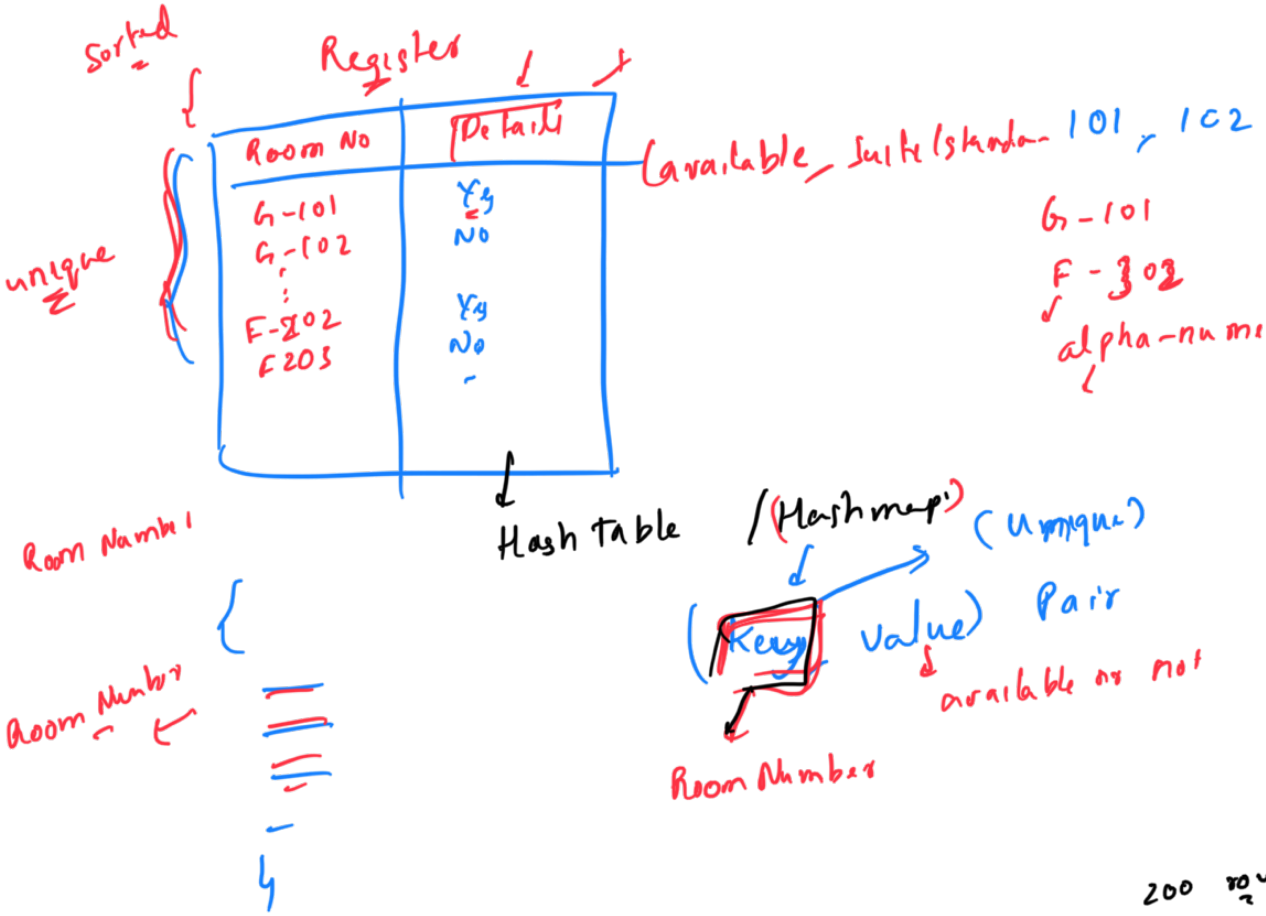
INTRODUCTION TO HASHING



200 rows

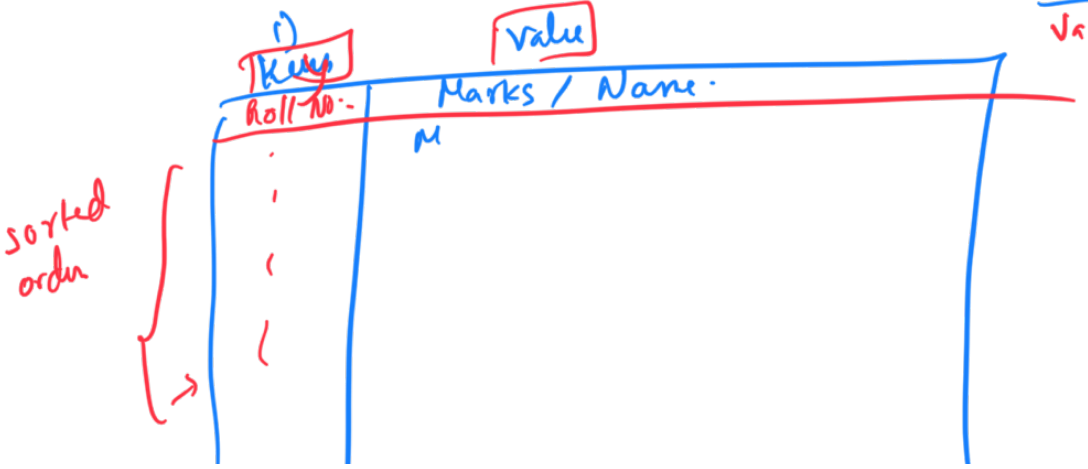
EXL Class of 200 students

Sorted Manner

Manveer => 95
Bhubha => 73
Sabiha => 82
Anish => 63

Strings than Number

Roll number / id



Key: Roll No.
Value: { Name, marks }

=> O(1) are

Binary search => O(log n)

Hash table / Hashmap:

Hashing:

Does this operation in average O(1)

Arrays

1) Insert
2) Update
3) Delete
4) Search

} => O(1) amortized

100 operations

→ 98 operation take O(1) time

→ 2 operations take O(n) time.

Each Operation: O(1) Amortized

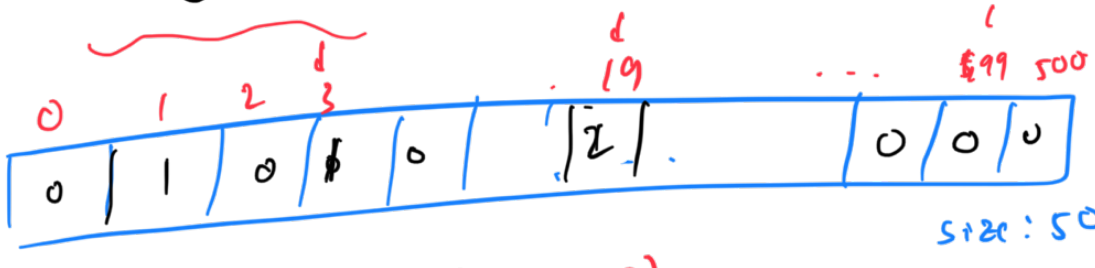
Question: Getting numbers in range (1-500)

No. of times this number has occured before.

1 19 3 19 2 19 3 1 19

0 0 0 1 0 2 1 1 3

Eg: (Size: 500)



Freq?

Range was (0 - 500)

create an array of size 500

→ [0 - 10^8) => Memory Limit Error

(Range 0-500)

15 99 ...

(1 ... 10)

Key: Number

Value: Frequency

Question: Numbers in range (1-100) (Almost 10 different)

Constraint: Array size can be 10

[1-100] → 10

modulo

x % 10 → [0, 9]

[1-100]

x % 10

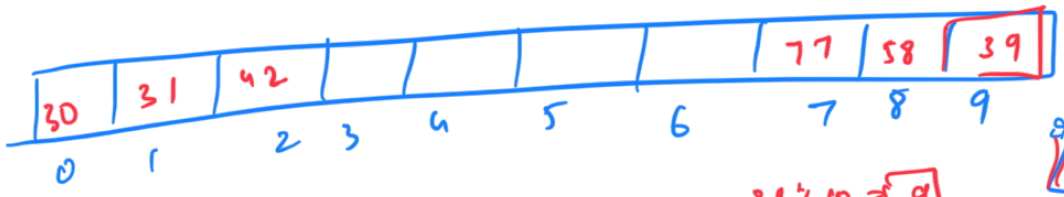
Hash code

Map 100 elements → 10 elements

x % 10

58, 31, 42, 77, 30, 39

x % 10



10

Search for 39?

39 % 10 = 9

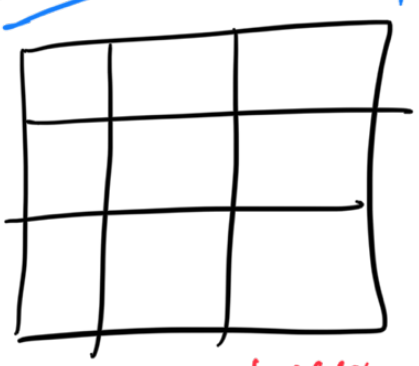
Ex: 58, 38, 28, 18, 68, 98

8

x % 10 =>

100 values → 10 values?

Pigeon Hole Principle.



=> 10 pigeons

bigger range → smaller range

x%10

there will be x%10 collisons (clashes)

Deal with collisons?

Maintain a Linked / Dynamic Array

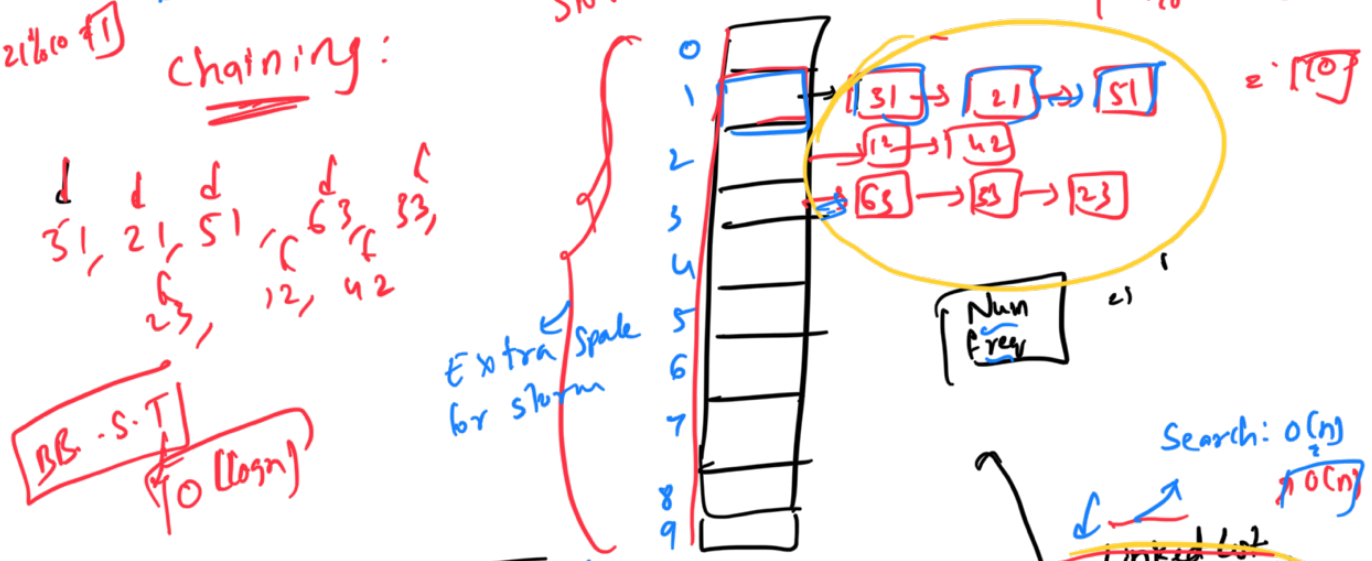
Store the Reference

x%10 = 1

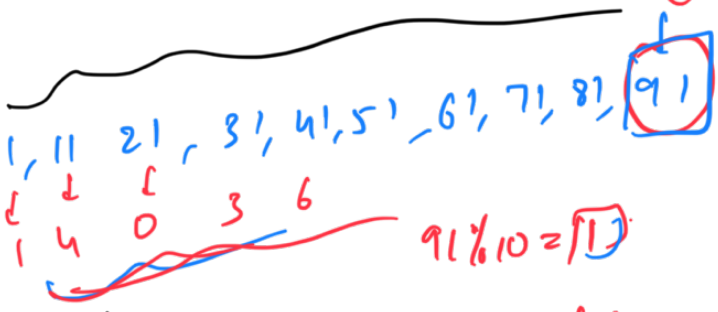
21%10 = 1

Chaining:

31, 21, 51, 63, 53, 23, 12, 42



BB-S.T → O(logn)

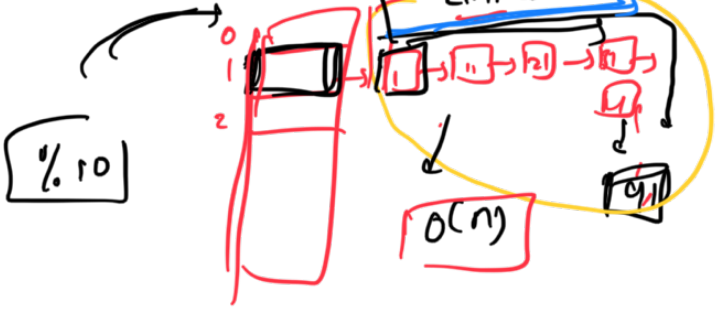


91%10 = 1

x%7

T.C: O(n)

→ Uniform Distribution
→ Less No. of collison



Array:

searching: O(1) ✗ any

Sorted Array:

O(logn)

Insertion:

2 3 6 7 9 10

O(n)

O(1)x

Can we do better than O(n) ?

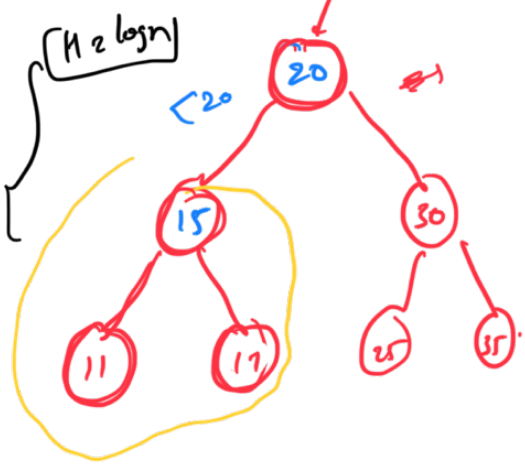
Java 7 → Java 8

Red Black / AVL

Balanced Binary Search Tree

↓

Binary Search :



Insertion, Deletion, Search:

O(log n)

get Min : O(1) → Heap / Priority

: O(n)

Pointers to left, Right

Balanced BST

↓

H = log n

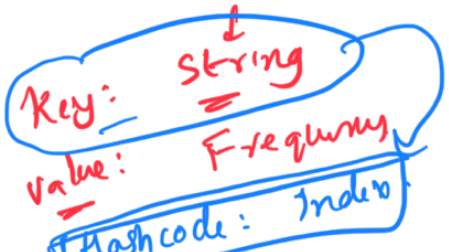
B.S.T

↓

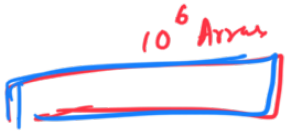
H = n

Number ✓

Frequency ✓



random strings



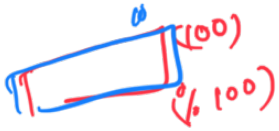
Question:

Getting

↓

abc bzy bca abc abc bzy Anish Anish

0 0 0 1 2 1 0 1



n % 10

(str) % 10

Integer to Integer

↓

(Index)

n%10

string%10

Random Access / Quick lookup => Array

Task: Map this string to Integer. !

Hash Function: string → Number

Option 1: Consider ASCII value of 1st character.

aab : 97 ✓ (97) 1) too many collisions

abc : 97 ✓ 2) Not using space

bca : 98 ✓

abb : 97 ✓

abb → ASCII : Integer

(abbbcxu......) → 97

26 entries

(a – z) => 26

Option 2: Sum of ASCII value. string →

aab :	97+97+98 =	292
abc :	97+98+99 =	294
bca :	98+99+97 =	294
abb :	97+98+98	293

Anagrams:

abcd, bcad, adbc...... Same Hash value

ac, bb =>

Option 3: Polynomial Rolling Hash Function

str = a b c d e f
      0 1 2 3 4 5

=> $S[0] + S[1]*p + S[2]*p^2 + S[3]*p^3 + \ldots$

First char

p: Prime Number > 26

10–15

Java: [p^i] s

s[i] → Ascii value of ith charact

F(n) = 2n+1

F(abcd) = (97 + 98 x p + 99 x p^2 + 100 x p^3 ....) %

= a + b x p + c x p^2

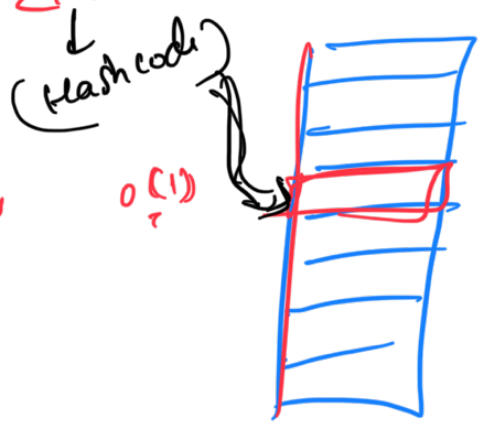
(very big)

Prime Number:

1) Lesser Collision
2) Uniform Distribution of keys

Characteristics of Good Hash Function

1) No / less Collisions
   ↓ Deal with it
2) Fast Computation



x % 10 ⇒

3) Even Distribution

F(n) = 2*n + 1

⇒ odd indices will be occupied

O(M)

Summary:

1) Hash Map: key - value pair
2) Int / String ⇒ int to use as an index of array
3) Hash code.
4) Functions: Hash Function is called Hashing

5) Proces

$F(x) = 2x+1$ → Function

→ base 2)

Prime

Hash value of string: $O(s)$

LL: $O(n)$

If we use a good Hash Function

T.C: of Insertion / Deletion / searching

$O(1)$ → Amortized

Hashmaps: Lot of Problems. → $O(1)$

100 operations:

98 operations: $O(1)$

2 operations: $O(n)$

$O(1)$ Amortized

Hashing Libraries

Hashmap : <key, value>

Frequency of string

key: string

value: int

Hashmap<string, int> mp;

Hash set : $O(1)$ $O(1)$

Key

Question: Give you some number, tell if it is occured or not.

$O(1)$ $O(1)$

Arr = 5 10 11 5

F F F T

O(1)

Dict

(Hash map)

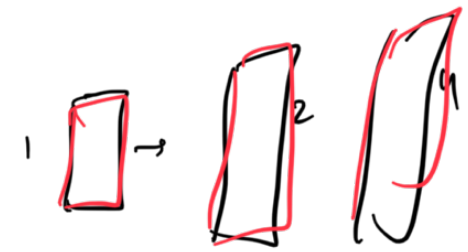
Set = { 5, 10, 11 }

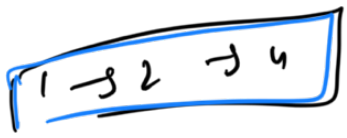
→ Store only key

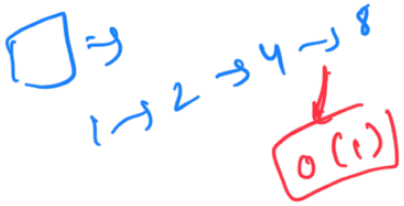
C++ ; unordered_set unordered_map.

Java : HashSet, HashMap } [Hash Table]

Dynamic Array







Hash map : Implemented using Hash table.

map = { 2 : 3, 3 : 2, 4 : 1 }

Question:

Size N

Array = [2 3 2 2 3 4 5 4 2 5]

Queries, q

(1 - 10^9)

{ 5 : 2 }

Hashmap <int, int>

Frequency

(% 10)

[3 : 2]

Number

Hashmap<int, int> mp;

for(int i=0; i<n; i++) {

check if a[i] in mp {

mp[a[i]]++;

} else {

put {a[i], 1} in mp:

} } ⇒ O(n)

GFG

O(1)

B = 5 4 3

(q < N)

5 5 5

q times

for(i=0; i<q; i++) {

ele = B[i];

if(ele in there in mp)

return mp[ele];

else

~~return 0;~~

queries[i] O(1)

queries = | 5 | 2 | 1 | 6 | 2 | 9 | 10 |

q

T.C:

(O(n) + O(q))

⇒ O(n+q)

Step:

T.C: O(n+q)

q = N² ⇒ O(n+q)

= O(n+n²) ⇒ O(n²)

(q < n)

Question:

First Non-Repeating element

A = [20, 1, 2, 10, 20, 1, 5, 2, 5]

0 1 2 3 4 5 6 7 8

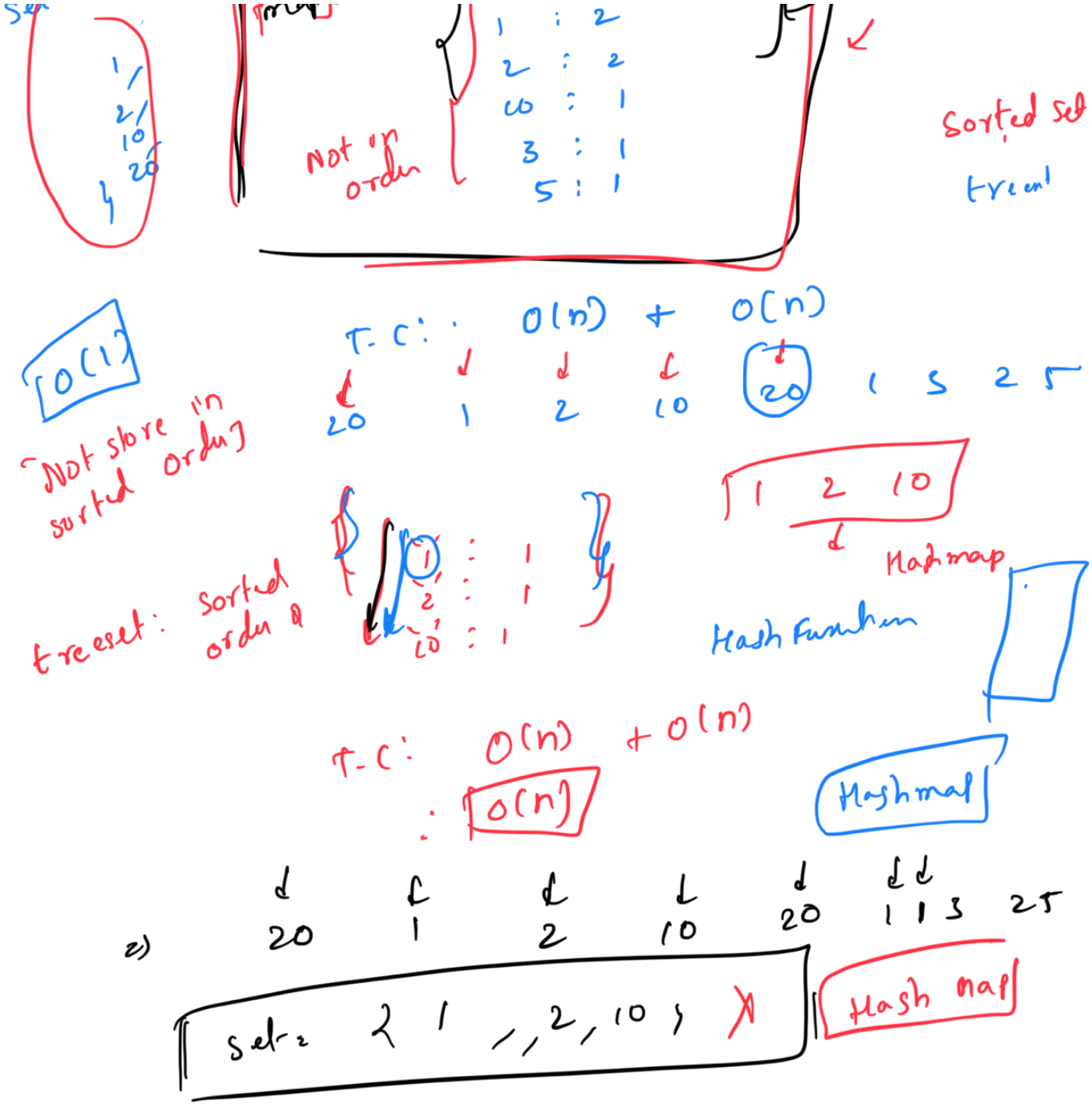
Freq = 1

ele = [0, 3, 5]

O(1)

(Key: 10)

[20 : 2


Not in order
Sorted set
tree
O(1)
"Not store in sorted order"
T.C: O(n) + O(n)
treeset: Sorted order
Hashmap
Hash Function
T.C: O(n) + O(n)
O(n)
Hashmap
Hash map
Question:
Given 2 arrays
(No Duplicates)
Ans

Hashset:

set = { 3, 2, 10, 7, 5, 15, 9 }

T.C: O(n+m)

Duplicacy:

Hash set → Hashmap ?

Hashmap = <key, value>

(value)

Question: Goldman Sachs, Google, Deshaw

7 + (K-7)

K = 9

A = [7, 4, 10, 2, 5, 16, 3]

0 1 2 3 4 5 6

a[i] + a[j] = = K ∀ i ≠ j

(7, 2) ✓

2 5 3 4 → (4+4)

3+3

Brute Force:

for a[i], (K - a[i]) ✓

T.C: $O(n^2)$

$O(n^2)$ → O(n)

Efficient Approach:

A = [1, 4, -2, 8, -9, 14, 25, 22, 17, 13]

K = 22

{8, 14}

set = { 1, 4, -2, 8, -9, 14, 25, 22, 17, 13 }

(14, 14) Yes

(1, 8) ✗

(4, 18) ✗

(-2, 24) ✗

(8,

Yes

K = 28

1, 27 ✗

4, 24 ✗ True

-2, 30 ✗

8, 20 ✗

-9, 37 ✗

14 → (14) ≥ 2

14 ⇒ 1

Hashmap ⇒ < Num, Frequency>

Frequency

a[i] ⇒ K - a[i]

(a[i] = K - a[i]) only if freq(a[i] ≥ 2) True

Hashset solution:

k = 22

k = 28

14 + 14 = 28

A = 1, 4, -2, 8, -9, 14, 14, 25, 22, 17, 13

set = { 1, 4, -2, 8, -9, 14 }

(-2)

(24)

-9,

(Hash Function)

(Integers)

(14, 8)

K = 28

(14)

1) Pair s.t a[i] - a[j] = K

2) pair s.t a[i] ^ a[j] = k

(Goldman, M.S, Amazon)

Question:

→ Given array of strings, group all the anagrams together. (Permutations)

A = {nlmo abc xyz cab lmno }

pqr xpq omn nmol

array of arrays (vector of vectors)

=> → [nlmo, lmno, omln, nmol]

vector<string>

[abc, cab]

[pqr, xpq]

[xyz]

1) Sum of Ascii (Block)

ac = bb

(abc, def)

[abc, cba]

Task: Map all the Anagrams to a single Hashcode

ascii-97

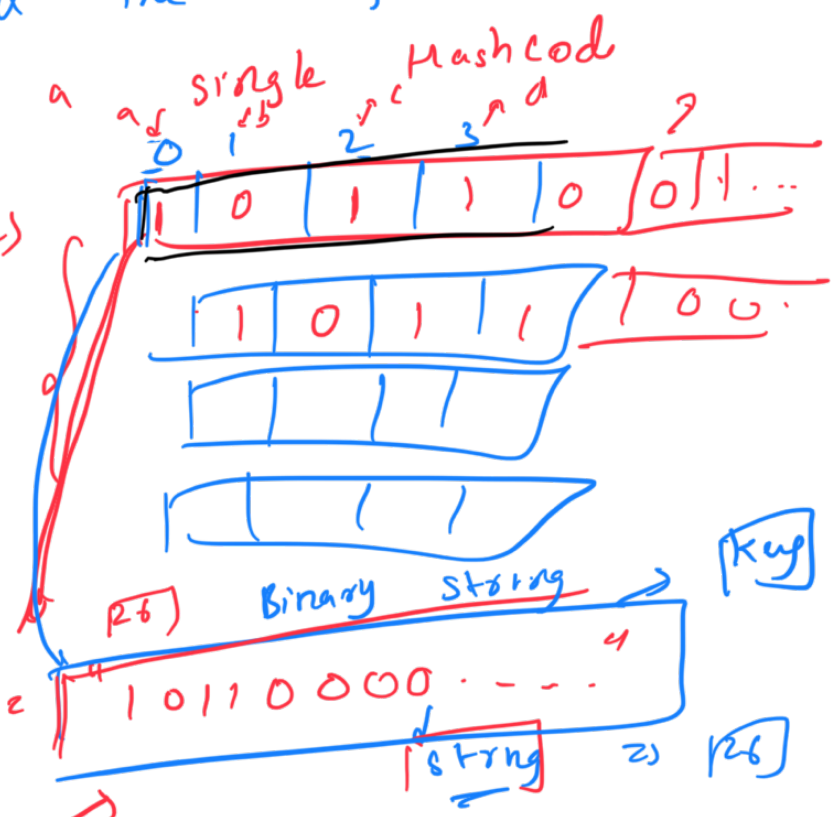
97 98

a d

d c a

c a d

c d a



Hashmap<String, vector<string>>

Hashmap < String, vector <string> > .

Approach 2:

acd, dca, cad, cda

acd acd acd acd

Use sorted string as the key:

HashMap<string, vector<int>>

key: sorted string

size 's' string: slogs × n

O(n.s.logs)

Can we sort Faster?

Count Sort (Bucket Sort)

s = b d a b c d e

26 size

O(s)

T.C: O(s)

O(s) + O(s) = O(s)

abbcdde

T.C: O(s.N)

O(26) = O(1)

O(1)

26 size

10^{18}

Chaining

Linear Probing

Quadratic Probing 2)

$x\%10$, $(x+1)\%10$, $(x+2)\%10$, $(x+3)\%10$

23 44 34 67 79 65 13 11

Array = 10

$(1-00)$

$x^2 \% 10$

43

Elog = T

Delete : 13

13

13%10

Search : 27

Delete : 44

$x\%p$

→ Less Collision
→ Uniform Distributed

GCD / HCF

H(x) = x % 12

Multiples of 4:

4, 8, 12, 16, 20, 24, 28

12 / gcd(12,4)

Only 3 buckets be used

2) Multiples of 6:

0 6 12 18 24 30 36

{0, 6}

12/6 = 2 gcd(12,4) = 4

12/4 = 3

gcd (...
(6)

12 = gcd(12, x) = 12

gcd(2, x) = 1 → prime

p

gcd(p, x) = 1

Generalised Dataset

Multiple

{ a_1, $2a_1$, $3a_1$ .... }

$a_1 = 2$
{2, 4, 6, 8 ... }

{ a_2, $2a_2$, $3a_2$ .... }

$a_2 = 3$
{3, 6, 9, 12 ...}

gcd(p, 2) = 1

Multiple a2 | Multiple a3

4 | 8 | 6

gcd(p, 3) = 1

gcd(p, 6) = 1

gcd(p, 8) = 1

gcd(p, 4) = 1

p ?

p is prime

Size of Hash table

gcd(12, x) = 12

[1]

gcd(12, x) = 1

p

(0 - 9)

{0-9}

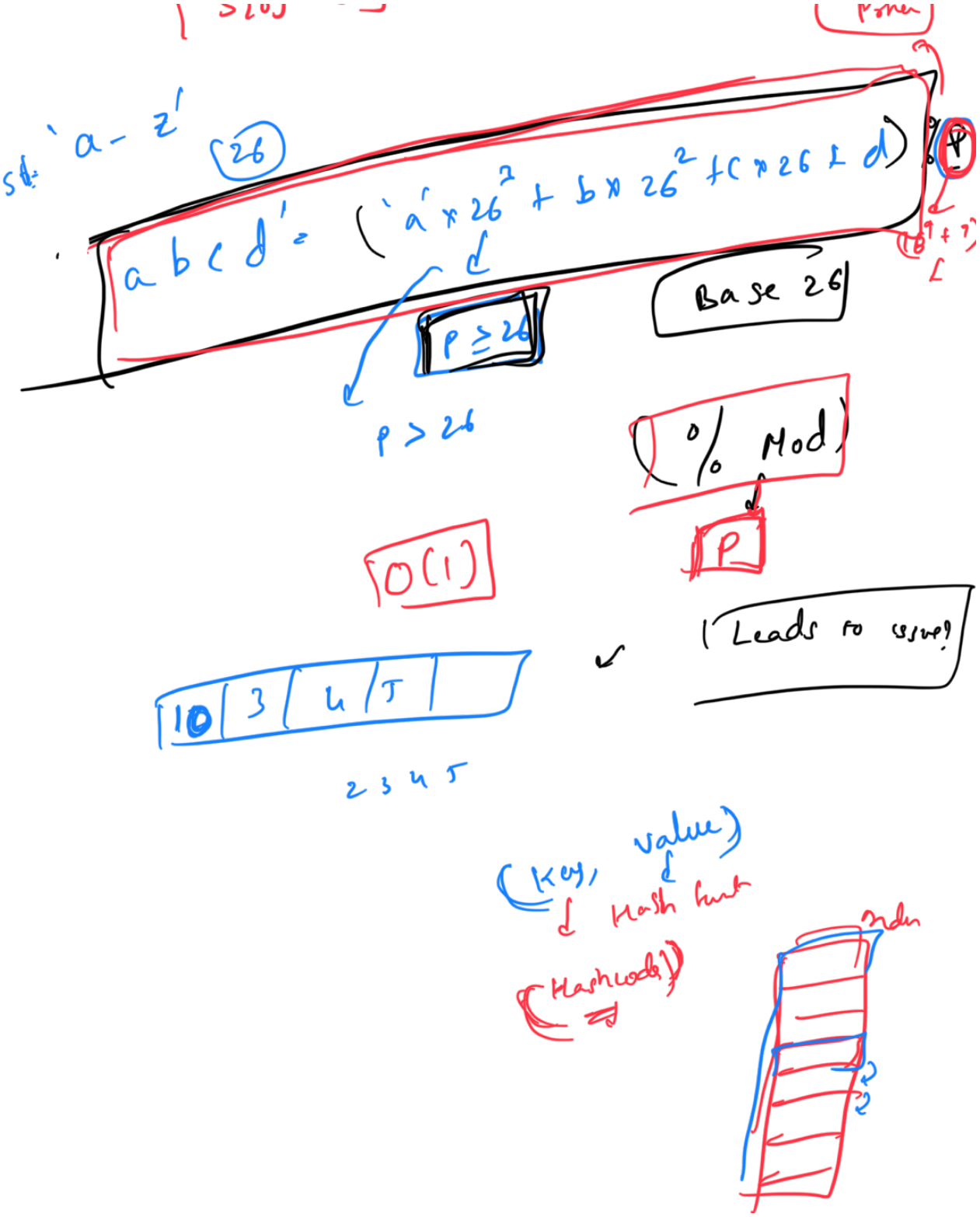
s = 3 7 8 9 : ⇒ 3 7 8 9

$3 \times 10^3 + 7 \times 10^2 + 8 \times 10 + 9$ ⇒ Base 10

[s[0] - '0'] ⇒ 3

[s[0] - '0'] × 10^3 + [s[1] - '0'] × 10^2 ....

[illegible]


'a - z'
26
abcd = ('a' x 26^3 + b x 26^2 + c x 26 + d)
P
Base 26
P ≥ 26
P > 26
% Mod
P
O(1)
Leads to collision
10 3 4 5
2 3 4 5
(Key, value)
Hash function
Hashcode
Index

Pair with sum = k

```
bool checkSum(int a[], int n, int k){
      unordered_set<int> st;
      for(int i=0;i<n;i++){
            if(st.find(k - a[i]) != st.end()){
                  return true;
            }
            else
                  st.insert(a[i]);
      }
      return false;
}
```