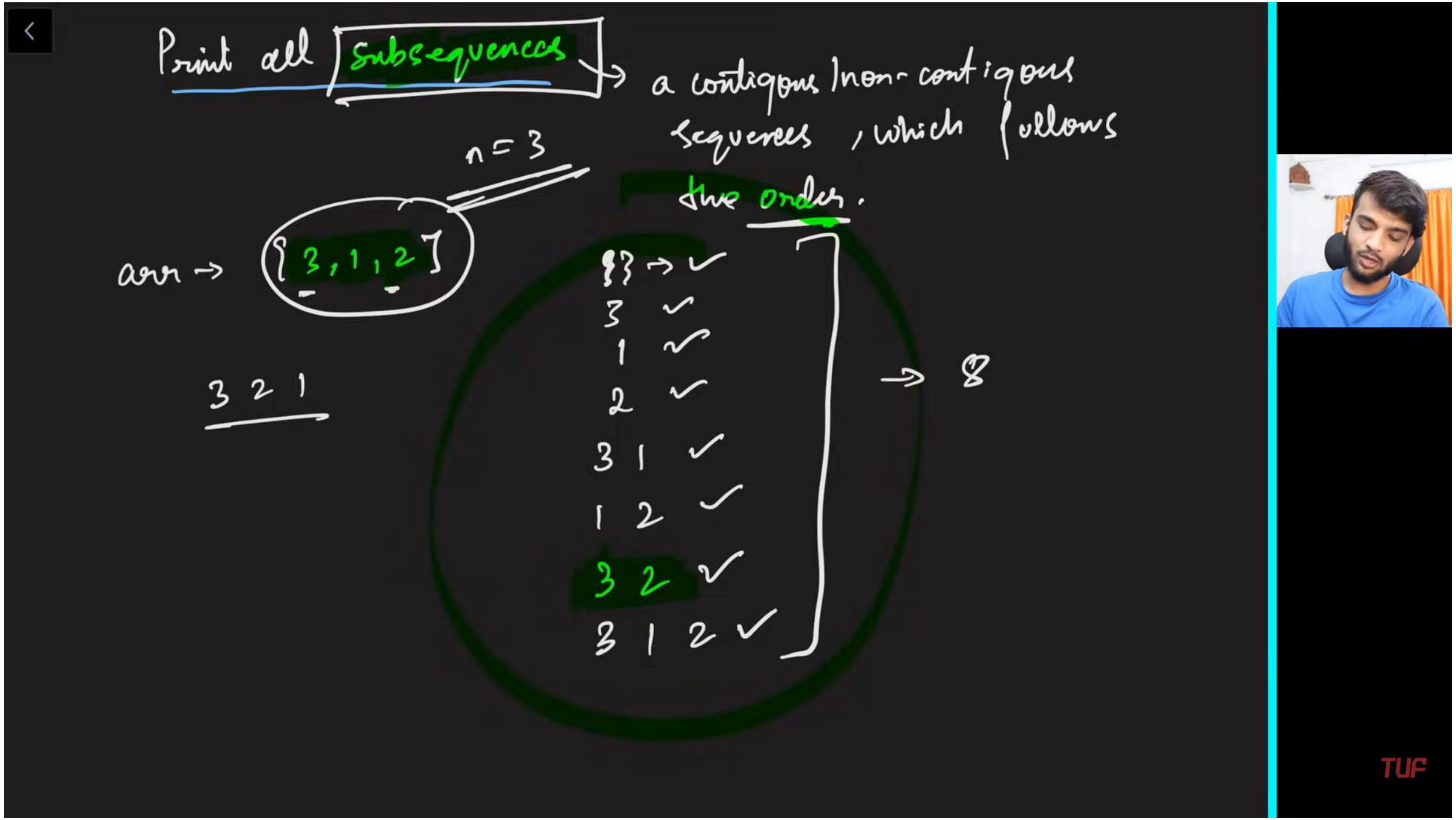
L6. Recursion on Subsequences | Printing Subsequences

Print all subsequences



Print all $\boxed{\text{subsequences}}$ → a contigous /non-contigous sequences, which follows the order.

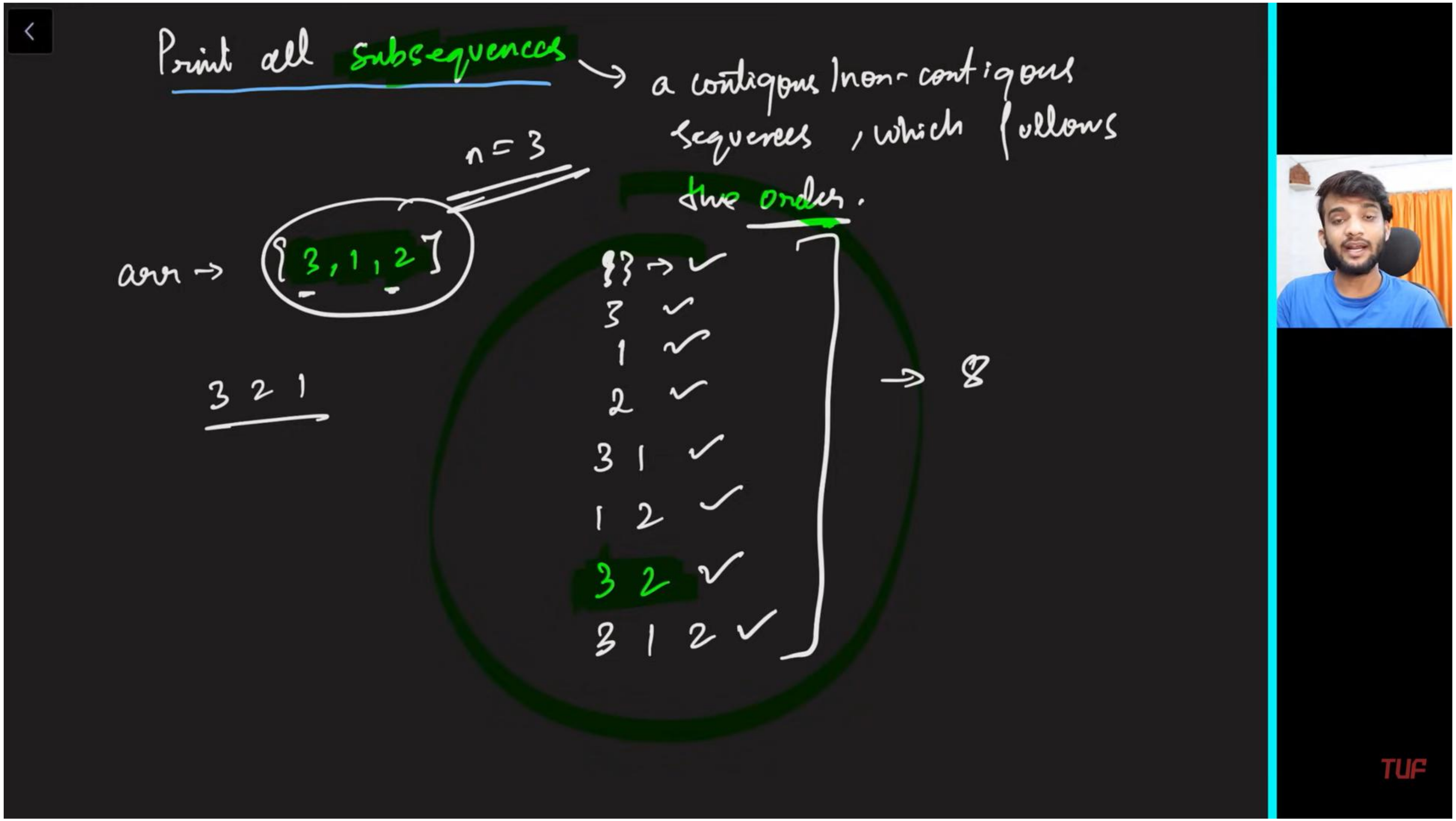
n = 3

arr → [3, 1, 2]

3 2 1

{} → ✓
3 ✓
1 ✓
2 ✓
3 1 ✓
1 2 ✓
3 2 ✓
3 1 2 ✓

→ 8



Print all subsequences → a contigous / non-contigous sequences, which follows the order.

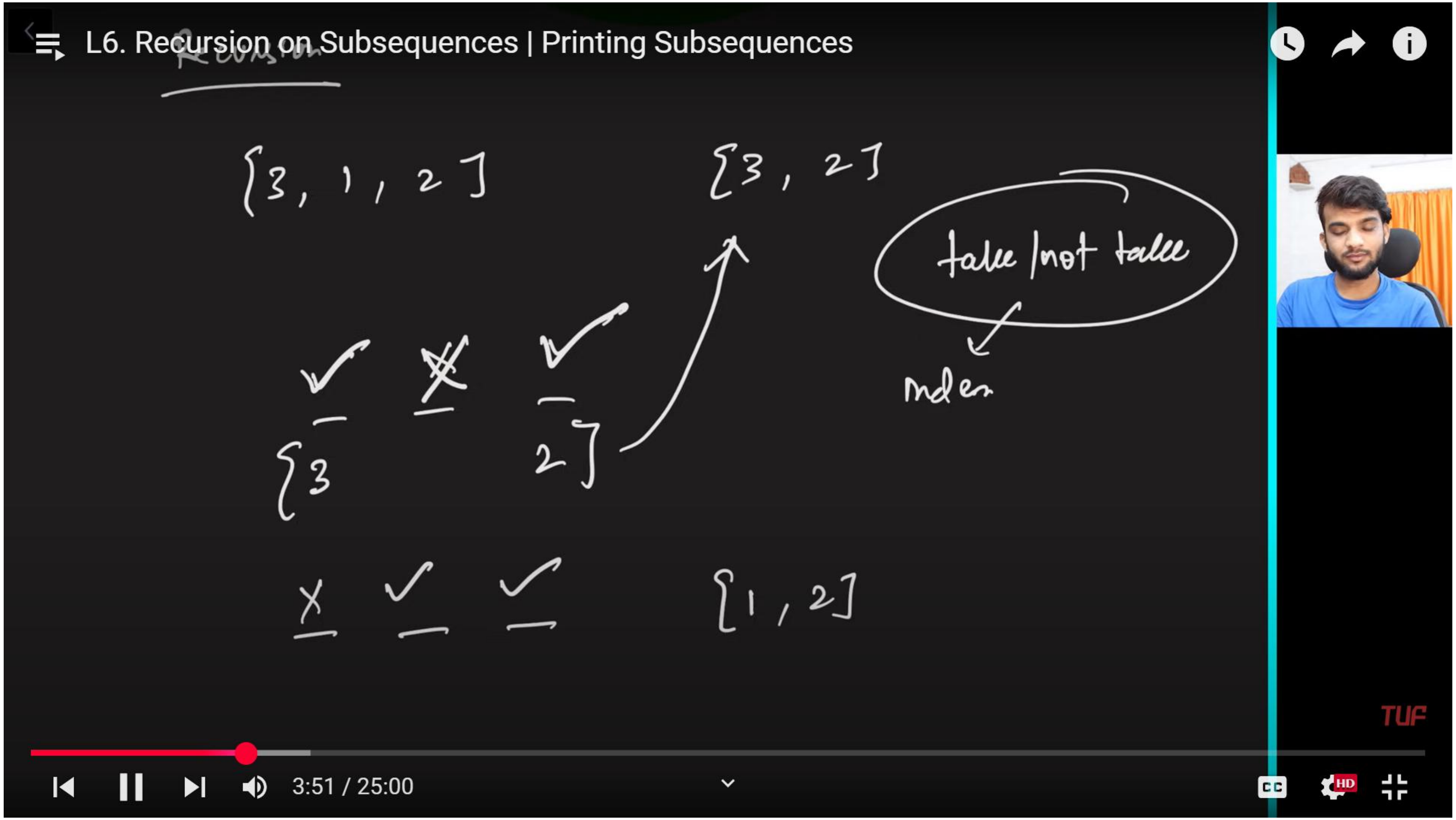
n = 3

arr → [3, 1, 2]

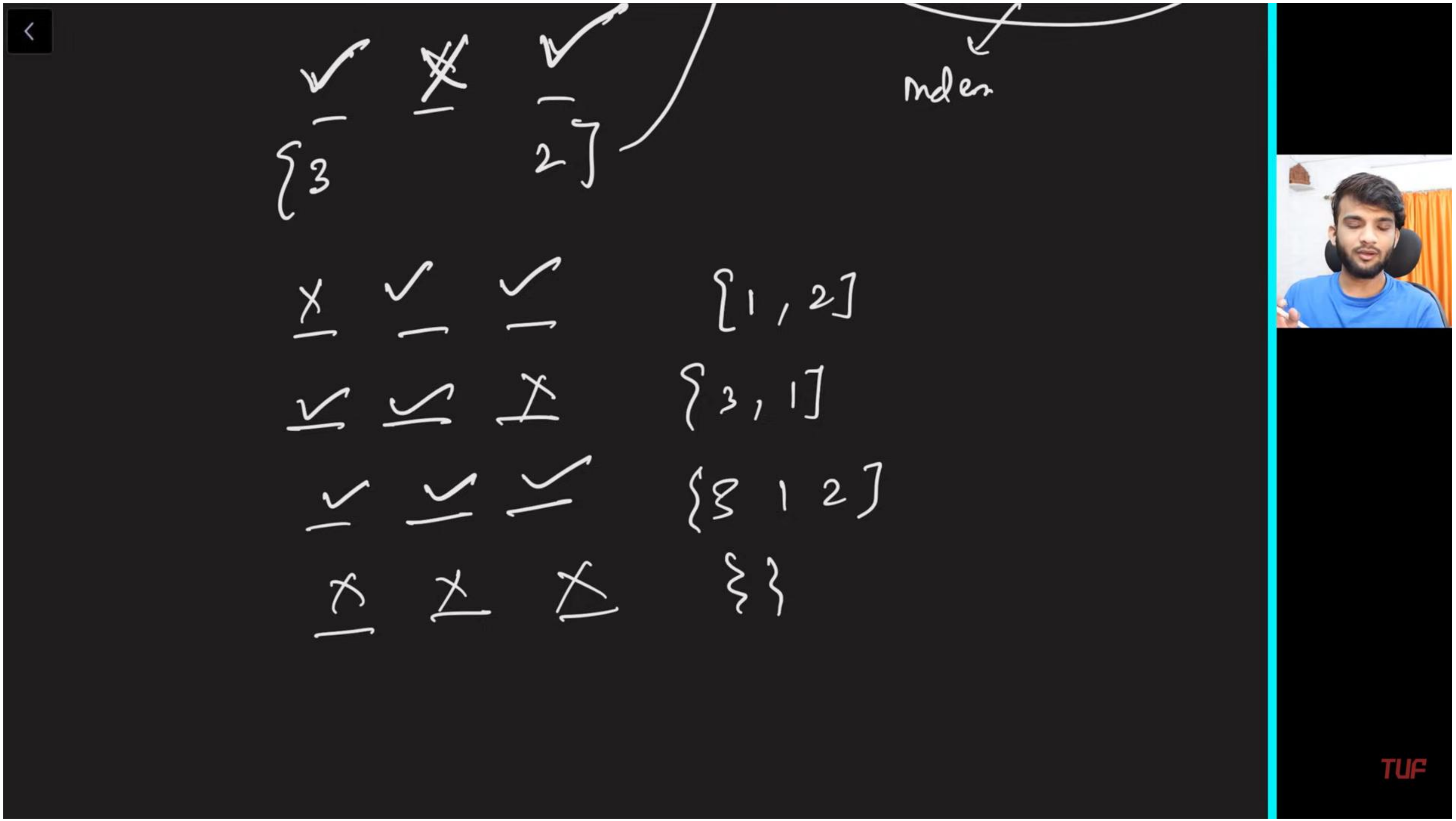
3 2 1

{} → ✓
3 ✓
1 ✓
2 ✓
3 1 ✓
1 2 ✓
3 2 ✓
3 1 2 ✓

→ 8



L6. Recursion on Subsequences | Printing Subsequences



Index

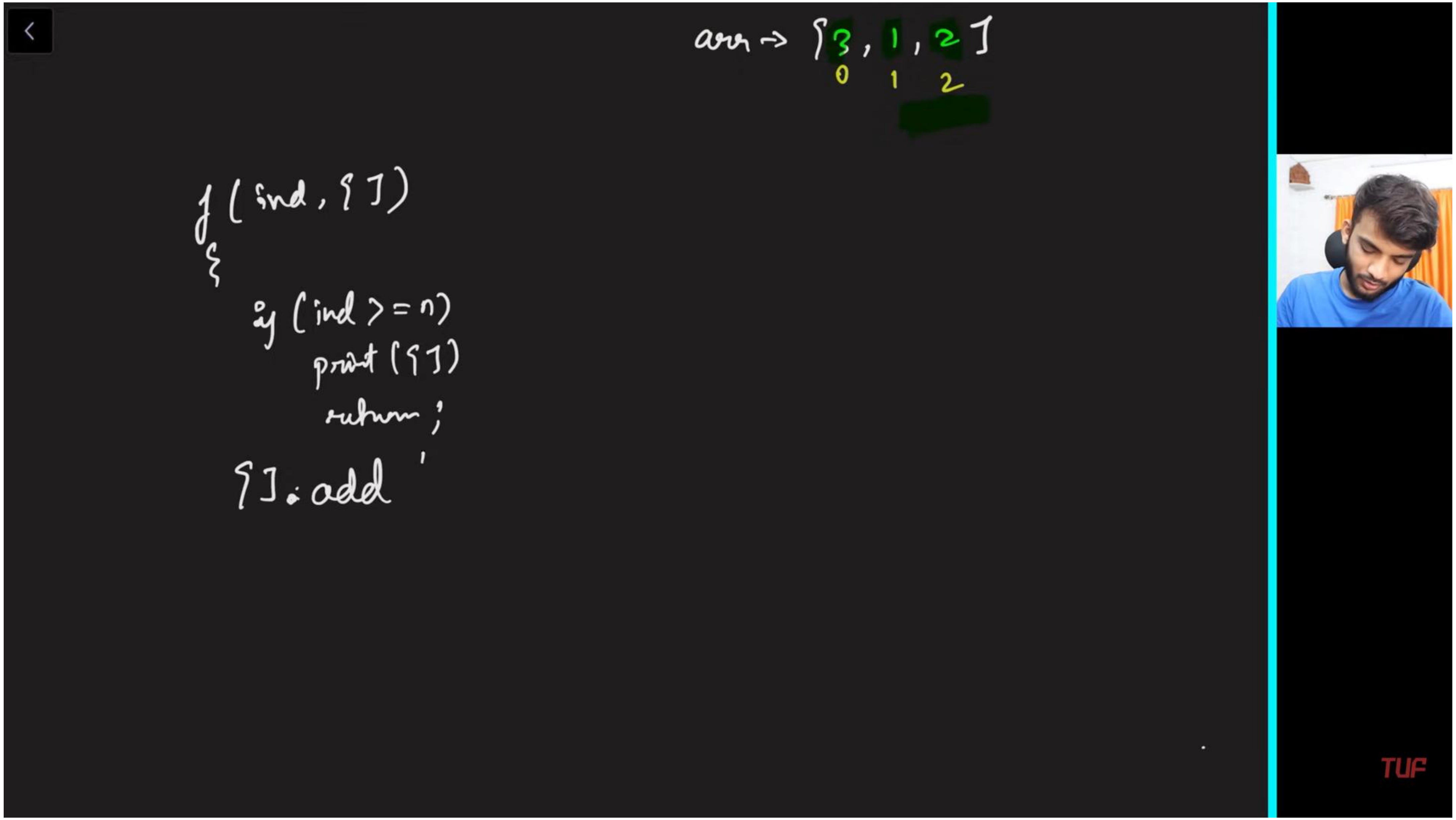
✓ ✗ ✓ {3 2]

✗ ✓ ✓ [1, 2]

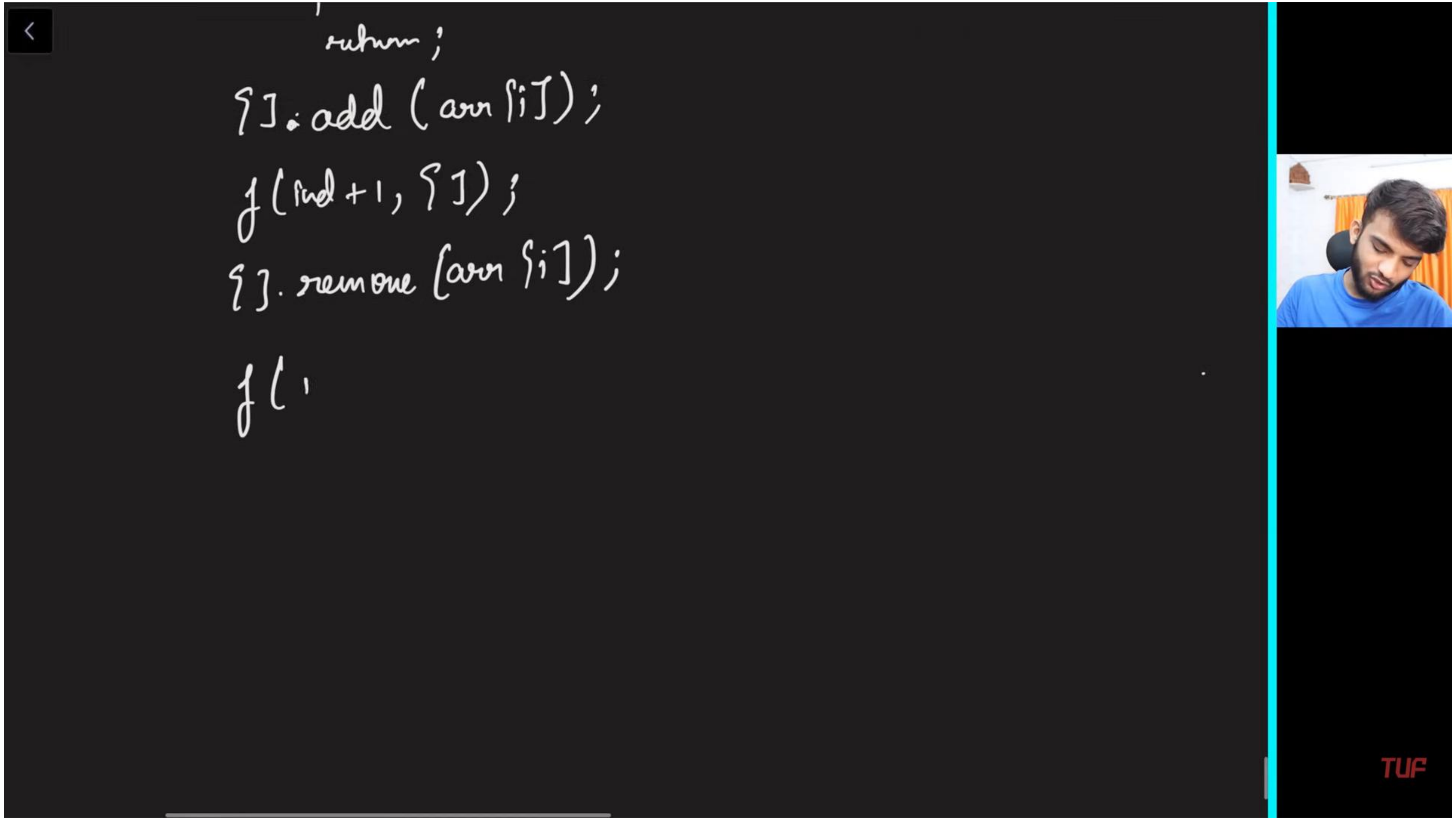
✓ ✓ ✗ {3, 1]

✓ ✓ ✓ {3 1 2]

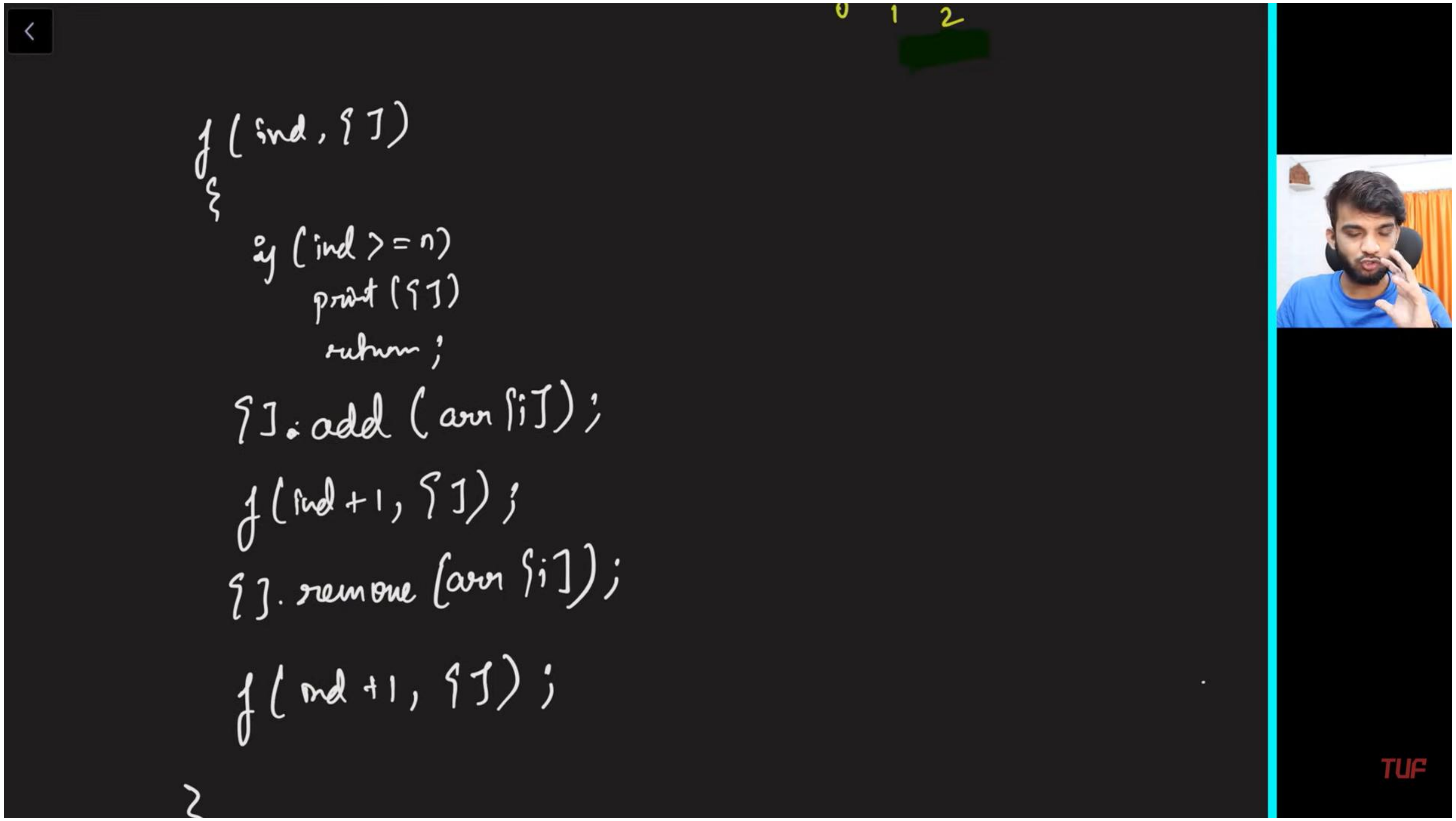
✗ ✗ ✗ {}



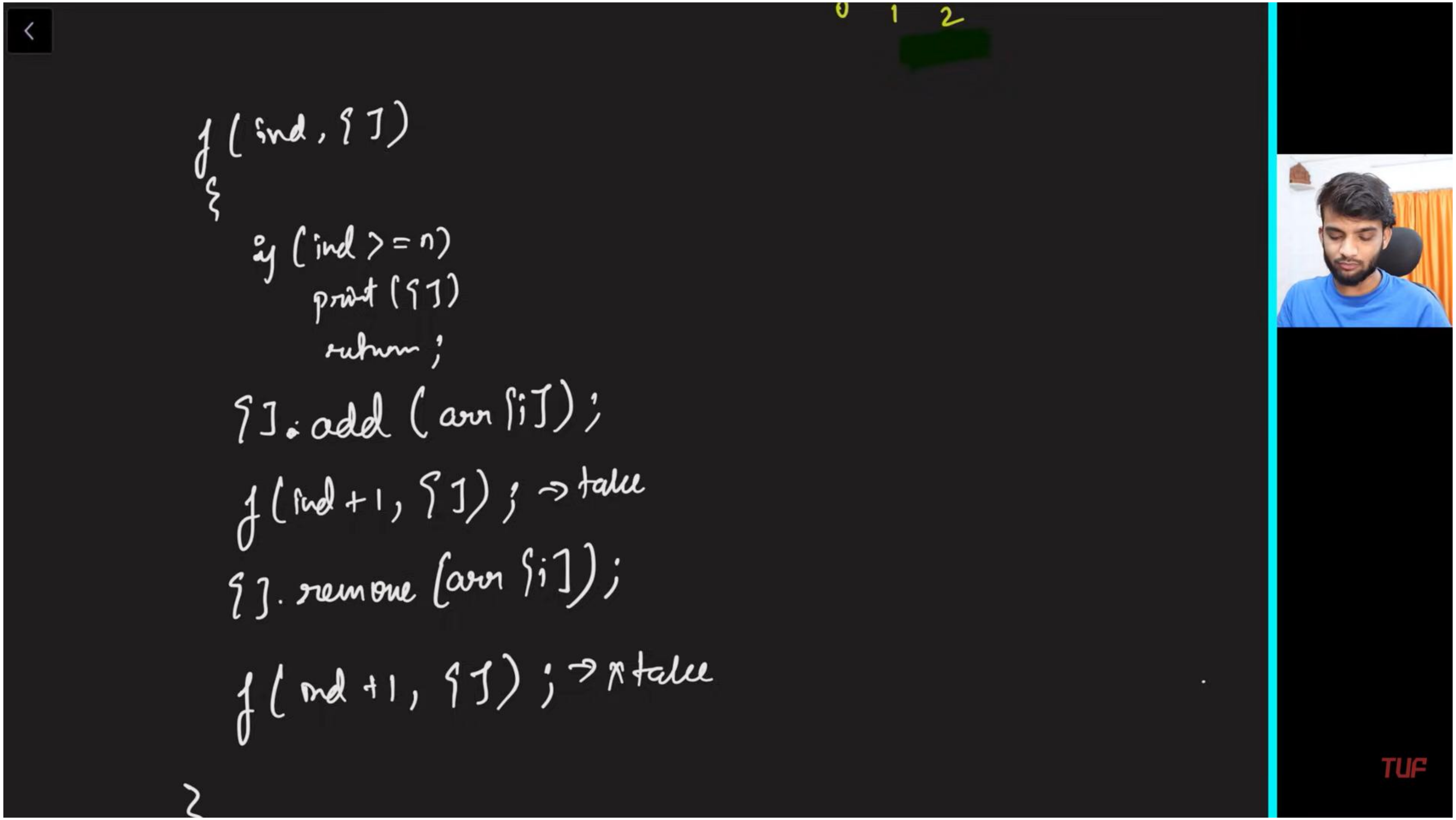
```
arr -> [3, 1, 2]
        0  1  2

f(ind, [])
{
    if (ind >= n)
        print([])
        return;

    [].add
```



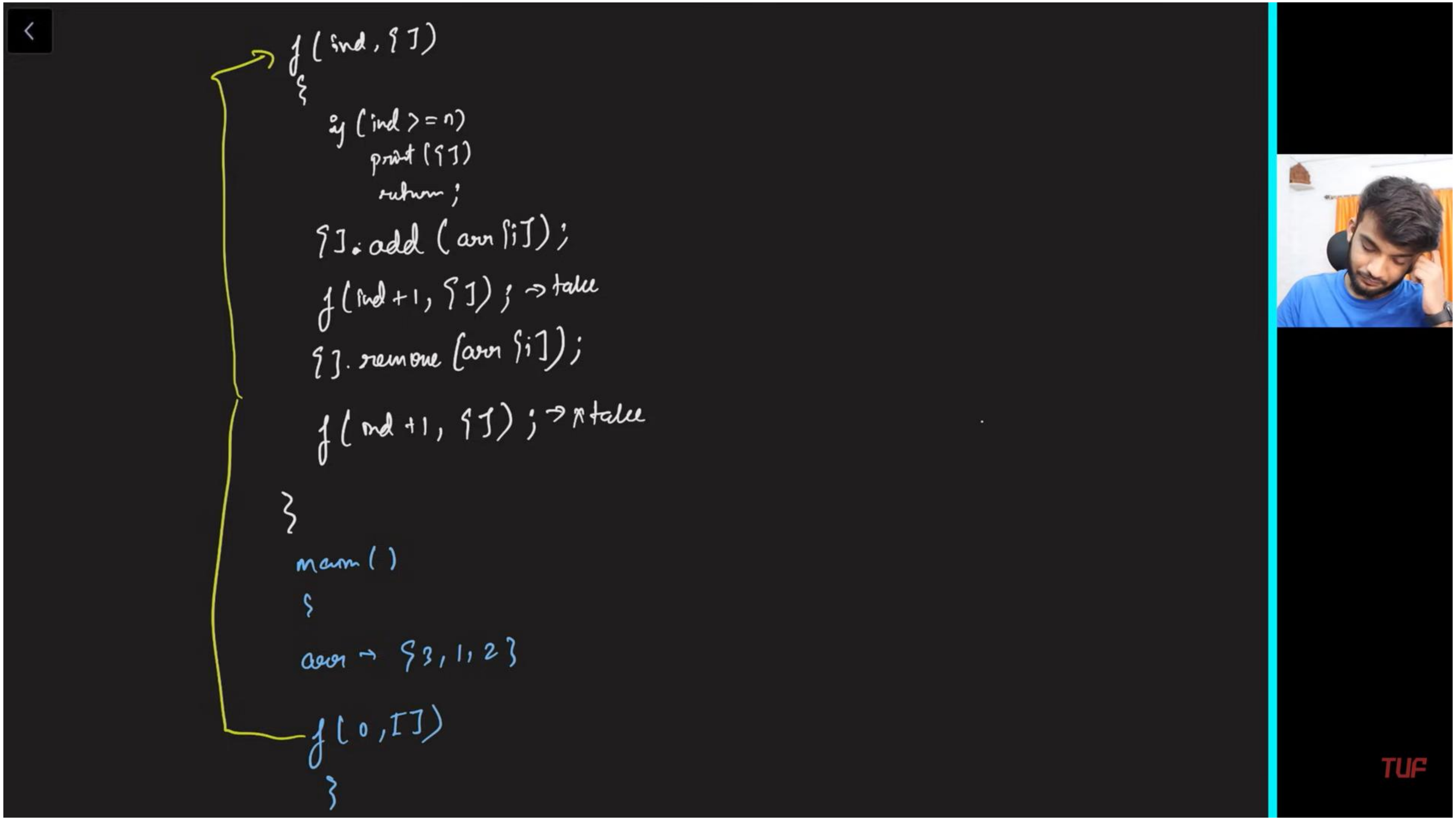
```
        return;
    [].add(arr[i]);
    f(ind+1, []);
    [].remove(arr[i]);

    f(,
```



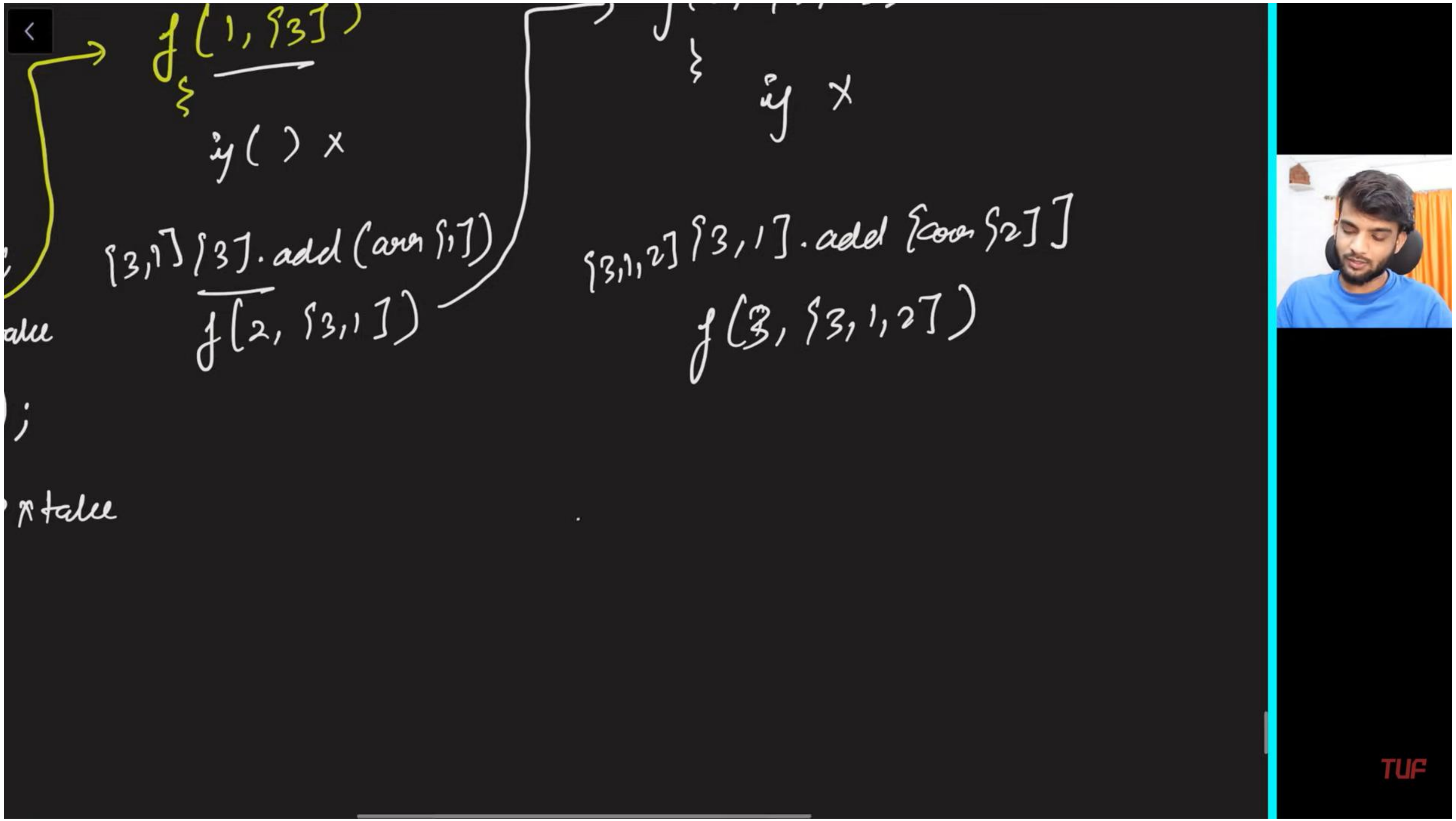
```
0   1   2

f(ind, [])
{
    if(ind >= n)
        print([])
        return;
    [].add(arr[i]);
    f(ind+1, []);
    [].remove(arr[i]);

    f(ind+1, []);
}
```



```
0   1   2

f(ind, [])
{
    if(ind >= n)
        print([])
        return;
    [].add(arr[i]);
    f(ind+1, []); -> take
    [].remove(arr[i]);
    f(ind+1, []); -> x take
}
```



```
f(ind, [])
{
    if(ind >= n)
        print([])
        return;
    [].add(arr[i]);
    f(ind+1, []); → take
    [].remove(arr[i]);
    f(ind+1, []); → not take
}
main()
{
arr → {3,1,2}
f(0,[])
}
```



alue

);

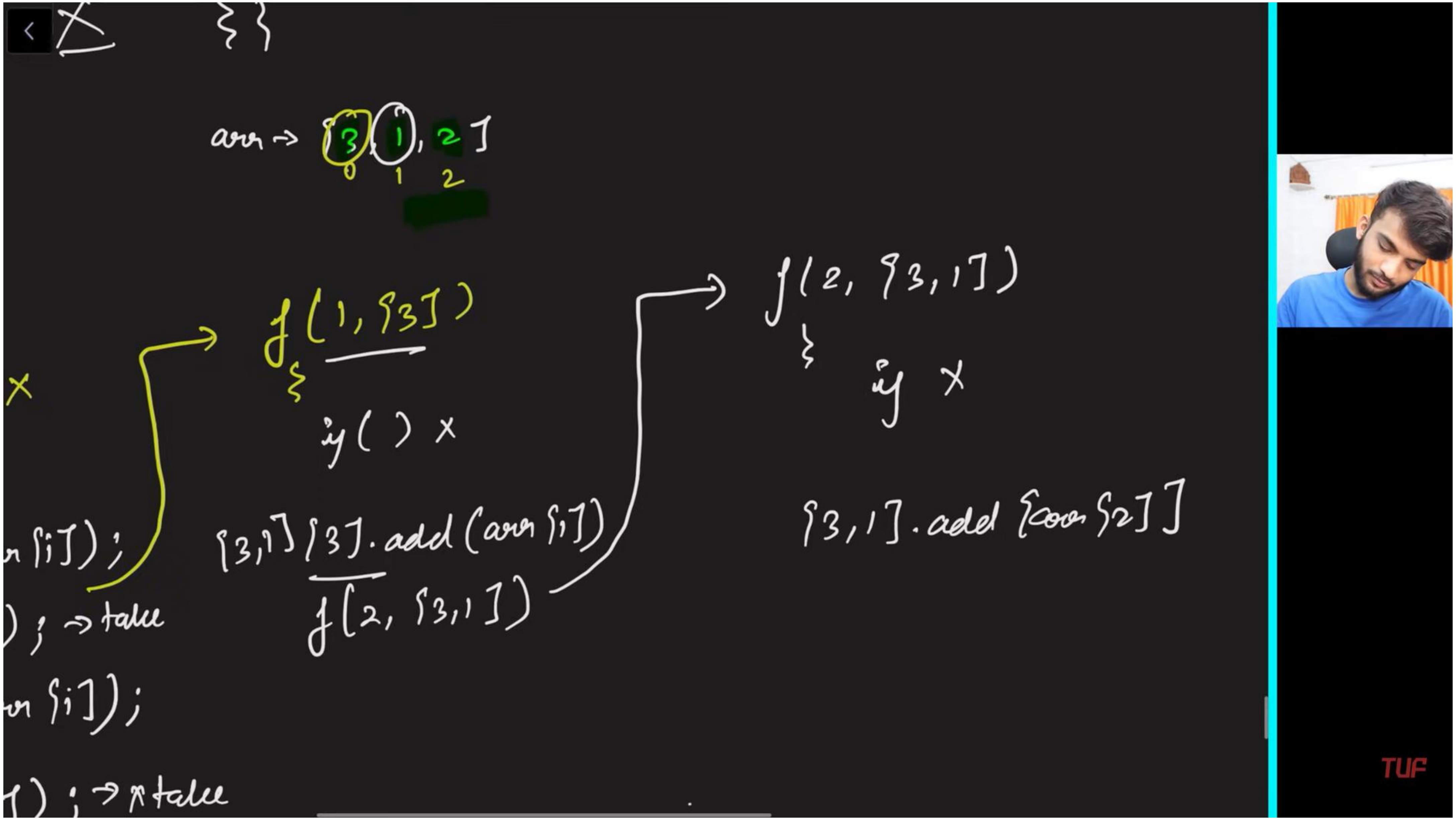
x take

f(1, [3])
{
if ( ) x

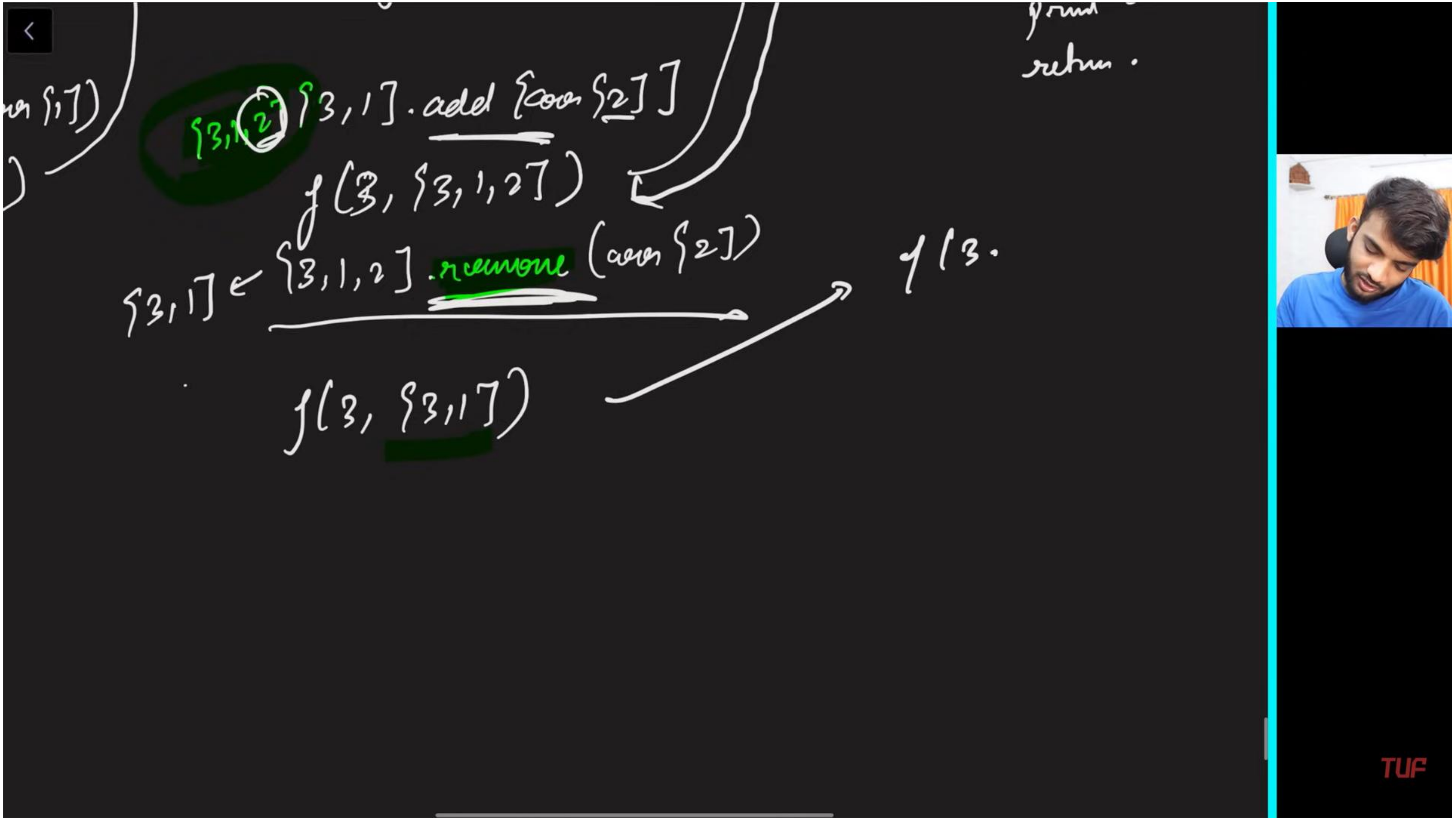
[3,1] [3].add(arr[1])
f[2, [3,1])

{
if x

[3,1,2] [3,1].add[arr[2]]
f(3, [3,1,2])



arr -> [3, 1, 2]
0 1 2
f(1, [3])
if ( ) x
[3,1] [3].add(arr[i])
f[2, [3,1]]
f(2, [3,1])
if x
[3,1].add [arr[2]]
r[i]);
); -> take
r[i]);
(); -> ~~n~~take
TUF



print

return.

rr [1])

)

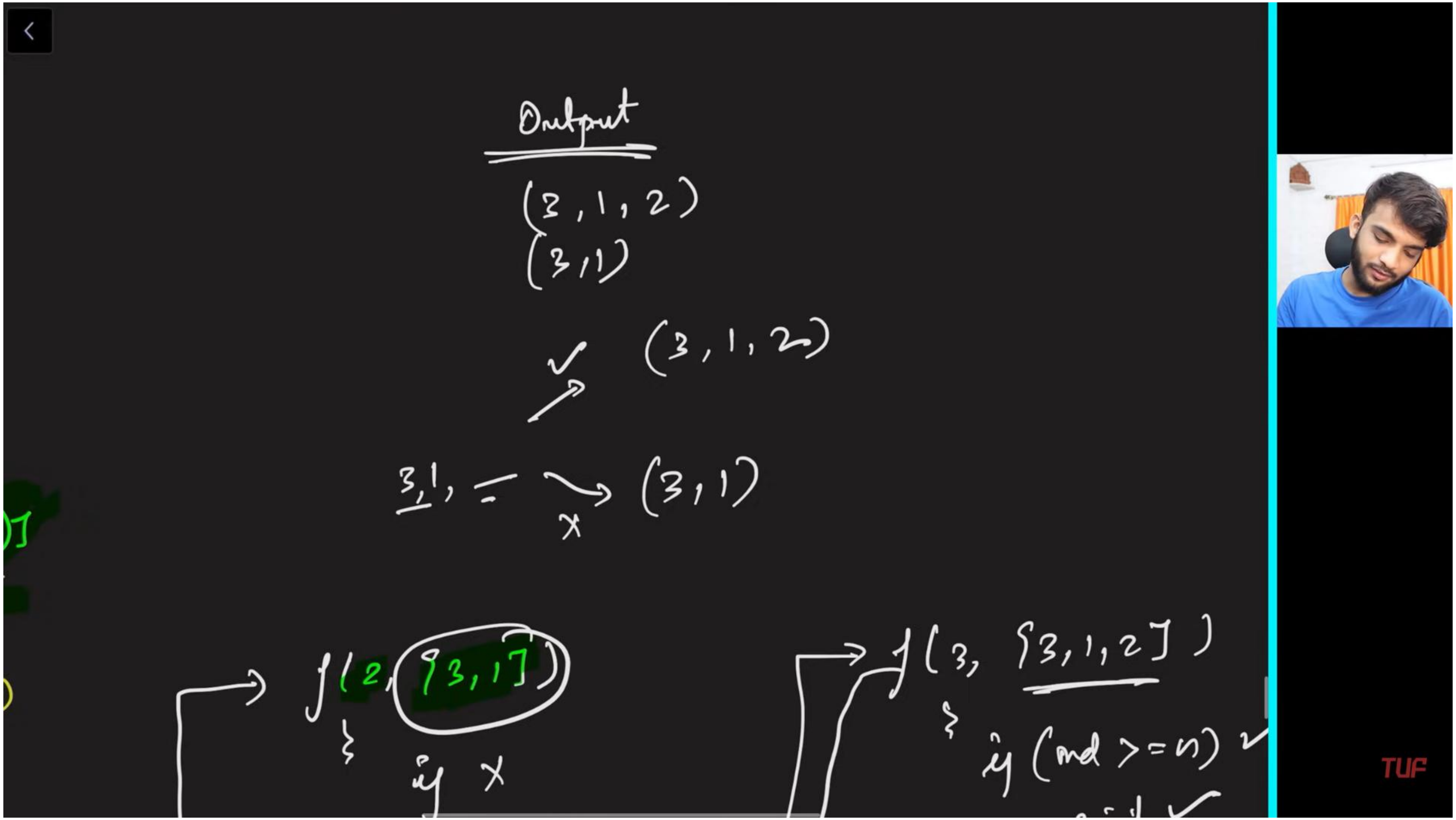
[3,1,2] [3,1].add [arr[2]]

f(3, [3,1,2])

[3,1] ← [3,1,2].remove (arr[2])

f(3, [3,1])

f(3.



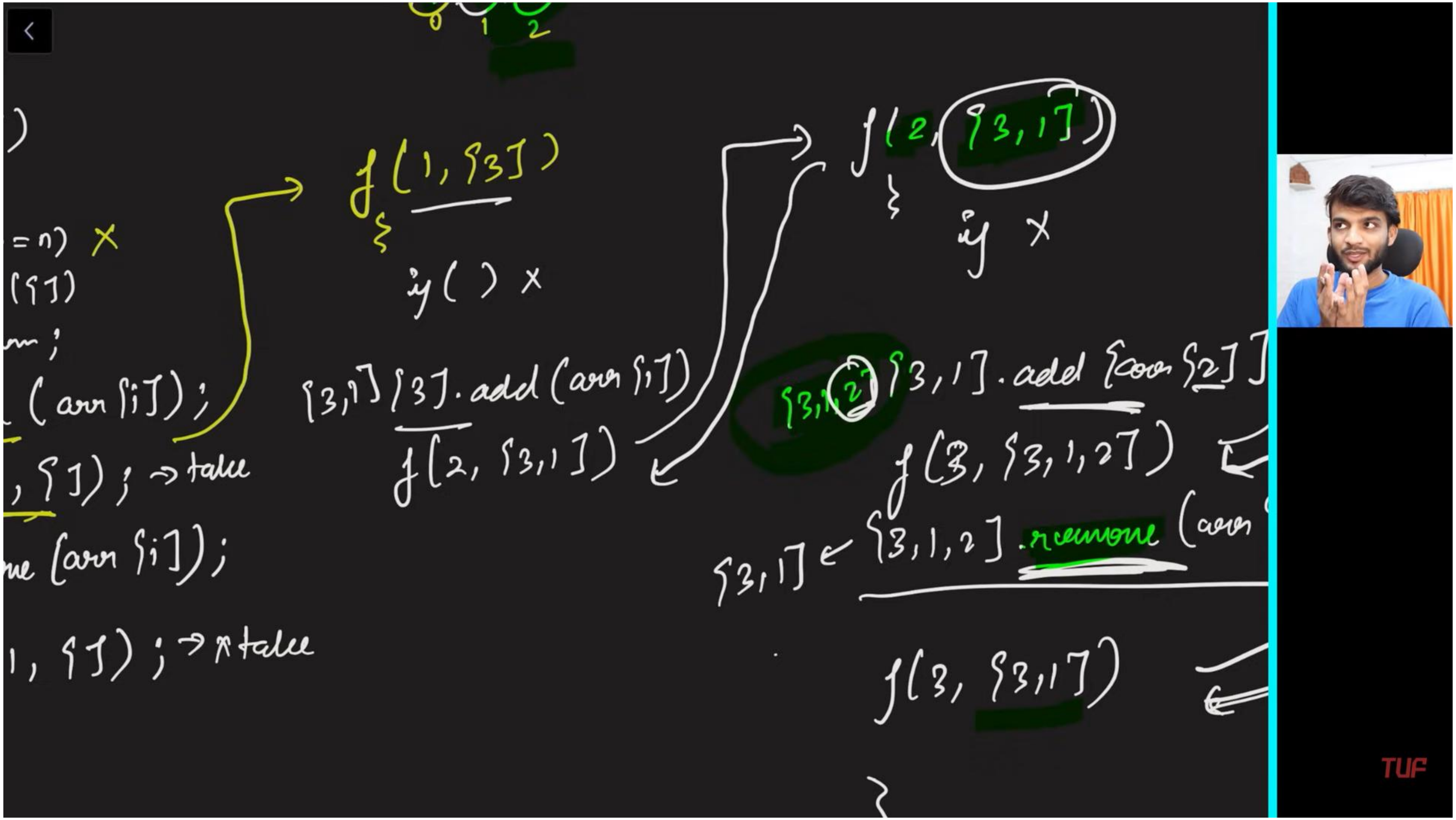
Output

(3, 1, 2)
(3, 1)

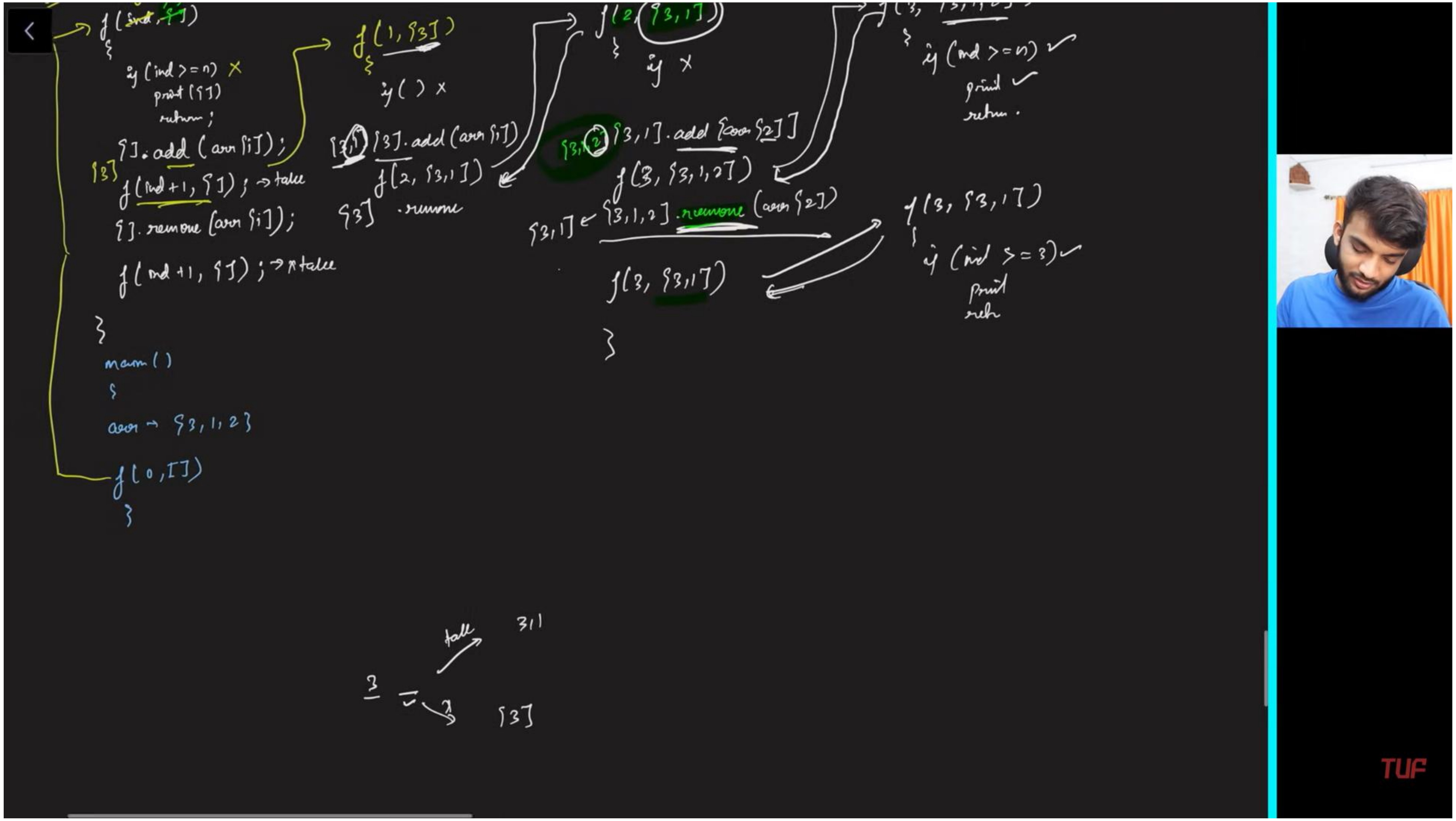
$$\begin{tikzcd} \checkmark & (3,1,2) \\ 3,1,\_ \arrow[ur] \arrow[r, "x"'] & (3,1) \end{tikzcd}$$

f(2, [3, 1])
{
if x

f(3, [3, 1, 2])
{
if (ind >= n)



0 1 2
)
= n) ×
([ ])
m;
(arr[i]);
, [ ]); → take
ne (arr[i]);
1, [ ]); → ~~n~~take
f(1, [3])
{
if( ) ×
[3,1] [3].add(arr[i])
f[2, [3,1])
f(2, [3,1])
{
if ×
[3,1,2] [3,1].add [arr[2]]
f(3, [3,1,2])
[3,1] ← [3,1,2].remove (arr
f(3, [3,1])
}
TUF



```
f(~~ind~~, ~~[]~~)
{
    if (ind >= n) X
        print([])
        return;
[3] [].add(arr[i]);
    f(ind+1, []); -> take
    [].remove(arr[i]);

    f(ind+1, []); -> not take

}
main()
{
    arr -> {3,1,2}

    f(0,[])
}
```

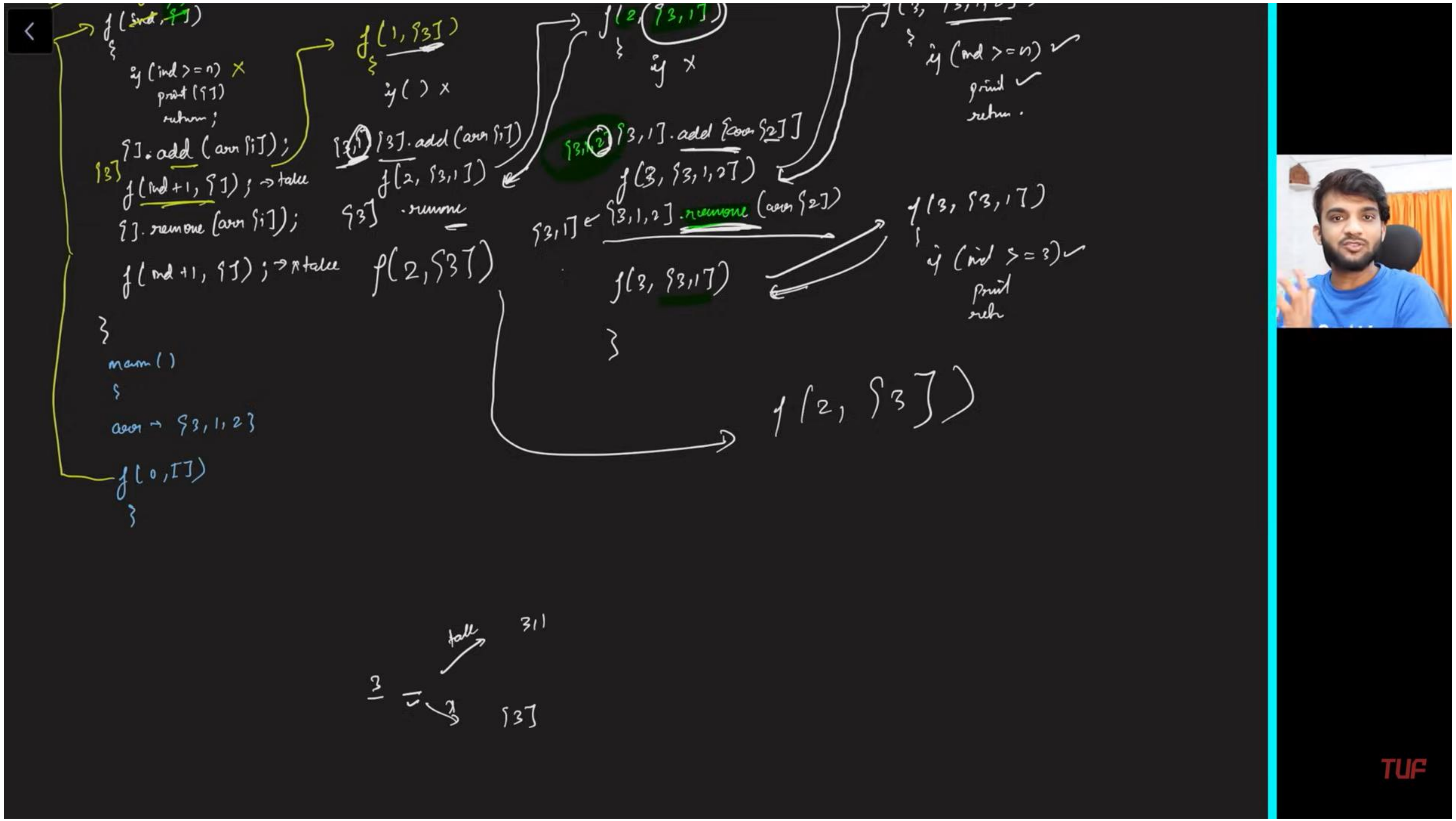
```
f(1, [3])
{
    if ( ) X
[3,1] [3].add(arr[i])
    f(2, [3,1])
[3] .remove
```

```
f(2, [3,1])
{
    if X
[3,1,2] [3,1].add[arr[2]]
    f(3, [3,1,2])
[3,1] <- [3,1,2].remove(arr[2])

    f(3, [3,1])

}
```

```
f(3, [3,1,2])
{
    if (ind >= n) ✓
        print ✓
        return.
```

```
f(3, [3,1])
{
    if (ind >= 3) ✓
        print
        ret
```

```
        take   3,1
3 =
        X      [3]
```



```
f(~~ind~~, [])
{
    if (ind >= n) ×
        print([])
        return;
[3] [].add(arr[i]);
    f(ind+1, []); → take
    [].remove(arr[i]);

    f(ind+1, []); → not take

}
main()
{
    arr → {3,1,2}

    f(0,[])
}
```

```
f(1, [3])
{
    if ( ) ×

    [3,1] [3].add(arr[i])
    f(2, [3,1])
    [3] .remove

    f(2, [3])
```

```
f(2, [3,1])
{
    if ×

    [3,1,2] [3,1].add[arr[2]]
    f(3, [3,1,2])
    [3,1] ← [3,1,2].remove(arr[2])

    f(3, [3,1])

}
```

```
f(3, [3,1,2])
{
    if (ind >= n) ✓
        print ✓
        return.
```

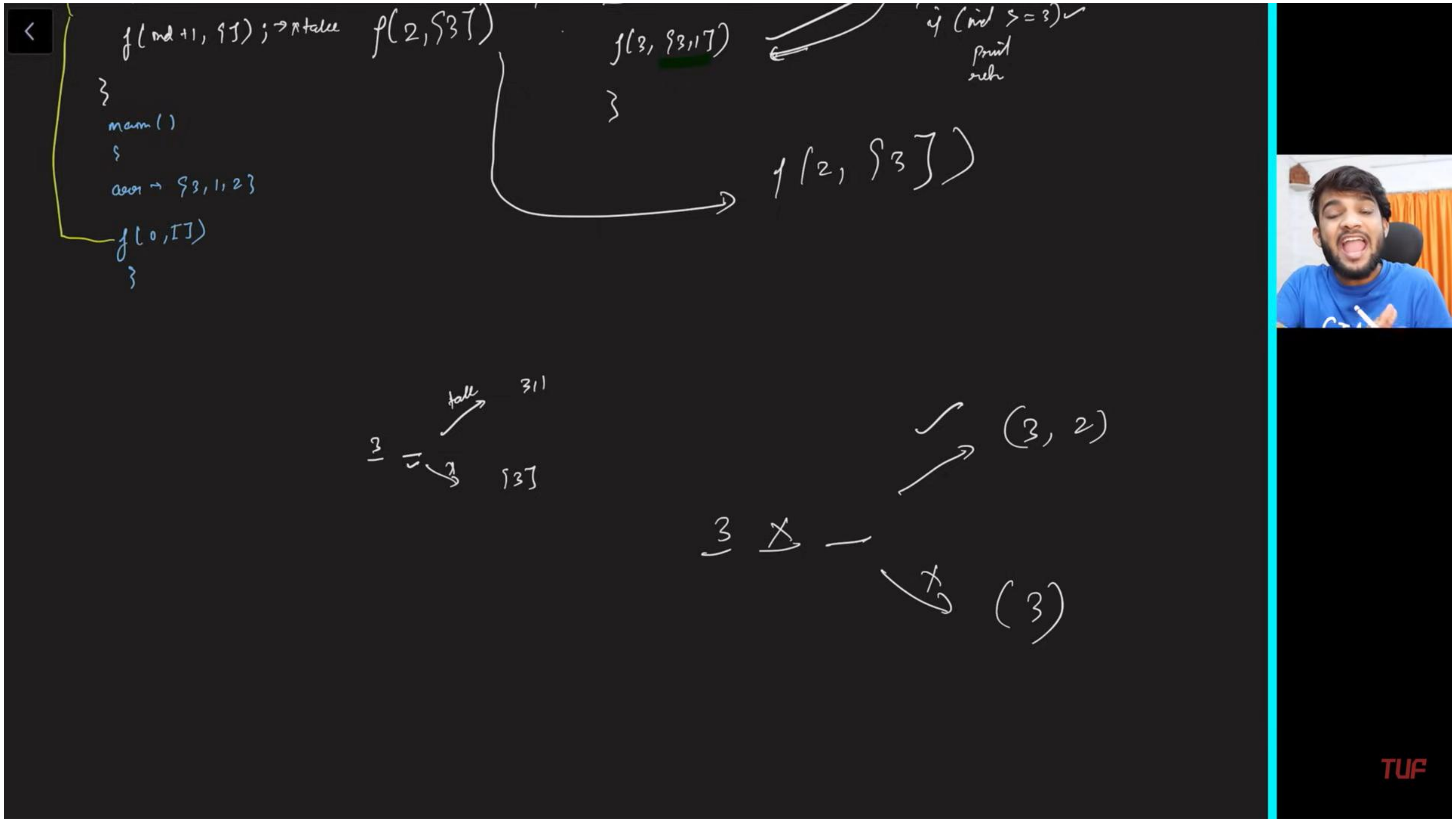
```
f(3, [3,1])
{
    if (ind >= 3) ✓
        print
        ret
```

```
f(2, [3])
```

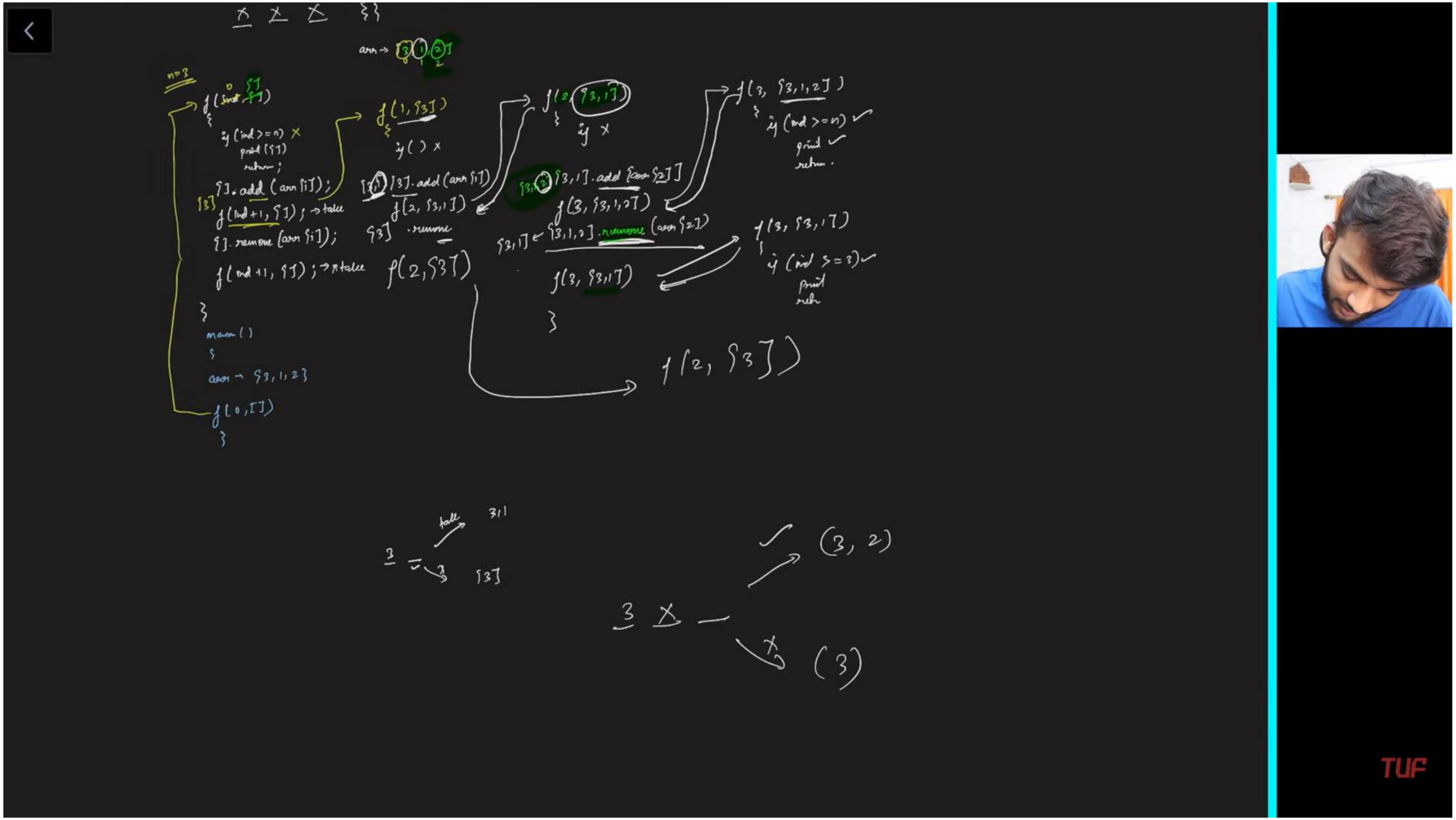
take 3,1

3

[3]



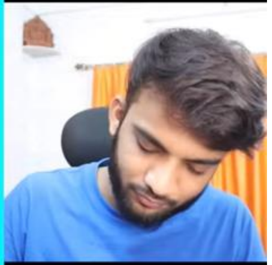
```
    f(ind+1, {}); → not take        f(2, {3})              f(3, {3,1})        if (ind >= 3)
                                                                                   print
}                                                          }                       return
main()
{                                                                    f(2, {3})
arr → {3,1,2}
f(0,[])
}
```

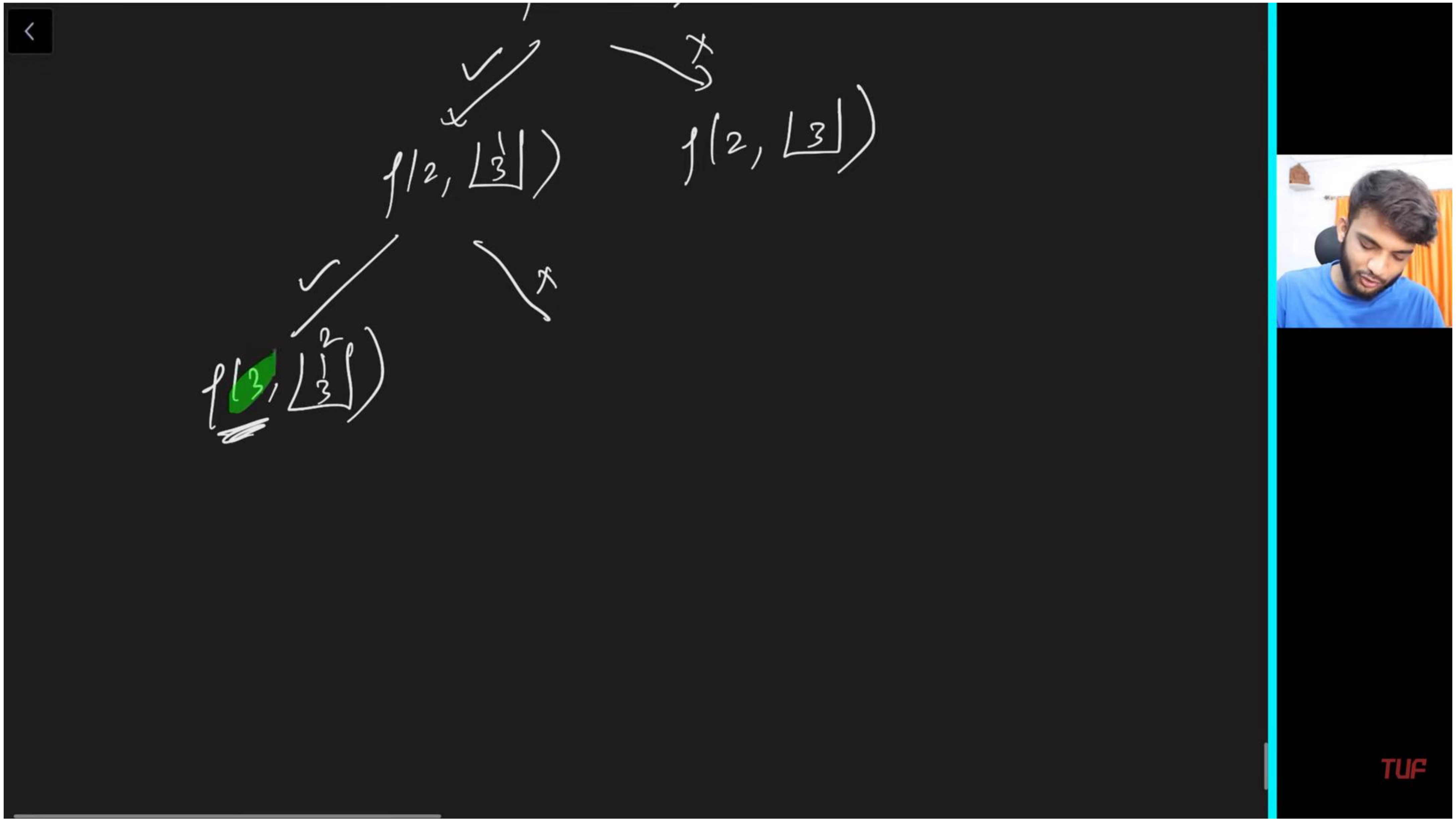


```
n=3
f(ind, [])
{
    if (ind >= n)
        print([])
        return;
    [].add(arr[i]);
    f(ind+1, []); -> take
    [].remove(arr[i]);
    f(ind+1, []); -> not take
}
main()
{
    arr -> {3,1,2}
    f(0,[])
}
```

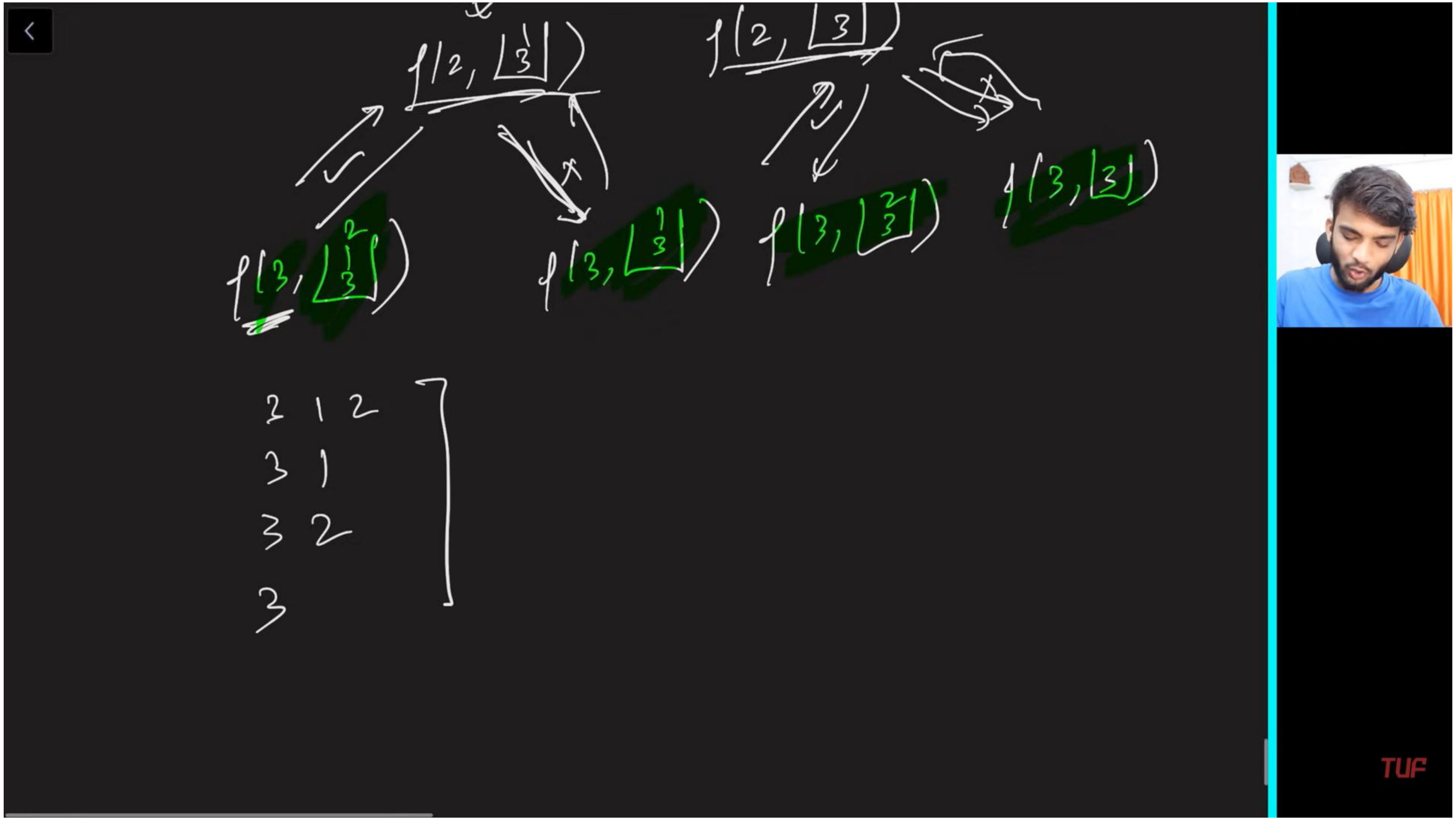
```
arr -> [3, 1, 2]

f(1, [3])
    if ( ) x
    [3,1]  [3].add(arr[1])
    f(2, [3,1])
    [3]  .remove
    f(2,[3])

f(2, [3,1])
    if x
    [3,1,2]  [3,1].add[arr[2]]
    f(3, [3,1,2])
    [3,1] <- [3,1,2].remove(arr[2])
    f(3, [3,1])
}

f(3, [3,1,2])
    if (ind >= n) ✓
    print ✓
    return.

f(3, [3,1])
    if (ind >= 3) ✓
    print
    ret

f(2, [3])
```

$$\begin{tikzcd}
 & f(0, \boxed{\ }) \arrow[dl, "\checkmark"'] \arrow[dr, "\times"] & \\
f(1, \boxed{3}) \arrow[d, "\checkmark"'] \arrow[dr, "\times"] & & f(1, \boxed{\ }) \\
f(2, \boxed{3}) & f(2, \boxed{3} &
\end{tikzcd}$$

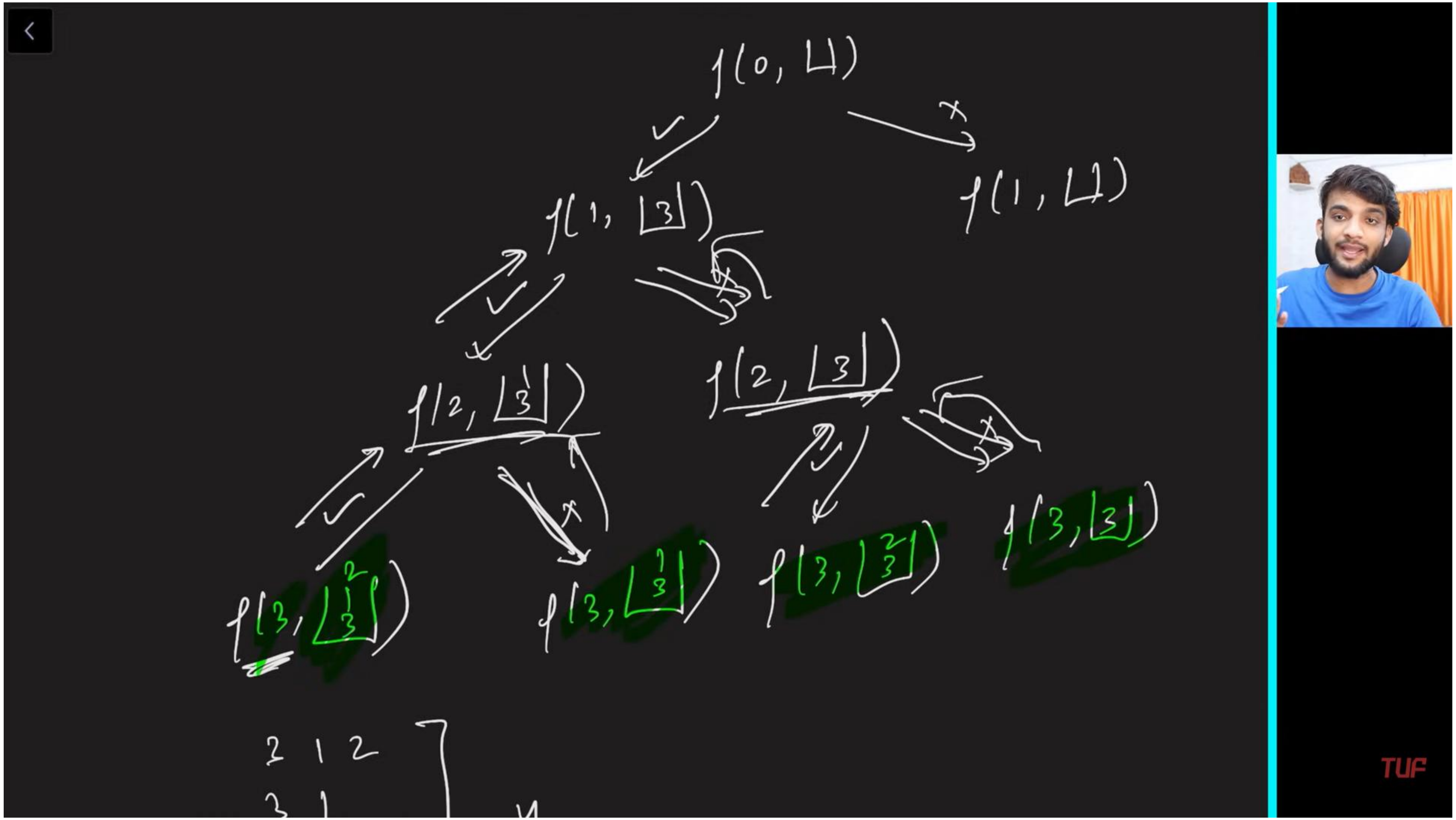




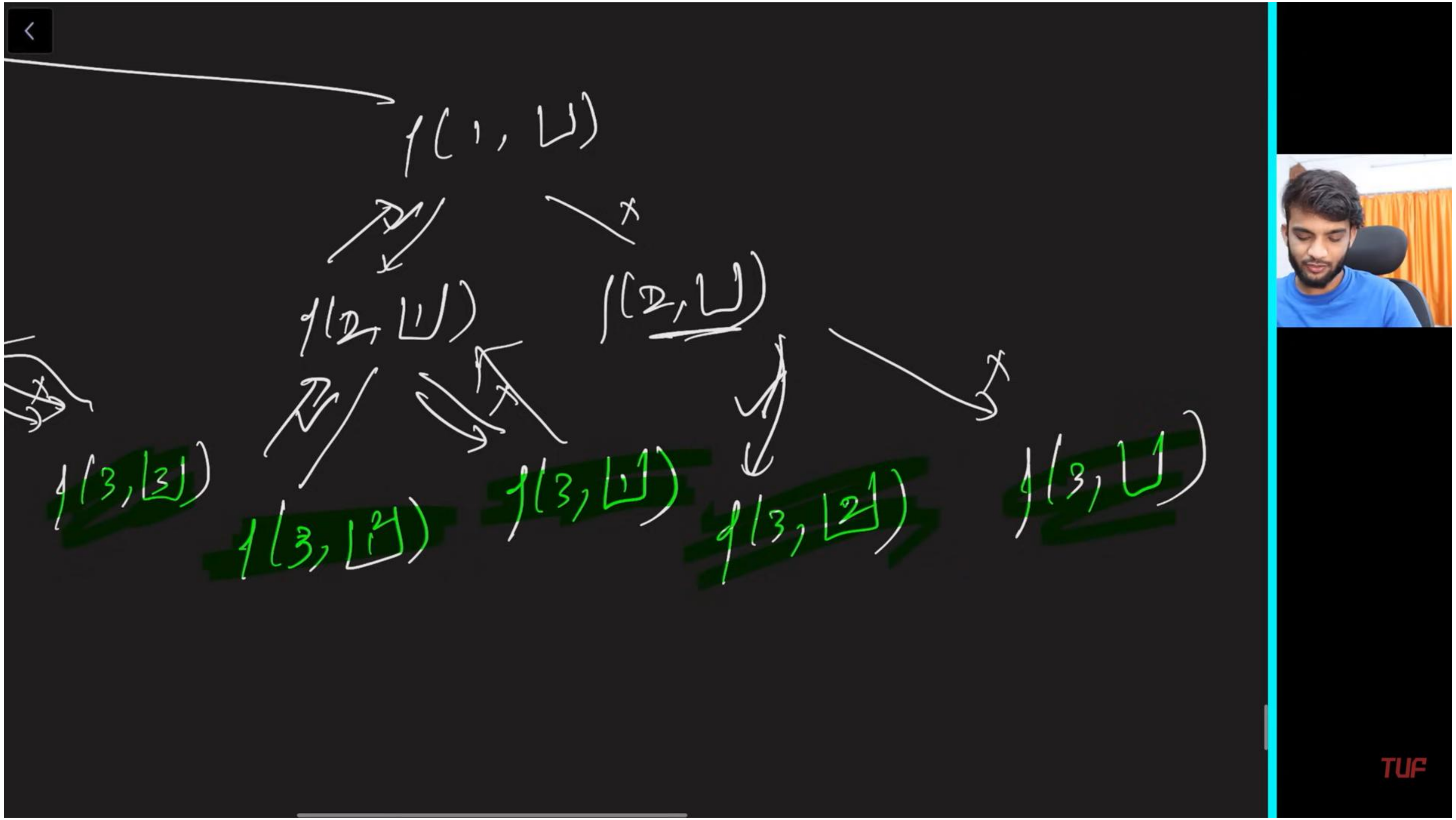
✓
×
f(2, [1 3])
f(2, [3])
✓
×
f(3, [2 1 3])
TUF

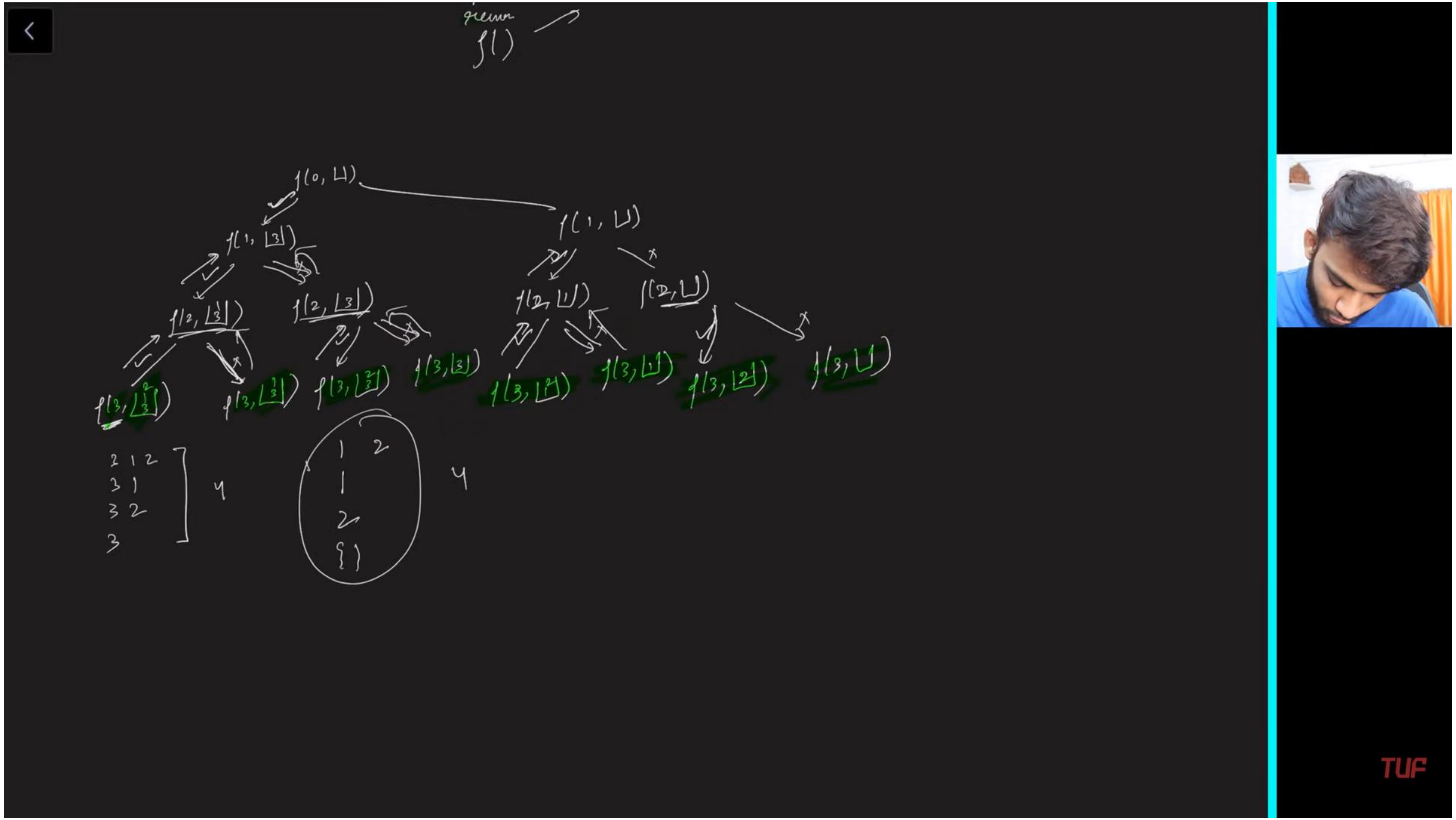


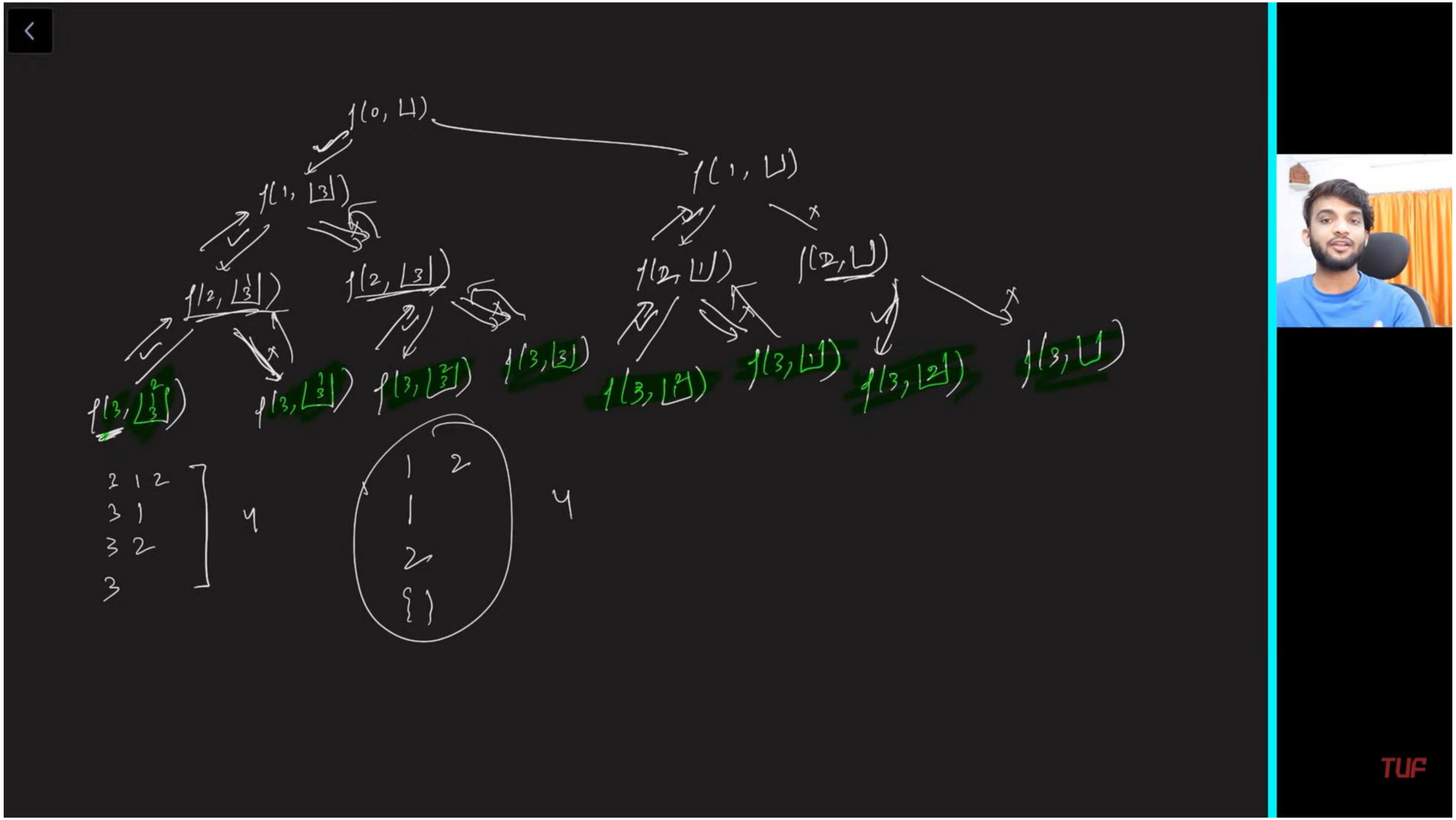
f(2, [3])
f(2, [3])
f(3, [2 1 3])
f(3, [1 3])
f(3, [2 3])
f(3, [3])
2 1 2
3 1
3 2
3
TUF



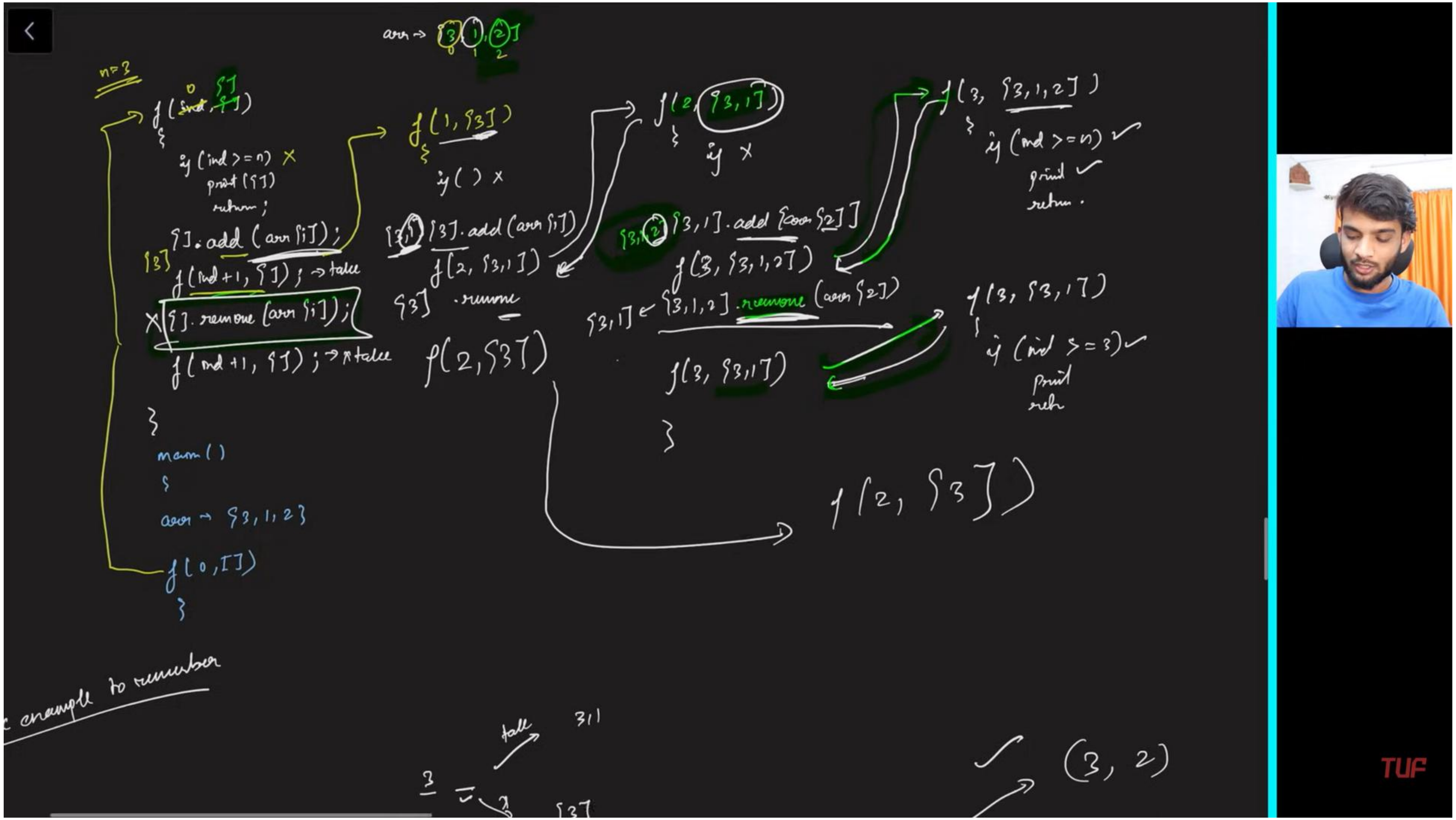
f(0, [])
✓
x
f(1, [3])
f(1, [])
✓
f(2, [1 3])
f(2, [3])
✓
x
x
x
f(3, [2 1 3])
f(3, [1 3])
f(3, [2 3])
f(3, [3])
3 1 2
3 1
TUF


f(1, [ ])
f(2, [1])
f(2, [ ])
f(3, [3])
f(3, [2 1])
f(3, [1])
f(3, [2])
f(3, [ ])
x
TUF

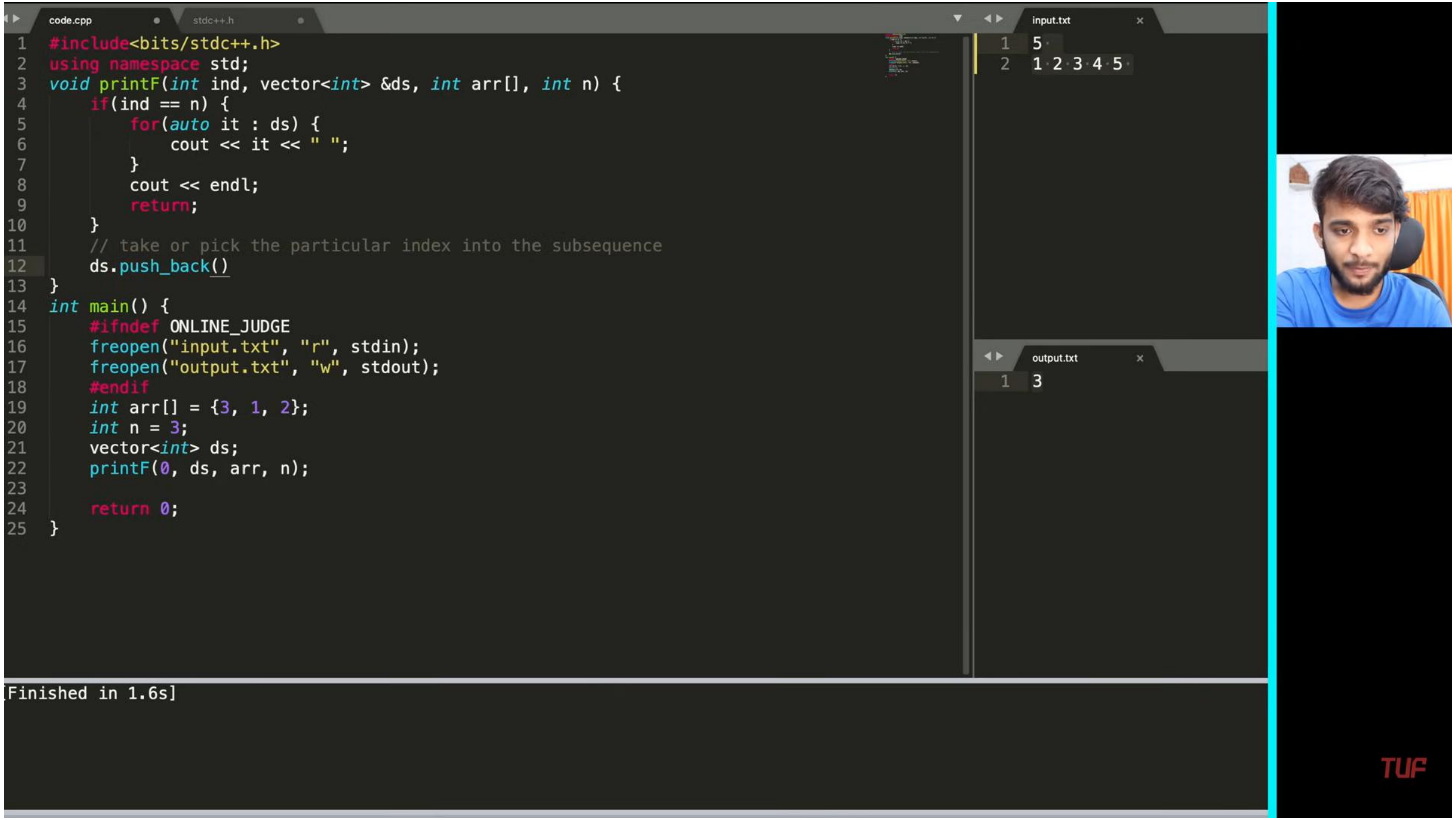

return
f()
f(0, [])
f(1, [3])
f(1, [])
f(2, [3])
f(2, [3])
f(2, [1])
f(2, [])
f(3, [2 1 3])
f(3, [1 3])
f(3, [2 3])
f(3, [3])
f(3, [2 1])
f(3, [1])
f(3, [2])
f(3, [])
2 1 2
3 1
3 2
3
4
1 2
1
2
{}
4
TUF



f(0, [])
f(1, [3])
f(1, [])
f(2, [3])
f(2, [3])
f(2, [])
f(2, [])
f(3, [2 3])
f(3, [1 3])
f(3, [2 3])
f(3, [3])
f(3, [2])
f(3, [1])
f(3, [2])
f(3, [])
2 1 2
3 1
3 2
3
4
1 2
1
2
{ }
4
TUF



```
arr -> [3, 1, 2]
         0  1  2

n=3

f(~~ind~~ 0, ~~[]~~ [])
{
   if (ind >= n) X
      print([])
      return;
   [].add(arr[i]);
[3]
   f(ind+1, []); -> take
X  [].remove(arr[i]);
   f(ind+1, []); -> not take
}

main()
{
   arr -> {3, 1, 2}
   f(0, [])
}

f(1, [3])
{
   if ( ) X
[3,1] [3].add(arr[i])
   f(2, [3,1])
[3]  .remove
   f(2, [3])

f(2, [3,1])
{
   if X
[3,2] [3,1].add(arr[2])
   f(3, [3,1,2])
[3,1] <- [3,1,2].remove(arr[2])
   f(3, [3,1])
}

f(3, [3,1,2])
{
   if (ind >= n) ✓
      print ✓
      return.

f(3, [3,1])
{
   if (ind >= 3) ✓
      print
      ret

f(2, [3])

example to remember

          take   3,1
3  =
          X      [3]

✓   (3, 2)
```

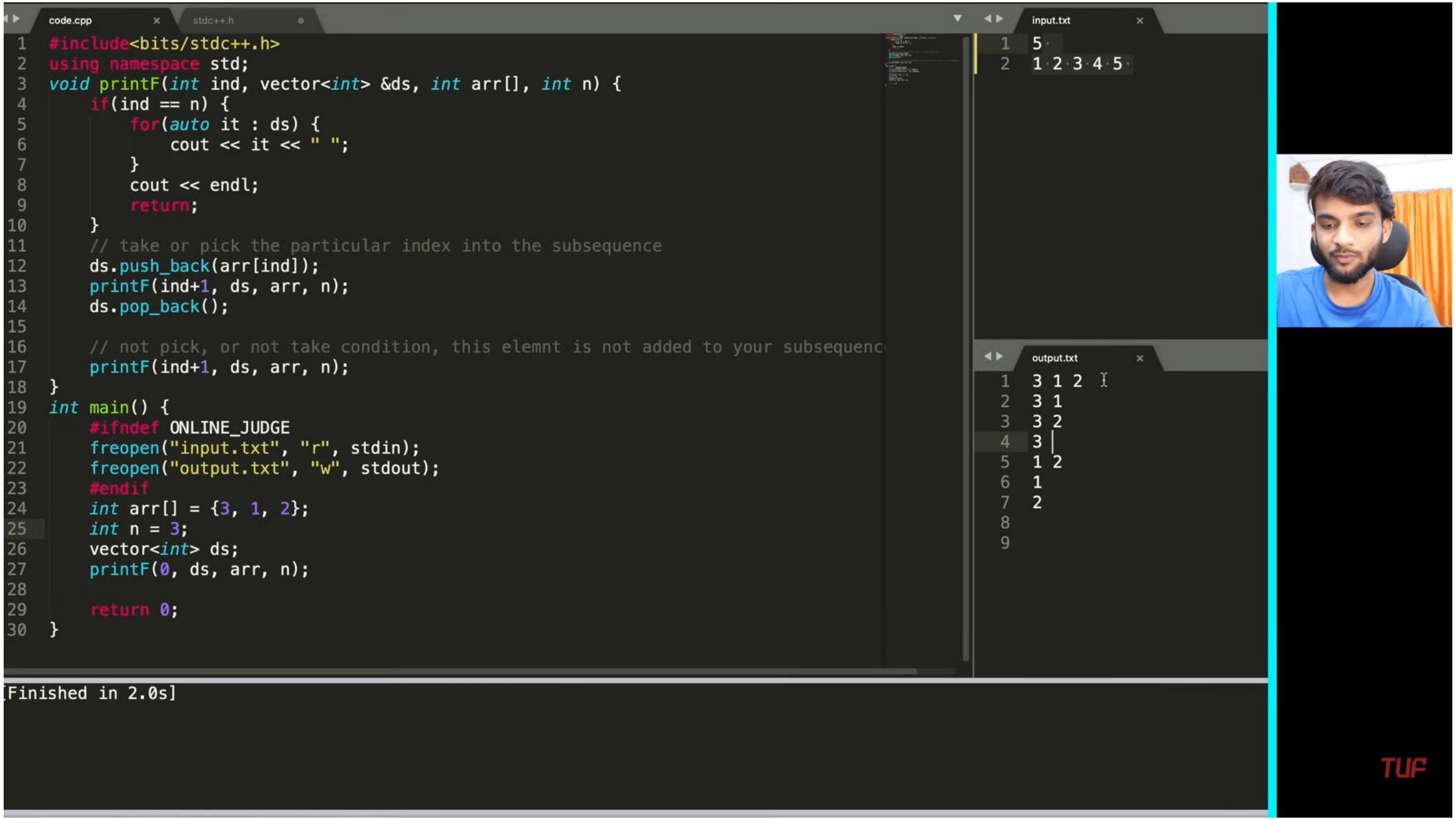


```cpp
#include<bits/stdc++.h>
using namespace std;
void printF(int ind, vector<int> &ds, int arr[], int n) {
    if(ind == n) {
        for(auto it : ds) {
            cout << it << " ";
        }
        cout << endl;
        return;
    }
    // take or pick the particular index into the subsequence
    ds.push_back()
}
int main() {
    #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    #endif
    int arr[] = {3, 1, 2};
    int n = 3;
    vector<int> ds;
    printF(0, ds, arr, n);

    return 0;
}
```

```
5
1 2 3 4 5
```

```
3
```

```
[Finished in 1.6s]
```

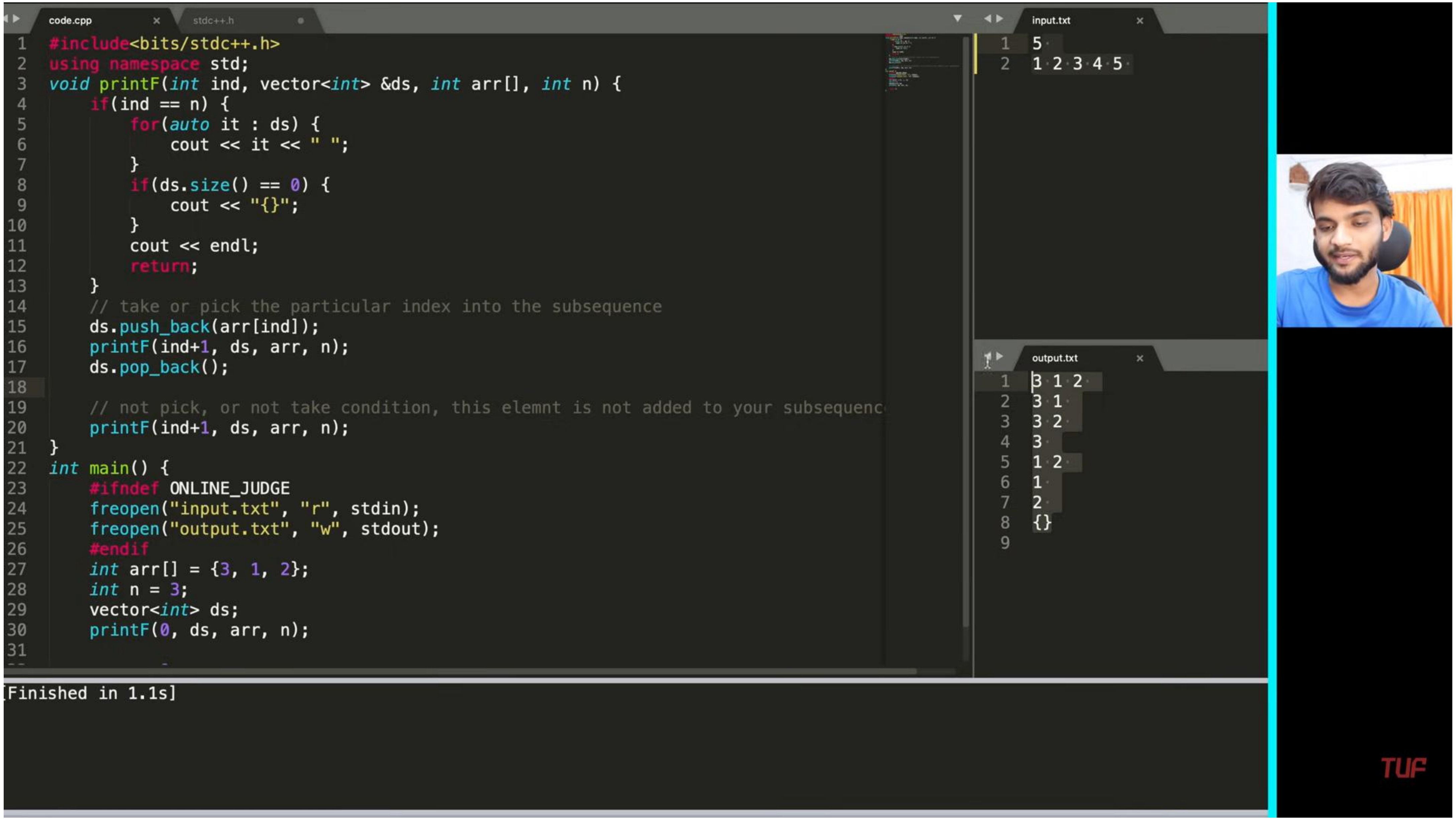


```cpp
#include<bits/stdc++.h>
using namespace std;
void printF(int ind, vector<int> &ds, int arr[], int n) {
    if(ind == n) {
        for(auto it : ds) {
            cout << it << " ";
        }
        cout << endl;
        return;
    }
    // take or pick the particular index into the subsequence
    ds.push_back(arr[ind]);
    printF(ind+1, ds, arr, n);
    ds.pop_back();

    // not pick, or not take condition, this elemnt is not added to your subsequenc
    printF(ind+1, ds, arr, n);
}
int main() {
    #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    #endif
    int arr[] = {3, 1, 2};
    int n = 3;
    vector<int> ds;
    printF(0, ds, arr, n);

    return 0;
}
```

```
5
1 2 3 4 5
```

```
3 1 2
3 1
3 2
3
1 2
1
2
```

```
[Finished in 2.0s]
```

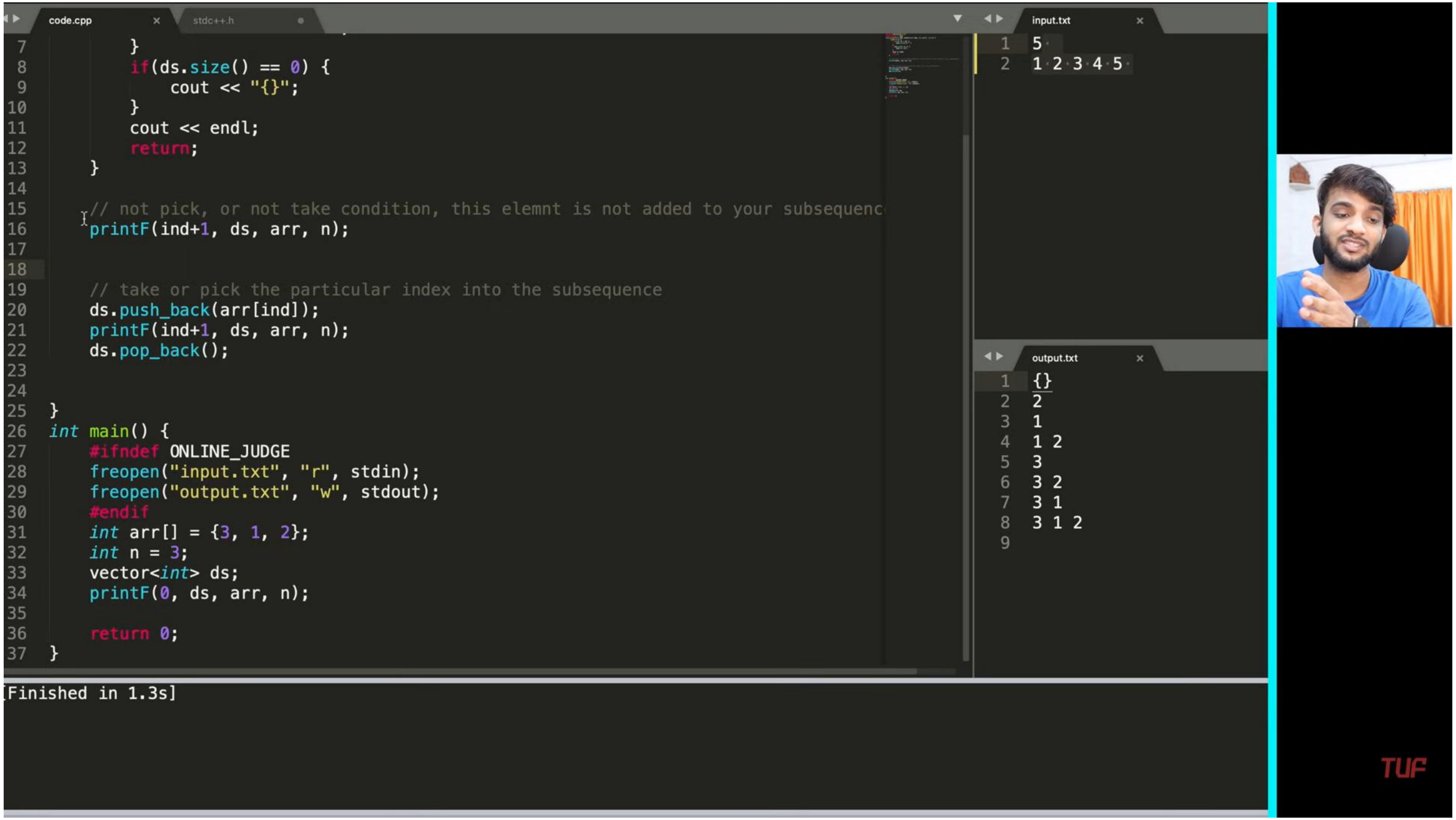


```cpp
#include<bits/stdc++.h>
using namespace std;
void printF(int ind, vector<int> &ds, int arr[], int n) {
    if(ind == n) {
        for(auto it : ds) {
            cout << it << " ";
        }
        if(ds.size() == 0) {
            cout << "{}";
        }
        cout << endl;
        return;
    }
    // take or pick the particular index into the subsequence
    ds.push_back(arr[ind]);
    printF(ind+1, ds, arr, n);
    ds.pop_back();

    // not pick, or not take condition, this elemnt is not added to your subsequenc
    printF(ind+1, ds, arr, n);
}
int main() {
    #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    #endif
    int arr[] = {3, 1, 2};
    int n = 3;
    vector<int> ds;
    printF(0, ds, arr, n);
```

```
5
1 2 3 4 5
```

```
3 1 2
3 1
3 2
3
1 2
1
2
{}
```

```
[Finished in 1.1s]
```



code.cpp · stdc++.h

```cpp
        }
        if(ds.size() == 0) {
            cout << "{}";
        }
        cout << endl;
        return;
    }

    // not pick, or not take condition, this elemnt is not added to your subsequenc
    printF(ind+1, ds, arr, n);


    // take or pick the particular index into the subsequence
    ds.push_back(arr[ind]);
    printF(ind+1, ds, arr, n);
    ds.pop_back();


}
int main() {
    #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    #endif
    int arr[] = {3, 1, 2};
    int n = 3;
    vector<int> ds;
    printF(0, ds, arr, n);

    return 0;
}
```

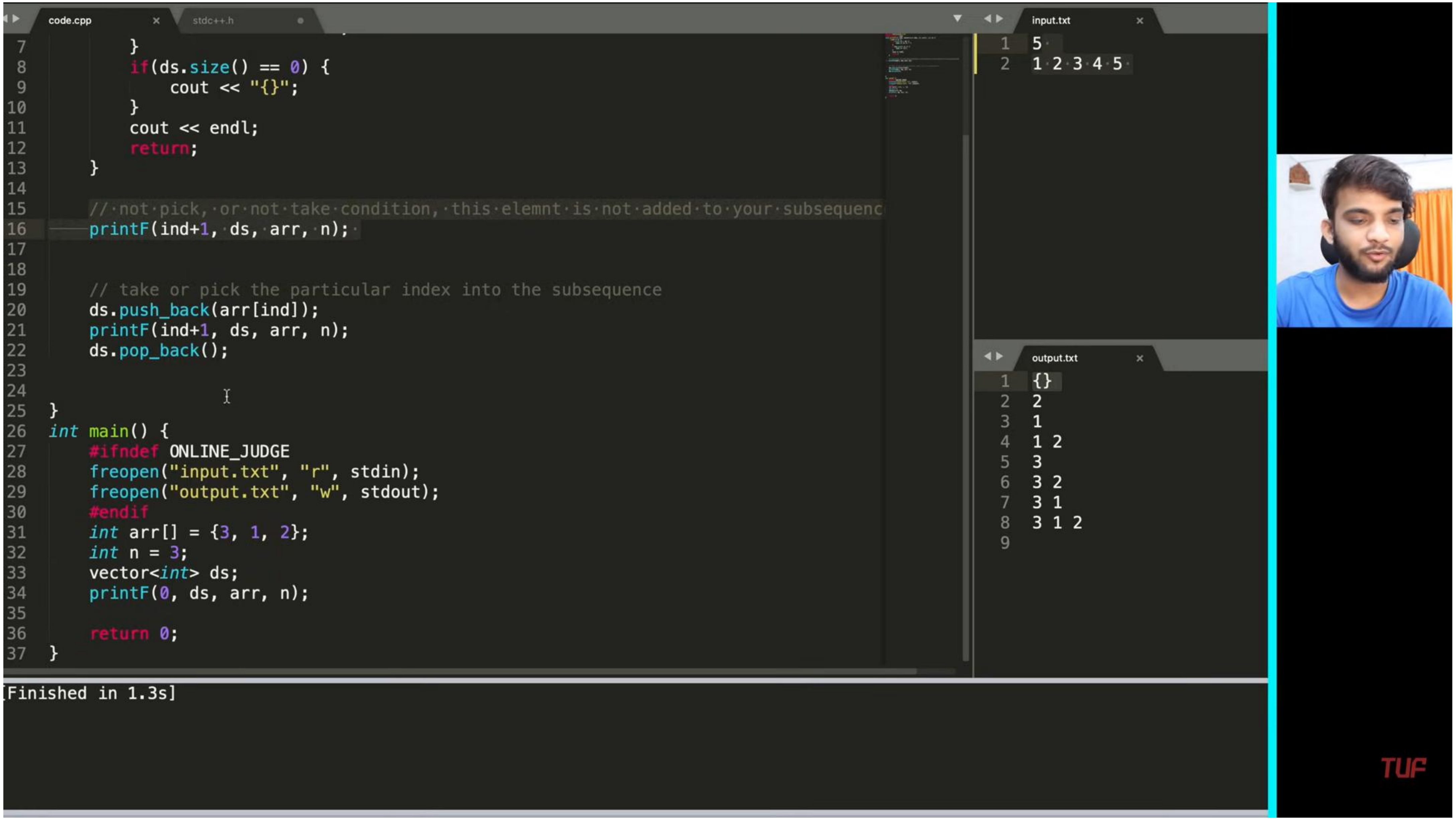
input.txt

```
5
1 2 3 4 5
```

output.txt

```
{}
2
1
1 2
3
3 2
3 1
3 1 2

```

```
[Finished in 1.3s]
```

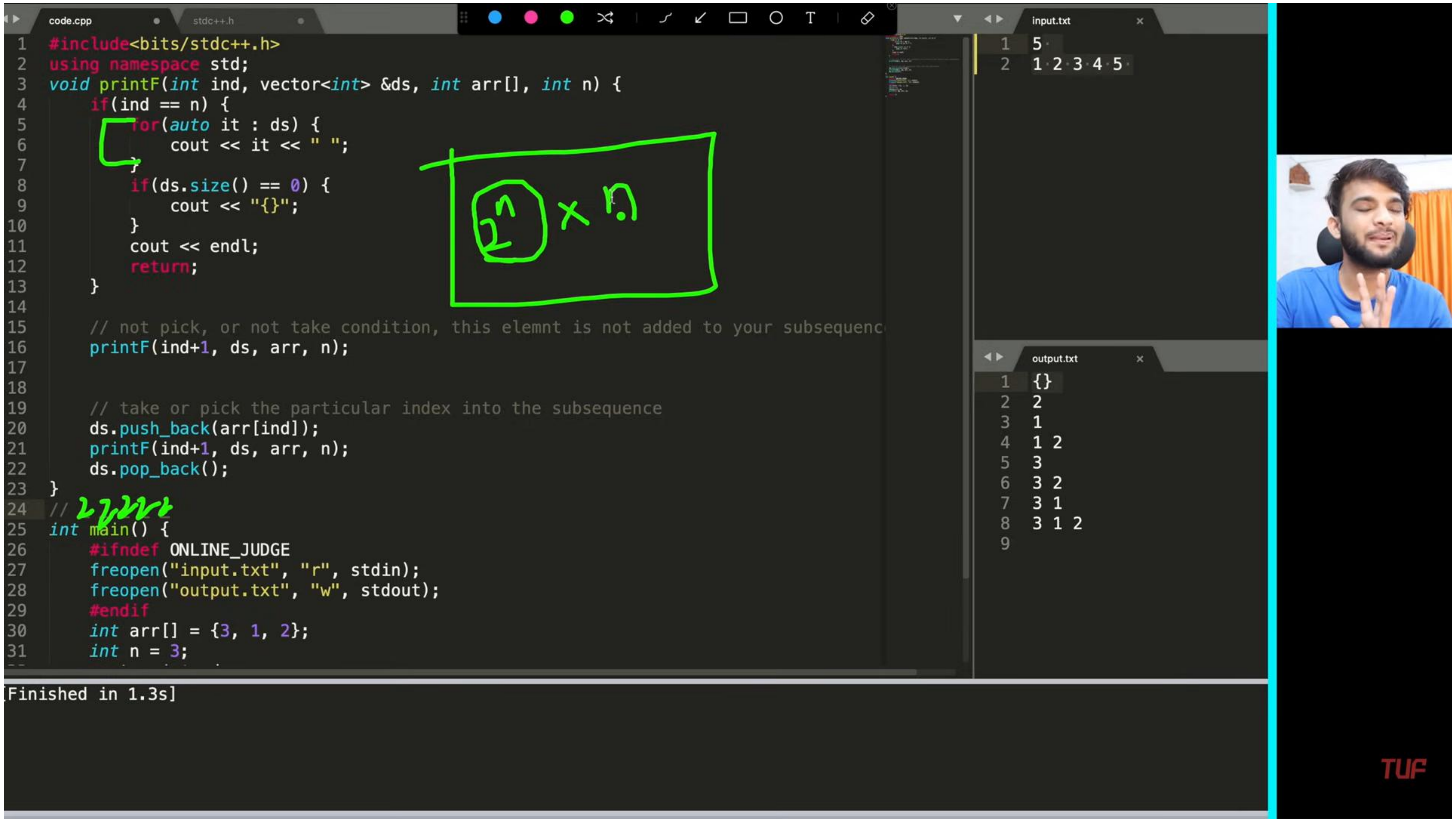


```cpp
        }
        if(ds.size() == 0) {
            cout << "{}";
        }
        cout << endl;
        return;
    }

    // not pick, or not take condition, this elemnt is not added to your subsequenc
    printF(ind+1, ds, arr, n);


    // take or pick the particular index into the subsequence
    ds.push_back(arr[ind]);
    printF(ind+1, ds, arr, n);
    ds.pop_back();


}
int main() {
    #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    #endif
    int arr[] = {3, 1, 2};
    int n = 3;
    vector<int> ds;
    printF(0, ds, arr, n);

    return 0;
}
```

```
5
1 2 3 4 5
```

```
{}
2
1
1 2
3
3 2
3 1
3 1 2
```

```
[Finished in 1.3s]
```



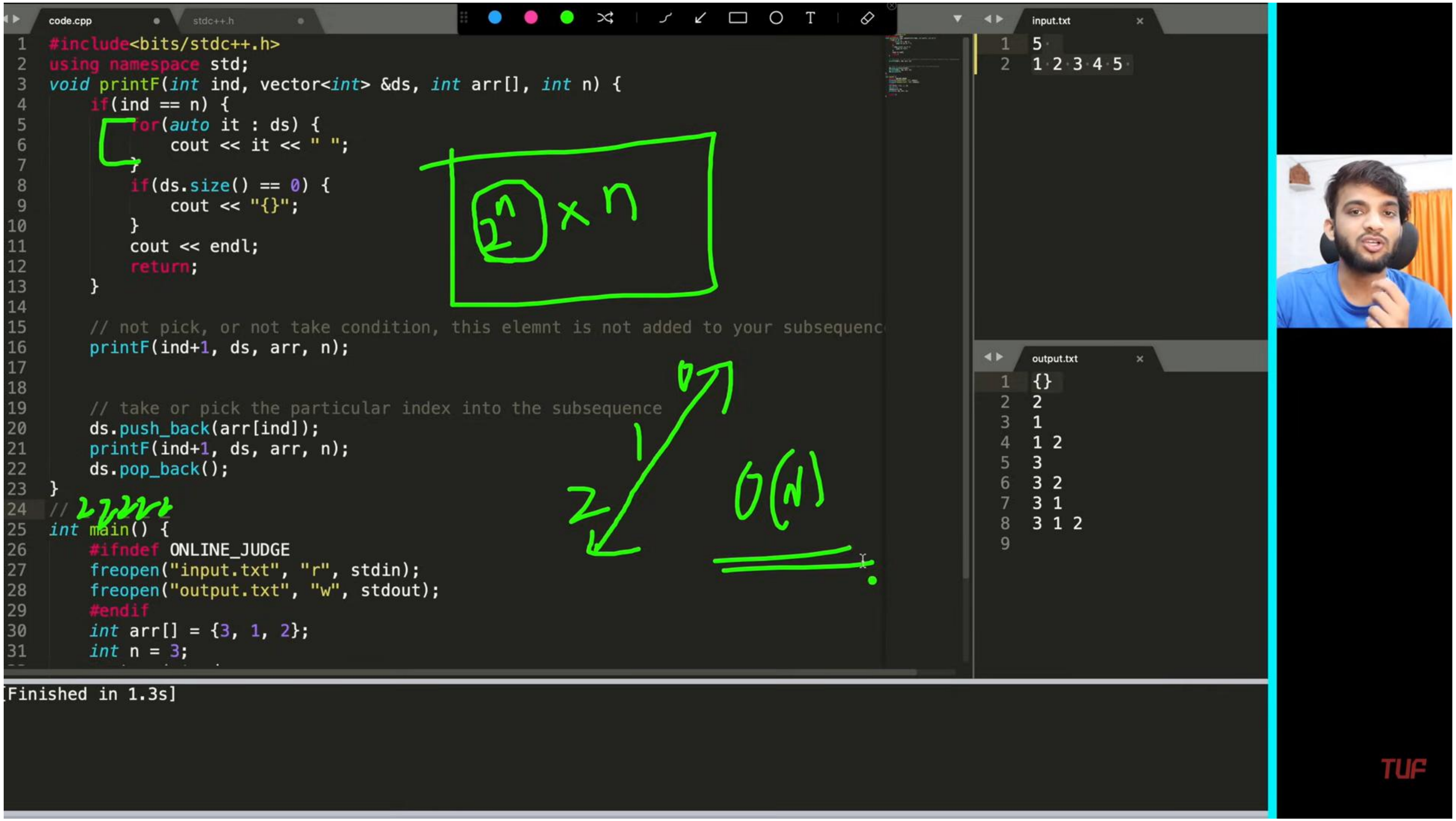
```
#include<bits/stdc++.h>
using namespace std;
void printF(int ind, vector<int> &ds, int arr[], int n) {
    if(ind == n) {
        for(auto it : ds) {
            cout << it << " ";
        }
        if(ds.size() == 0) {
            cout << "{}";
        }
        cout << endl;
        return;
    }

    // not pick, or not take condition, this elemnt is not added to your subsequenc
    printF(ind+1, ds, arr, n);


    // take or pick the particular index into the subsequence
    ds.push_back(arr[ind]);
    printF(ind+1, ds, arr, n);
    ds.pop_back();
}
//
int main() {
    #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    #endif
    int arr[] = {3, 1, 2};
    int n = 3;
```

$\boxed{2^n} \times n$

input.txt
```
5
1 2 3 4 5
```

output.txt
```
{}
2
1
1 2
3
3 2
3 1
3 1 2
```

```
[Finished in 1.3s]
```

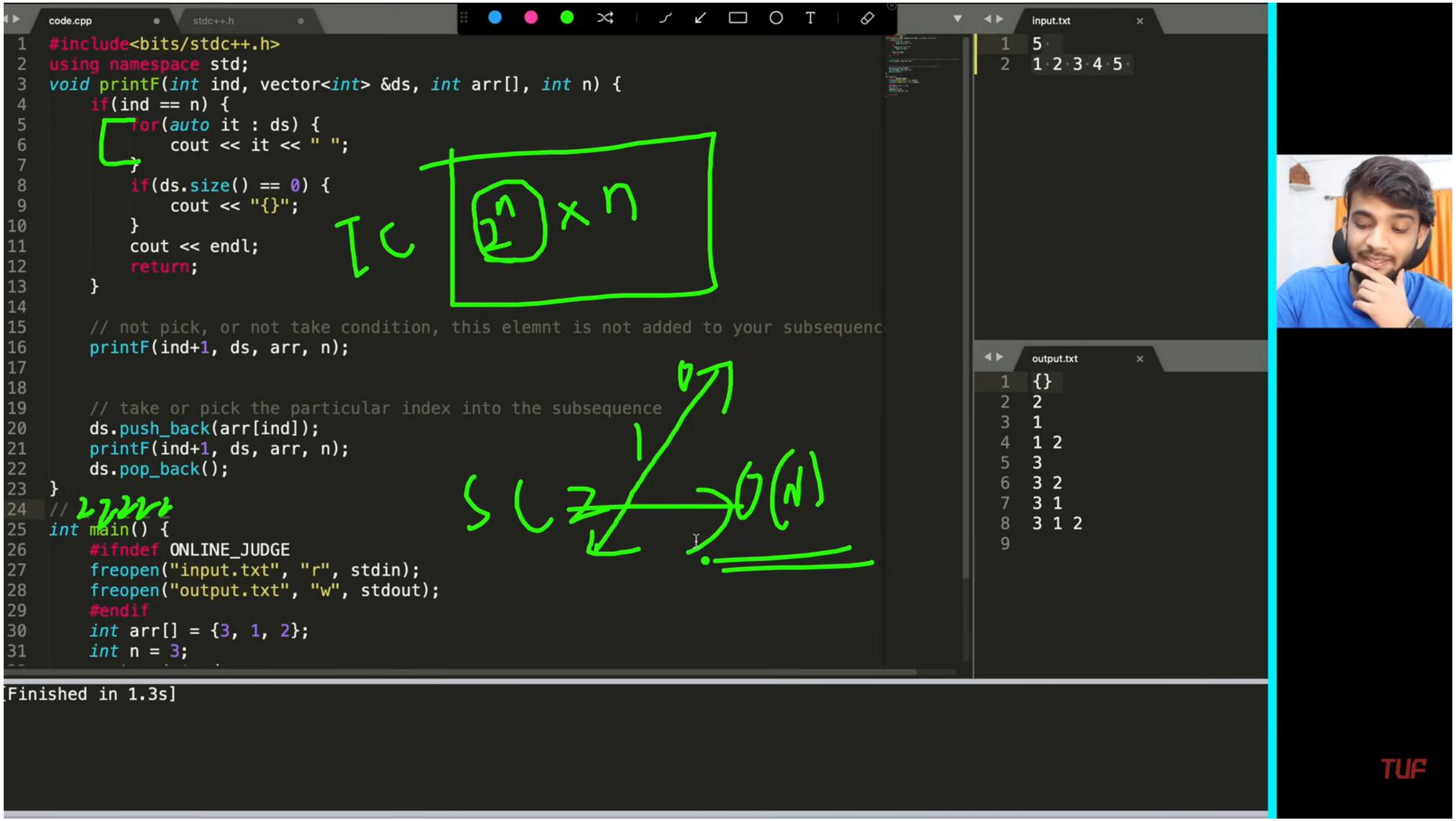


```cpp
#include<bits/stdc++.h>
using namespace std;
void printF(int ind, vector<int> &ds, int arr[], int n) {
    if(ind == n) {
        for(auto it : ds) {
            cout << it << " ";
        }
        if(ds.size() == 0) {
            cout << "{}";
        }
        cout << endl;
        return;
    }

    // not pick, or not take condition, this elemnt is not added to your subsequenc
    printF(ind+1, ds, arr, n);


    // take or pick the particular index into the subsequence
    ds.push_back(arr[ind]);
    printF(ind+1, ds, arr, n);
    ds.pop_back();
}
//
int main() {
    #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    #endif
    int arr[] = {3, 1, 2};
    int n = 3;
```

```
5
1 2 3 4 5
```

```
{}
2
1
1 2
3
3 2
3 1
3 1 2
```

```
[Finished in 1.3s]
```



```cpp
#include<bits/stdc++.h>
using namespace std;
void printF(int ind, vector<int> &ds, int arr[], int n) {
    if(ind == n) {
        for(auto it : ds) {
            cout << it << " ";
        }
        if(ds.size() == 0) {
            cout << "{}";
        }
        cout << endl;
        return;
    }

    // not pick, or not take condition, this elemnt is not added to your subsequenc
    printF(ind+1, ds, arr, n);


    // take or pick the particular index into the subsequence
    ds.push_back(arr[ind]);
    printF(ind+1, ds, arr, n);
    ds.pop_back();
}
//
int main() {
    #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    #endif
    int arr[] = {3, 1, 2};
    int n = 3;
```

```
5
1 2 3 4 5
```

```
{}
2
1
1 2
3
3 2
3 1
3 1 2
```

```
[Finished in 1.3s]
```