Why hashing?

The concept of hashing was Adopted to make Searching, Insertion, and deletion more time efficient Basically in O(1)...

finding a (unique index) for all element to store them yahi to hai hashing

→ hashValue

Ek Example dekho:

[682, 761, 494, 567, 869]

Suppose ki humko (682) ko ek Array me put karna hai...

Ek primitive Approach ye ho sakti hai ki hum 0-999 size tak ka Array Banale And fir Array * 682 index per 682 ko insert kar de AB we can check for 682 at index 682... And yeo (1) time complexiety. me bhi ho jayega.

Problem :-

Ab suppose humko is Array me 8210882557... store Karna hua to ab kya kya AB 9999999999 Itna Bada Array Banana hoga ye to Bahut Space Ka wastage cause karega As itna Bada Array to banaliye But fill to bas 4-5 element hi ho raha hai na...

Q. What is HASHING?

⇒ [682, 761, 494, 567, 869]

Ek Tariqa of finding hash value...

Number % 10 Jo number aaye usko usi Index per insert karo and yeah all set!

↳ HASH VALUE

Hash Table

0	
1	761
2	682
3	
4	494
5	
6	
7	567
8	
9	869

[682, 761, 494, 567, 869]

Hash Value ⇒ 2 1 4 7 9

h(K) = K % 10

↳ Hash function to determine hash value...

Now After insertion

Search(761)

↳

to search for 761

Pehle find its hash value 761 % 10 == 1

Now we have to check only 1st index

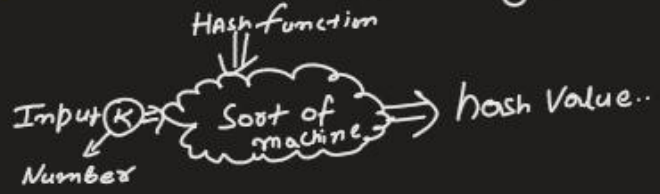
to search for 761 in the whole Array

O(1) time complexity me kaam Khatam...

Same Approach to delete...

Q. What are hash functions?

Ans: Hash value nikalne k liye function.



Different method :- (DIVISION METHOD)

$h(k) = k \bmod m$ → Bucket...

for example..... $m = 11$

$1276 \bmod 11 = 0$

Isiko hash value bolte hai And isi index par wo would store (1276)

2.) Mid Square method :- $h(k) = k^2$ & extract middle digit..

eg. $k = 60$

$h(k) = k^2 = (60)^2 = 3600$ → 60 is the middle 60th index par 60 store karo

eg2) $k = 13$

$h(k) = k^2 = (13)^2 = 169$ 6th index par store (13).

3. Digit folding method :-

fold Key into equal size parts :-

$K_1\ K_2\ K_3\ K_4\ K_5$

$K_1 K_2$

$+ K_3 K_4$

$+ K_5$

↳ HashValue

Example :-

$K = 12345$

$12 + 34 + 5 = 51$

51 ↓ HASH VALUE

4) Multiplication method :

$h(k) = floor(M(KA \% 1))$

M → Size of hash table...

$0 < A < 1$

eg. $K = 12345$ $M = 10$

$A = 0.01$

$K \times A = 123.45$

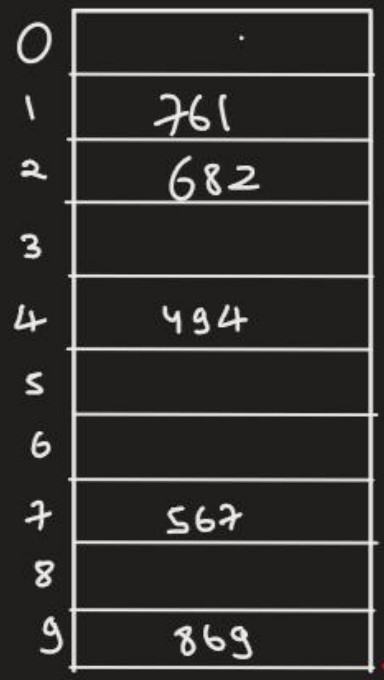
$KA \bmod 1 = (12345 \bmod 1) = 0.45$

↳ $\times M = 0.45 \times 10 = 4.5$

$floor(M(KA \bmod 1)) = floor(4.5) = 4$

↳ Greatest Possible Integral Value...

COLLISIONS:



hash value

[682, 761, 494, 567, 869, 634]

2 1 4 7 9

↳ New Number we are trying to insert in the data structure

$h(k) = k \% 10$

Step 1) hash value of 634

$h(634) = k \% 10 \Rightarrow 634 \% 10 = 4$

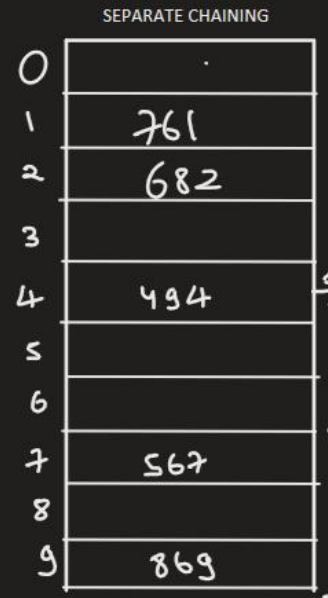
But 4th index per to already ek value hai

This situation is called collision...

Collision handling techniques:-

Collision handling techniques:-

1) OPEN HASHING: (Close Addressing)



[682, 761, 494, 567, 869, 634]

hash value: 2 1 4 7 9

By Linked list...

634 → 794

$h(k) = k \bmod 10$

$634 \% 10 = 4$

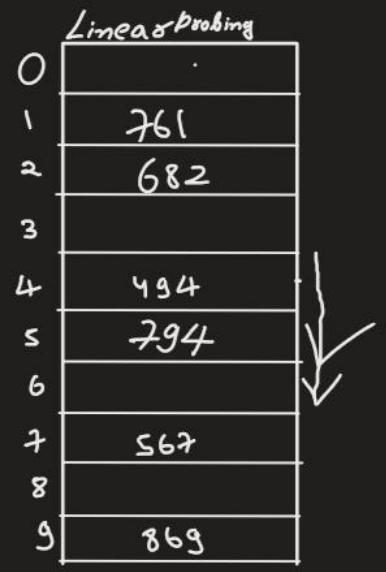
Similarly 794 AAya...

Now After list is created the time complexiety required to search for that particular element would be O(length of the linked list)...

Note: Agar collision hua to linked list k Sahayta se element ko insert KARO...

PROBLEM IN USING THIS METHOD : HO SAKTA HAI KI BOHOT SE NUMBERS KA HASH VALUE SAME AAYE TO EK LAMBI SI LINKED LIST BAN JAYEGI JISKE karan searching bhi tough ho jayegi means usi complexiety badh jayegi saath hi saath bahut saare buckets khali reh jayege

Closed Hashing (OPEN Addressing)



[682, 761, 494, 567, 869, 634]
2 1 4 7 9

M → size of HASH TABLE

$h(k) = k \bmod 10$

$at(h(k)+i) \bmod 10 \quad 0 \le i \le 9$

$h(794) = (4+0) \bmod 10 = 4$ → Already 494 hai waha

$= (4+1) \bmod 10 = 5$ → Abhi koi nahi hai waha per to insert kardo...

for 634 $at(h(k)+i) \bmod 10 \quad 0 \le i \le 9$

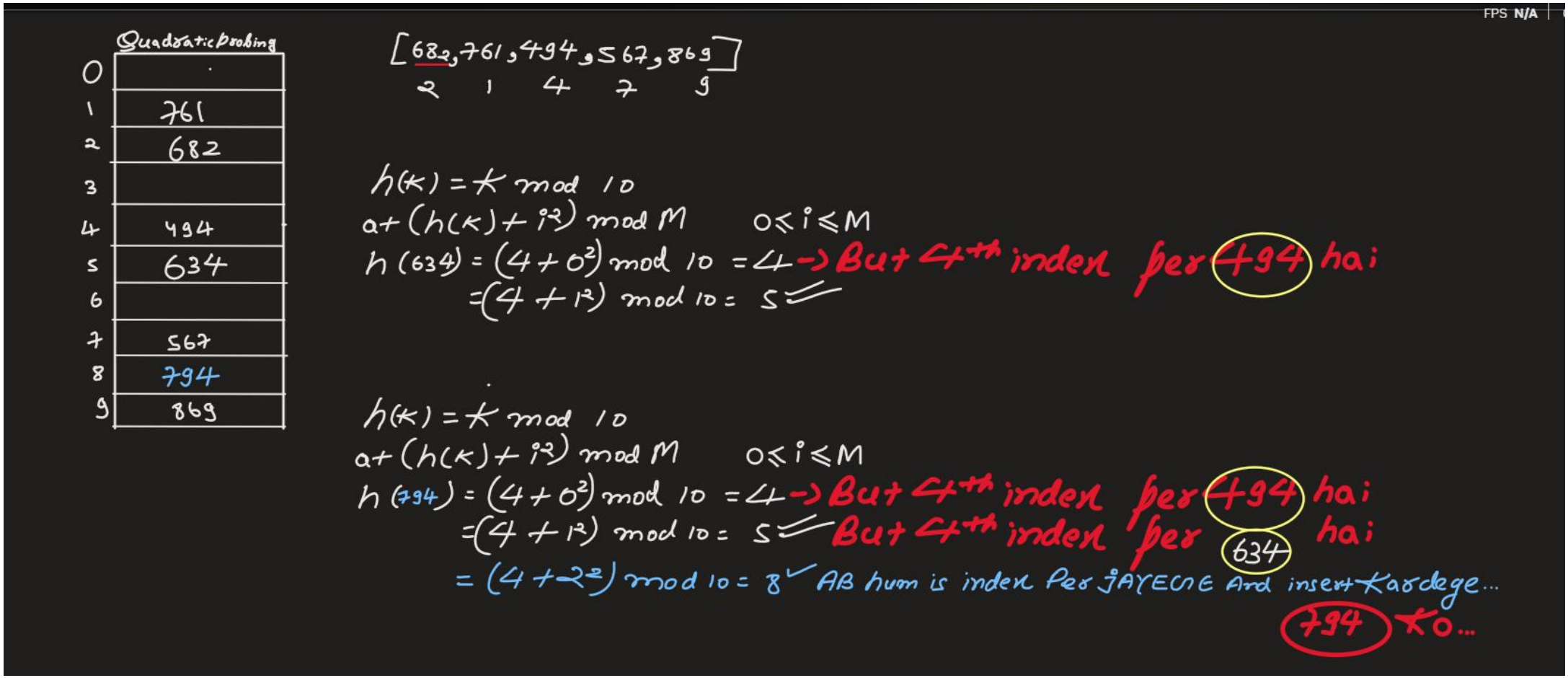
$h(634) = (4+0) \bmod 10 = 4$ → Ye index Khali Nahi hai...

$\Rightarrow (4+1) \bmod 10 = 5$ → Ye index Khali Nahi hai...

$\Rightarrow (4+2) \bmod 10 = 6$ → Ye index Abhi Khali hai yaha Bhardo...

Limitation dekh hi le na AB Maan le 784 insert KARNA HOGA To KAHA Bharega!

Cluster wala issue AARAHA HAI...


Quadratic probing
0
1 761
2 682
3
4 494
5 634
6
7 567
8 794
9 869
[682, 761, 494, 567, 869]
2 1 4 7 9
h(k) = k mod 10
a + (h(k) + i²) mod M 0 ≤ i ≤ M
h(634) = (4 + 0²) mod 10 = 4 -> But 4th index per 494 hai
= (4 + 1²) mod 10 = 5
h(k) = k mod 10
a + (h(k) + i²) mod M 0 ≤ i ≤ M
h(794) = (4 + 0²) mod 10 = 4 -> But 4th index per 494 hai
= (4 + 1²) mod 10 = 5 But 4th index per 634 hai
= (4 + 2²) mod 10 = 8 AB hum is index per JAYENGE And insert karenge...
794 ko...

Closed Hashing (Open Addressing)

SKILLS

Double Hashing

0	
1	761
2	682
3	
4	494
5	
6	
7	567
8	634
9	869

682, 761, 494, 567, 869, 634

$h_2(k) = k \bmod 5$

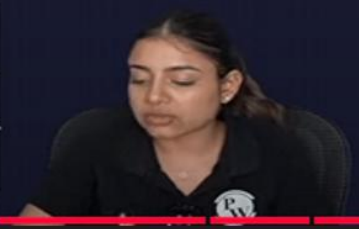
$h_1(k) = k \bmod 10$

store at $(h_1(k) + i \cdot h_2(k)) \bmod M$ $\quad 0 \le i < M$

$h(634) = h_1(634) + 0 \cdot 4$

$= [h_1(634) + 1 \times h_2(634)] \bmod 10$

$= [4 + 1 \times 4] \bmod 10 = 8$



COLLEGE WALLAH

LOAD FACTOR :

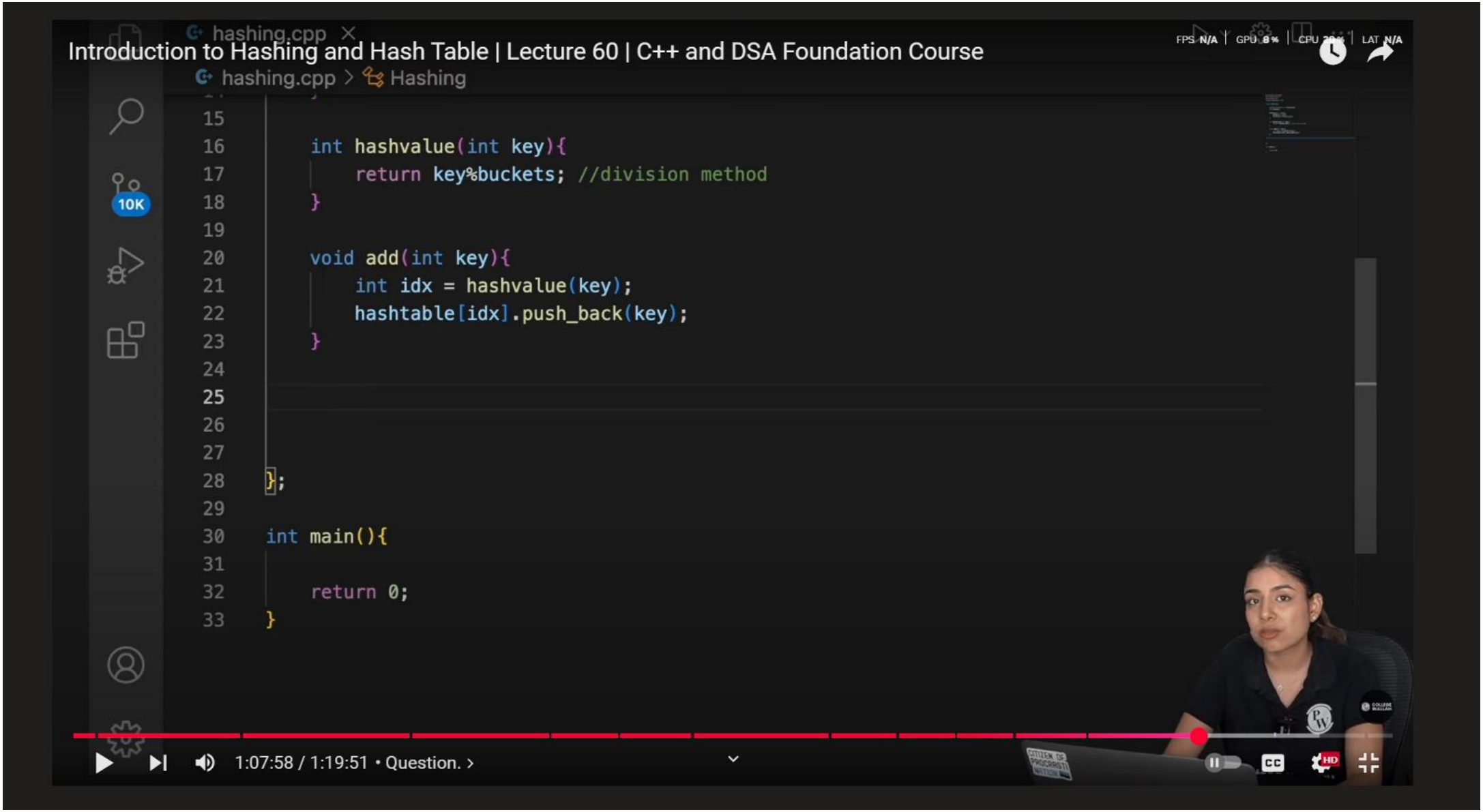
n = no. of elements
m = no of buckets
Load factor = n/m
↳ Average entities in one bucket

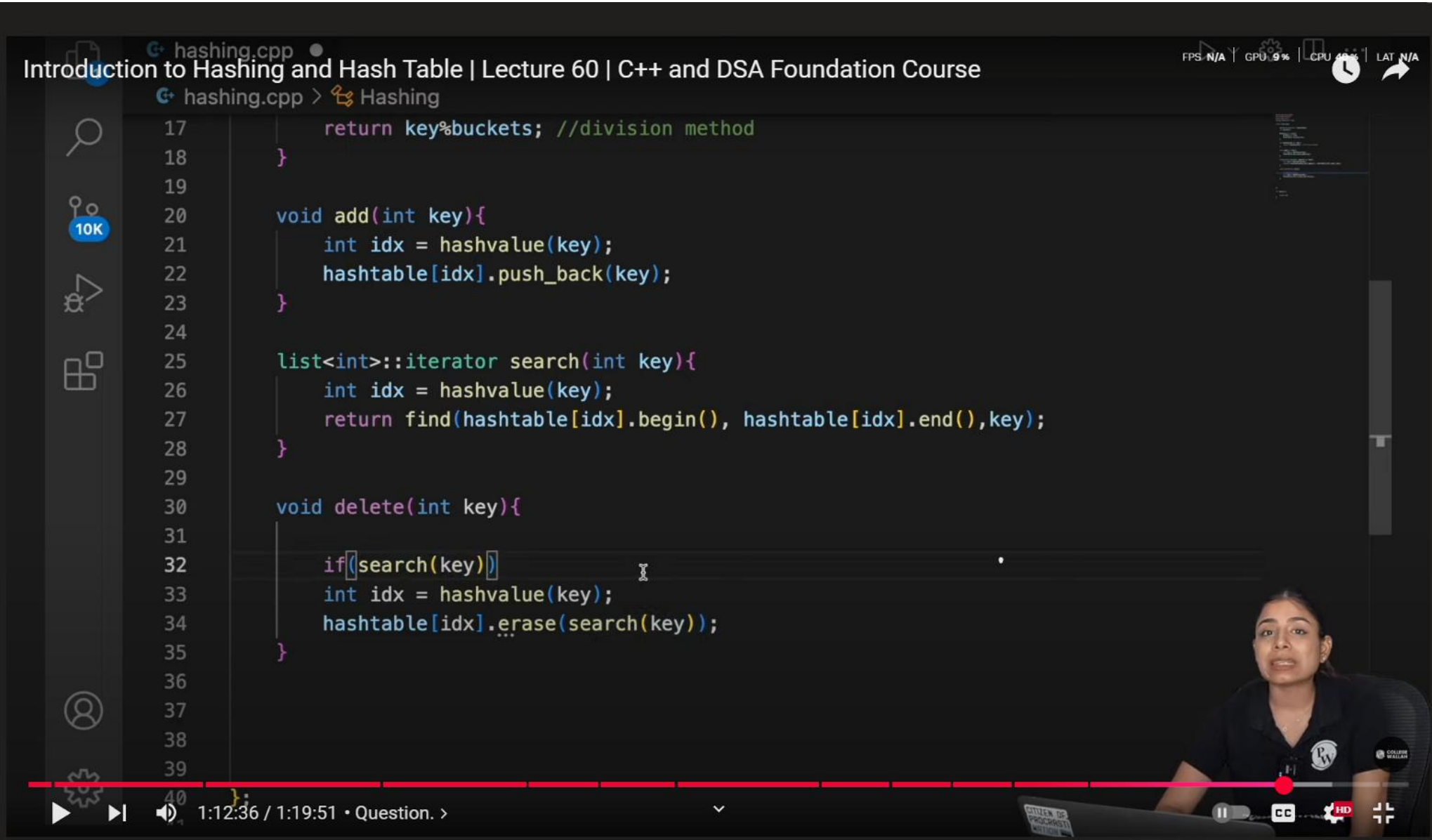
Rehashing...

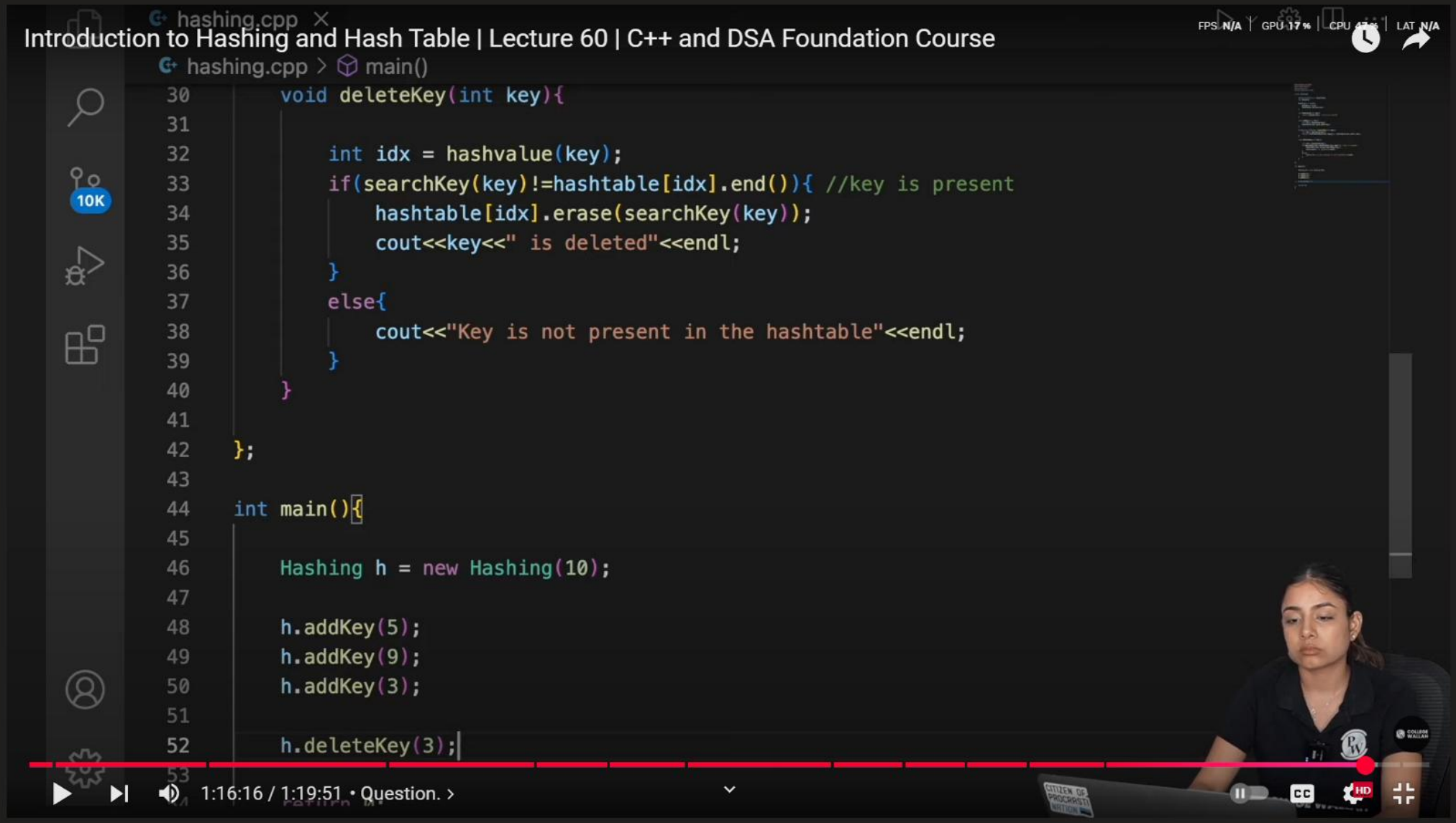
whenever Load factor limit = 0.75
$Lf > limit$

Increasing size of hash table & redistributing elements in it

Coding Implementation...


Introduction to Hashing and Hash Table | Lecture 60 | C++ and DSA Foundation Course
hashing.cpp
hashing.cpp > Hashing
FPS N/A
GPU 8%
LAT N/A
10K
15
16 int hashvalue(int key){
17 return key%buckets; //division method
18 }
19
20 void add(int key){
21 int idx = hashvalue(key);
22 hashtable[idx].push_back(key);
23 }
24
25
26
27
28 };
29
30 int main(){
31
32 return 0;
33 }
1:07:58 / 1:19:51 • Question. ›
CC
HD


Introduction to Hashing and Hash Table | Lecture 60 | C++ and DSA Foundation Course
hashing.cpp
hashing.cpp > Hashing
return key%buckets; //division method
}
void add(int key){
int idx = hashvalue(key);
hashtable[idx].push_back(key);
}
list<int>::iterator search(int key){
int idx = hashvalue(key);
return find(hashtable[idx].begin(), hashtable[idx].end(),key);
}
void delete(int key){
if(search(key))
int idx = hashvalue(key);
hashtable[idx].erase(search(key));
}
1:12:36 / 1:19:51 • Question. ›


Introduction to Hashing and Hash Table | Lecture 60 | C++ and DSA Foundation Course
hashing.cpp
hashing.cpp > main()
void deleteKey(int key){
int idx = hashvalue(key);
if(searchKey(key)!=hashtable[idx].end()){ //key is present
hashtable[idx].erase(searchKey(key));
cout<<key<<" is deleted"<<endl;
}
else{
cout<<"Key is not present in the hashtable"<<endl;
}
}
};
int main(){
Hashing h = new Hashing(10);
h.addKey(5);
h.addKey(9);
h.addKey(3);
h.deleteKey(3);
1:16:16 / 1:19:51 • Question. ›