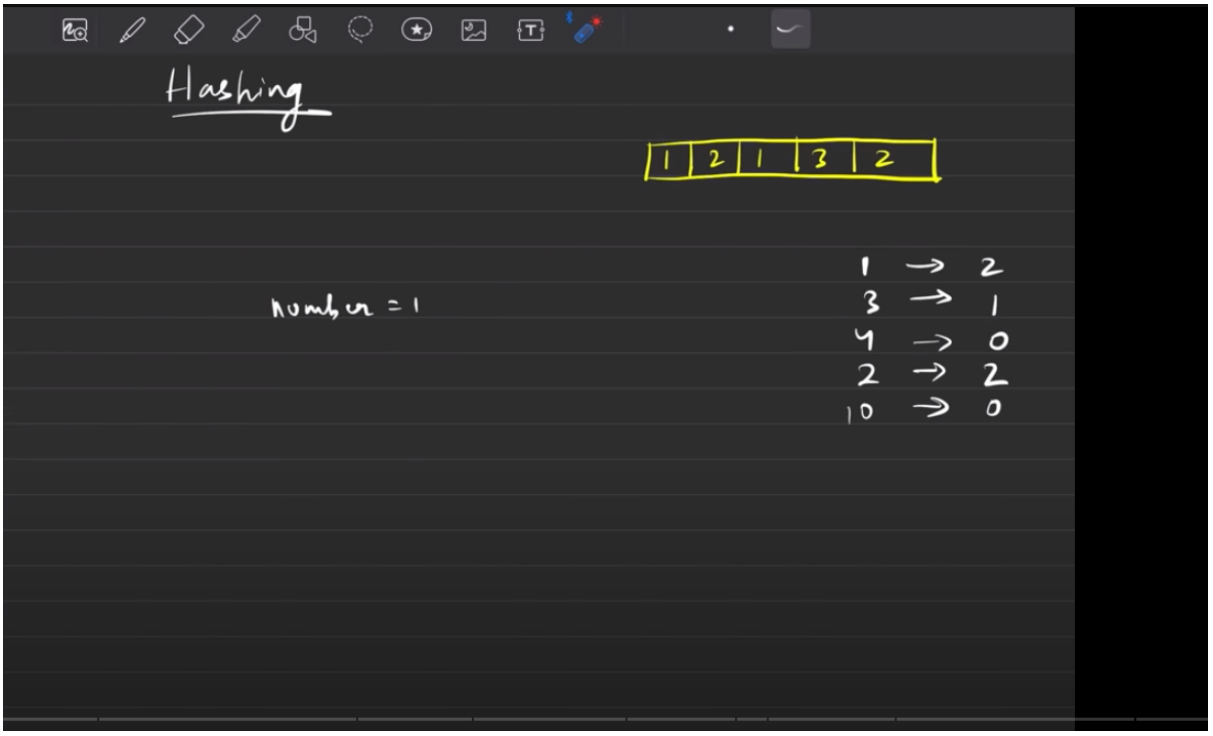
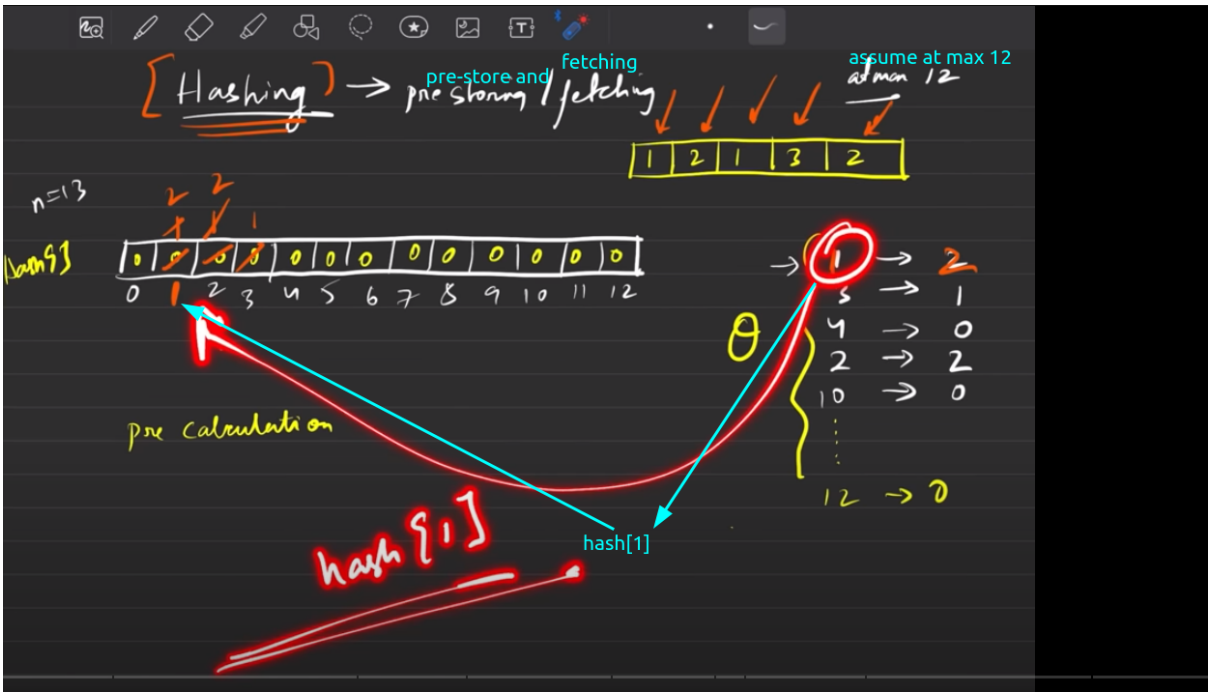
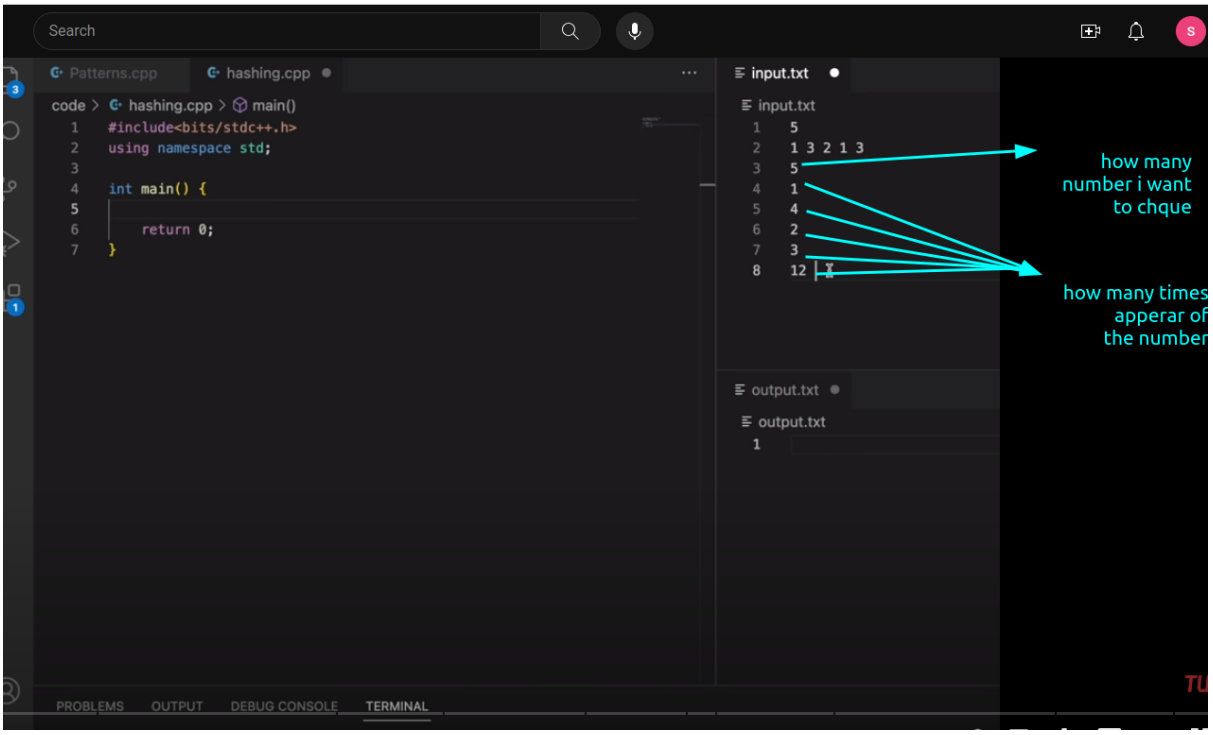
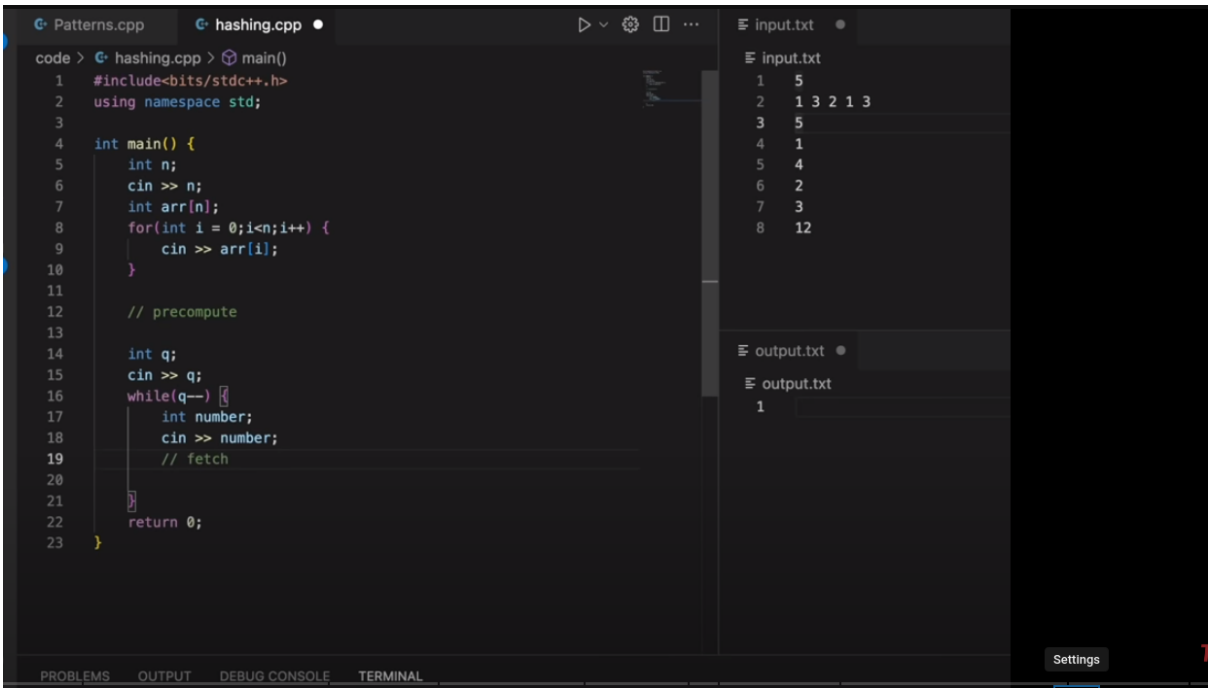
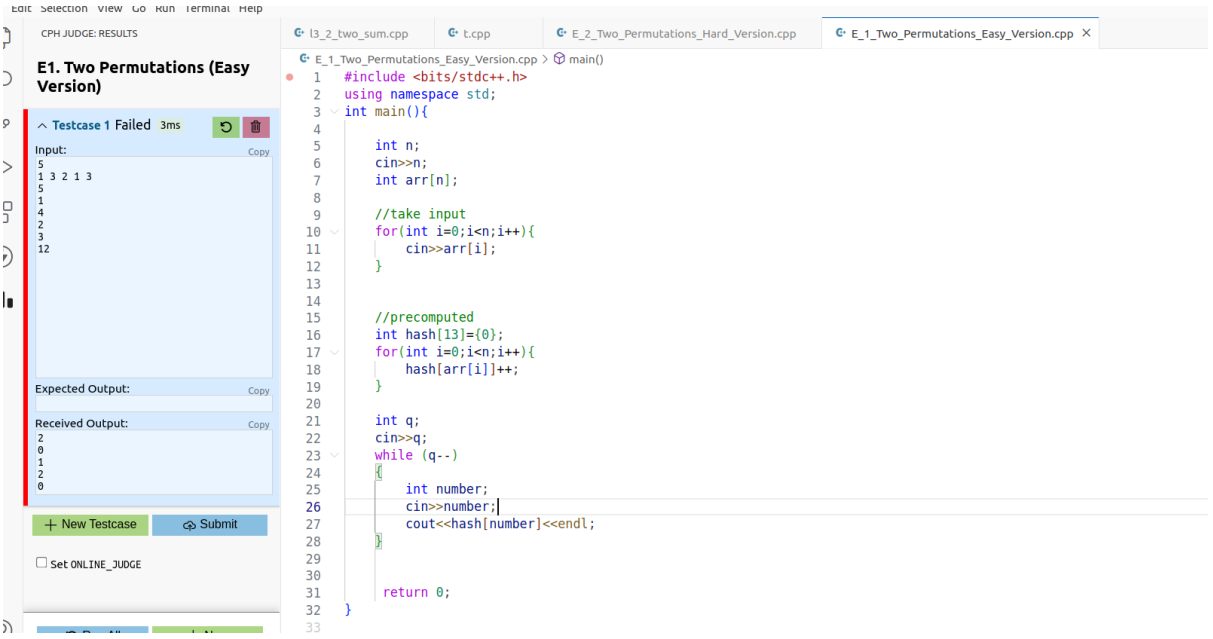
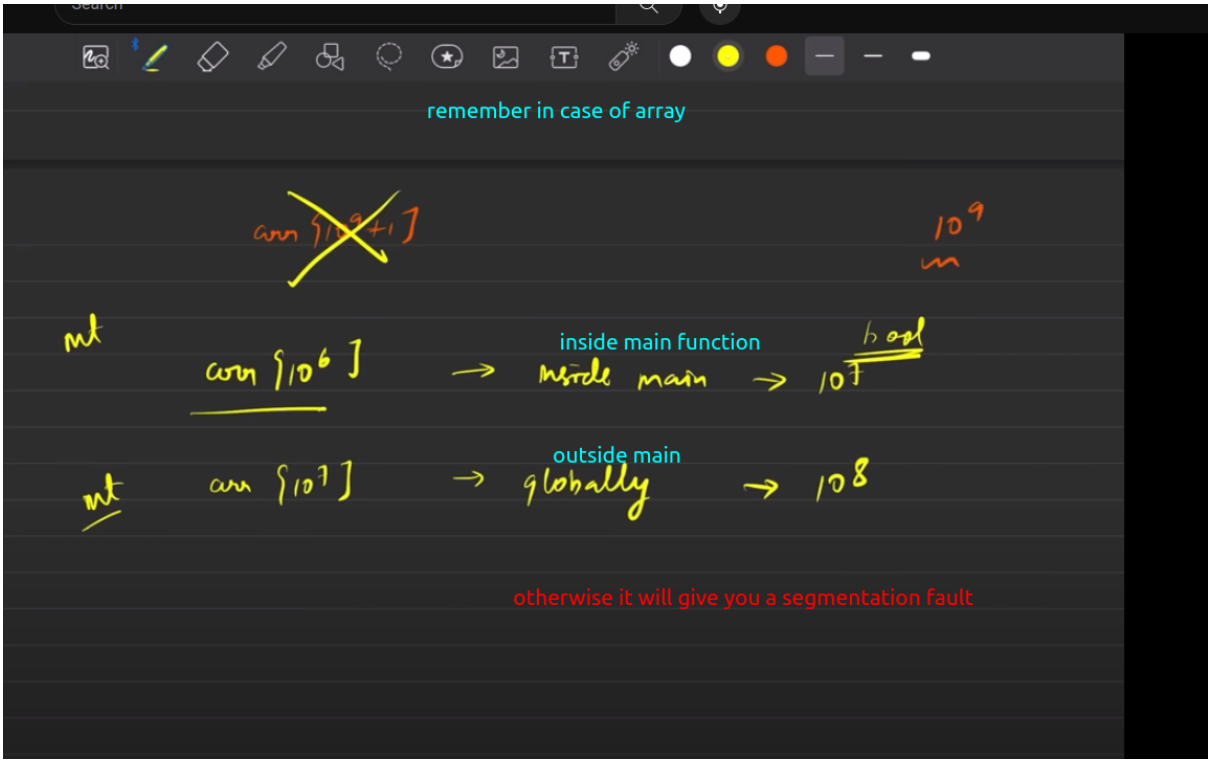

Hashing
1 2 1 3 2
number = 1
1 → 2
3 → 1
4 → 0
2 → 2
10 → 0

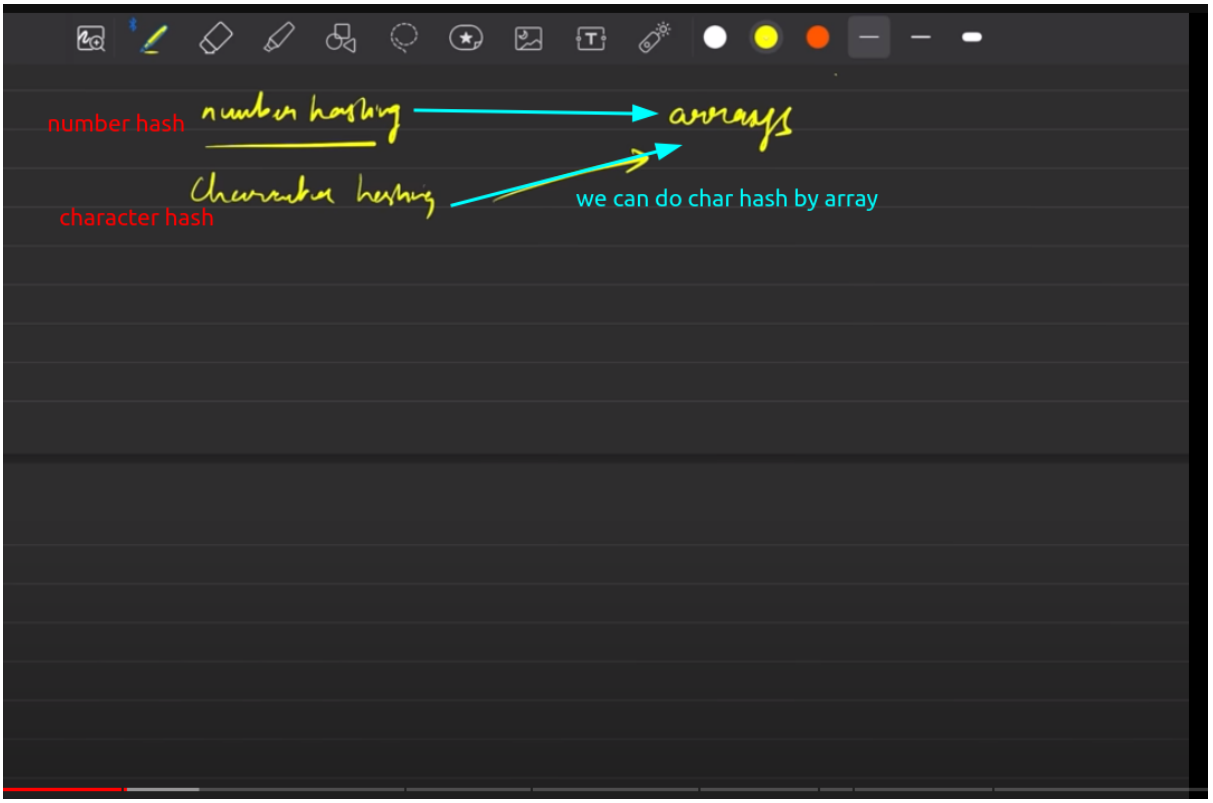

Hashing → pre storing / fetching
fetching
pre-store and
assume at max 12
1 2 1 3 2
n=13
hash[]
0 1 2 3 4 5 6 7 8 9 10 11 12
pre calculation
hash[1]
hash[1]
1 → 2
3 → 1
4 → 0
2 → 2
10 → 0
12 → 0

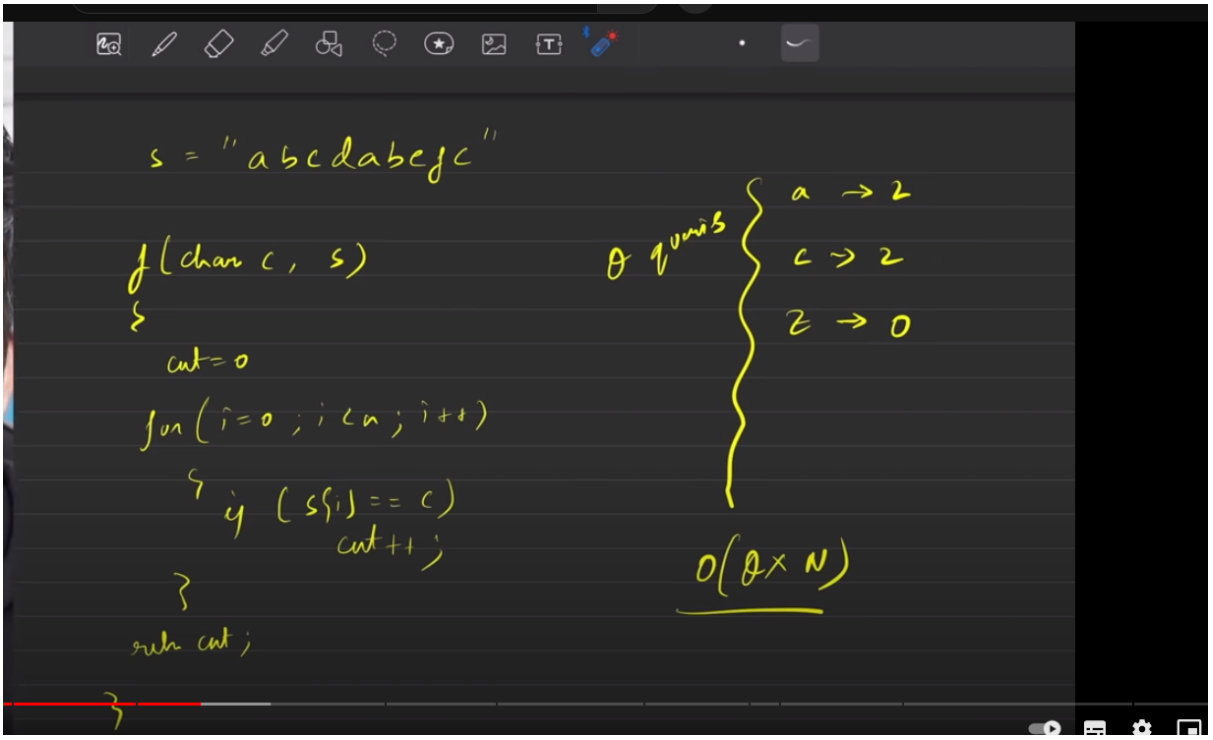

Search
Patterns.cpp
hashing.cpp
code > hashing.cpp > main()
#include<bits/stdc++.h>
using namespace std;
int main() {
return 0;
}
input.txt
input.txt
5
1 3 2 1 3
5
1
4
2
3
12
how many number i want to chque
how many times apperar of the number
output.txt
output.txt
1
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL

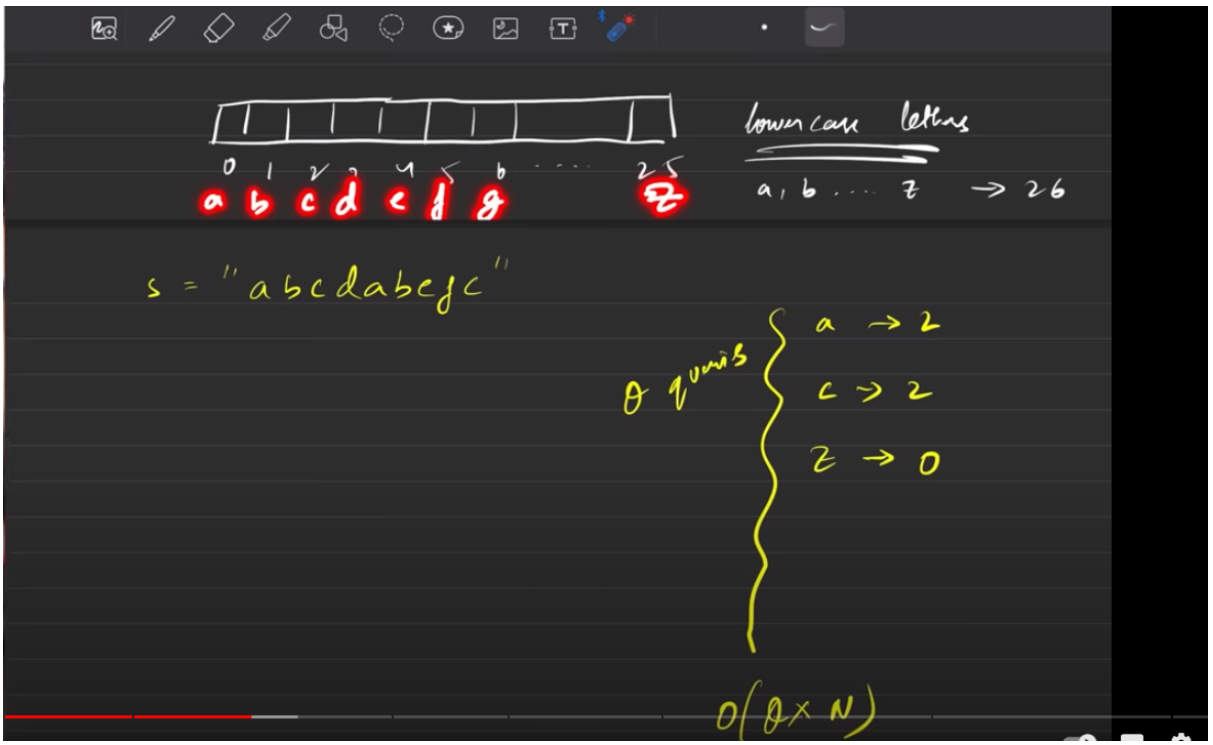

Patterns.cpp
hashing.cpp
code > hashing.cpp > main()
#include<bits/stdc++.h>
using namespace std;
int main() {
int n;
cin >> n;
int arr[n];
for(int i = 0;i<n;i++) {
cin >> arr[i];
}
// precompute
int q;
cin >> q;
while(q--) {
int number;
cin >> number;
// fetch
}
return 0;
}
input.txt
input.txt
5
1 3 2 1 3
5
1
4
2
3
12
output.txt
output.txt
1
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Settings

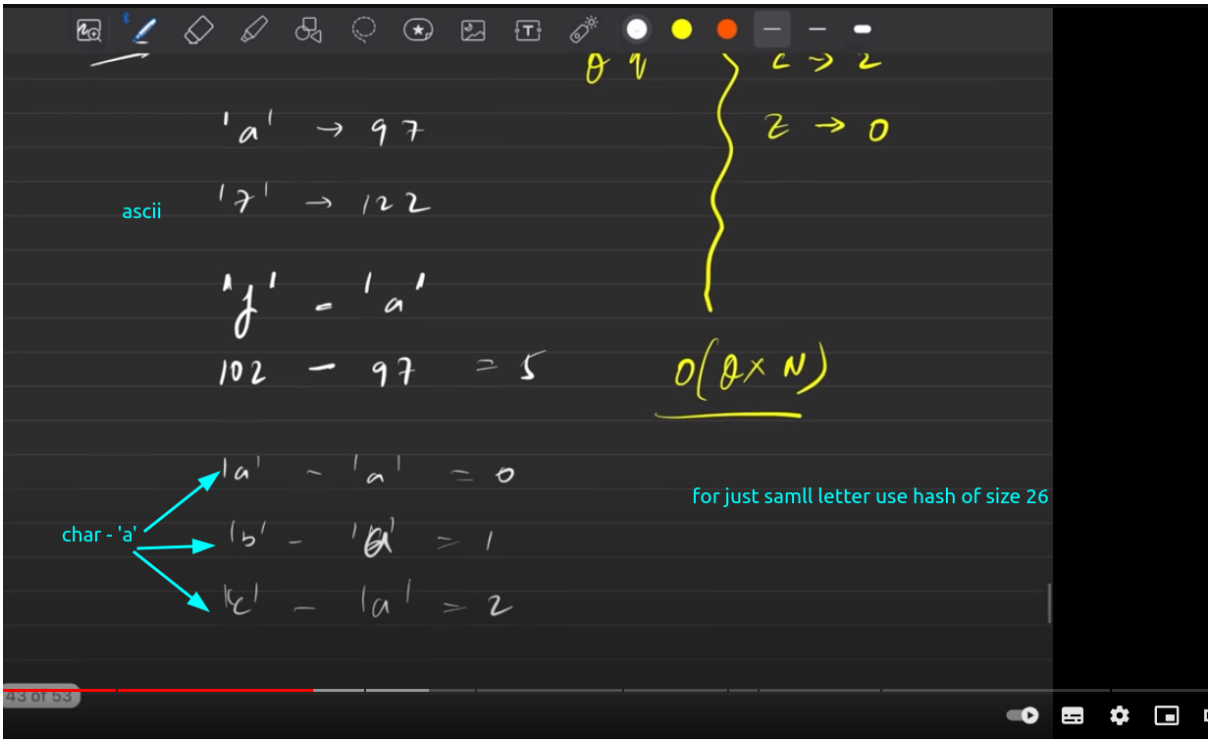

CPH JUDGE: RESULTS
E1. Two Permutations (Easy Version)
Testcase 1 Failed 3ms
Input:
5
1 3 2 1 3
5
1
4
2
3
12
Expected Output:
Received Output:
2
0
1
2
0
New Testcase
Submit
Set ONLINE_JUDGE
l3_2_two_sum.cpp
t.cpp
E_2_Two_Permutations_Hard_Version.cpp
E_1_Two_Permutations_Easy_Version.cpp
E_1_Two_Permutations_Easy_Version.cpp > main()
#include <bits/stdc++.h>
using namespace std;
int main(){
int n;
cin>>n;
int arr[n];
//take input
for(int i=0;i<n;i++){
cin>>arr[i];
}
//precomputed
int hash[13]={0};
for(int i=0;i<n;i++){
hash[arr[i]]++;
}
int q;
cin>>q;
while (q--)
{
int number;
cin>>number;
cout<<hash[number]<<endl;
}
return 0;
}

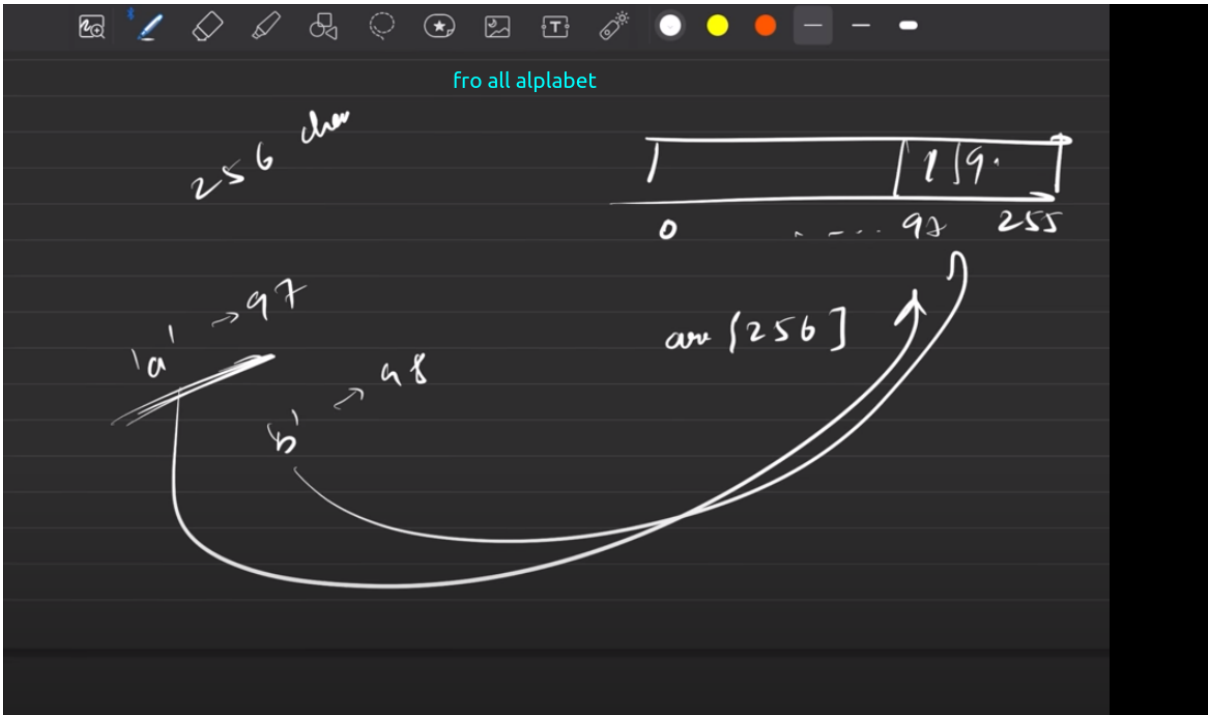

remember in case of array
inside main function
outside main
globally
otherwise it will give you a segmentation fault


number hash
number hashing
arrays
character hash
character hashing
we can do char hash by array


s = "abcdabcjc"
f(char c, s)
{
cnt = 0
for(i=0; i<n; i++)
{ if (s[i] == c)
cnt++;
}
return cnt;
}
Q queries
a → 2
c → 2
z → 0
O(Q × N)


lowercase letters
a, b .... z → 26
s = "abcdabcjc"
a → 2
c → 2
z → 0
O(Q × N)


'a' → 97
'z' → 122
ascii
'j' - 'a'
102 - 97 = 5
O(Q × N)
'a' - 'a' = 0
'b' - 'a' = 1
'c' - 'a' = 2
char - 'a'
for just samll letter use hash of size 26
43 of 53



CPH JUDGE: RESULTS

Local: t

Testcase 1

Input: Copy

abcdabehf
5
a
g
h
b
c

Expected Output: Copy

New Testcase

Set ONLINE_JUDGE

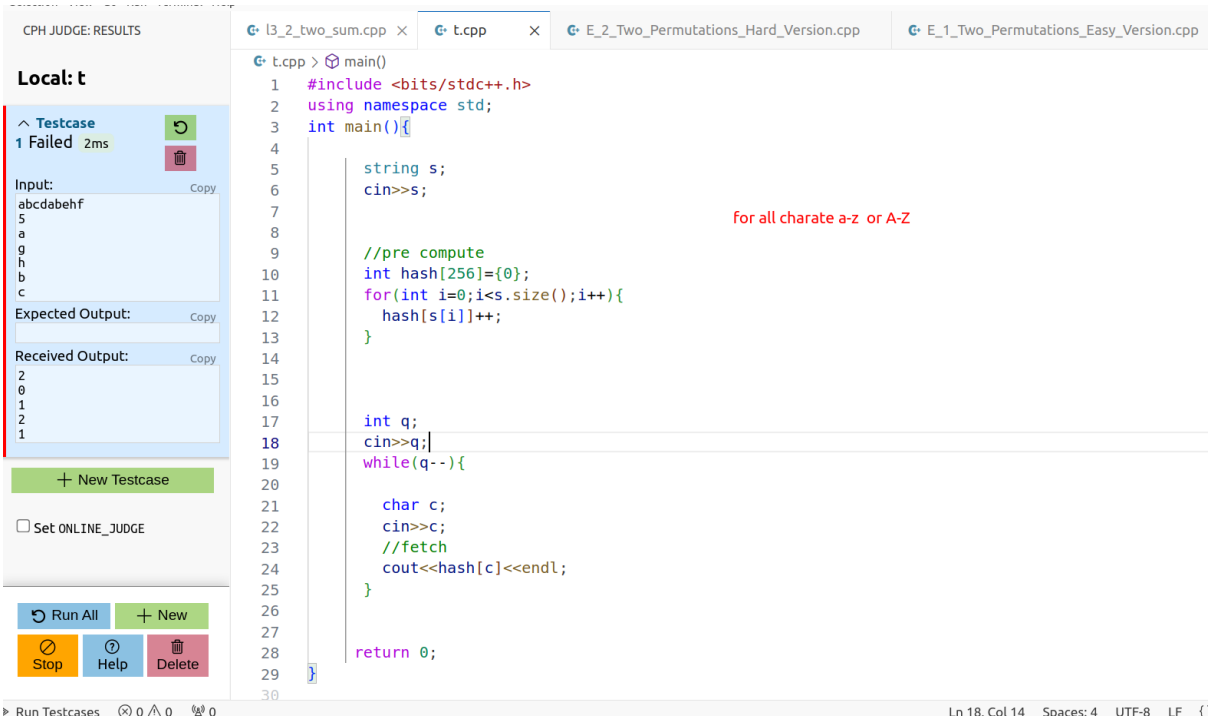
Run All New

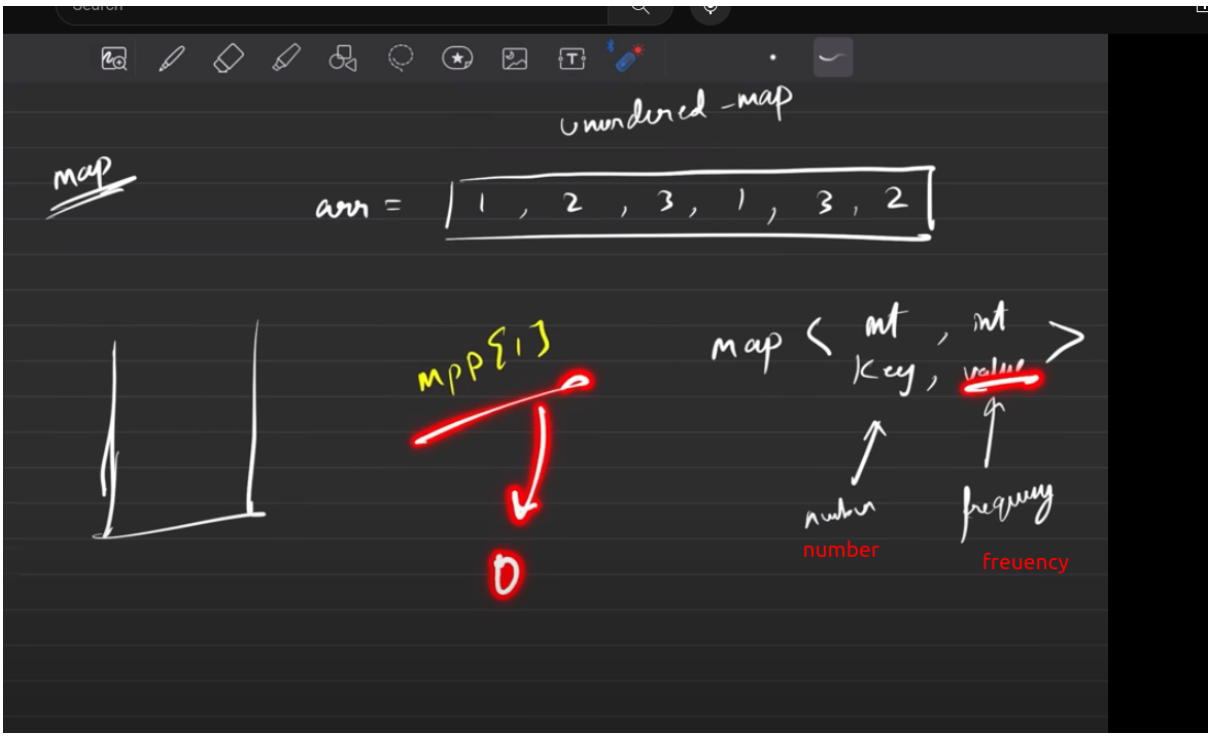
l3_2_two_sum.cpp | t.cpp | E_2_Two_Permutations_Hard_Version.cpp | E_1_Two_Permutations_Easy_Version.cpp

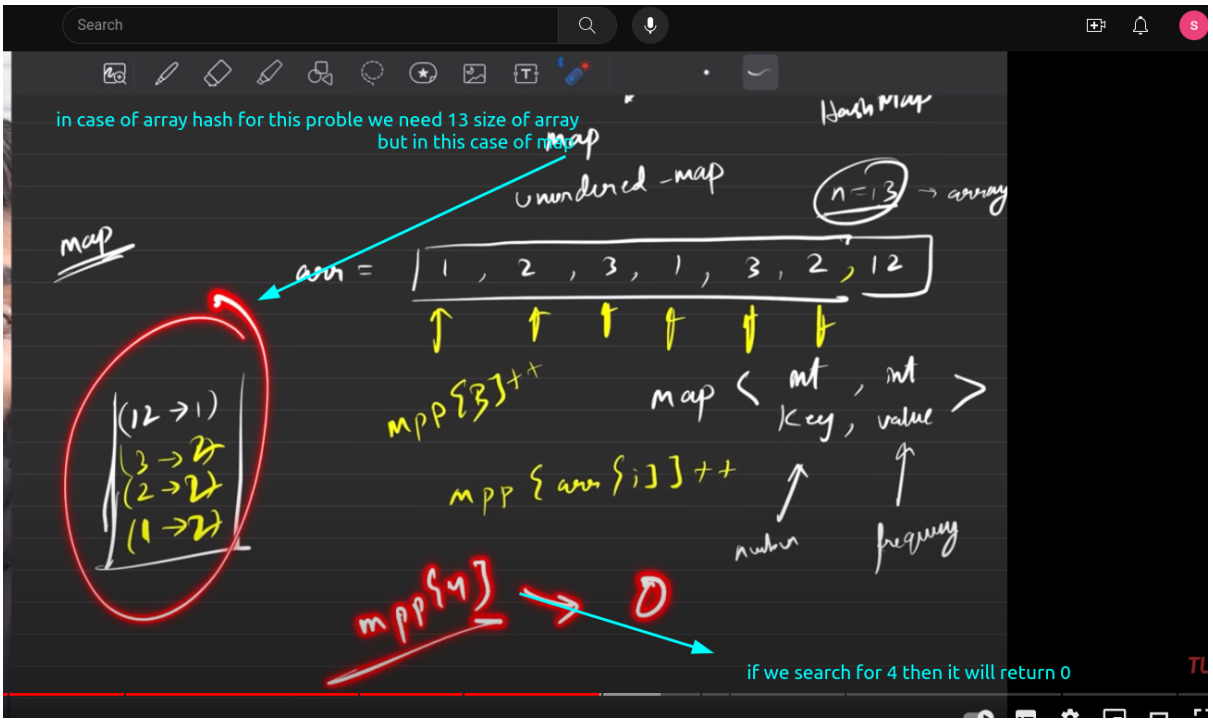
t.cpp > main()

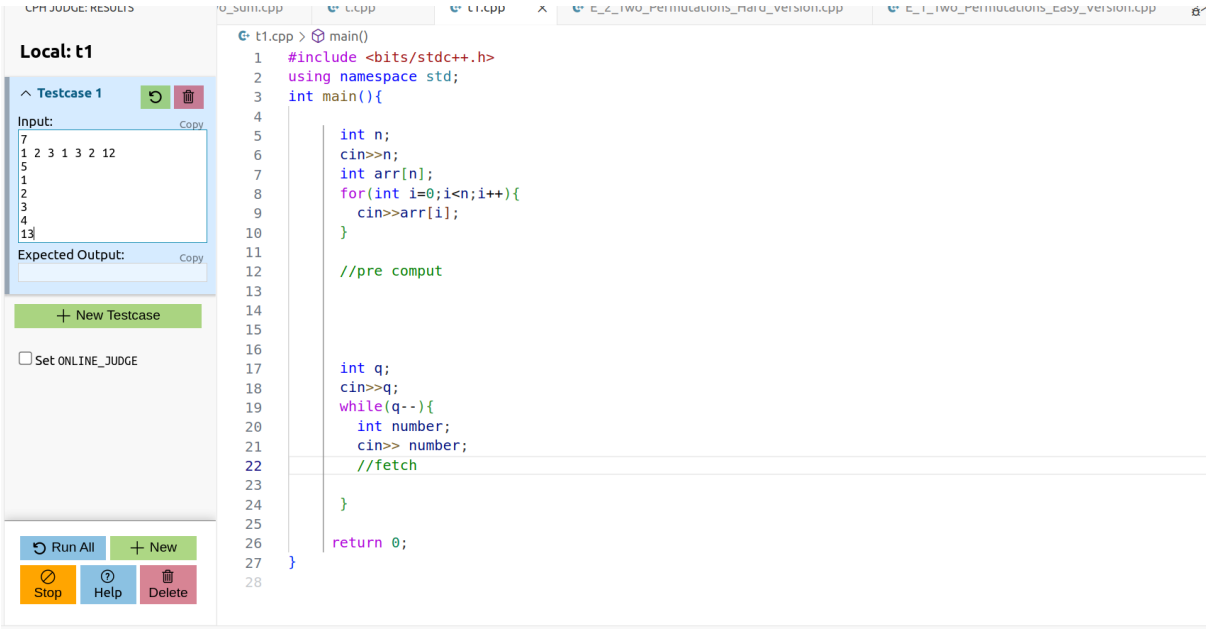
```cpp
#include <bits/stdc++.h>
using namespace std;
int main(){

    string s;
    cin>>s;


    //precouputed



    int q;
    cin>>q;
    while(q--){

      char c;
      cin>>c;
      //fetch
    }


    return 0;
}

```

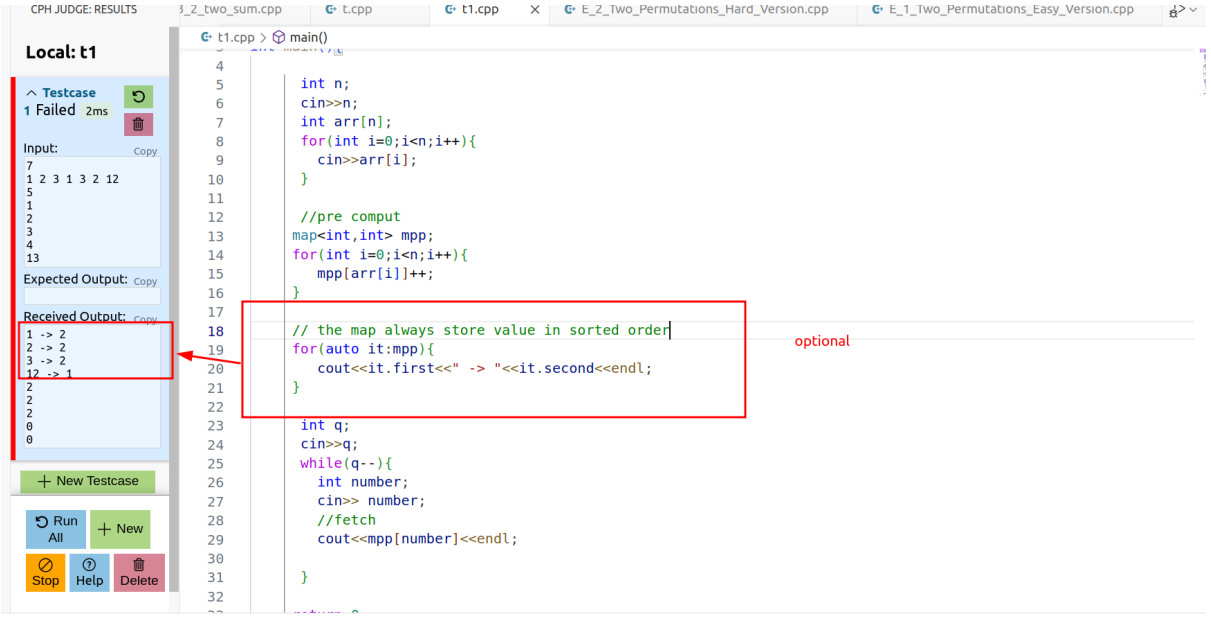

Local: t
Testcase
1 Failed 5ms
Input:
Copy
abcdabehf
5
a
g
h
b
c
Expected Output:
Copy
Received Output:
Copy
2
0
1
2
1
New Testcase
Set ONLINE_JUDGE
Run All
New
Stop
Help
Delete
t.cpp > main()
#include <bits/stdc++.h>
using namespace std;
int main(){
string s;
cin>>s;
//pre compute
int hash[26]={0};
for(int i=0;i<s.size();i++){
hash[s[i]-'a']++;
}
int q;
cin>>q;
while(q--){
char c;
cin>>c;
//fetch
cout<<hash[c-'a']<<endl;
}
return 0;
}


CPH JUDGE: RESULTS
Local: t
Testcase
1 Failed 2ms
Input:
Copy
abcdabehf
5
a
g
h
b
c
Expected Output:
Copy
Received Output:
Copy
2
0
1
2
1
New Testcase
Set ONLINE_JUDGE
Run All
New
Stop
Help
Delete
l3_2_two_sum.cpp
t.cpp
E_2_Two_Permutations_Hard_Version.cpp
E_1_Two_Permutations_Easy_Version.cpp
t.cpp > main()
#include <bits/stdc++.h>
using namespace std;
int main(){
string s;
cin>>s;
for all charate a-z or A-Z
//pre compute
int hash[256]={0};
for(int i=0;i<s.size();i++){
hash[s[i]]++;
}
int q;
cin>>q;
while(q--){
char c;
cin>>c;
//fetch
cout<<hash[c]<<endl;
}
return 0;
}
Run Testcases
Ln 18, Col 14
Spaces: 4
UTF-8
LF


unordered -map
map
arr = 1 , 2 , 3 , 1 , 3 , 2
mpp[i]
0
map < int , int >
key , value
number
frequency
number
freuency


Search
in case of array hash for this proble we need 13 size of array
but in this case of map
unordered -map
map
arr = 1 , 2 , 3 , 1 , 3 , 2 , 12
(12 → 1)
(3 → 2)
(2 → 2)
(1 → 2)
mpp[3]++
map < int , int >
key , value
mpp[arr[i]]++
number
frequency
mpp[4]
0
if we search for 4 then it will return 0


CPH JUDGE: RESULTS
Local: t1
Testcase 1
Input:
Copy
7
1 2 3 1 3 2 12
5
1
2
3
4
13
Expected Output:
Copy
New Testcase
Set ONLINE_JUDGE
Run All
New
Stop
Help
Delete
t1.cpp
t.cpp
E_2_Two_Permutations_Hard_Version.cpp
E_1_Two_Permutations_Easy_Version.cpp
t1.cpp > main()
#include <bits/stdc++.h>
using namespace std;
int main(){
int n;
cin>>n;
int arr[n];
for(int i=0;i<n;i++){
cin>>arr[i];
}
//pre comput
int q;
cin>>q;
while(q--){
int number;
cin>> number;
//fetch
}
return 0;
}


CPH JUDGE: RESULTS
Local: t1
Testcase
1 Failed 2ms
Input:
Copy
7
1 2 3 1 3 2 12
5
1
2
3
4
13
Expected Output:
Copy
Received Output:
Copy
1 -> 2
2 -> 2
3 -> 2
12 -> 1
2
2
2
0
0
New Testcase
Run All
New
Stop
Help
Delete
t1.cpp > main()
int n;
cin>>n;
int arr[n];
for(int i=0;i<n;i++){
cin>>arr[i];
}
//pre comput
map<int,int> mpp;
for(int i=0;i<n;i++){
mpp[arr[i]]++;
}
// the map always store value in sorted order
for(auto it:mpp){
cout<<it.first<<" -> "<<it.second<<endl;
}
int q;
cin>>q;
while(q--){
int number;
cin>> number;
//fetch
cout<<mpp[number]<<endl;
}
optional

CPH JUDGE: RESULTS

Local: t1

Testcase
1 Failed 1ms

Input: Copy
```
6
lalmia
3
l
a
s
```

Expected Output: Copy

Received Output: Copy
```
a -> 2
i -> 1
l -> 2
m -> 1
2
2
0
```

+ New Testcase

Set ONLINE_JUDGE

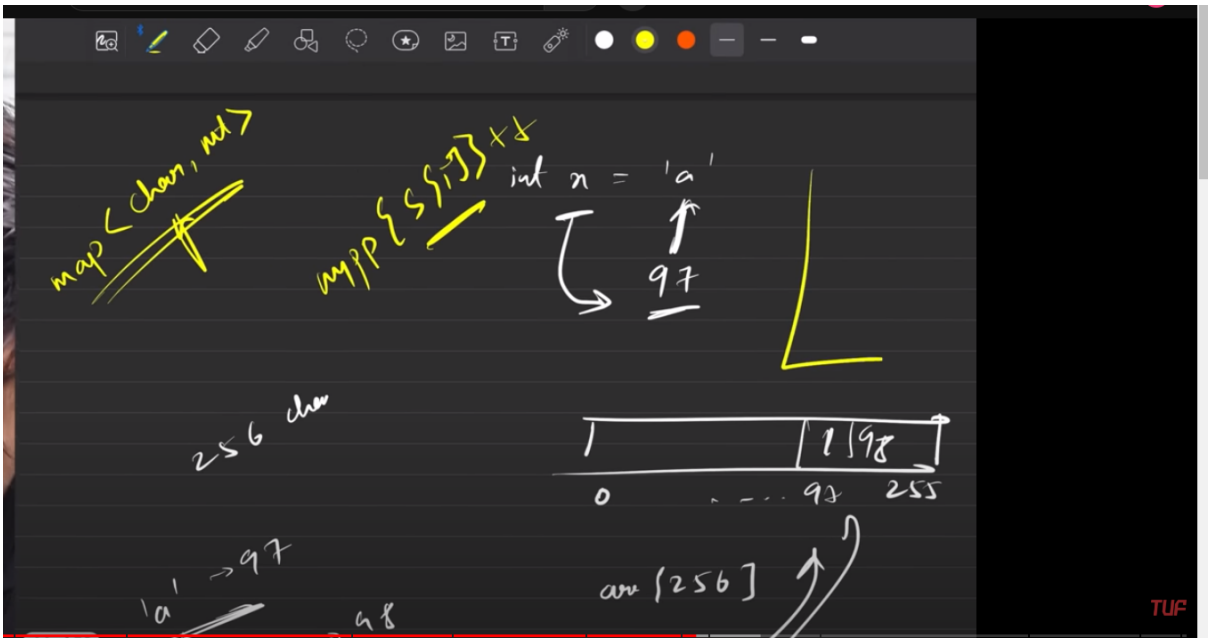
Run All | + New | Stop | Help | Delete

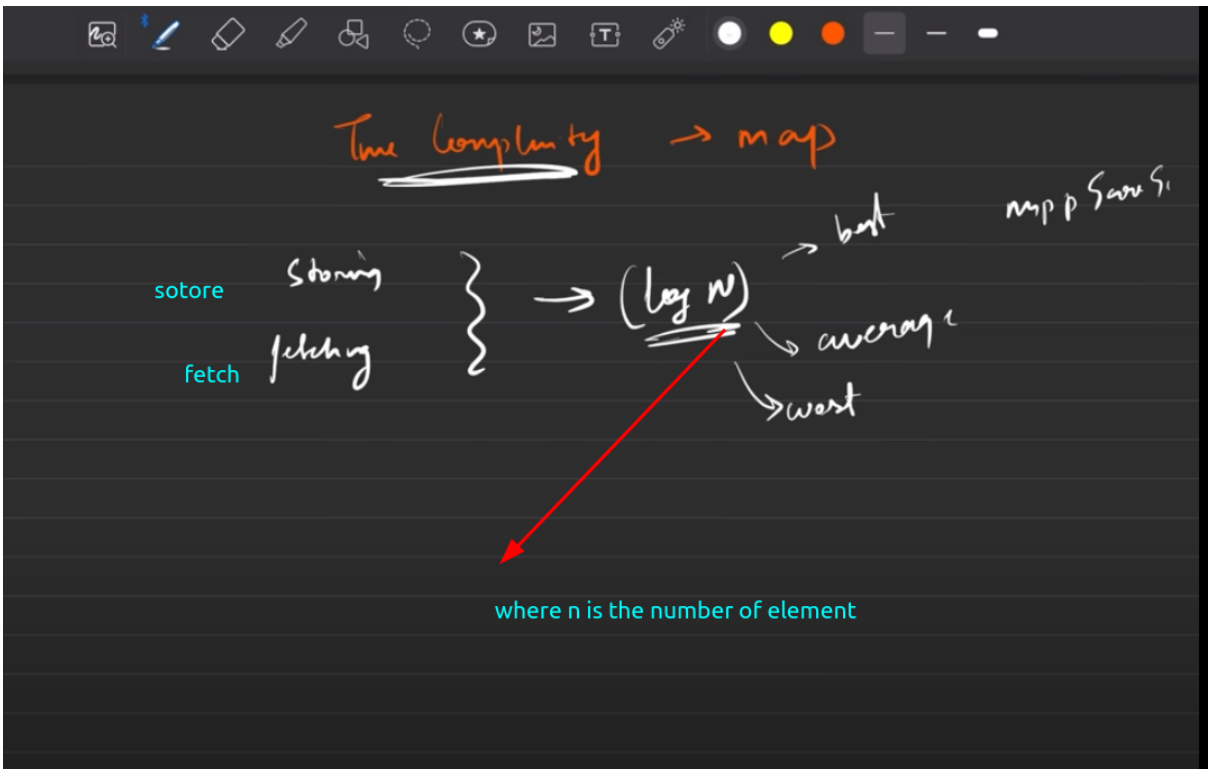
3_2_two_sum.cpp | t.cpp | t1.cpp | E_2_Two_Permutations_Hard_Version.cpp | E_1_Two_Permutations_Easy_Version.

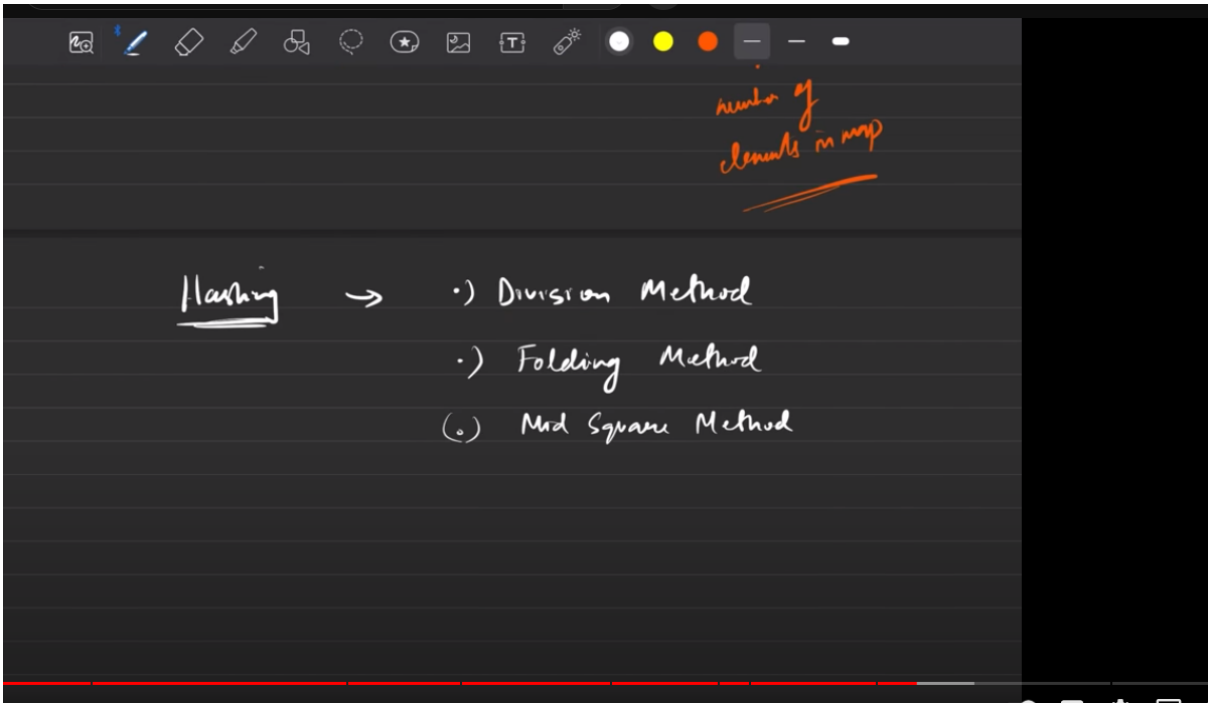
t1.cpp > main()

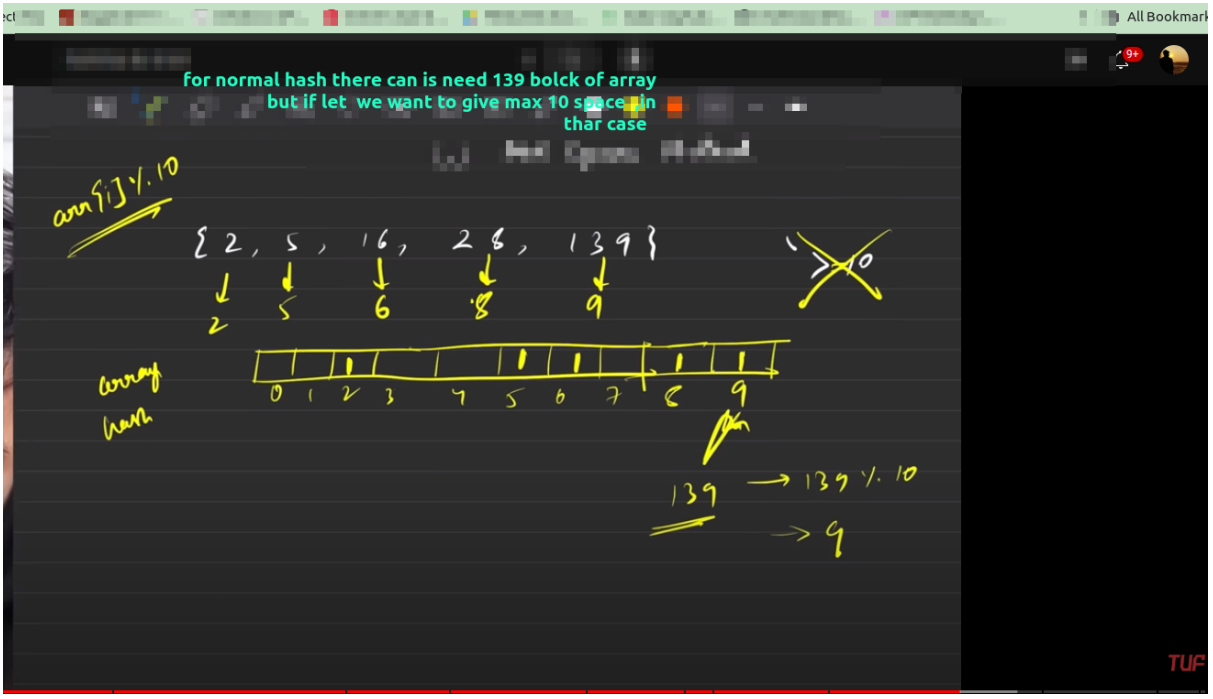
for char hash map

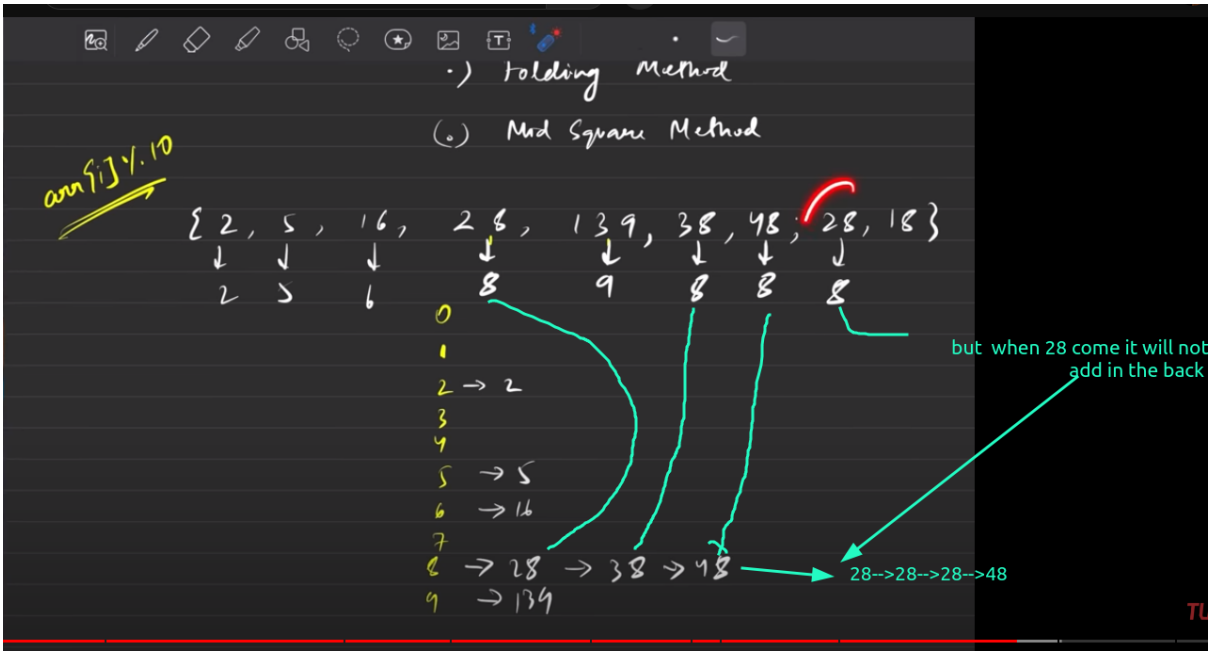
```cpp
int n;
cin>>n;
char arr[n];
for(int i=0;i<n;i++){
  cin>>arr[i];
}

//pre comput
map<char,int> mpp;
for(int i=0;i<n;i++){
   mpp[arr[i]]++;
}

// the map always store value in sorted order
for(auto it:mpp){
   cout<<it.first<<" -> "<<it.second<<endl;
}

int q;
cin>>q;
while(q--){
  char number;
  cin>> number;
  //fetch
  cout<<mpp[number]<<endl;

}
```

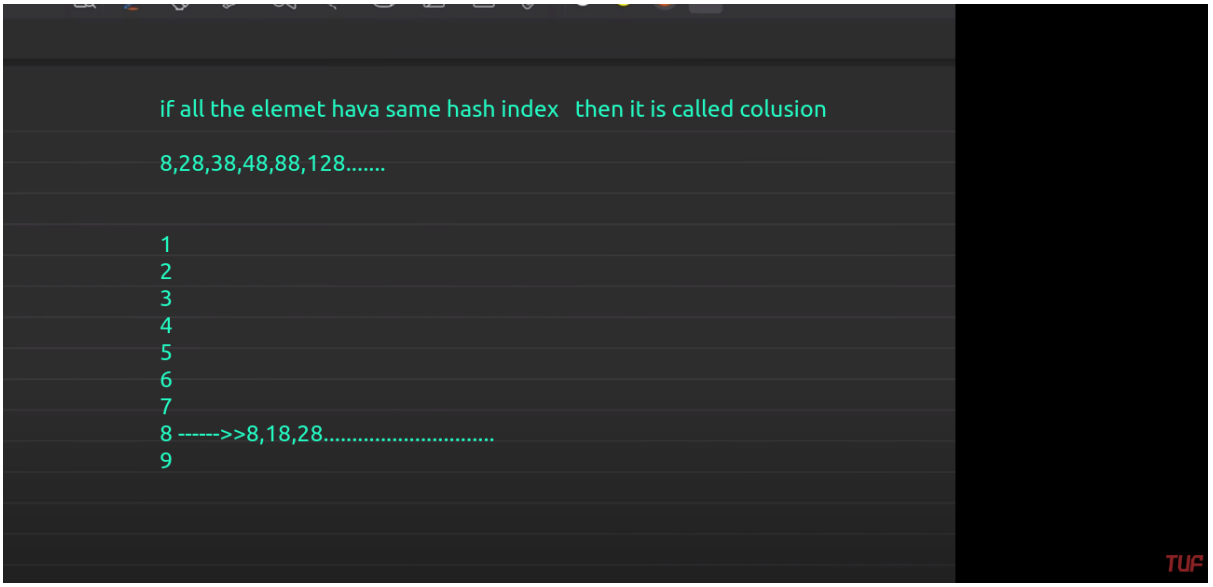



Time Complexity → map
sotore
Storing
fetch
fetching
(log n)
best
average
worst
where n is the number of element


number of elements in map
Hashing →
•) Division Method
•) Folding Method
(•) Mid Square Method


for normal hash there can is need 139 bolck of array
but if let we want to give max 10 space in
thar case
arr[i] % 10
{ 2, 5, 16, 28, 139 }
2
5
6
8
9
array
hash
0 1 2 3 4 5 6 7 8 9
139
139 % 10
9
TUF


·) Folding Method
(.) Mid Square Method
arr[i]%.10
{2, 5, 16, 28, 139, 38, 48, 28, 18}
2 5 6 8 9 8 8 8
0
1
2 → 2
3
4
5 → 5
6 → 16
7
8 → 28 → 38 → 48
9 → 139
but when 28 come it will not add in the back
28-->28-->28-->48


if all the elemet hava same hash index then it is called colusion
8,28,38,48,88,128.....
1
2
3
4
5
6
7
8 ------>>8,18,28.....
9