Theme: Hashing

Date: / /

Sat Sun Mon Tue wed Thu Fri

Imagine given array 1 2 1 3 2

Give me how many times '1' appears in array ?

Method 1:- Looping iterations & taking count

TC → 5 × O(N)

1 → 2
3 → 1
4 → 0
2 → 2
10 → 0

But what is 'x' x O(N)

x is too long → O(Q × N)

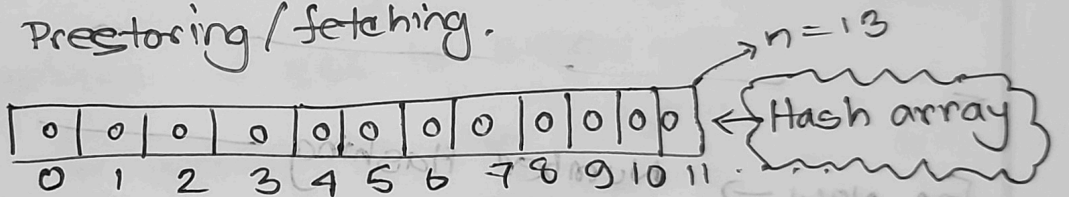
∴ O($10^5 \times 10^5$) = O(10^{10})

10^8 → 1s

1 → $\frac{1}{10^8}$

10^{10} → $\frac{10^{10}}{10^8}$ = 100s

Hashing → Preestoring / fetching.



Precalculation →

```
main() { input (size + array)
        int q ; ⟶ How many nums
        cin >> q ;   while (q--) { int number ; ← input }
        int hash [13] = {0} ;
// Precompute ⟶ for ( i=0 , i<n ; i++ ) { hash [arr[i]] ± 1 }
⟶ // fetch   cout << hash [number] << endl ; ⟶ Inside while
```

BD SCHOOL SHOP
ONE-STOP SCHOOL SHOPPING

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

* what if array size / max elems → 10^9

int max_size_of_array => 10^6

otherwise segmentation fault.

But → Globally we can define globally till 10^7.

int hashh [10^7] ;

	int	Bool
main	10^6	10^7
Global	10^7	10^8

for NOW → Number Hashing ——— Arrays ✓

Character " ——— Arrays ✓

s = "abcdea" ; → a=2 just like frequencies

a	b	...

0 1 ... 25

ASCII

a → 97

f → 102

f - a → 5

a - a = 0

b - a = 1

...

→ formula → char - 'a'

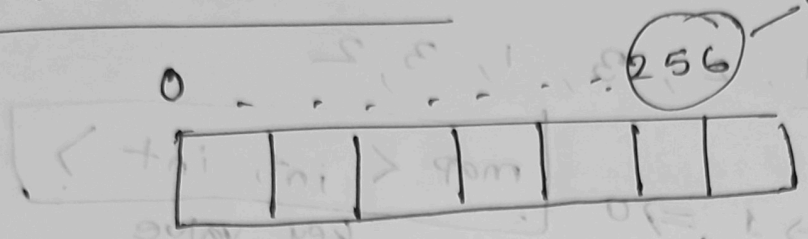
4

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

smaller — Capital



a → 97
b → 98

```
main() { string s; → input
//Precompute
int hash[256] = {0};
for(i=0; i<s.size(); i++) { hash[s[i]]++; }

int q; → input
while(q--) { char c; ← input
//fetch
cout << hash[c] << endl;
}
```

No complications → 256 is size.

- for smaller → char - 'a'
- for Caps → char - 'A'
- everything is autocasted.

number hashing → STL → map / unordered map.

BD SCHOOL SHOP
ONE-STOP SCHOOL SHOPPING

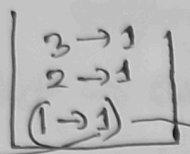
Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Map Concepts => stores in sorted order

arr = 1, 2, 3, 1, 3, 2



mpp → 1 => 0

mpp[i] ++

map < int, int >

key, value

num, frequency

1 → 2

To do this → mpp[arr[i]]++ ;

So, map takes little memory and takes only what we have.

```
main () { input (size, arr)

    //Pre Comp
    map < int, int > mpp;
    for (i = 0 , i < n , i++) { mpp[arr[i]]++ ; }

    input (query)
    while (q--) { input number;
                  //fetch
                  cout << mpp[number] << endl;
    }
```

// iterate in map

```
for (auto it : mpp) { cout << it.first << "->" << it.second << endl
```

* Pre compute inside the array input is Ok!
it will save us one for loop. Basically same.

BD SCHOOL SHOP
ONE-STOP SCHOOL SHOPING

Theme: Any datatype will be a key

Date: / /

Sat Sun Mon Tue wed Thu Fri

Map in string hashing

map<char, int> ⟶ char - key, int → value

mpp [s[i]]++; → O(logN)

even if we take unordered_map it will work.

Unordered → only int, double, string...

Advantages

Storing / fetching > TC O(1) {Avg best}

why? worst case → O(N) → Happens once in a blue moon
↳ Interal collisions.

Hashing → (i) Division Method → linear chaining

Not Req → (ii) Folding Method

Not Req → (iii) Mid square w

(i) Division Method

{2, 5, 10, 28, 139, 38, 48, 28, 16} but array > 10 X

arr[i] % 10

Linked List + Binary search

0 —
1 —
2 — 2
3 —
4 —
5 — 5
6 — 16
7 —
8 — (48) → 28 → 28 → 38 → 48
9 — 139

sorted

BD SCHOOL SHOP
ONE-STOP SCHOOL SHOPPING