algorithms

Sort( )

- lower_bound
- upper_bound
- binary search

] sorted arrays or vectors

containers

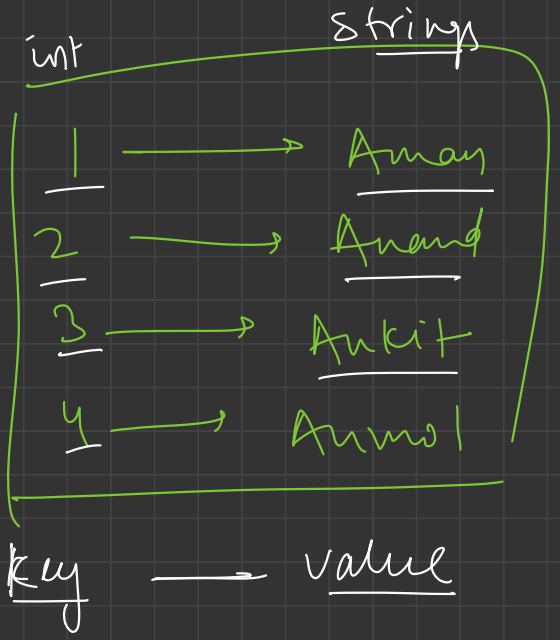
array

1. Vector
2. String
3. Map
4. Set
5. Pair

] unordered multi

6. Stack
7. Queue
8. Priority_queue (Heap)

Map

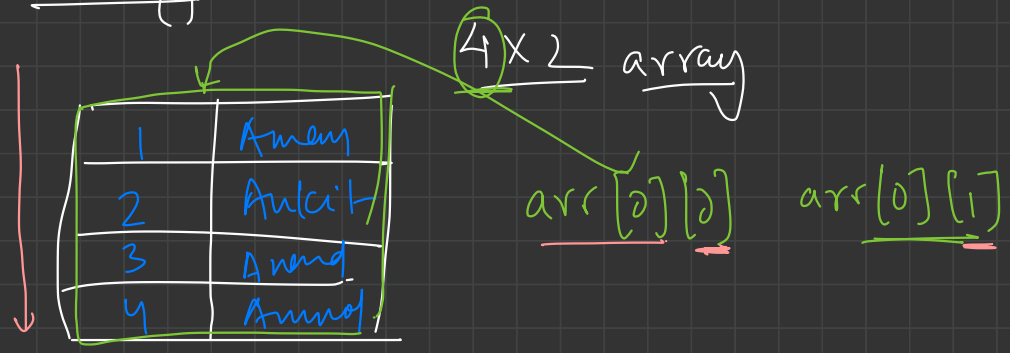


Phone book

[Name → Phone number.

[Aadhar number → Demographic details

2D array



	Sorted	Unsorted	≈ Map (STL)
Search	O(log n) Binary Search	O(n) linear	O(log n)
insert	O(n) shifting of elements	O(1) push_back	O(log n)
delete	O(n)	pop_back → O(1) O(n)	O(log n)

read/iterate O(n) O(n) O(n)

Self-balancing Binary Search Trees

Container: with no duplicat

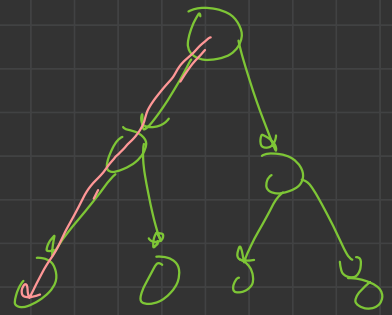
	Sort	unsort-	Set
search	O(logn)	O(n)	O(logn)
insert	O(n)	O(n)	O(logn)
delete	O(n)	O(n)	O(logn)
iterate	O(n)	O(n)	O(n)

first need to search for duplicate

Iterators are pointers used to point at the memory of elements of STL containers.

unique
Rooted
efficient

map
set



List:

Iterators: to iterate over a container

↳ begin()
↳ end()

for(it = m.begin(); it != m.end(); it++)

↳ valid for any container (MT)

vector<int>

vector<int>::iterator it;

CPP

Ordered_map

Ordered_set

Java

TreeMap

TreeSet

C++

unordered_map

unordered_set

Java

HashMap

HashSet

→ keys are not sorted

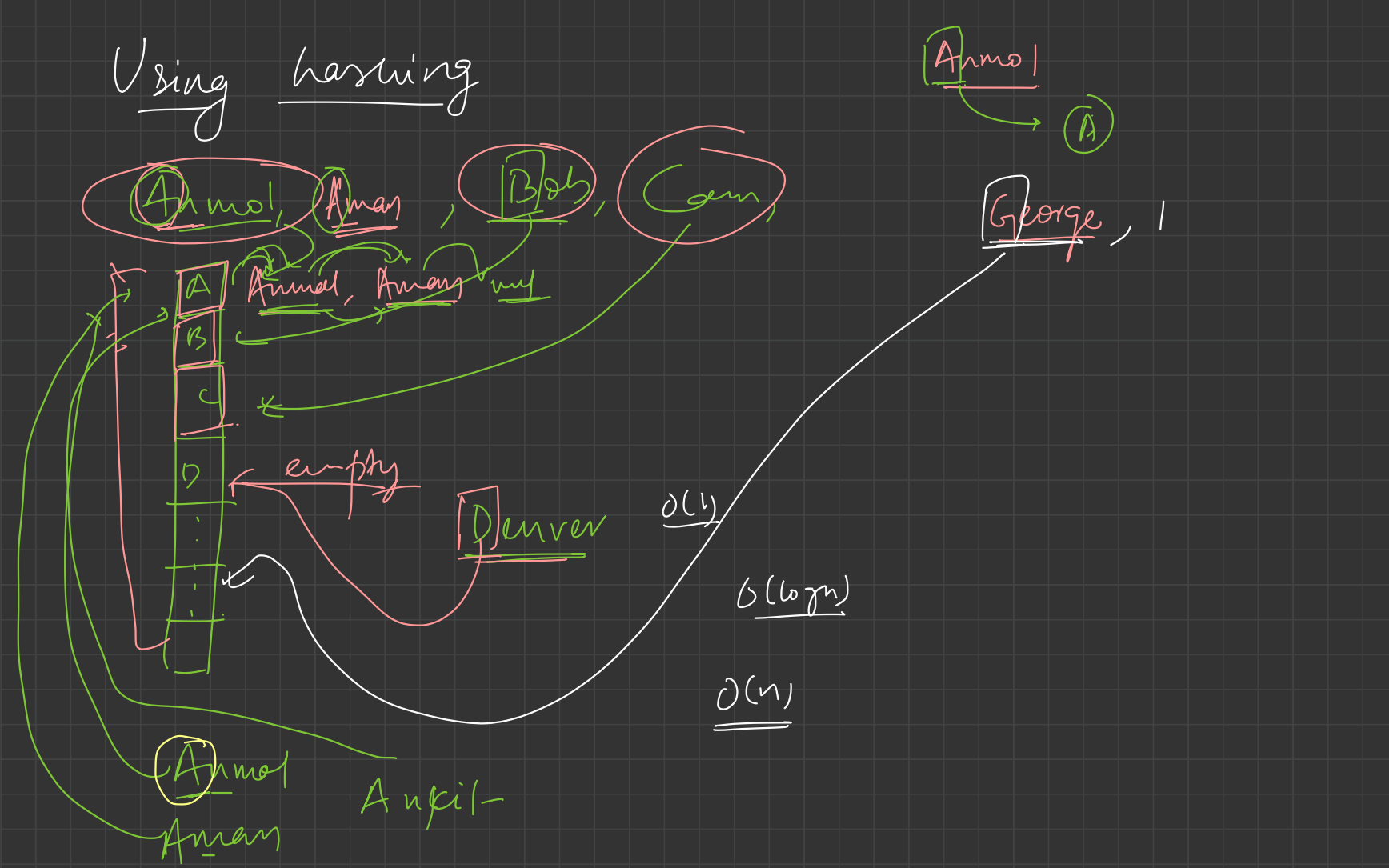
keys are unique

map (ordered)

O(log n)

insertion
deletion
search
→ O(1)

Using hashing



Hash chaining

O(1) → O(n²)

Hash algorithm

very distributive

64/128

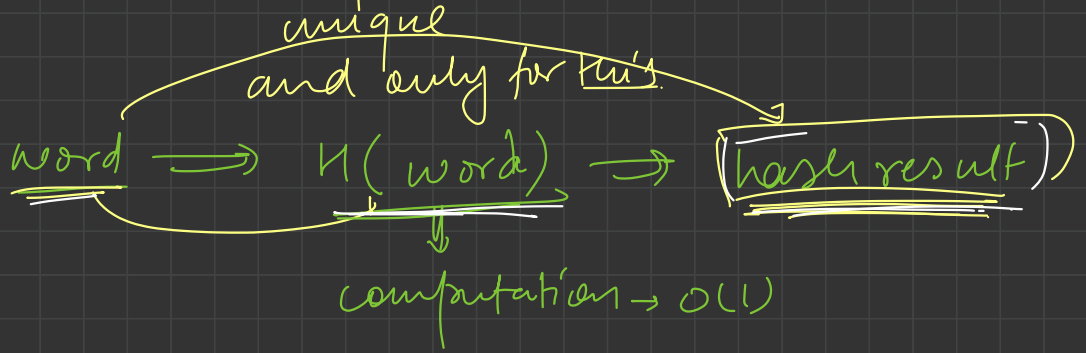
MD5 break

2^{256}

256-bit

SHA-256

2^{256}



"Anmol" ⇒ getfirstChar(Anmol) ⇒ A

linear: array, linked list, stacks queue

What is the fastest searching algorithm!?

any linear data structure : O(n)

sorted " : O(log n)

any structure : O(1)

with O(n) pre-comput: making hashmap

[~~x~~multimap →]
[~~x~~multiset →] } O(log n) tree

- uniqueness X
- can have duplicates
- sorted

Linked List : STL list