Topics

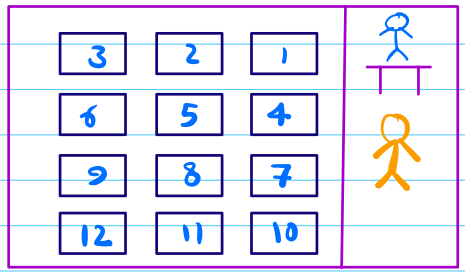
Intro to hashmaps

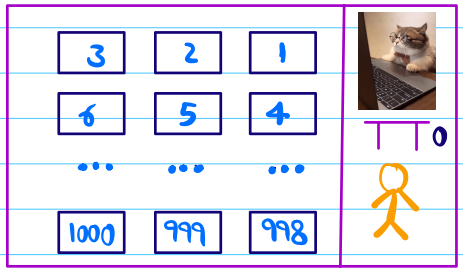
problems

1. Frequency of each number
2. First non-repeating number
3. No. of distinct elements
4. ⚠ No. of subarrays with sum=0

Intro to hashmaps:

hotel rooms example





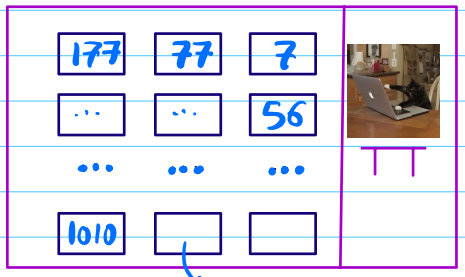
bool

0 1 2 ... 999 1000

O(1) fetch
O(1) update
SC: O(n) for n rooms
TC: search O(n)

not exist

int { -1, 0, 1 }



1000

$\frac{10^3}{10} \cdot 9 = 10^6$

1,000,000,000

0 1 2 3 4 5 6 7 8 ... 77 10^9

O(1) fetch
O(1) update
SC: O(max) for n rooms
TC: search O(max)

hashMap<key, value>

donald knuth

just psudocode

⚠ not syntaticaly correct in any programming language

<77, true>
<177, false>

Storing data in hash map

① store population of every country:

HM<String, long>

HM["India"] = 1,000,000,000
HM["US"] = 300,000,000

---

② # of states/provinces of each country:

HM<string, int> noOfStates["US"] = 30

⋮

---

③ For every country we want to store all state names:

HM<string, list<string>>

---

④ For every country store the population of each state:

HM<string, HM<string, int>>

list

HM["US"]["WA"] → 8,000,000

① keys type: primitive type int, float, double, string, ... → otherwise for complex objects we need to define extra functions so we can use them as key

② values type: any type

hashSet <Key>

Time Complexity

	hash map operations		hashSet operations	
get size		O(1)		O(1)
insert ← put	(key, value)	O(1)	(key)	O(1)
← containsKey	(key)	O(1)	(key)	O(1)
V ← HM[k] ← get	(key)	O(1)	- - - -	
remove	(key)	O(1)	(key)	O(1)
update?	(key)	O(1)	- - - -	

* what if we insert n <key, val> to hash map?

TC: O(n)

SC: O(n)

Hashmap libs in different programming languages:

	Psudo code	Java	C++ STL	python	JS	C#
HM	hash Map	hashMap	unordered_map	Dictionary	Map	Dictionary
HS	hash Set	hashSet	unordered_set	Set	Set	HashSet

P1 Given an array of integers a[N] (N: size), and Q queries find the frequency of each number that is queried

$1 <= N <= 10^5$
$1 <= Q <= 10^5$
$1 <= a[i] <= 10^9$

ex a[] = {2, 6, 3, 8, 2, 8, 2, 3, 8, 10, 6}
(indices 0 1 2 3 4 5 6 7 8 9 10)

Q	freq?
2	3
8	3
3	2
5	0
6	2

} ans

brute force

idea1:

TC: O(Q×N)
SC: O(1)

```
for(q=0; q<Q; q++){
    k = Query[q] ; Count=0
    for(i=0; i<n; i++){
        if(a[i]==k) Count++
    }
    print("k :" + Count)
}
```

idea2:

hashMap<int, int>
(int: number, int: freq)

Quiz

TC: O(N+Q)

SC: O(N)

```
void printFreqForQuery(int a[], int q[]){
    int n = a.Len
    hashMap hm<int,int> = ...
    for(i=0;i<n;i++){                          n
        key = a[i]
        if(!hm.containsKey(key)){              O(n)
            hm[key]=0
        }
        hm[key]++
    }
    for(q=0;q<Q;q++){                          Q
        k = Query[q]                           O(Q)
        if(!hm.containsKey(k)) print("invalid")
        else print(hm[k])
    }
}
```

from left close to index 0

P2 Find the first non_repeating element in a given array

ex a[]: { 1, 2, 3, 1, 2, 5 } (0 1 2 3 4 5) ans=3

Quiz a[]: { 4, 3, 3, 2, 5, 6, 4, 5 } ans=2

Quiz a[]: { 2, 6, 8, 4, 7, 2, 9 } ans=6

ideas: ① freq[ ]

wrong ✗

```
foreach entry<K,V> in freq
    if( V==1) ret k
}
```

HashMap doesn't keep the same order

{2,2,3,3}

② Freq[ ]..

```
for(i=0;i<n;i++){
  if( freq[ a[i] ]==1) ret a[i]
}
```

(a[i] = key)

ret ... (not exist)

O(n+n)

TC: O(n)

SC: O(n)

P3 Given an array a[n], find number of distinct elements:

ex as {3, 5, 6, 5, 4} ans=4

Quiz a{6, 3, 7, 3, 8, 6, 9} ans=5

Quiz a={1, 1, 1, 2, 2} ans=2

ex {3, 3, 3} ans=1

idea:

```
hashSet<int> hs=...
for(i=0;i<n;i++){
   hs.put(a[i])
}
ret hs.size()
```

true/false

P3.2 Given an array a[n], check if all elements are distinct?

ex {6, 8, 3, 2, 7} → true

ex {3, 1, 6, 1, 4, 9, 6} → false

ret hs.size() == arr.len

true/false

P4 Given an array a[n], check if there is a subarray with sum ≥ 0.

ex a[]: { 2, 2, 1, -3, 4, 3, 1, -2, -3, 2 }
(indices 0 1 2 3 4 5 6 7 8 9)

idea 1

```
void sumOfEachSubarray1(int a[]){
  n=a.len
  for(i=0;i<n;i++){
    for(j=i;j<n;j++){        TC: O(n^3)  SC: O(1)
      sum=0
      for(k=i;k<=j;k++){
        sum+=a[k]
      }
      if(sum>=0) ret true
    }
  }
  ret false
}
```

Carry Forward
$O(n^2)$

idea 2

---

ex a[]: { 2, 2, 1, -3, 4, 3, 1, -2, -3, 2 }
(indices 0 1 2 3 4 5 6 7 8 9)

PS[]: { 2, 4, 5, 2, 6, 9, 10, 8, 5, 7 }

case I

obs
PS[2] = 5 → sum[0,2]
PS[8] = 5 → sum[0,8] = sum[0,2] + sum[3,8]
sum[0,8]: 5
sum[0,2]: 5
sum[3,8]: ? 0

special case

a { 2, -5, 3, 6 }
(0)

case II

0 PS { 2, -3, 0, 6 }

Quiz

TC: O(n)

SC: O(n+n)
≈ O(n)

```
bool subArrZero(int a[]){
    n = a.Len
    int[] PS = ...          // PS  O(n)
    hashSet<int> hs = ...

    for(i=0; i<n; i++){
        if(PS[i] == 0) ret true   Case II
        if(hs.contains(PS[i])) ret true
        hs.put(PS[i])
    }
    ret false
}
```