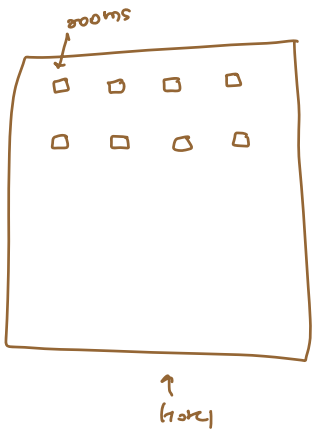
Hashing - 1



100 rooms (Register)

↓

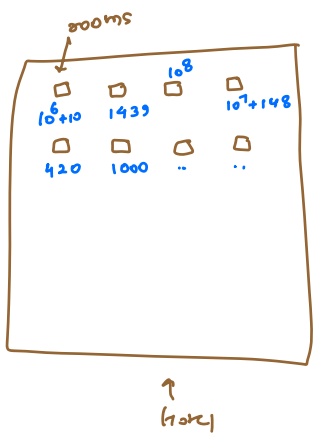
1000 rooms

Boolean arr [1001]

↓

70th room

a[70]



1000 special numbers.

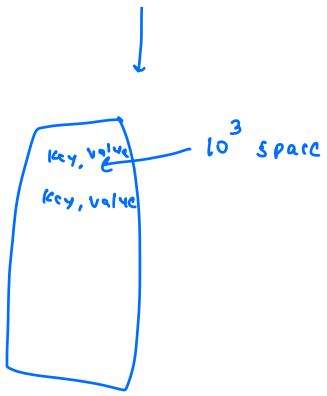
↓

Each number $[1 - 10^9]$

arr $[10^9]$ → 10^9 space

10^2 is actually used

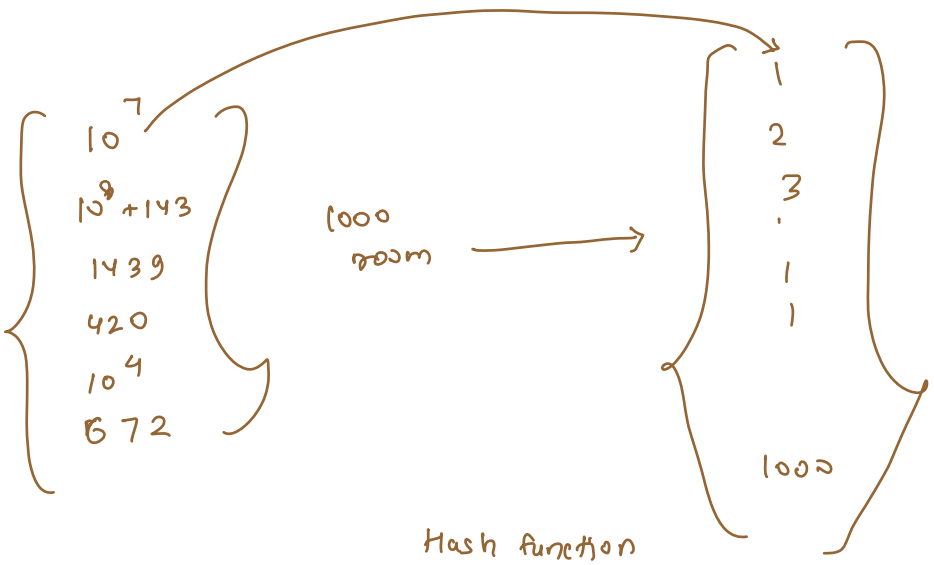
Hashing

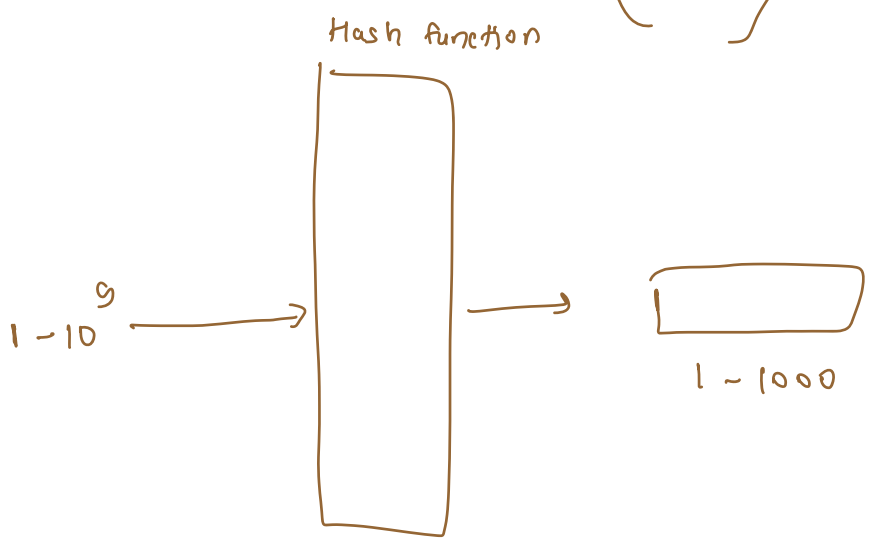


TC: O(1)

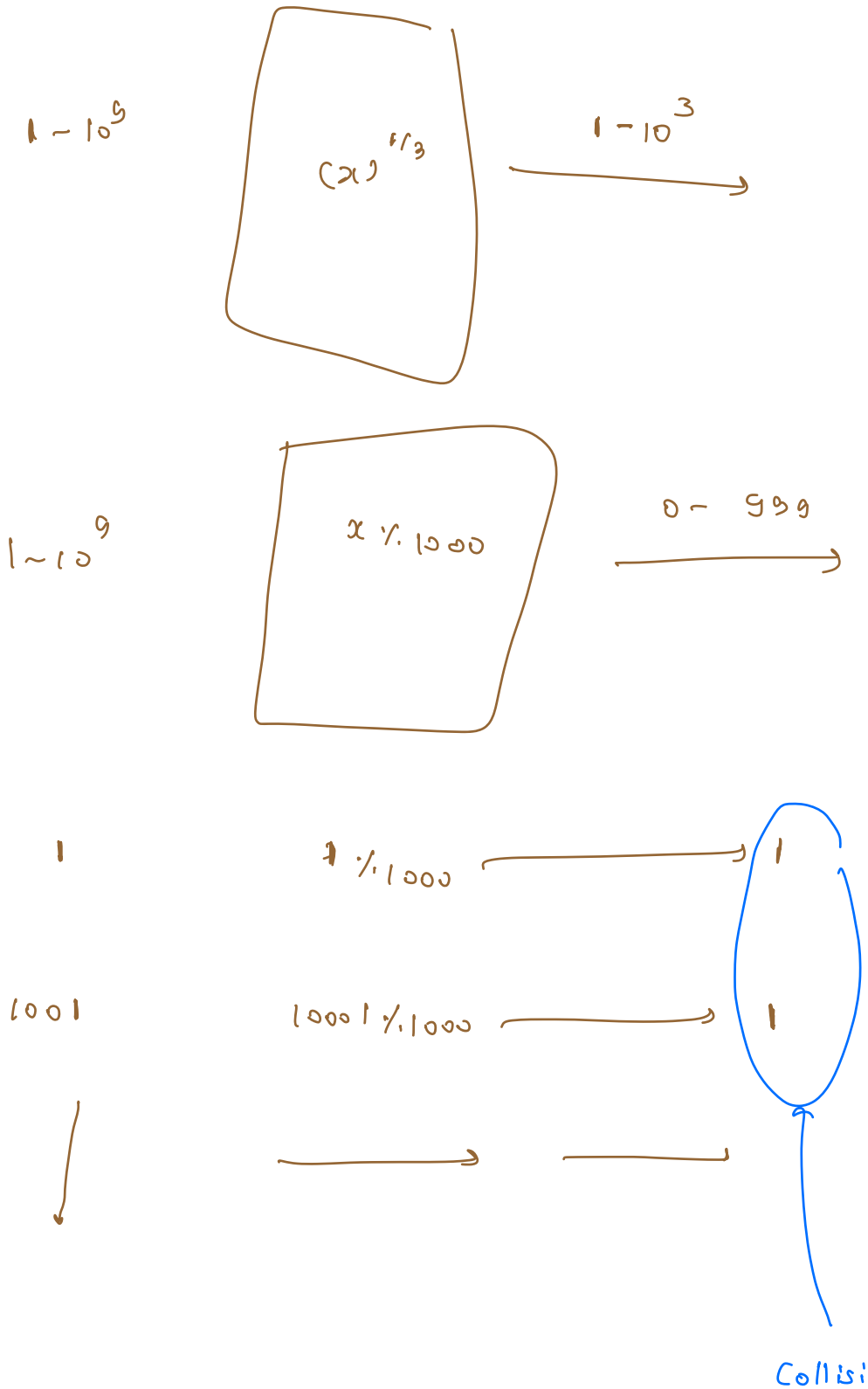
Insertion
deletion
searching

Hashmap
Hashed



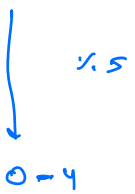


$y = f(x)$


1 - 10^5
(x)^1/3
1 - 10^3
1 ~ 10^9
x % 0.001
0 - 999
1
1 % 1000
1001
1000 / 1000
Collisions

Chaining

A = [2, 5, 10, 7, 3, 4, 7, 12, 17, 12, 1]



0 → 5, 10
1 → 1
2 → 2, 7, 7, 12, 17, 12
3 → 3
4 → 4

worst each operation
↓
TC: O(N)

1. Hash function very good. ST chances of collision is very rare.

10000 values → map [1 - 1000] in range

[1 - 10^9]

101 pigeons

100 holes

Open addressing

TC: O(1) ← Each function in HashFunction

	Avg / TC practical	Theoretically (worst & rare)
Hashmap	O(1)	O(N)
Hashset	O(1)	O(N)

ordered set
ordered map [self-balancing trees]

		Avg	worst
C++ map	Java Treemap	O(log N)	O(log N)
set	Treeset	O(log N)	O(log N)

Problems

$i \neq j$

Q. Given an array[N]. Find pair (i, j) (index)

S.T $A[i] == A[j]$ & $|j-i|$ is minimum

return $|j-i|$

Ex

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

ans	
4	[0, 4]
2	[5, 7] →

ans = 2

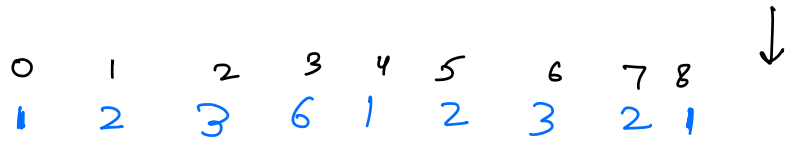
BF Check each pair

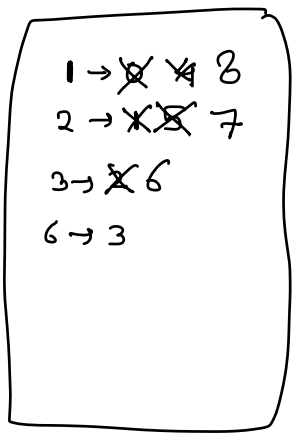
If values are equal

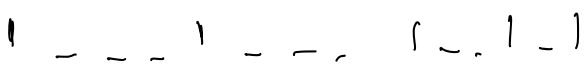
update ans

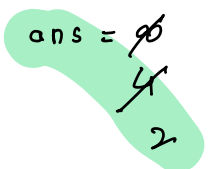
Tc: $O(N^2)$

Sc: $O(1)$









Hashmap < int, int > hm

ans = INT_max

for (i = 0 ; i < N ; i++)

{

if (arr[i] in hm)

{ ans = min (ans, i - hm [arr[i]])

update/add arr[i] → i

}

return ans

TC: O(N)

SC: O(N)

Problems

$i \neq j$

Q. Given an array[N]. Find pair (i, j) (index)

S.T $A[i] == A[j]$ & $|j-i|$ is maximum.

Ex

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

ans = 8 - 0 = 8

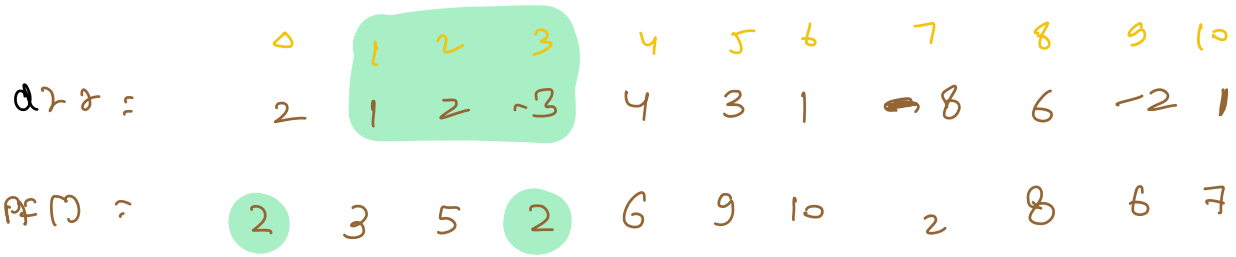
```
                ↓   ↓
Hashmap < int, int > hm

ans = 0

for ( i=0 ; i<N ; i++ )
{
    if ( arr[i] in hm )
    {   ans = max(ans, i - hm[arr[i]])
    }
    else
    {   add  arr[i] → i
    }
}

return ans
```

TC: O(N)
SC: O(N)

Q. Given array. Find longest subarray with

sum = 0.

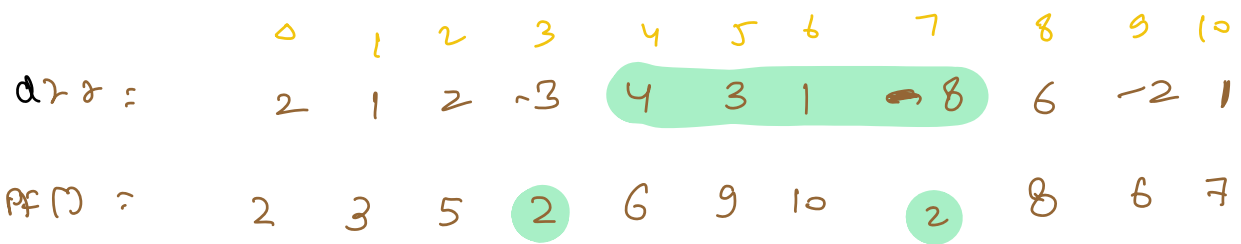
return len of subarray.

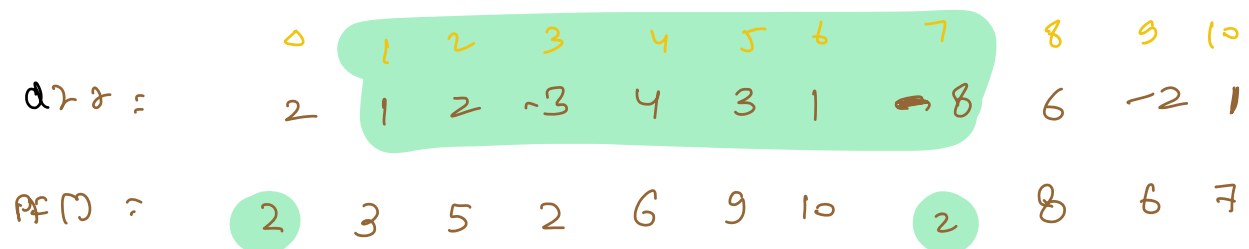


$$sum[L-R] = PF[R] - PF[L-1] = 0$$

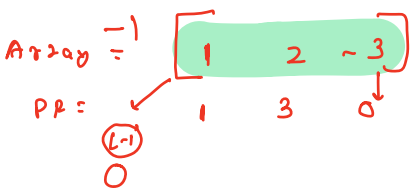
$$\Downarrow$$

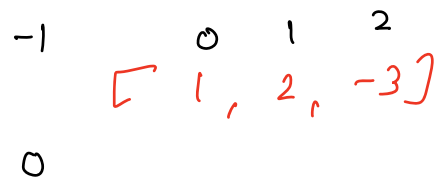
$$PF[R] = PF[L-1]$$





previous question





↓ ↓

Hashmap < int, int > hm

hm.add { 0, -1 }

ans = 0

for (i = 0 ; i < N ; i++)

{

if (~~arr~~[i] PF in hm)

{ ans = max(ans, i - hm [~~arr~~[i] PF])

else

{ add ~~arr~~[i] PF → i

}

return ans

TC: O(N)

SC: O(N)

Break

8:31 8:35

Q. Given a array[N]. Find length of longest subsequence which can be reordered into consecutive elements

arr[] : 101, 4, 3, 6, 10, 20, 11, 5, 100

ans = 4

100, 101 → 100, 101 (2)

10, 11 → 10, 11 (2)

20 → 20 (1)

4, 3, 6, 5 → 3, 4, 5, 6 (4)

Idea 1: Sort the array

3, 4, 5, 6, 10, 11, 20, 100, 101

4 2 1 2 ans = 4

TC : O(N log N + N) : O(N log N)

SC : O(1)

Idea 2 : add Hashmap

arr[] : 101, 4, 3, 6, 10, 20, 11, 5, 100

101, 4, 3, 6, 10, 20, 11, 5, 100

101 → x

4 → 5, 6 → x

3 → 4, 5, 6, → x

6 → x

10 → 11, x

20 → x

11 → x

5 → 6, x

100 → 101, x

1

3

4 → ans = 4

1

2

1

1

2

2

TC: O(N²)

SC: O(N)

↓

hs

arr[] : 101, 4, 3, 6, 10, 20, 11, 5, 100 ↓

101, 4,
3, 6, 10,
20, 11, 5,
100

101 → X (1)

4 → X (1)

3 → 4, 5, 6, (4) → ans = 4

6 → X (1)

10 → 11 (2)

20 → X (1)

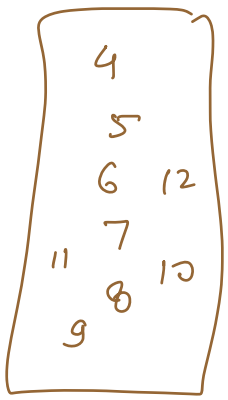
11 → X

5 → X (1)

100 → 101, X (2)

TC: O(N²) SC: O(N) ⟶ TC: O(N²) SC: O(N)

arr = 4, 4, 4, 4, 4, 4, 4, 4, 5, 6, 7, 8, 9, 10, 11, 12



4 → 5, 6, 7, 8, 9, 10, 11, 12

4 → 5, 6, 7, 8, 9, 10, 11, 12

4 →

4 →

4 →

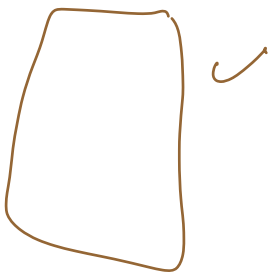
4

4

Array ⟶ array

(without duplicate)

Put all element in hs



ans = 0

for (iterate of hs.keySet())

x

if (x - 1 is not in hs)

cnt = 1

curr = x

while (curr+1 in hs)

{ cnt++
curr++

ans = max(ans, cnt)

else

nothing

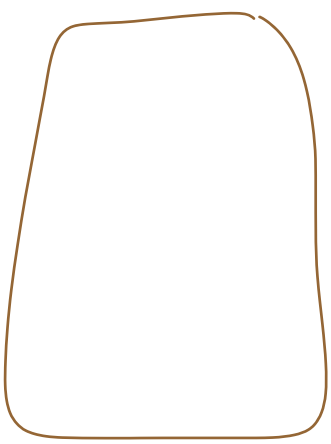
return ans

TC : O(N + N) : O(N)

SC : O(N)

4 → 5 → 6 → 7 → 8 → x

3 → 4 → 5 → 6 → 7 → 8 → x



4 interview

45 mins + 15 min

DSA

Sooglyness