Hashing 1

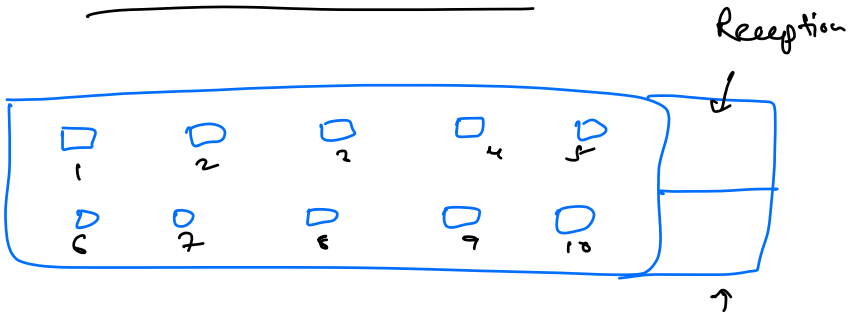
Agenda

- Introduction
- Frequency Count
- First Non Repeating Element
- Subarray with sum=0

Arrays

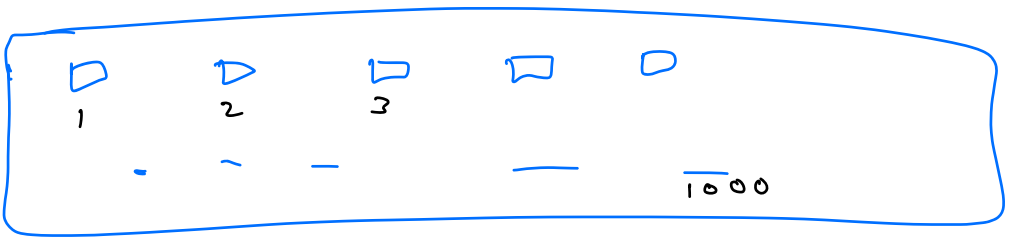
What is Hashing ?

Bruce & Clark



10 rooms

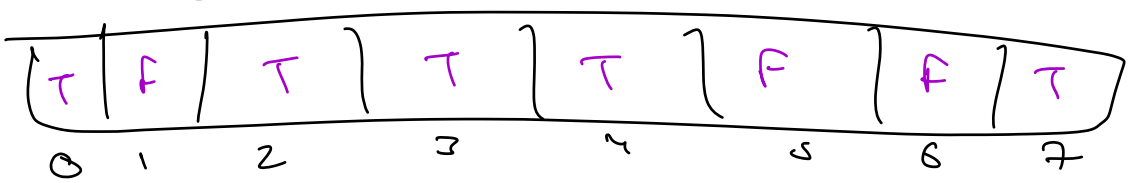
→ Maintain a register



1000 rooms

boolean array

boolean occupied [1000]



Covid Pandemic

Went to a numerologist.

Lucky Numbers between $[1, 10^9]$

Numbers - 5, 9, 12, 13, 21, 25, ... 1002, 10^5, 10^6, 10^2 -

boolean occupied $[10^9]$ ← very large space

Advantage: O(1) time
fast access

---

5 → T

19 → F

12 → T

13 → T

21 → F

86 → T

Hashmap

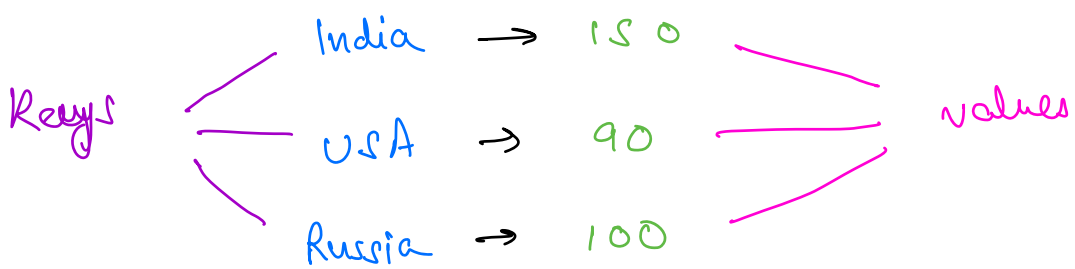
key → value

Time - O(1)

Space - O(N)

Understanding key-value

Q1. Store population of every country



Country → Population

HashMap < str , int >

Q2. No of states in each country

India → 29

USA → 50

Russia → 60

keys values

HashMap <str , int>

Q3. For every country, store all state names

Country → List [states]

Hashmap < str, Array < str > >

Q4. For every country, store population of each state.

Country → Hashmap < str, int >

Population of each state

Hashmap < string, Hashmap < str, int > >

India → {
Punjab: 1000
TN: 2000
Kerala: 3000
}

Hashmap Operations

India → 29
USA → 50
Russia → 60

- insert (key, value)
- search (key)
- remove (key)
- update (key, newValue)
- size ( ) ← No of entries

Time : O(1)

Space : N operations → O(N) space

Internal Implementation (Hashing) → Advanced Batch

Hashset Operations

It only stores keys.

{ 2, 5, 1, 6, 7, 8 }

- insert (key)
- search (key)
- remove (key)
- ~~update (key)~~ → remove (oldKey) → insert (newKey)
- size ( )

O(1) time

Space : O(N)

Unique keys

1, 1, 2, 3, 2, 2, 3, 4, 4, 4, 4

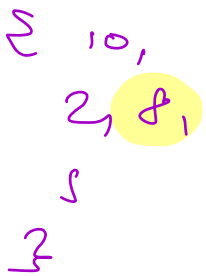
↳ {1, 2, 3, 4}

Quiz 1

Arrays	Sets
→ Ordered DS	→ Unordered DS
→ Indexing	→ No indexing
→ Duplicate data	→ Unique keys

→ Time Complexities
- O(N)

→ Time Complexities
- O(1)



Language Implementation

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

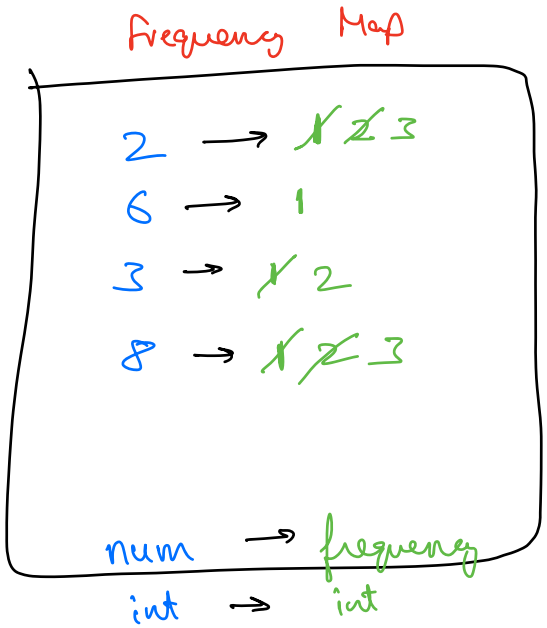
Q1 Frequency of Numbers

Given N array elements & Q queries. For each query find frequency of each element in array.

arr[10] = 2 6 3 8 2 8 2 3 8

Q = 4

query
- 2 : 3
- 8 : 3
- 3 : 2
- 5 : 0



Brute force

1) for each query, iterate & get the count.

TC: O(Q * N)

SC: O(1)

Optimised Approach

```
Hashmap <int, int> freq;

// Iterate & store the frequencies
for(i=0; i<N; i++) {
        // Insert ar[i] in my hashmap
        if (ar[i] in freq)
                update value by +1
        else
                insert (ar[i] → 1)
}
// Answer the queries
for (i=0; i<Q; i++) {
        query = input()
        if (query is in freq)
                print( freq[query])
        else
                print (0)
}
```

Java

```java
import java.util.Scanner;
import java.util.HashMap;

public class Frequency {
    public static void main(String[] args) {
        int[] arr = { 2, 5, 6, 1, 7, 8, 9, 2, 7, 5, 10, 8 };
        HashMap<Integer, Integer> freq = new HashMap<Integer, Integer>();

        for (int i = 0; i < arr.length; i++) {
            if (freq.containsKey(arr[i])) {
                freq.put(arr[i], freq.get(arr[i]) + 1);
            } else {
                freq.put(arr[i], 1);
            }
        }

        // Answer Q queries
        Scanner sc = new Scanner(System.in);
        int Q = sc.nextInt();
        while (Q-- > 0) {
            int query = sc.nextInt();
            if (freq.containsKey(query)) {
                System.out.println(freq.get(query));
            } else {
                System.out.println(0);
            }
        }
    }
}
```

Python

```python
arr = [2, 5, 6, 1, 7, 8, 9, 2, 7, 5, 10, 8]

freq = {}    <- dict

for i in range(len(arr)):
    if arr[i] in freq:
        freq[arr[i]] += 1
    else:
        freq[arr[i]] = 1

# Answer Q queries
Q = int(input())
for _ in Q:
    query = int(input())
    if query in freq:
        print(freq[query])
    else:
        print(0)
```

O(N)

O(Q)

Time = O(N+Q)

Space = O(N)

Break till 10:21 PM

Q2 Find the first non repeating element.

sprinklr

snapdeal

Example 1 Quiz 2

arr[6] = 1 2 3 1 2 5

1 → 2

2 → 2

3 → 1

5 → 1

Ans = 3

Example 2

arr[8] = 4 3 3 2 5 6 4 5

4 → 2

3 → 2

2 → 1

5 → 2

6 → 1

Ans = 2

Example 3

arr[7] = 2 6 8 4 7 2 9

Ans = 6

Idea & Pseudocode

Brute force — N^2

1) Construct the frequency map

2) Iterate over the array, get the first element with freq = 1

TC : O(N)

SC : O(N)

Java

```java
import java.util.HashMap;

public class FirstNonRepeating {
    public static void main(String[] args) {
        int[] arr = { 1, 2, 3, 1, 2, 5 };
        HashMap<Integer, Integer> freq = new HashMap<Integer, Integer>();
        for (int i = 0; i < arr.length; i++) {
            if (freq.containsKey(arr[i])) {
                freq.put(arr[i], freq.get(arr[i]) + 1);
            } else {
                freq.put(arr[i], 1);
            }
        }

        for (int i = 0; i < arr.length; i++) {
            if (freq.get(arr[i]) == 1) {
                System.out.println(arr[i]);
                break;
            }
        }
    }
}
```

Python

```python
arr = [1, 2, 3, 1, 2, 5]

freq = {}

for i in range(len(arr)):
    if arr[i] in freq:
        freq[arr[i]] += 1
    else:
        freq[arr[i]] = 1

for i in range(len(arr)):
    if freq[arr[i]] == 1:
        print(arr[i])
        break
```

Q3 Given N array elements, find no. of distinct elements.

Example 1

arr[5] = 3 5 6 5 4

{ 3, 5, 6, 4 }

⇒ 4

Example 2 Quiz 3

arr[7] = 6 3 7 3 8 6 9

{ 6, 3, 7, 8, 9 }

⇒ 5

Hashset

```
Hashset < int > hs;
for (i=0; i<N; i++) {
        hs.insert ( arr[i] )
}
return hs.size()
```

Java

```java
import java.util.Arrays;
import java.util.HashSet;

public class UniqueElements {
    static int countOfUnique(Integer[] arr) {
        HashSet<Integer> set = new HashSet<Integer>();
        for (int i = 0; i < arr.length; i++) {
            set.add(arr[i]);
        }
        return set.size();
    }

    static int countOfUniqueShort(Integer[] arr) {
        HashSet<Integer> set = new HashSet<Integer>(Arrays.asList(arr));
        return set.size();
    }

    public static void main(String[] args) {
        Integer[] arr = { 6, 2, 4, 7, 4, 4, 2, 6, 1 };
        System.out.println(countOfUnique(arr));

        System.out.println(countOfUniqueShort(arr));
    }
}
```

Python

```python
def countOfUnique(arr):
    s = set()
    for i in range(len(arr)):
        s.add(arr[i])
    return len(s)


def countOfUniqueShort(arr):
    s = set(arr)
    return len(s)


arr = [6, 2, 4, 7, 4, 4, 2, 6, 1]
print(countOfUnique(arr))
print(countOfUniqueShort(arr))
```

TC: O(N)

SC: O(N)

Q4 Given N array elements, check if they are all distinct or not.

Example 1

2 5 4 9 ⇒ True

Example 2

2 9 8 2 6 ⇒ False

Example 3

5 8 6 10 8 6

⇒ False

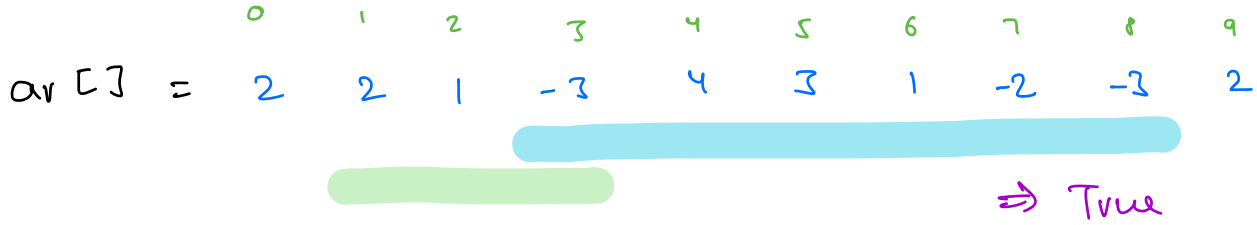
if (countOfUnique(arr) == N)

return True

else

return False

Q5 Given N array elements, check if there exists a subarray with sum=0.



No of subarrays = $\frac{N(N+1)}{2}$

Quiz 4

Idea

1) For every subarray, check if sum = 0
   ↳ $O(N^3)$

2) optimisation: Carry forward
   ↳ $O(N^2)$

3) Prefix Sum

Quiz 5

$sum[L\ R] = pf[R] - pf[L-1]$

	0	1	2	3	4	5	6	7	8	9
ar[] =	2	2	1	-3	4	3	1	-2	-3	2
pf[] =	2	4	5	2	6	9	10	8	5	7

$pf[2] = 5 = sum[0\ \ 2]$

$pf[8] = 5 = sum[0\ \ 8]$

$\Rightarrow 5 = sum[0\ \ 2] + sum[3\ \ 8]$

$\Rightarrow 5 = 5 + sum[3\ \ 8]$

$\Rightarrow 5 - 5 = sum[3\ \ 8]$

$\Rightarrow 0 = sum[3\ \ 8]$

$pf[0] = 2 = sum[0 \quad 0]$

$pf[3] = 2 = sum[0 \quad 3]$

$\Rightarrow 2 = sum[0 \quad 0] + sum[1 \quad 3]$

$\Rightarrow 2 = 2 + sum[1 \quad 3]$

$\Rightarrow 2 - 2 = sum[1 \quad 3]$

$\Rightarrow 0 = sum[1 \quad 3]$

Idea

If we have repetition in the prefix sum array, ans → True

1) Construct pf[] away ← TODO

2) Create a hashset & insert all values of pf[] in it.

3) if (hs.size() < N or (0 in hs))

Repetition → True

else

No repetition → false

This will NOT WORK

	0	1	2	3
arr[] =	-1	4	-3	2
pf[] =	-1	3	0	2

pf[2] = 0 ⇒ sum[0 2] = 0

Java

```java
boolean checkSubarrayWithZeroSum(int[] arr) {
    HashSet<Integer> s = new HashSet<Integer>();

    int prefixSum = arr[0];
    s.add(prefixSum);

    for (int i = 1; i < arr.length; i++) {
        prefixSum += arr[i];
        s.add(prefixSum);
    }

    if (s.size() < arr.length || s.contains(0)) {
        return true;
    }

    return false;
}
```

Python

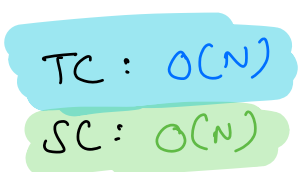
```python
def checkSubarrayWithZeroSum(arr):
    s = set()

    prefixSum = arr[0]
    s.add(prefixSum)

    for i in range(1, len(arr)):
        prefixSum += arr[i]
        s.add(prefixSum)

    if len(s) < len(arr) or (0 in s):
        return True

    return False
```



Doubts

Thank
You

Keys → Immutable

int, str, tuple, float, bool

~~list, set, dict~~

values → Anything

Good Night

Thank You

Wednesday