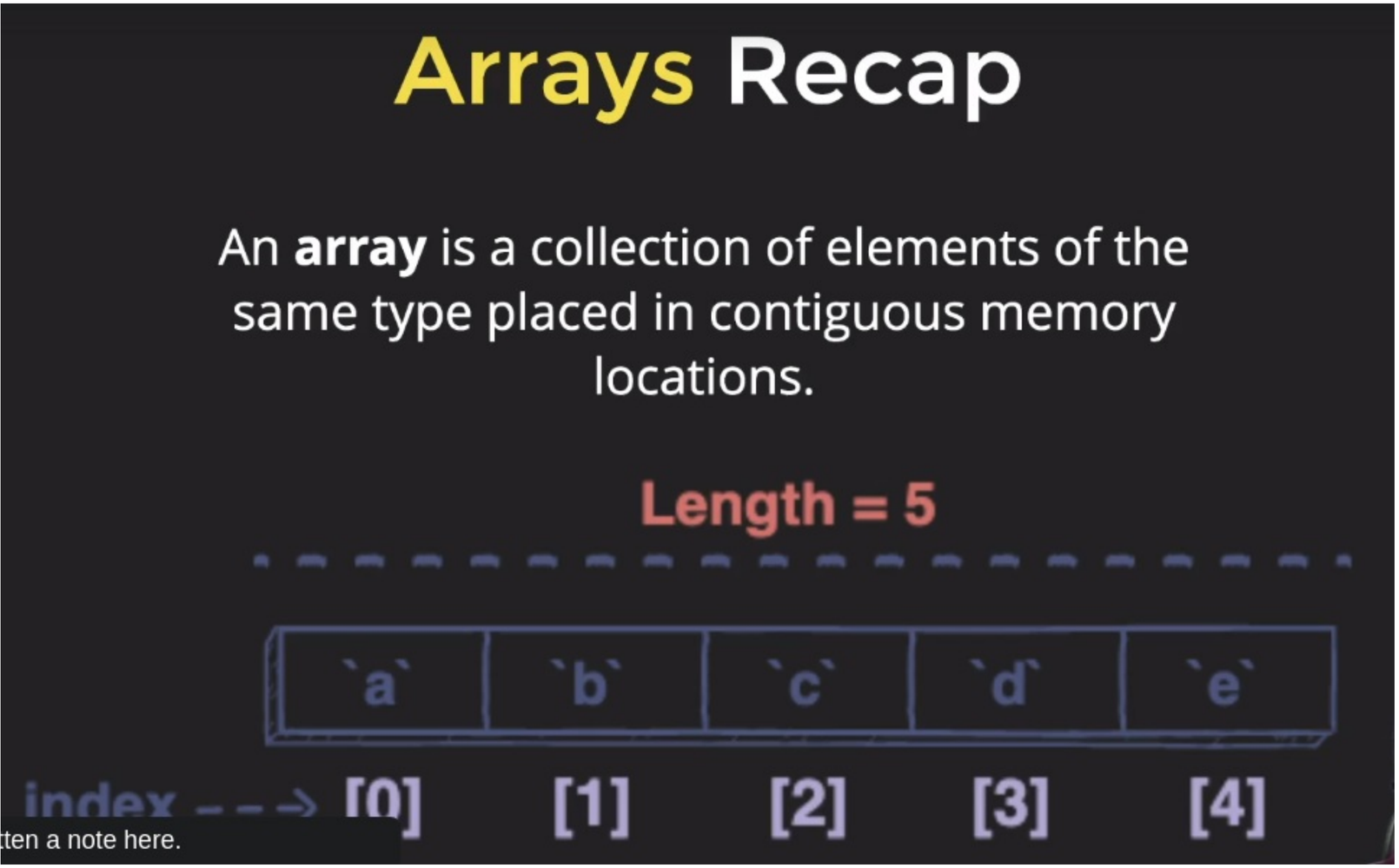
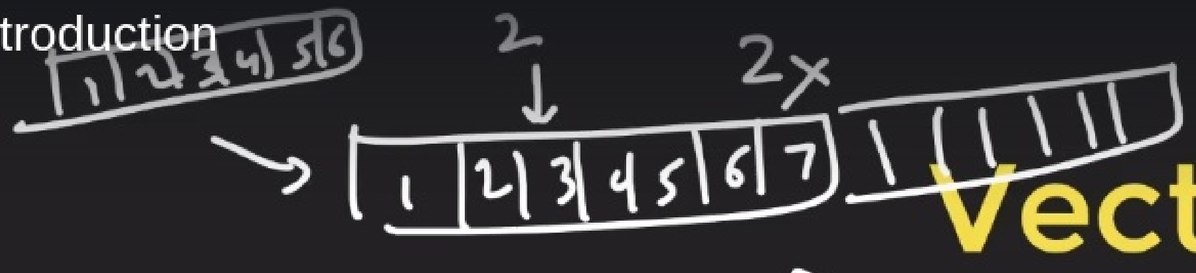
Arrays Recap

An **array** is a collection of elements of the same type placed in contiguous memory locations.



tten a note here.

```
1  //Init an Array
2
3  int a[100];
4
5  int a[100] = {0};
6
7  int a[100] = {1,2,3};
8
9  int a[] = {1,2,3};
10
11 string fruits[4] = {"apple","mango","guava"}
```



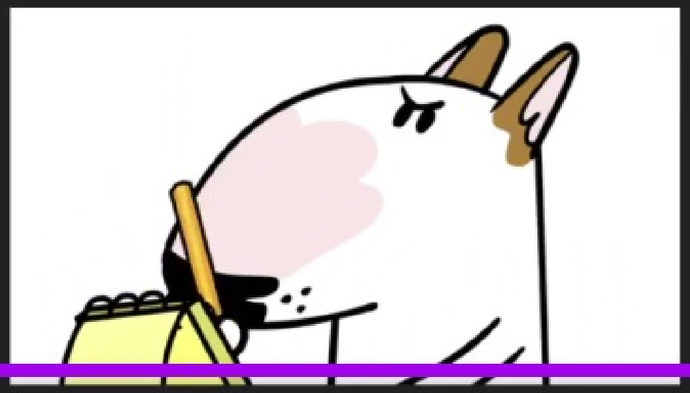
- Vectors are sequence containers representing arrays that can change in size.
- Just like arrays, vectors use contiguous storage locations for their elements, which means that their elements can also be accessed directly in O(1) time.
- But unlike arrays, their size can change dynamically, with their storage being handled automatically by the container.

Vector changes size ∵ capacity :- Doubles when full.

```
void fn (vector<int> &v) {

}
```

Points to Note

- Vectors can be used to create dynamic **1D & 2D Arrays**
- We will be using vectors throughout this course in place of a regular array.
- Unlike arrays, vectors are passed by value to a function, you may can still pass them by reference if need arises.



7. Vector STL - Demo

CODING MINUTES

```cpp
#include<iostream>
#include<vector>
using namespace std;

int main(){
    //Demo Vector
    vector<int> arr = {1,2,10,12,15};

    // Push_Back O(1)
    arr.push_back(16);

    //Print all the elements
    for(int i=0;i< arr.size(); i++){
        cout << arr[i] <<endl;
    }


    //Size of the vector (No of elements) present currently.
    cout<<arr.size() <<endl;

    //Capacity of the vector
    cout << arr.capacity() <<endl;
```

// Initially size is zero if no elements are added.

// adds element to the end of array (right side)

] → printing.

→ actual elements ko store karne ki capacity

Here, cap → 10
size → 6

[Finished in 0.5s]

1.75× 3:27 / 5:23

```cpp
vector<int> arr = {1,2,10,12,15};

//Pop_back
arr.pop_back();
// Push_Back O(1)
arr.push_back(16);

//Print all the elements
for(int i=0;i< arr.size(); i++){
    cout << arr[i] <<endl;
}
```

→ Removes the last element

```cpp
//Fill Constructor
vector<int> arr(10,7);
```

→ Creates a vector of size = 10 and fills every elem with val → 7

o/p → 7 7 7 7 7 7 7 7 7 7

8. Vector of Vector - Demo

{ Vector of Vector ⇒ 2D vector }

```cpp
#include<iostream>
#include<vector>
using namespace std;


int main(){
    //2D Vector
    vector< vector<int> > arr =  {
        {1,2,3},
        {4,5,6},
        {7,8,9,10},
        {11,12}};

    arr[0][0] += 10; //update


    for(int i=0;i< arr.size(); i++){

        for(int number : arr[i]){
            cout << number;
        }
        cout <<endl;


    }

    //Print all the elements
    for(int x : arr){
        cout << x <<",";
    }
```

Note:- We're having different no. of elements unlike in arrays.

arr[0] ←

→ for each loop

→ Similarly for 1D-arrays!

A Note on Sorting!

Inbuilt Sort Function

Some problems may require the use of a sorting algorithm, and I hope you are already familiar with basic sorting algorithms like Bubble Sort, Selection Sort, Insertion Sort which have `O(n^2)` complexity. Better algorithms include Merge Sort, Quick Sort and Heap Sort which have `O(nLogn)` complexity.

For most array problems we can use the inbuilt sort function which also offers `O(nlogn)` complexity. This is how you can use the inbuilt sort function in case you are not aware of. Also depending upon the data in the problem, you may also require Counting Sort, or Bucket Sort or Radix Sort sometimes. We will discuss more problems in Sorting & Searching section.

Sort an array containing N integers

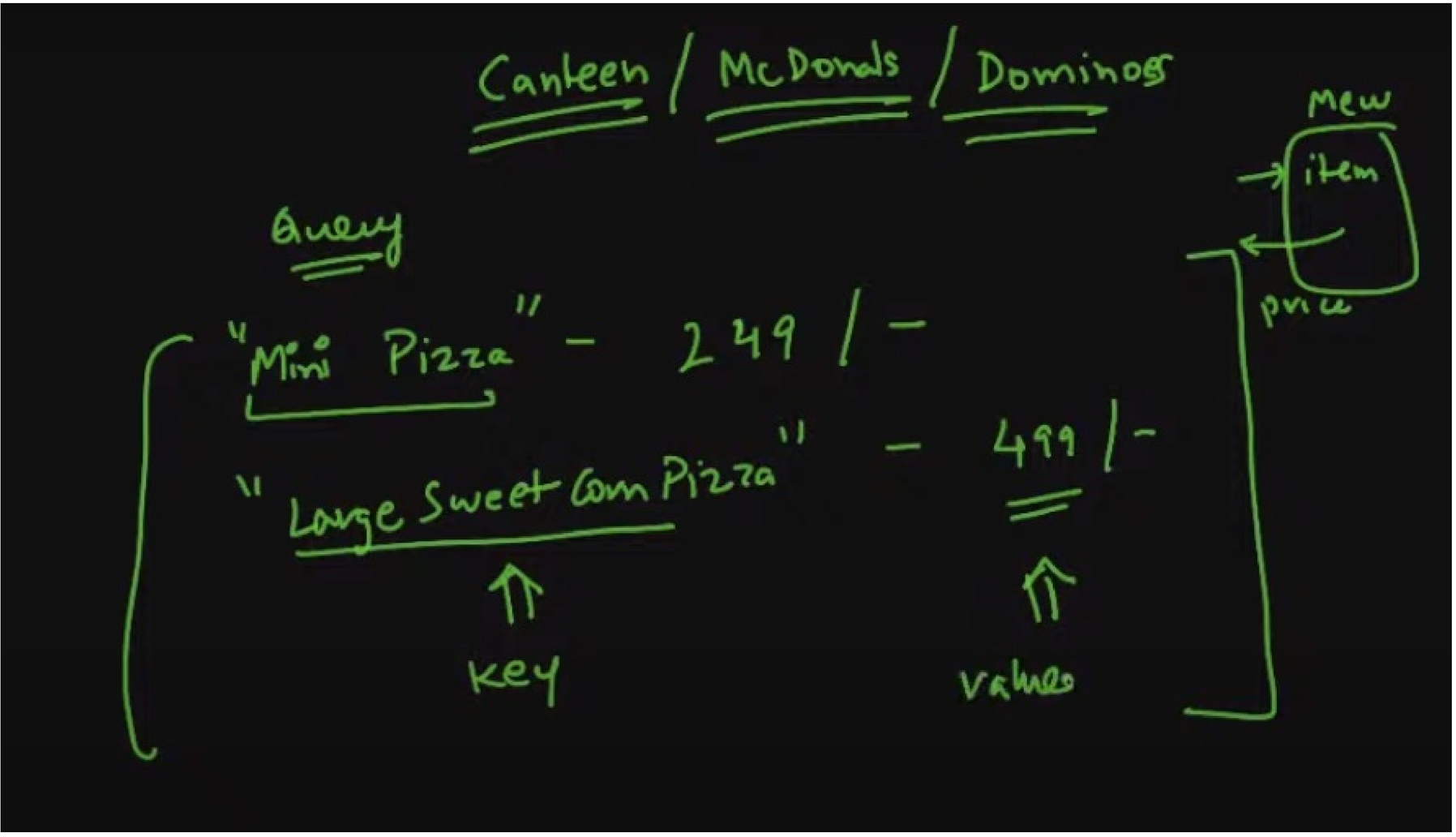
```
sort(arr, arr +n)
```

Sort an vector containing N integers

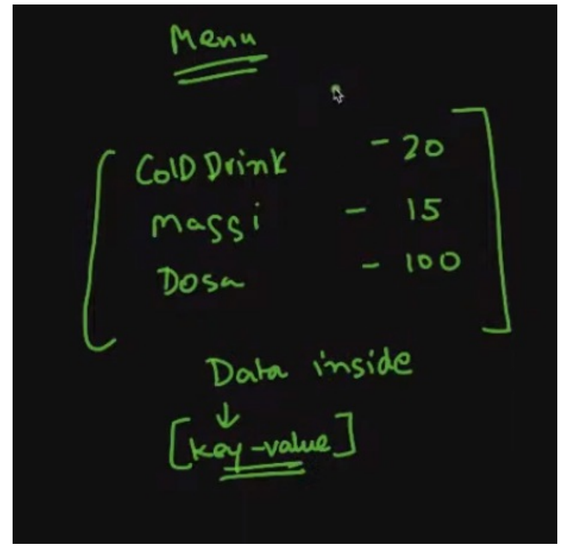
```
sort(arr.begin(), arr.end())
```

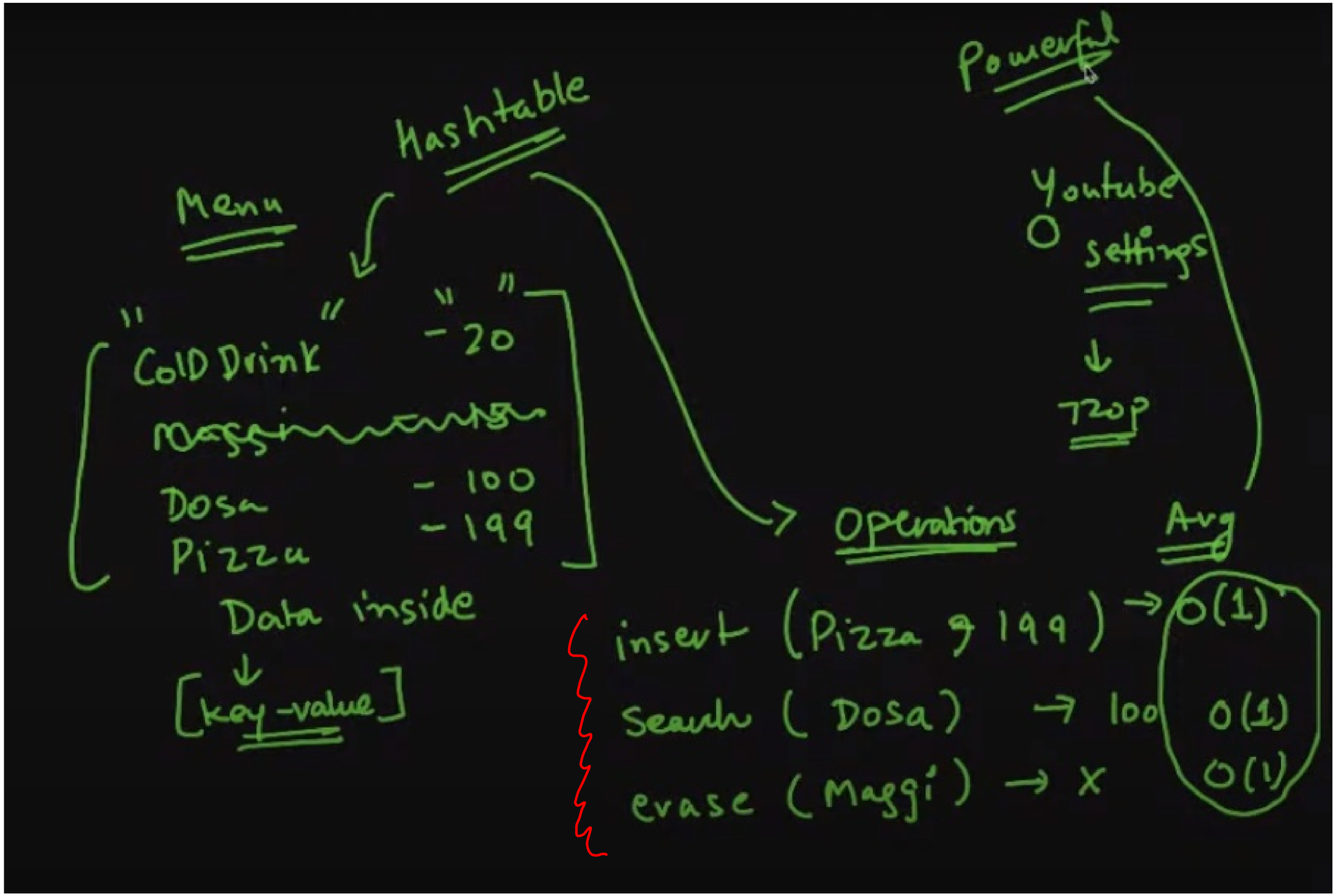
Hashtables, Sets and Maps

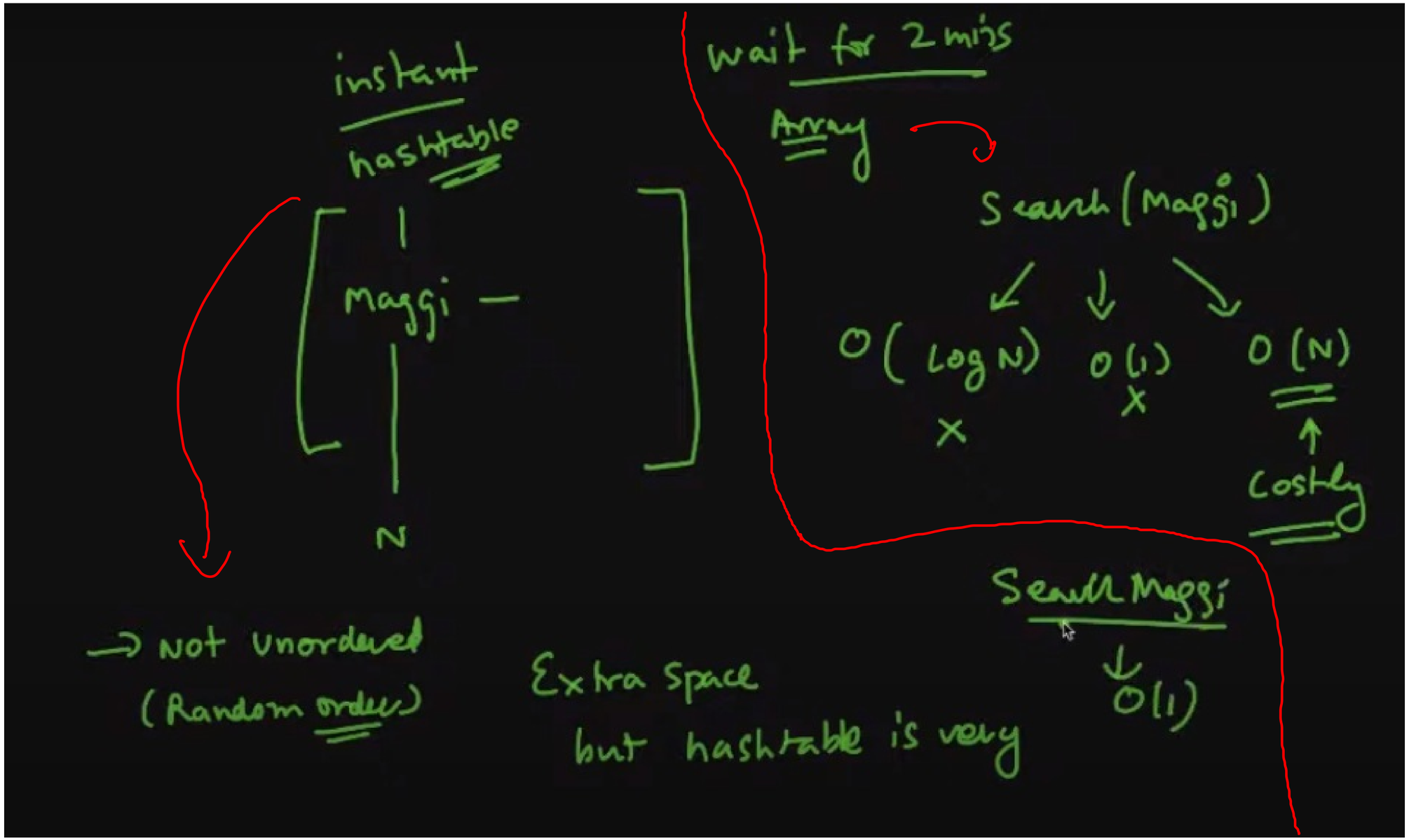
Hashtable is a key-value type DSS storage



Query is being done to person standing at reception.

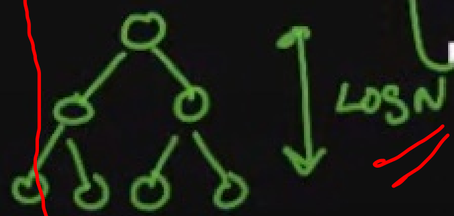



Hashtable
Menu
" "
" "
Cold Drink
- 20
Maggi
- 100
Dosa
- 199
Pizza
Data inside
[key-value]
Powerful
Youtube
settings
720p
Operations
Avg
insert (Pizza , 199) → O(1)
Search (Dosa) → 100 O(1)
erase (Maggi) → x O(1)


instant
hashtable
1
Maggi
N
→ not Unordered
(Random order)
Extra space
but hashtable is very
wait for 2 mins
Array
Search (Maggi)
O(Log N)
O(1)
O(N)
Costly
Search Maggi
O(1)

Container Types?

- Maps & Unordered_Maps
- Sets & Unordered_sets
- Multimaps & Unordered Multimaps
- These containers are used in -
  - Array Problems
  - Linked Lists Problems
  - Sliding Window
  - Graphs



Key-value

hashtable in C++

O(1) on avg

Keys

Balanced BSTs

O(LogN)

LogN

Dijkstra's

Splitwise

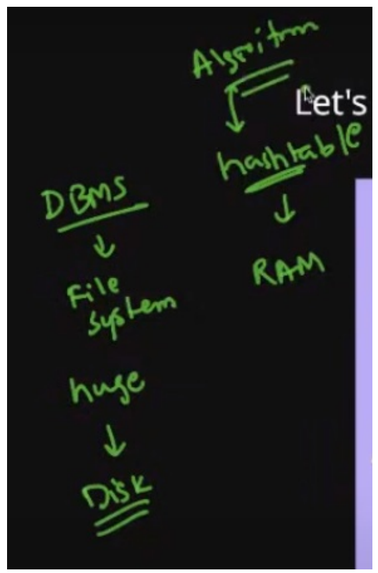
Unordered Maps

Key-Value

Unordered sets

Keys only

Both these are simplest



```cpp
#include<iostream>
#include<unordered_map>
using namespace std;


int main(){


    unordered_map<string,int>  menu;

    //Insert Key Value pairs insdie th
    menu["maggi"] = 15;
    menu["colddrink"] = 20;
    menu["dosa"] = 99;

    cout<< menu["dosa"] <<
    //Search
    string item;
    cin>>item;

    if(menu.count(item)==0){
        cout<<item<<" is not available";
    }
    else{
        cout<<item <<" is available, and its cost is " << menu[]
    }
```

Include

declaration

Pair {
first → key
second → value

This is

em] <<endl;

Very

Count > 0 if item is present

```cpp
//Deletion
menu.erase("dosa");
```

```cpp
//update query
menu["dosa"] = (1 + 0.1)*menu["dosa"];
```

O(1)
↳ avg case
99.1%

in for each loop.

```
//we can iterate over all the key-values pair that are present
for(pair<string,int>  item: menu ){
    cout<<item.first <<" - " <<item.second<<endl;
}
```

Left side me are things jo ki return karega right wala.

inbuilt class in C++

Pair {
first → key
second → value
}

```
em] <<endl;
```

```
Hashing — -zsh — 80×24
prateeknarang@192 Hashing % ./example_hashtable
dosa
dosa is not available
indianplatter - 399
pizza - 499
colddrink - 20
maggi - 15

prateeknarang@192 Hashing %
```

4 items O(N)

ORDER ???

Random → unordered_map

Internally
↓
(hashing with Separate chaining)

```cpp
#include<iostream>
#include<map>
using namespace std;


int main(){


    map<string,int>  menu;


    //Insert Key Value pairs insdie the hashtable O(1) operation
    menu["maggi"] = 15;
    menu["colddrink"] = 20;
    menu["dosa"] = 99;
    menu["maggi"] = 15;
    menu["pizza"] = 499;
    menu["indianplatter"] = 399;
```

O(1) → O(LogN)

Sorting is done for keys and not values.

```
prateeknarang@192 Hashing % ./example_map
dosa
dosa is not available
colddrink - 20
indianplatter - 399
maggi - 15
pizza - 499

prateeknarang@192 Hashing %
```

"Keys" ordered ?

↓

Lexicographic sorted order

- If we want order we aage sab toh use <map>
- Everything remains same just operations same O(1) → O(logN) ho jayenge
- ∵ internally BST se implement kiya hai map!.

```
not available
atter - 399
499
k - 20
15

arang@192 Hashing % ./example_map

not available
k - 20
atter - 399
15
499

arang@192 Hashing %
```

Phonebook

A → (Vi 98100) (Jio 99100)
B → (99) ( ) (0181---)
C →
D
⋮

Print ⇒ Data Container

C → Key →
Value →

Ques:- Phonebook banani ho jisme DS use here?
Ek bande pass multiple mobile nos ho sakte.

Ans:- use map (if names are unique)

↳ map < string, vector <int> phoki;

vector <int>
or
list <int>

or
vector <string>
∴ 4 + 9 + ... 4

cycle_detect.cpp | example_hashtable.cpp | example_map.cpp | phonebook.cpp

```cpp
#include<map>
#include<list>
#include<string>
using namespace std;


int main(){

    map<string, list<string> > phonebook;

    phonebook["Prateek"].push_back("1111111");
    phonebook["Prateek"].push_back("1111112");
    phonebook["Prateek"].push_back("1111113");

    phonebook["Krishna"].push_back("2311111");
    phonebook["Krishna"].push_back("2411112");
    phonebook["Krishna"].push_back("2511113");
```

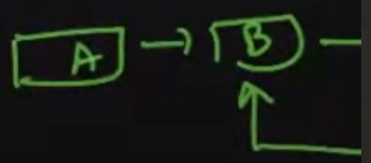
Note :- key is same to value pair jake append karege

Hashtable

Node {
int data;
Node * next;

Example-I 🤦 3

Write a function to cl
contains a cycle!




```cpp
bool containsCycle(node *head){
    //Complete this function using hashtable
    unordered_map<node*,bool> hashtable;

    node  temp = head;
    while(temp!=NULL){

        //check if temp already exists in the hashtable
        if(hashtable.count(temp)!=0){
            return true;
        }
        //insert in the hashtable
        hashtable[temp] = true;
        temp = temp->next;
    }
    return false;
}
```

My approach - Since addresses are distinct
we'll treat them as keys.

same

CORRECT
2

, bool ? Hashtable;

And as an alternative approach we can use Floyd's algo
↳ 2-pointer approach
fast and slow one

Downside of using maps instead of floyd is that it'll take up extra space!

```cpp
bool containsCycle(node *head){
    //Complete this function using hashtable
    unordered_map<node*,bool> hashtable;

    node  temp = head;
    while(temp!=NULL){

        //check if temp already exists in the hashtable
        if(hashtable.count(temp)!=0){
            return true;
        }
        //insert in the hashtable
        hashtable[temp] = true;
        temp = temp->next;
    }
    return false;
}
```

→ if key already there then return found cycle

→ tab bhi nikali so pakka nahi hai

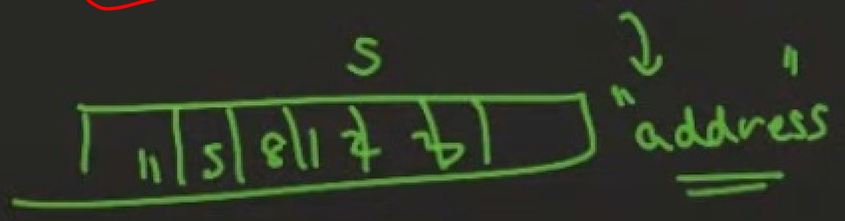
```cpp
using namespace std;


int main(){


    //set is just a collection of keys

    unordered_set<int> s{1,2,3,8,11,15,0};

    int key;
    cin>>key;

    if(s.find(key)!=s.end()){
        cout<<key <<"is present";
    }
    else{
        cout<<"Not found";
    }



    return 0;
```

array → O(N)
sorted array → O(log N)

O(1) time

Awesome → Array me search karna hota toh O(N) logy jaate!
→ soft karke bhi O(logN) but here O(1)

s.end()



#include <unordered_set>

s.end() → return some address (not exactly the last elem of a set but beyond that).

```cpp
//set is just a collection of keys

unordered_set<int> s{1,2,3,8,11,15,0};

int key;
cin>>key;

if(s.find(key)!=s.end()){
    cout<<key <<"is present";
}
else{
    cout<<"Not found";
}

//let us print all the elements of unordered set
for(auto x : s){
    cout << x <<","<<endl;
}
```

Printing of a set

```cpp
//erase O(1)
s.erase(11);
```

→ erase(11) is delete

```cpp
//insert
s.insert(111);
```

auto taaki kisi bhi return type wala value le...

Example-II

Pair Sum Problem

[2, 1, 8, 6, -2, 5, 3, 0]

⇒ Print pairs whose sum is equal to K.

Possible ~~Appln~~ Approaches

1.
2.
3.

1:06:44 / 1:31:48

Approach 1:- Brute force : note PAIR chahiye to two for loops lagake lagayega $O(N^2)$
agar sum equals k toh print else continue...

Approach 2:- Sort the array and use two pointer approach. $O(N \log N)$ ⟵ sorting

[-2, 0, 1, 2, 3, 5, 6, 6]

Suppose sum is k → 6 → -2 + 6 = 6] ⇒ less than expected to slowly move karenge increase left ptr.

0 + 6 → 6 // print
1 + 6 → 7 7 > 6 → decrease right ptr
1 + 6 → 7 7 < 6 → increase left ptr
6 + 2 → 6 // print
3 + 6 → 9 9 > 6 → decrease right
3 + 5 → 6 // print.

Ans:- {0,6}, {6,2}, {3,5}

Note:- left wala ptr increase karo jab pair milliye kuch aisa hai nahi but just bahar steps

Approach 3:- Sorting + Binary Search.

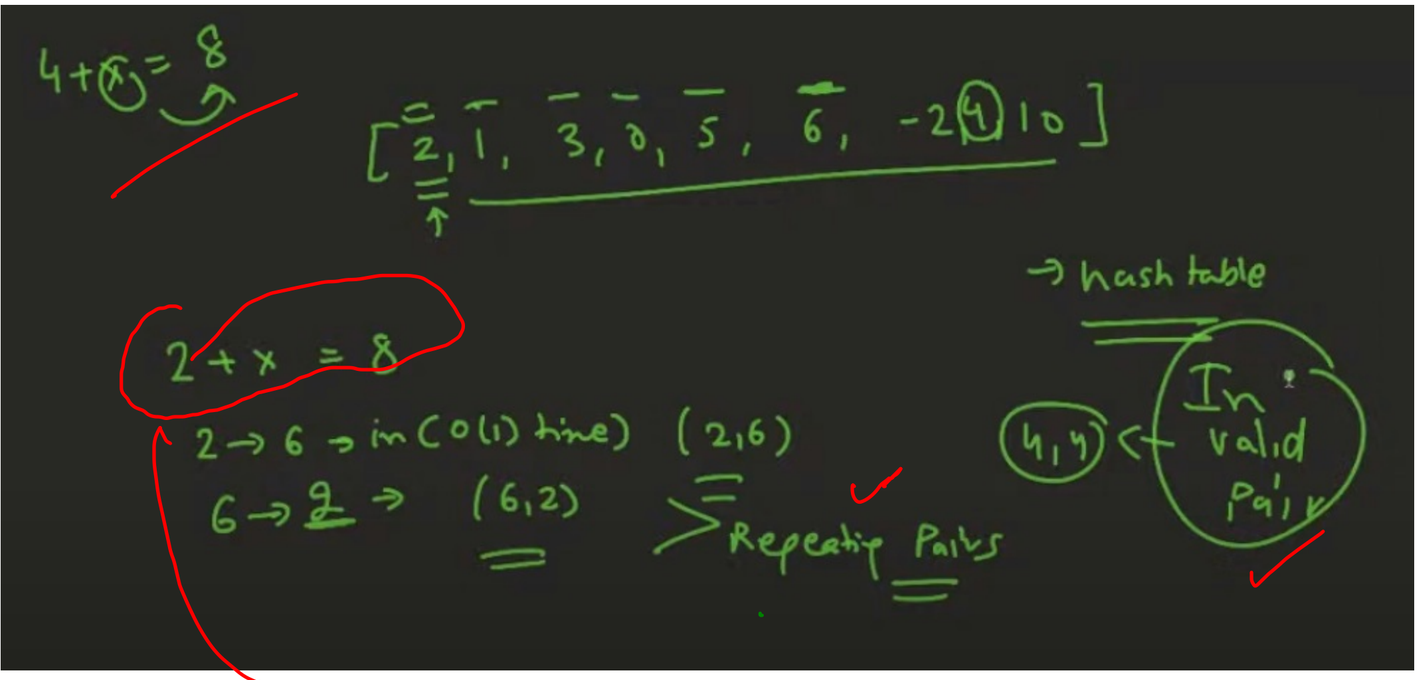
Suppose k = 8 and left ptr is -2 toh

BS se search for (10) :: -2 + 10 = 8

[-2, 0, 1, 2, 3, 5, 6, 8]

Approach:- Solve using Hashing (Best approach) (Use unordered set)

O(N)



Logic formula used:-

Note:-

Agar sharp me hi dual dig save elements for repetition ho sakti hai humare soln me

∴ (6,2) and (2,6) are both valid pairs and jisme hi unique chaige

Soln:- do hashing as you go - To save order hi hi hoga

```cpp
void pairSum(vector<int> arr,int Sum){
    //Logic
    unordered_set<int> s;


    for(int i=0;i<arr.size();i++){

        int x = Sum - arr[i];
        if(s.find(x)!=s.end()){
            cout<<x <<"-"<<arr[i]<<endl;
        }
        //insert the current no inside set
        s.insert(arr[i]);
    }

}
```

finding an element jisko sum me hatake result bach jaye

Pause

agar vo already set me hai to bachiya pair mil gaya

ni to hai ni present elem ko to daal hi do

```cpp
vector<int> arr{10, 5, 2 , 3, -6, 9 , 11};
```

input

→ { 10, 5, 2, 3, -6, 9, 11 }

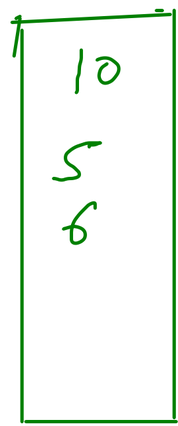
$x = 8 - 10 \to -2$ ✗

$x = 8 - 5 \to 5$ ✗

$x = 8 - 2 \to 6$ ✗

$x = 8 - 3 \to 5$ ✓ found ∴ sol"

↳ $x \to 5$
arr[i] $\to 3$



```cpp
void pairSum(vector<int> arr,int Sum){
    //Logic
    unordered_set<int> s;


    for(int i=0;i<arr.size();i++){

        int x = Sum - arr[i];
        if(s.find(x)!=s.end()){
            cout<<x <<"-"<<arr[i]<<endl;
        }
        //insert the current no inside set
        s.insert(arr[i]);
    }

}

int main(){
```

O(N)

O(1)

Complexity

O(N)

O(1)

OVERALL COMPLEXITY !!