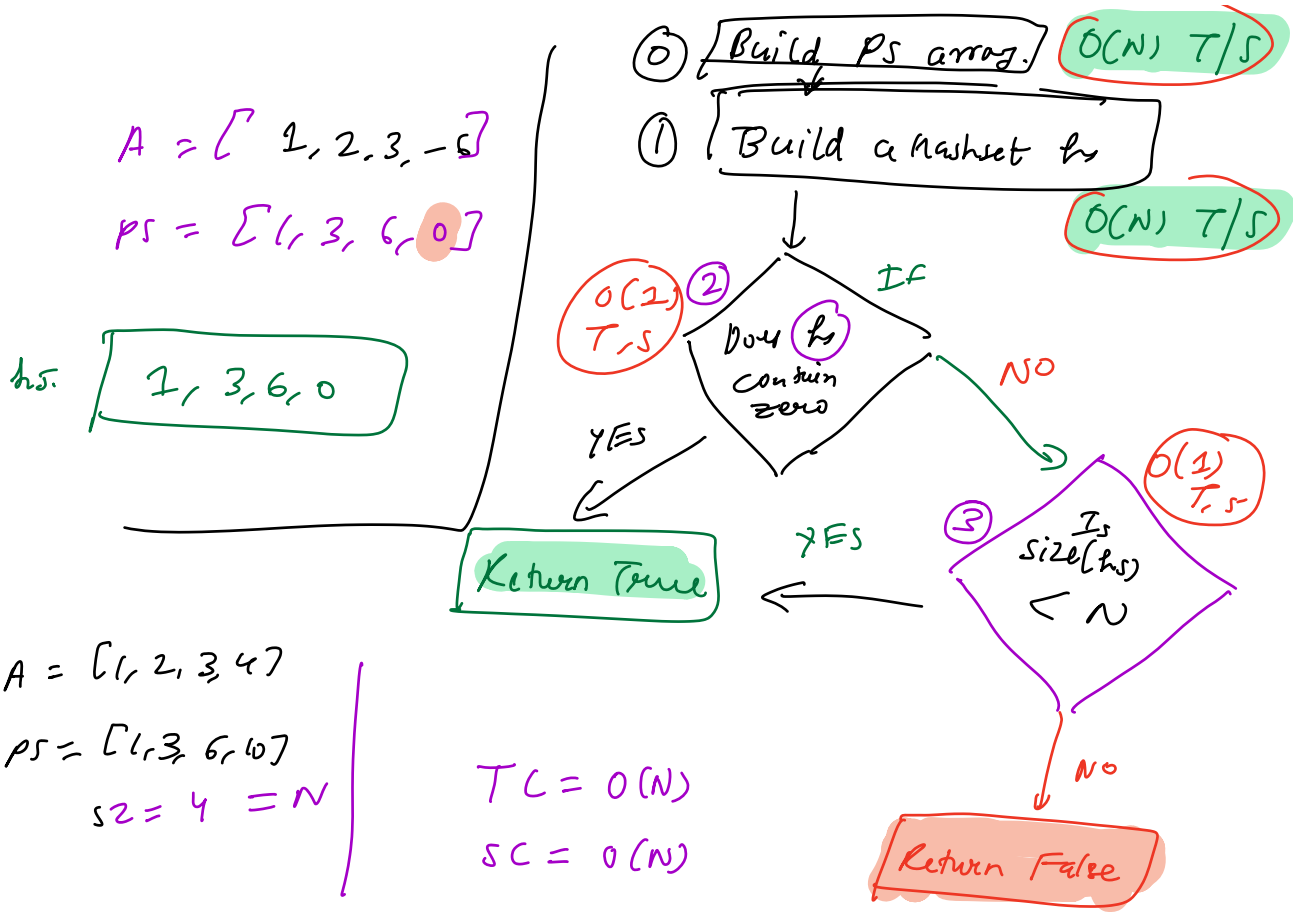
A = [1, -1, 2] (N)

PS = [1, 0, 2]

PS value itself zero but no repititions.

Hashset

hs {1, 0, 2} sz = 3 = (N)



A = [1, 2, 3, 4]

PS = [1, 3, 6, 10]

s2 = 4 = N

TC = O(N)

SC = O(N)

HW: Try to do it in one loop.

Building PS + hash set currently takes 2 loop.

Carry Forward running_sum

Doubts

N.

1, 2, 3, 4, 5, 6.

$N + 1 = O(N)$

HM

key | val:

0
1
2
3
4
5
1
1
10

K > N

```
def func(li, hm):
    [ ]
    return -
```

Aux. space = O(1)

---

a = { }

1) key → val

2) lookup quick
   ↳ Hashmap

O(1) avg.

W.C. = O(N)
collisions.

→ nums ⇄ access (with idx is known)

→ lookup not needed
then use a list

only store

sorted

O(log N)
Binary Search.

index not known

a = set()

Check for elt.

present | absent

O(1) TC

name

Array of LL

⇒ Array of BST

Handle collisions

↳ Chaining.

HashMap < Key, List of room No?

num

Any further questions

↓

Please ping on slack.

Respond

ETA 24 hrs.

Fri → Doubt/ Remedial

OOPs → Created