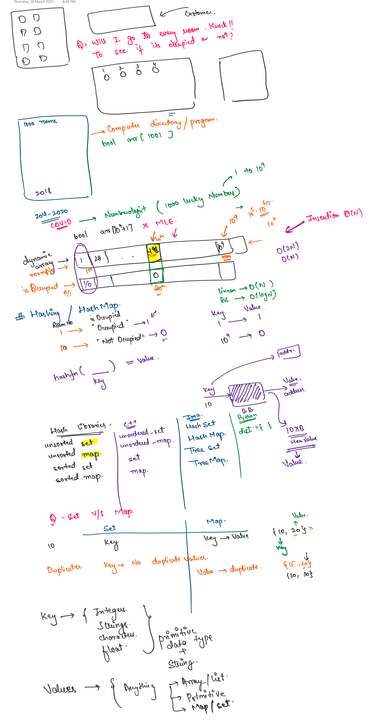

TC (searching) →

ordered - set/map (ordered) → O(log N) — why? → Implementation.

unordered set/map (not ordered) → O(1)

log N

Q Given an array, find no. of pairs where sum is present in the array.

A: [2 4 2 6 7]

Set: 2.

BF: 1. Take 2 loops for i & j → N^2

2. Check the sum of a[i] + a[j] = sum

Set → [2 4 5] O(N)

3. Iterate over the array & check if that element (sum) is present or not. → N

$O(N^2)$

C++:

unordered_set <int> s

for(i=0; i< A.size; i++)

s.insert(A[i]);

found=0

for(i) 0 to A.size()

for(j) i+1 to A.size()

if(s.find(A[i]+A[j]) != s.end())

found++;

Map.

key → Value.

arr[i] → 1

5 → 0

(arrs not sorted)

s.find(key) → Address/Area

→ Not present. (s.end())

Java.

HashSet<Integer> s = new HashSet<>();

for(i) 0 to A.length.

s.add (A[i])

found=0

for(i) 0 to A.size.

for(j) i+1 to A.size.

if(s.contains(A[i]+A[j]))

found++;

return found

Q - Find the first repeating elements: → O(N) N=10^5 266893

A: [8 3 7 5 12 18 5 3] 8 → set → O(1)

Repeating element is the one with least index.

BF Take every element & check for its occurance in rest of the array (after that index).

O(N²)

Key	Value
8	0
3	1
7	2
5	3
12	4
18	5

map[key] → arr[i] → Value

5

Ans. 0

O(N)

{18, 5, 3}

key list → Place

Key

"1·2·6·16" & "1·2·6·16" → String Value

String key

dynamic array →

110

100 101 102 103

111

20

O(N)

Insertion O(N)

10

1 + 1 + 1 + ... + N

O(1) × N

O(1)

N + N

2N

$= \frac{2N}{2M} = O\left(\frac{3}{2}\right)$

N

O(1)

O(1)

N = 1000

O(N)

1000

2000 → 1 times O(N)

all others O(1)

2000 → O(N) ⇒ 1 time

O(1) → (2000-1) times

O(1)

Hash Map / Sets

↳ collisions

address

collision.

key

2 diff key

Open Hashing.

Closed Hashing