Agenda

1. Introduction to HashMap & HashSet
2. Given queries, find frequency of an element
3. No.of Distinct Elements

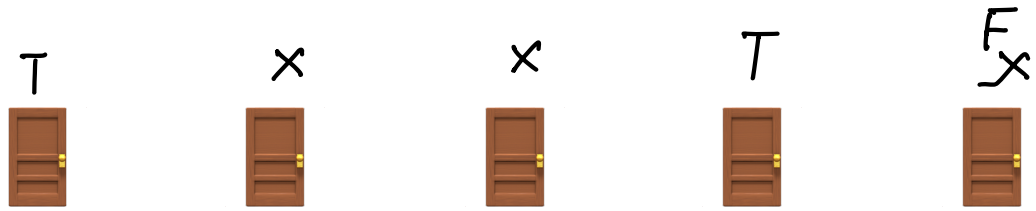
Agenda

- Introduction to HashMap & HashSet - Iterate over map [redacted]
- Given queries, find frequency of an element
- No. of Distinct Elements
- Pair sum
- Count of Pair Sum

Hotel

0	1	2	3	4
T	F	F	T	F

5 rooms



↓

1000 rooms

T - occupied

F → vacant

0	1	2	3	. . . .	999

10^9 hotel rooms

boolean Array $= 10^9 * 1B$

$= 10^6 * 10^3 B$

$= \underbrace{10^6 B}_{1MB} * 10^3$

$1MB * 10^3 = 1000 MB$

≈256 MB

Input → 10^6 hotel rooms $= 10^6 * 1B = 10^6 B$

$= 1MB$

	T	T	T	
Static array	1	1000	10000	False
		10^6		

100

1000

1	~~1000~~	10000

→ List → for searching

dynamic array contains → List of occupied rooms

O(1) lookup? ✗

arr → 10^6

arr[100] → T ✓ / F ✗

HashMap = 2 column table

Room Numbers

all elements are unique in Key

Key is unique	Value
1	True
1000	True
10000	True

O(1) lookup/search

Using Key → address → value

Key's value → address

Nardup → unique address

O(1)

Element Key value → formula (hashcode) → unique address

O(1) lookup

hashtable hashmap

dict

unordered

1. Population of every country

Key	Value
String	Long

10^9 1B

10^{10} 10^{11}

🇮🇳 India → 145

🇺🇸 USA → 65

🇷🇺 Russia → 14

🇨🇳 China → 147

2. Number of states of every country

String	Int

🇮🇳 India → 29

🇺🇸 USA → 50

🇨🇳 China → 25

🇷🇺 Russia → 21

SCALER

3. Name of all the states of every country

Key	Value
India →	Andhra Pradesh, Arunachal Pradesh, U.P, M.P, Karnataka,....
USA →	New York, Washington, Texas,...
Russia →	Moscow, Kazan, Samara,...
China →	X, Y, Z,...

String	List <String>

get → List

search True/false

4. Population of each state in each country

🇮🇳 **India**

S I

Karnataka → 50

Maharashtra → 72

Himachal Pradesh → 36

🇺🇸 **USA**

Texas → 18

Florida → 16

Washington → 32

🇷🇺 **Russia**

Kazan → 3

Samara → 5

Moscow → 7

String	HashMap<String vs Int>

SCALER

Key $\xrightarrow{\text{formula}}$ unique address

Hash - Map	Hash - Set
2 column table Key is unique Search in Keys : O(1) add(Key, value) size delete (Key) update (Key, value) search (Key) get (Key) O(1)	1 column table Key column Keys are unique Search in Keys : O(1) add (Key) size delete (Key) search (Key) O(1) No ~~get (Key)~~ — 1 col

implementation

	Java	C++	Python	JS	C#
Hash - Map	HashMap	unordered_map	dictionary	map	dictionary
Hash - Set	HashSet	unordered_set	set	set	Hashset

Hotel example →

Occupied Rooms

HashSet →

Key
1
~~1000~~
1000

1 column table $\xleftrightarrow[\text{same}]{\text{not}}$ list

1
1000
10000
100

O(1) lookup

1, 1000, 10000, 100

O(1) lookup?

SCALER

< Question > : Given N elements & Q queries. Find the frequency of elements provided in the query.

($1 \le N \le 10^6$)

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

N = 11

2	6	3	8	2	8	2	3	8	10	6
0	1	2	3	4	5	6	7	8	9	10

11

Q = 4

2 → 3 times

8 → 3 times

3 → 2 times

1 → 0 times

Idea 1

- For each query, find the frequency of the element in the Array.
- TC - *O(QN)** and SC - **O(1)**.

TC = O(Q * N) , SC = O(1)

Idea -2

arr[i] Int	count Int
8	~~1~~ ~~2~~ 3
2	~~1~~ ~~2~~ 3
3	~~1~~ 2
6	~~1~~ 2
10	1

2638282381 06

search the element →

element present → value+1

element not present → 1

SCALER

for every Query → O(1)

< / > Code

```
void answerQueries (int[] A, int[] Q) {
    HashMap<int, int> mp;
    for ( i=0  i<N  i++) {
        if (mp.search(A[i]) == true)
            mp.add (A[i], mp.get(A[i]) + 1)
        else
            mp.add (A[i], 1)
    }
    for ( i=0 ;  i < Q.length  i++) {
        if (mp.search (Q[i]) == true)
            print mp.get (Q[i])
        else
            print 0
    }
}
```

< Question > : Count of distinct elements

N = 5 [3 5 6 5 4] 4

N = 3 [3 3 3] 1

N = 5 [1 1 1 2 2] 2

1 2

< / > Code

Insert all elements in Hashset

return its size.

```
Function distinctCount(Array[])
{
  hashset<integer>set;
  for(i -> 0 to Array.length - 1 )
  {
    set.insert(Array[i])
  }
  return set.size
}
```

O(N)

O(1)

Complexity

Time Complexity: O(N)

Space Complexity: O(N) HashSet O(N)

Problem Statement

Given arr[N] and K, check if there exists a pair(i, j) such that,

```
arr[i] + arr[j] == K && i != j
```

Example

Let's say we have an array of 9 elements and K, where K is our target sum,

Index	0	1	2	3	4	5	6	7	8
Array	8	9	1	-2	4	5	11	-6	4

6

K = 6: arr[2] + arr[5] : such a pair exists ✓ true

K = 22: does not exist No

K = 8: arr[4] + arr[8]: such a pair exists ✓

Hence if K = `6` or `8` the answer is `true`, for K = `22`, it will be `false`.

Q. [3, 5, 1, 2, 1, 2]

7 true

Q. 3 5 1 2 1 2

false K = 10

ele 3 5 1 2 | 1 2

K=7

pair 4 2 6 5

true ✓

HashSet

3
5 ..... Yes
1

if pair is not found return false

K=8

⇒ 3 1 2 4 3

pair 5 7 6 4 5 No

first check and then add to hashset

3 ✓
1
2
4
~~5~~

false

Elements one by one

Code will work

pair = K - ele

if (pair is present)

return true

set.add(ele)

K = 8

	3	1	2	4	3
pair	5.	7	6	4?	
	No	No	No	Yes	

false positive

4, 4 — not present as a pair.

HashSet

3 ✓
1
2
4
~~3~~

all elements in 1 go.

K = 9.

	0	1	2	3	4	5	6	7	8
ele.	8	9	2	-2	4	5	11	-6	4
pair K-ele	1.	0	7	11	5	4			

Yes ✓

HashSet

8
9
2
-2
4

true.

is the pair

```
boolean targetSum(int[] arr, int K) {
    HashSet<int> bs;
    for (i=0; i<N; i++) {
        pair = K - arr[i]
        if (bs.search(pair) == true)
            return true
        bs.add(arr[i])
    }
    return false
}
```

TC: O(N)

SC = O(N)

Traversal — One by One

Count pairs, whose sum = K

0	1	2	3	4	5
3,	5,	1,	2,	1,	2

$\Rightarrow$

K = 3

4 pairs

2	3	2	5	3	4	4	5
1	2	1	2	2	1	1	2

hashMap < int , int >

element count

Traversal — Elements - One by One

K = 3

ele	3	5	1	2	1	2
pair	0	-2	2	1.	2	1

hm =

ele int	freq/count int
3	1
5	1
1.	~~1~~ 2
2	1

C = 1 + 1 + 2

4 pairs

K = 10

	0	1	2	3	4	5	6	7
ele =	2	5	2	5	8	5	2	8
pair =	8 (10−2)	5	8	5	2	5	8	2

2	~~1~~ ~~2~~ 3
5	~~1~~ ~~2~~ 3
8	~~1~~ 2

0 4 / 2 8   3 5 / 5 5   6 7 / 2 8

2 4 / 2 8   6 4 / 2 8

1 3 / 5 5   0 7 / 2 8

1 5 / 5 5   2 7 / 2 8

9

count = 0 + 1.
+ 2.
+ 2.
+ 1.
+ 3.

9 pairs

$K = 4$

	0	1	2	3	4	5
ele =	2	2	2	2	2	2
pair	2	2	2	2	2	2
	No	Y				

2	~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6

$c = +1 +2 +3 +4 +5$

15 pairs

PseudoCode :

```
int countPairSum(int[] arr, int K) {
    N = arr.length
    HashMap<int, int> hm

    c = 0

    for (i=0  i<N  i++) {
        pair = K - arr[i]
        if (hm.search(K-arr[i])) {
            c = c + hm.get(K-arr[i])
        }
        if (hm.search(arr[i])
            hm.add(arr[i], hm.get(arr[i]) + 1)
        else
            hm.add(arr[i], 1)
    }
    return c
}
```

TC

$O(N)$

SC

$= 2N$

$= O(N)$