Application and Other Issues

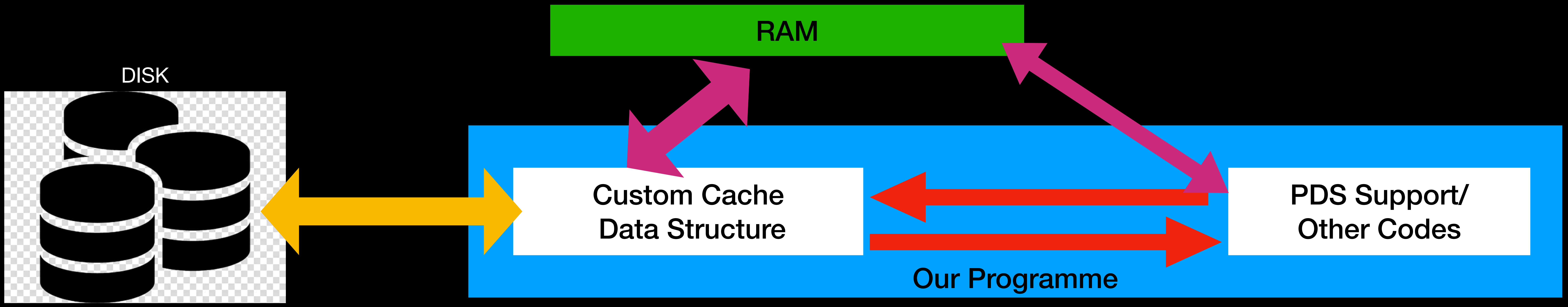
Persistent Data structure

PROJECT MEMBERS: ANANNYO DEY, SOUMYAJIT RUDRA SARMA , DEBASMIT ROY, KANKO GHOSH AND KUSHAL DAS

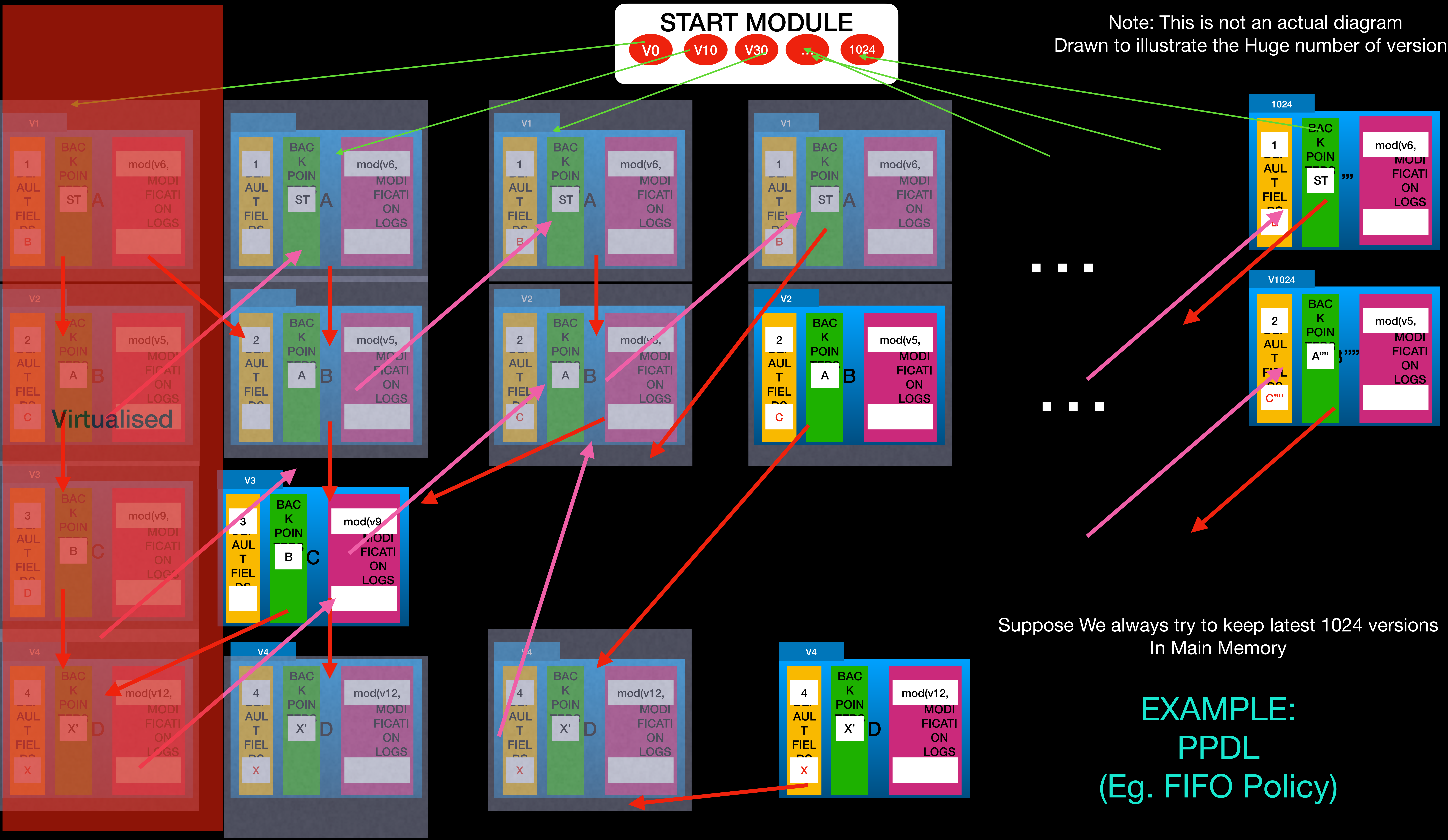
1. Try To Keep A Limited Number of Latest Version in Persistence

Because, the PDSs are very memory consuming.

We may store the Most Recently Used Versions in our Main Memory,
Here we can build a custom **Cache Data Structure**, with relevant replacement policy.
E.G. A LRU Policy will keep Least Recently Used versions in Disk



Note: This is not an actual diagram
Drawn to illustrate the Huge number of versions



Suppose We always try to keep latest 1024 versions
In Main Memory

EXAMPLE:
PPDL
(Eg. FIFO Policy)

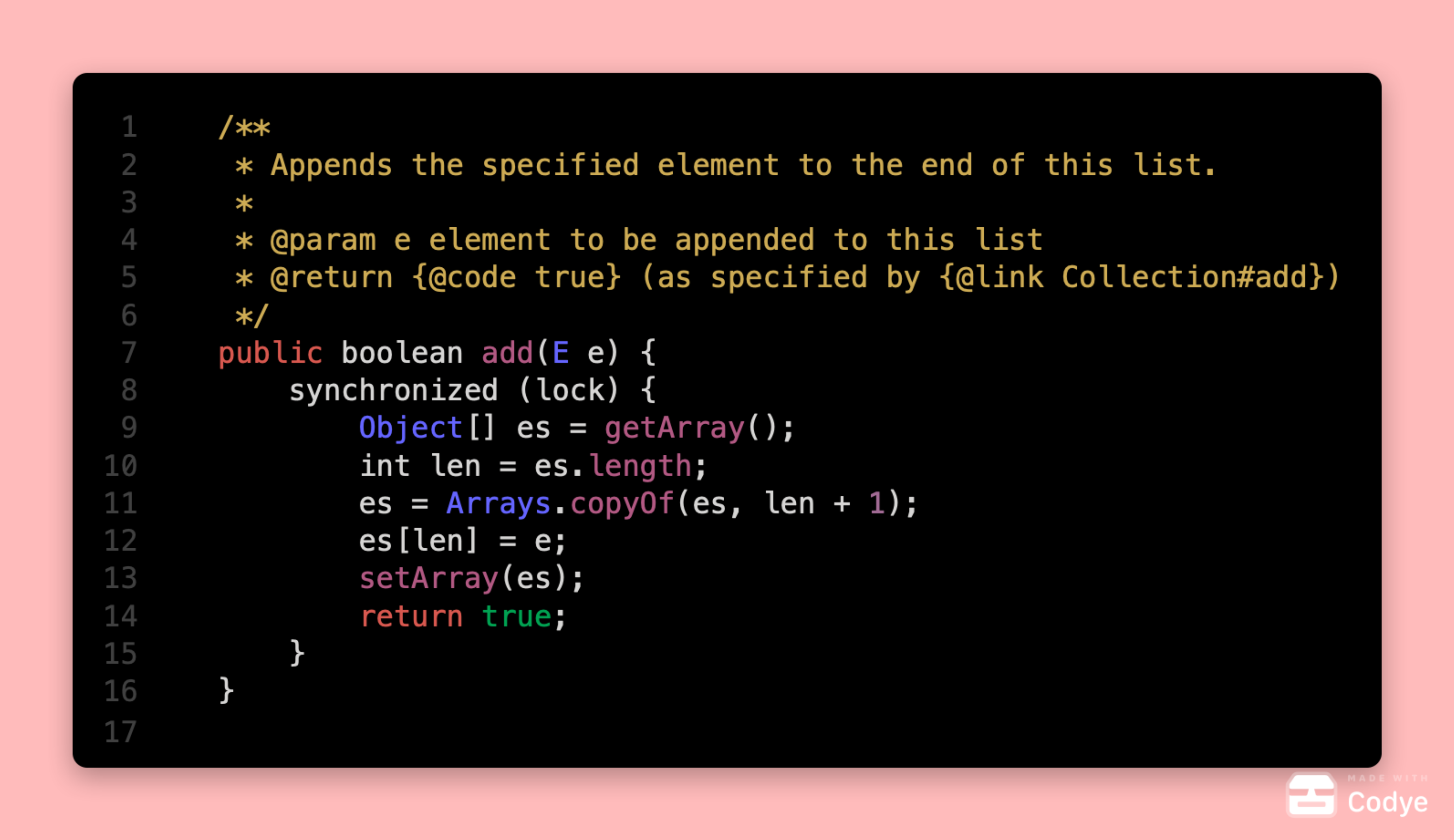
2. Application in JAVA

The Java programming language is not particularly functional. Despite this, the core JDK package java.util.concurrent includes CopyOnWriteArrayList and CopyOnWriteArraySet which are **partial persistent** structures, implemented using copy-on-write techniques

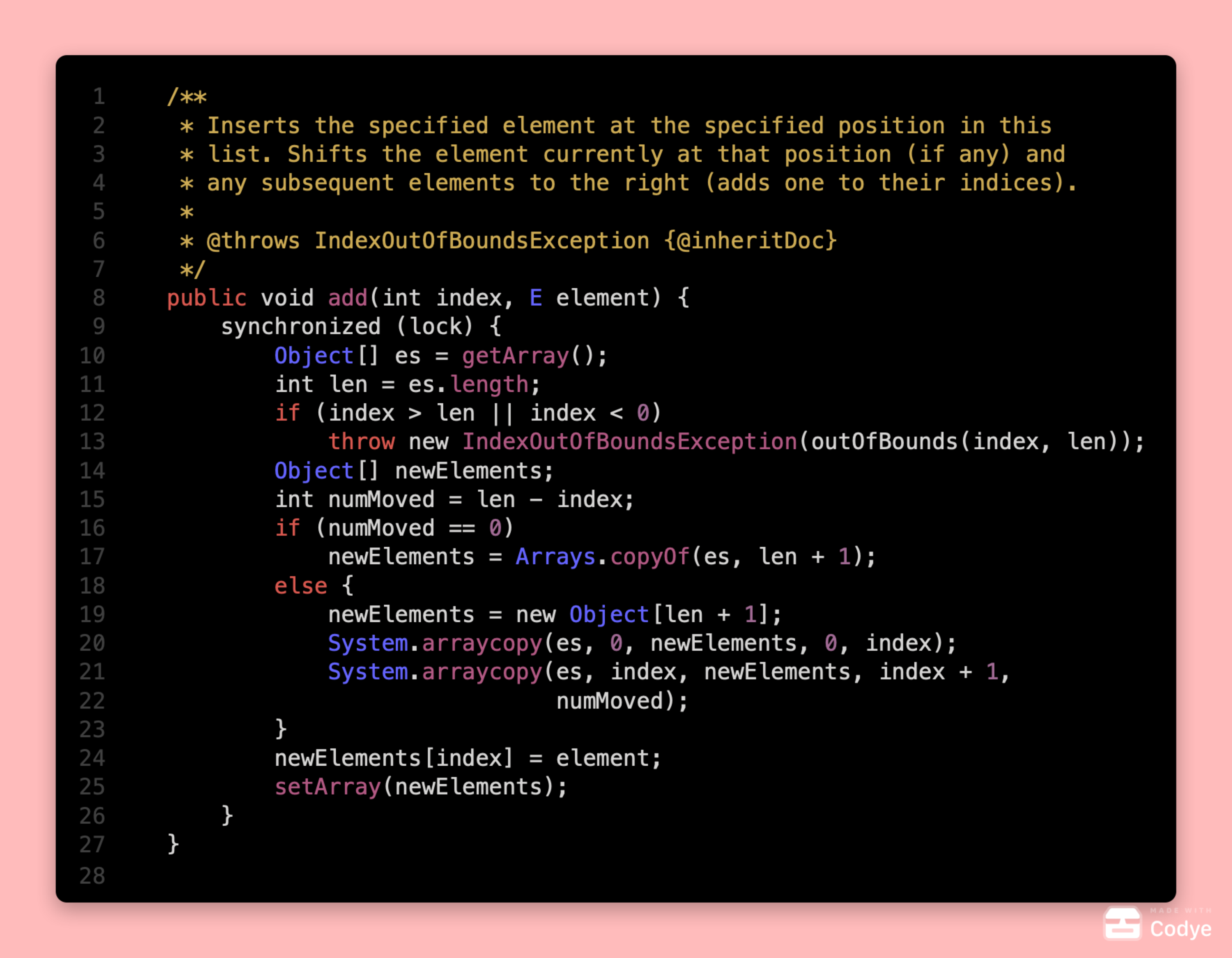
Class declaration

```
public class CopyOnWriteArrayList
   extends Object
implements List, RandomAccess, Cloneable, Serializable
```

1. CopyOnWriteArrayList is a thread-safe variant of ArrayList where operations which can change the ArrayList (add, update, set methods) creates a clone of the underlying array.
2. CopyOnWriteArrayList is to be used in a Thread based environment where read operations are very frequent and update operations are rare.
3. Iterator of CopyOnWriteArrayList will never throw ConcurrentModificationException.
4. Any type of modification to CopyOnWriteArrayList will not reflect during iteration since the iterator was created.
5. List modification methods like remove, set and add are not supported in the iteration. This method will throw UnsupportedOperationException. null can be added to the list.



```java
/**
 * Appends the specified element to the end of this list.
 *
 * @param e element to be appended to this list
 * @return {@code true} (as specified by {@link Collection#add})
 */
public boolean add(E e) {
    synchronized (lock) {
        Object[] es = getArray();
        int len = es.length;
        es = Arrays.copyOf(es, len + 1);
        es[len] = e;
        setArray(es);
        return true;
    }
}
```

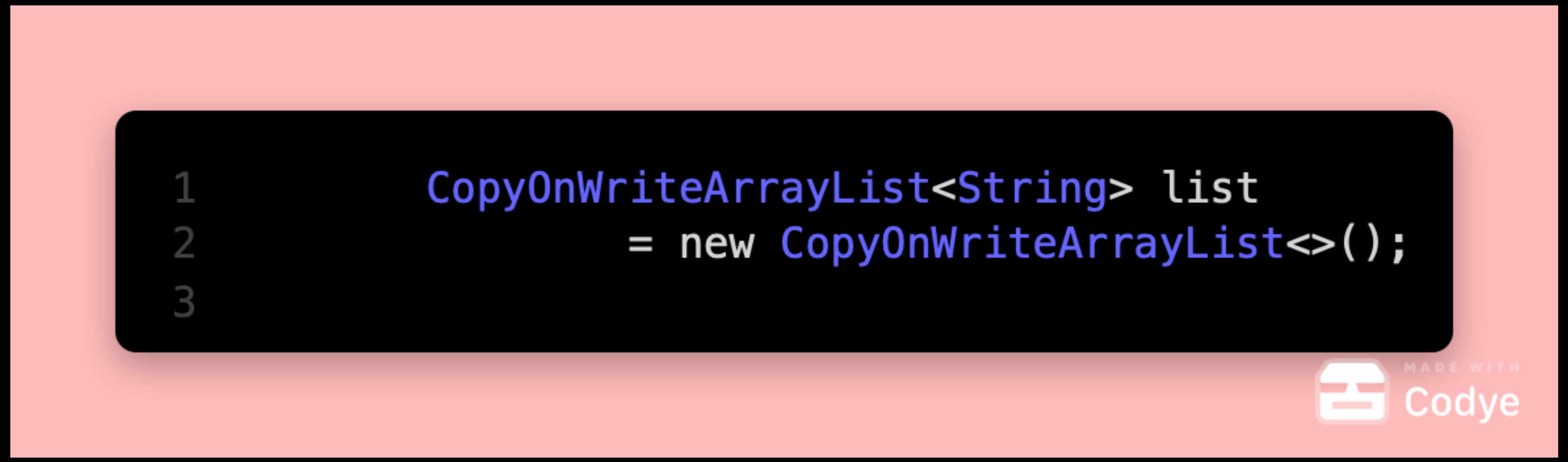


```java
/**
 * Inserts the specified element at the specified position in this
 * list. Shifts the element currently at that position (if any) and
 * any subsequent elements to the right (adds one to their indices).
 *
 * @throws IndexOutOfBoundsException {@inheritDoc}
 */
public void add(int index, E element) {
    synchronized (lock) {
        Object[] es = getArray();
        int len = es.length;
        if (index > len || index < 0)
            throw new IndexOutOfBoundsException(outOfBounds(index, len));
        Object[] newElements;
        int numMoved = len - index;
        if (numMoved == 0)
            newElements = Arrays.copyOf(es, len + 1);
        else {
            newElements = new Object[len + 1];
            System.arraycopy(es, 0, newElements, 0, index);
            System.arraycopy(es, index, newElements, index + 1,
                             numMoved);
        }
        newElements[index] = element;
        setArray(newElements);
    }
}
```

Code Taken From underlying source code of “CopyOnWriteArrayList” class

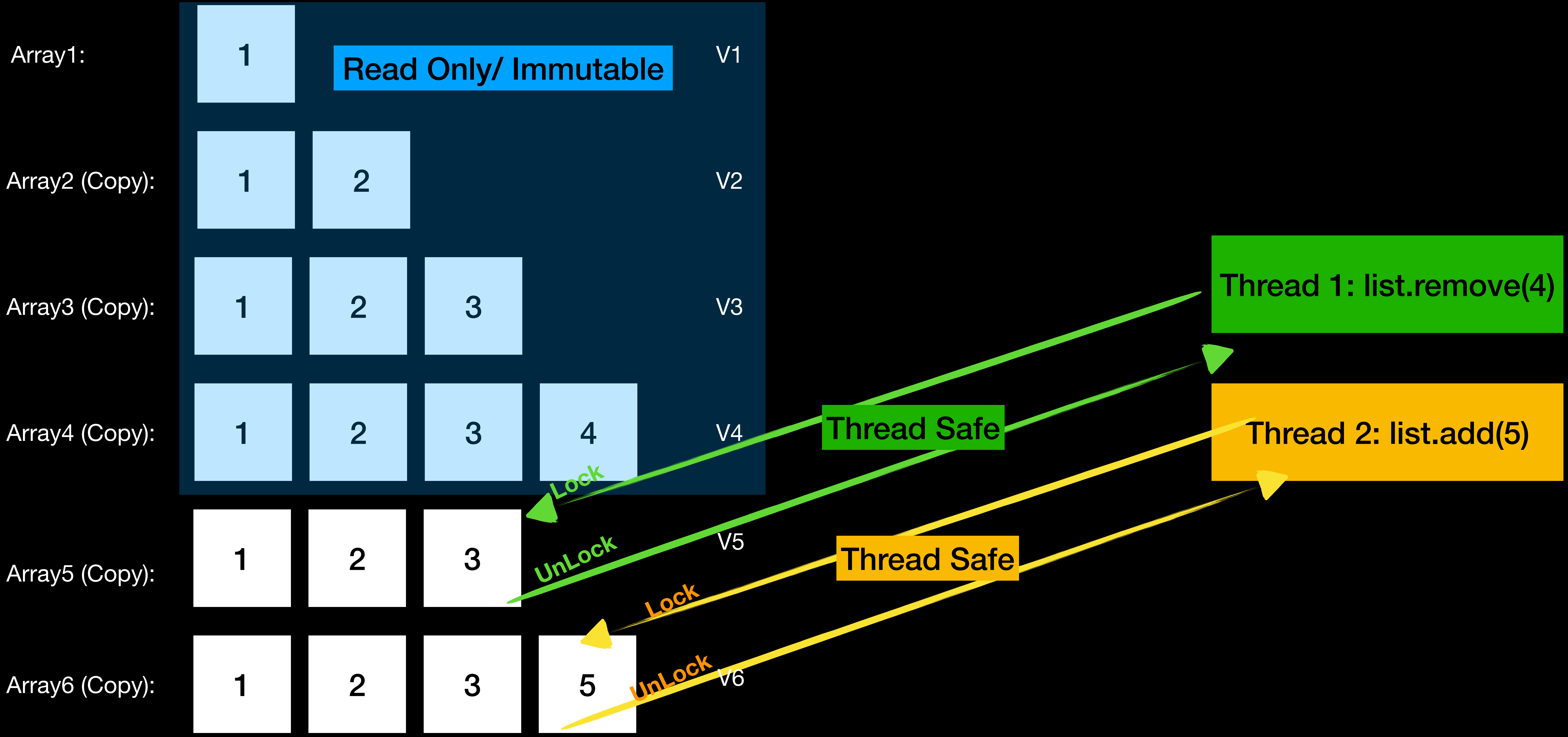
Note: We have tried to implement the same Functionality in C++.
[Code Attached]

Idea:



```
        CopyOnWriteArrayList<String> list
                = new CopyOnWriteArrayList<>();

```



Array1: 1 V1

Read Only/ Immutable

Array2 (Copy): 1 2 V2

Array3 (Copy): 1 2 3 V3

Array4 (Copy): 1 2 3 4 V4

Array5 (Copy): 1 2 3 V5

Array6 (Copy): 1 2 3 5 V6

Thread 1: list.remove(4)

Thread 2: list.add(5)

3. Application In Clojure

Clojure

Like many programming languages in the Lisp family, Clojure contains an implementation of a linked list, but unlike other dialects its implementation of a Linked List has enforced persistence instead of being persistent by convention.[19] Clojure also has efficient implementations of persistent vectors, maps, and sets based on persistent hash array mapped tries. These data structures implement the mandatory read-only parts of the Java collections framework.[20]

The designers of the Clojure language advocate the use of persistent data structures over mutable data structures because they have value semantics which gives the benefit of making them freely shareable between threads with cheap aliases, easy to fabricate, and language independent.[21]

These data structures form the basis of Clojure's support for parallel computing since they allow for easy retries of operations to sidestep data races and atomic compare and swap semantics.[22]

Persistence In Clojure is Achieved Through Bitmapped-HAMT
We Have Already Attached A Dedicated PPT
On HAMT

Note: A External Open Source Library Provides Persistence
In C++. It is done through Bitmapped-HAMT too.

Extract Taken from wikipedia

Now lets come to Javascript

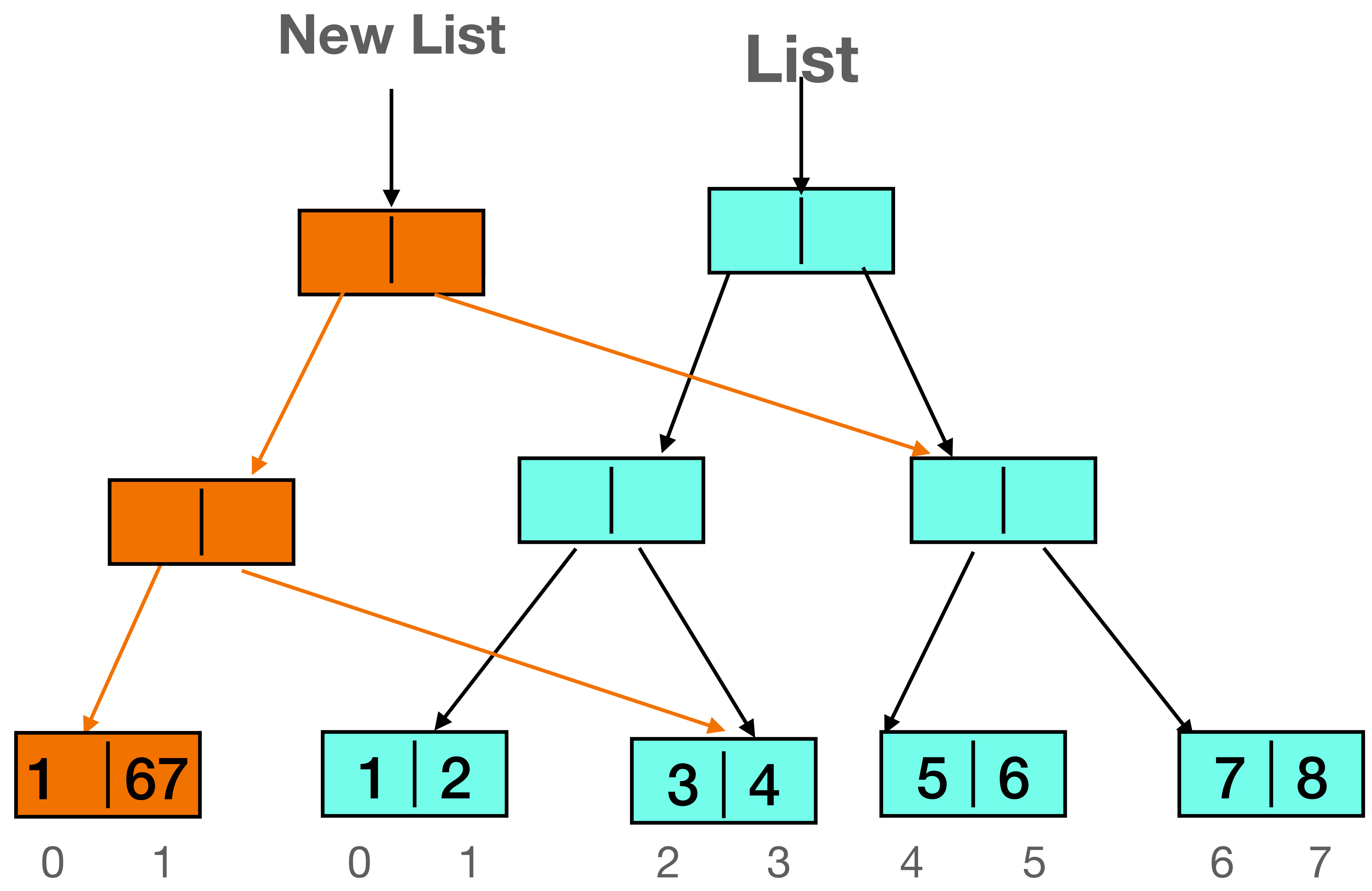
Suppose we want to make a list of some numbers

list=[1,2,3,4,5,6,7,8]

Now suppose we want to change the number at index 1 i.e. 2 to 67.

Ok so now lets break our list of numbers in size of 2 → 1,2 3,4 5,6 7,8

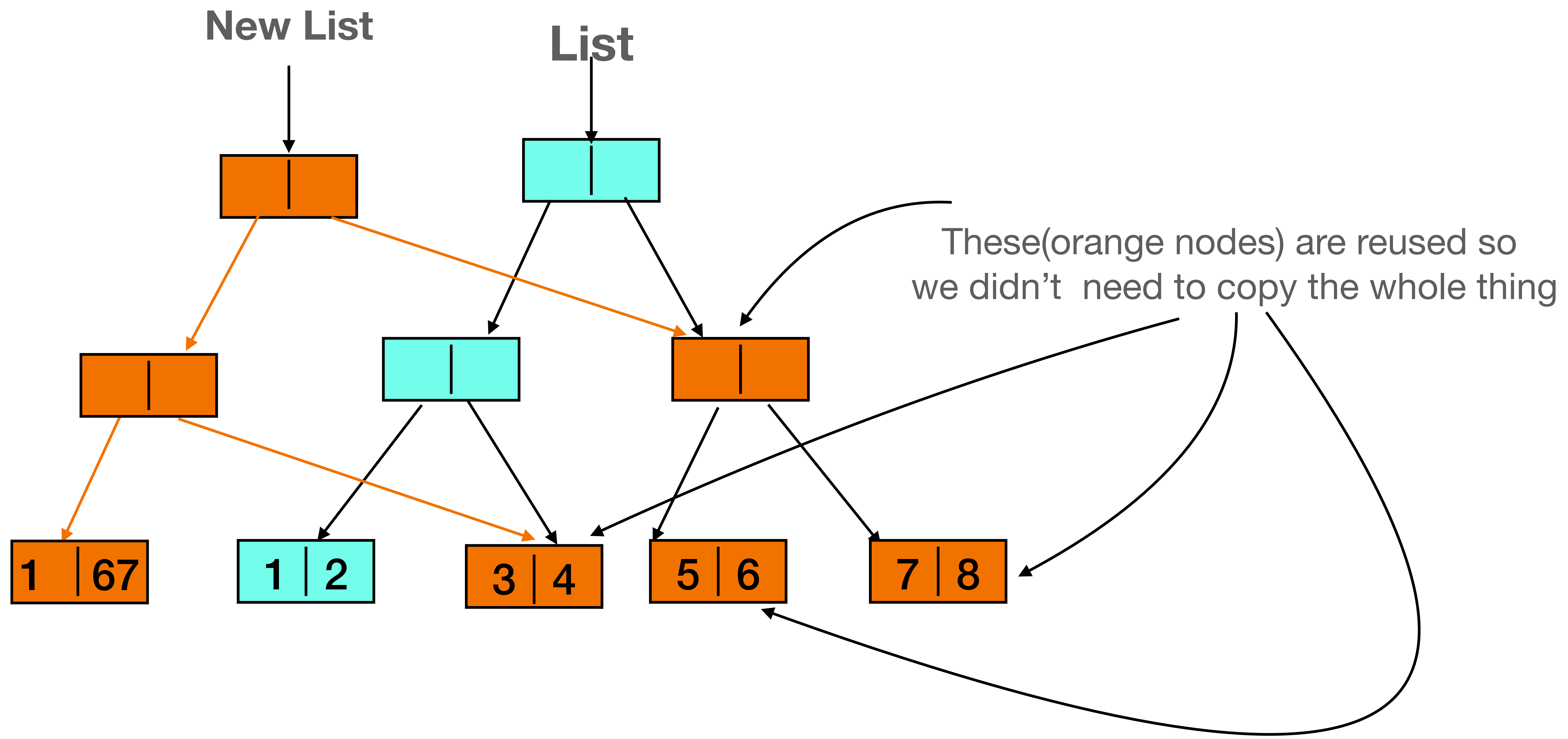
So now we will update the value 2 to 67



1st the node is copied

Then path copying happens

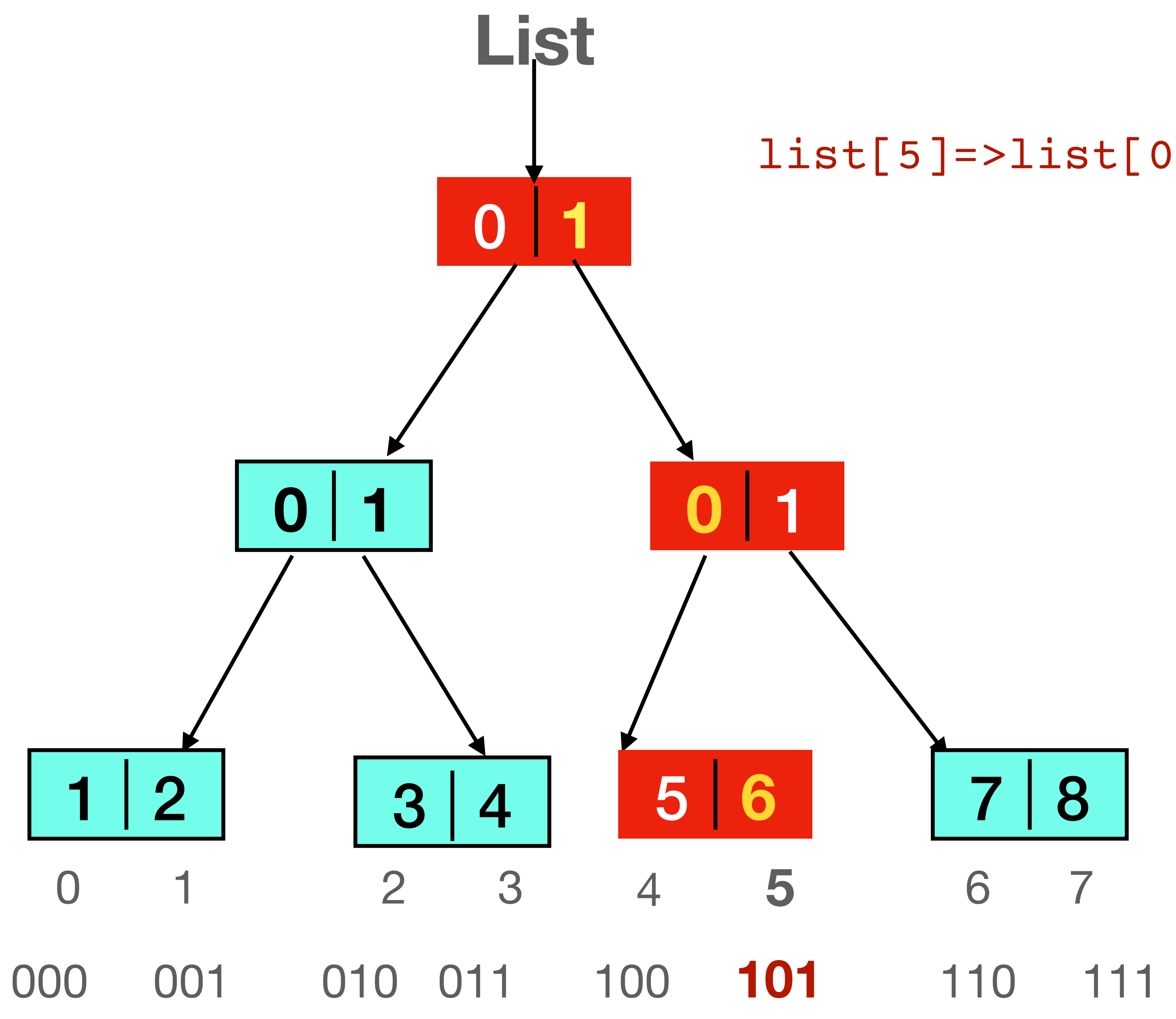
So now the list is



These(orange nodes) are reused so we didn’t need to copy the whole thing

So now if we want to get a value at any index say 5
We can do that easily using hashing 5 to its binary value 0b101

We will go from 1 ⟶ 0 ⟶ 1 to get
our value I.e. 5



```
list[5]=>list[0b101]=>list->1->0->1=>6
```

In this way we can reach to any node quickly and return the value in a less time using hashing

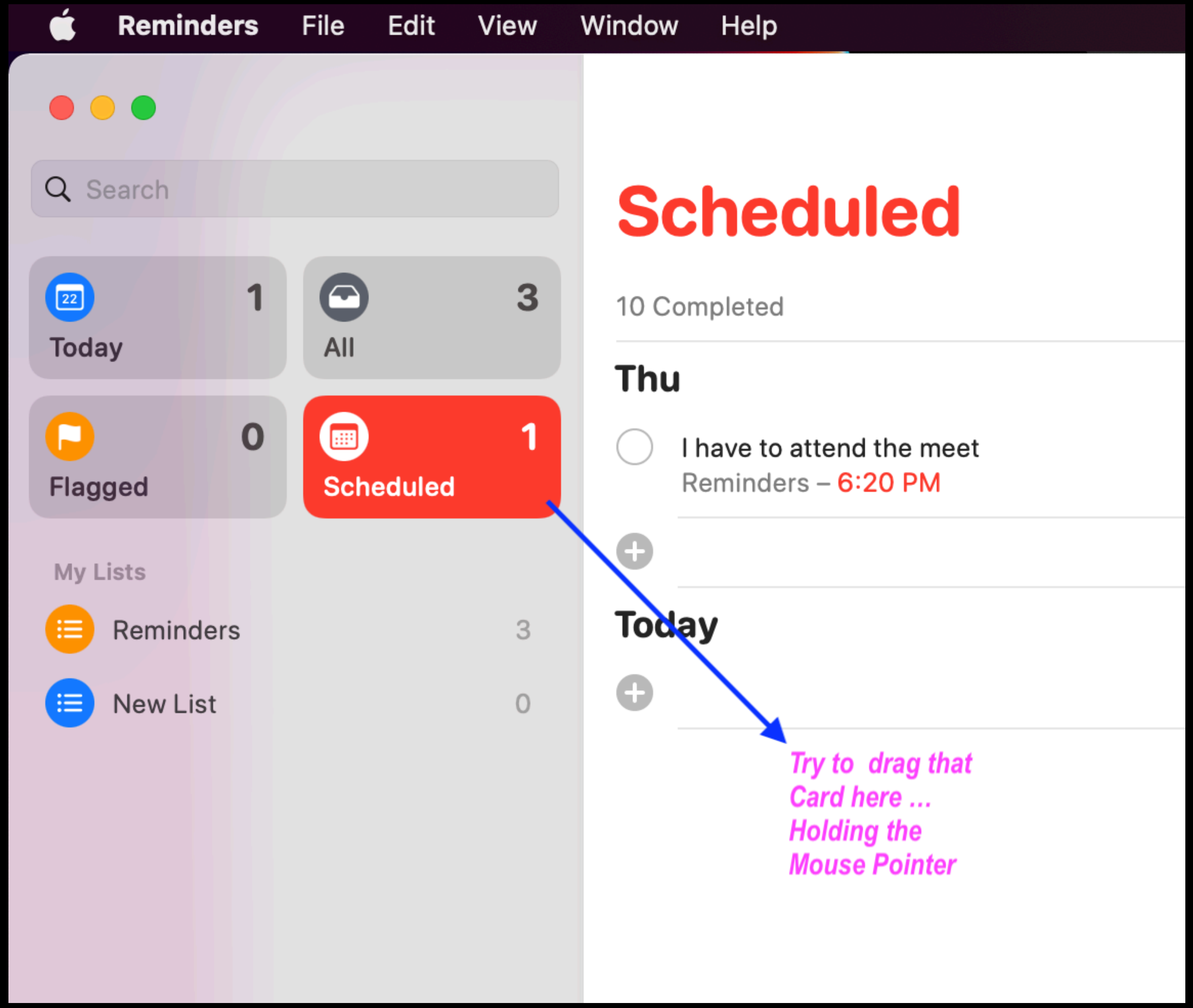
Some applications!!

Application

… so far we have seen

1. In addition to the obvious 'look-back'applications, we can use persistent data structures to solve problems by representing one of their dimensions as time.
2. To achieve Immutability we may use Persistent Data structure. In several languages, in immutability perspective several versions versions of that imm. data structure is generated rather hampering the same version.
3. Multithreading is much easier!! Because if different threads try to modify a same unlocked data-structure, the threads will independently modify the data-structure and will create different versions. Race Problem won't occur!!

4. Optimistic UI:



Scheduled

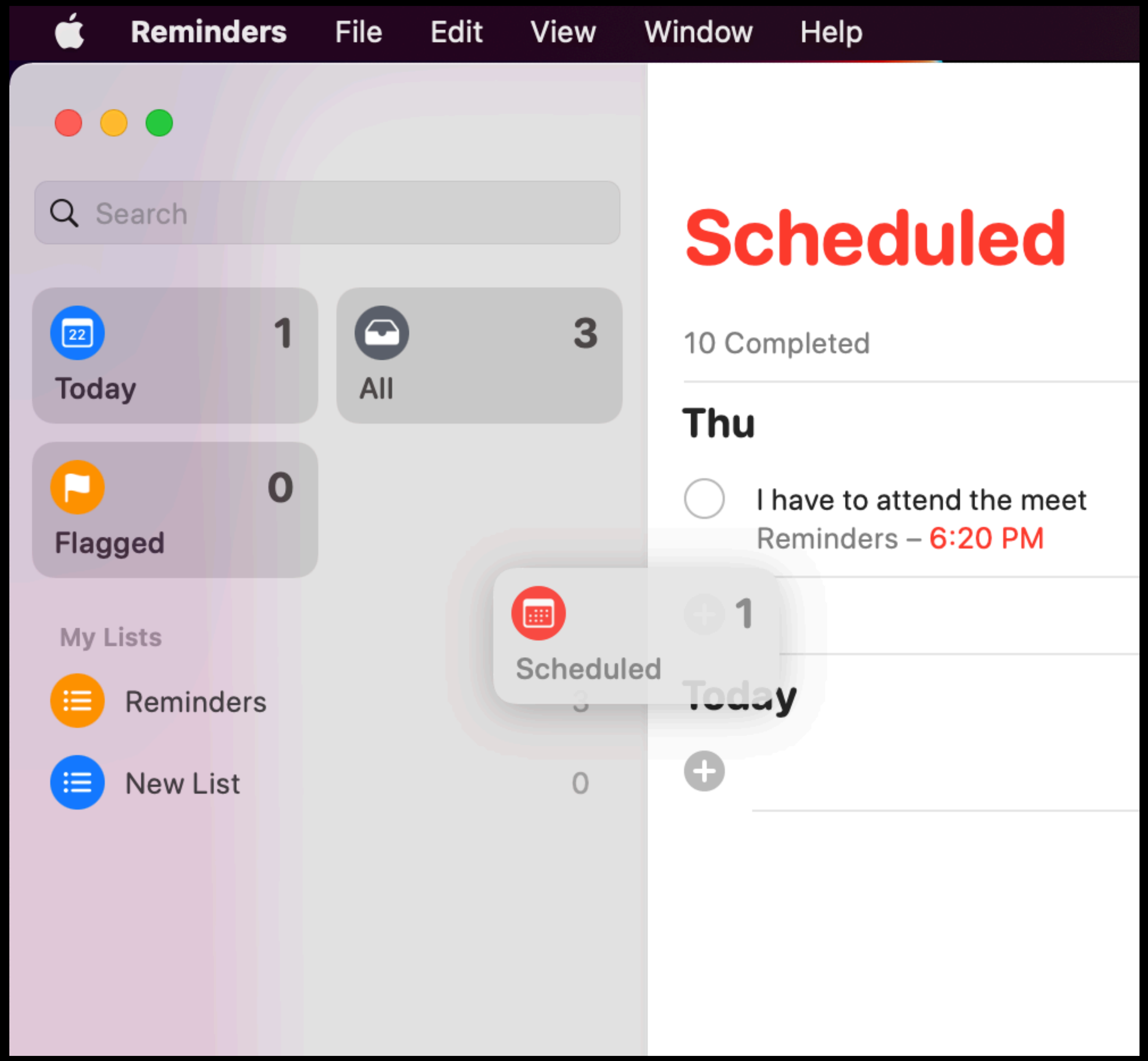
10 Completed

Thu

○ I have to attend the meet
Reminders – 6:20 PM

Today

Try to drag that Card here ... Holding the Mouse Pointer



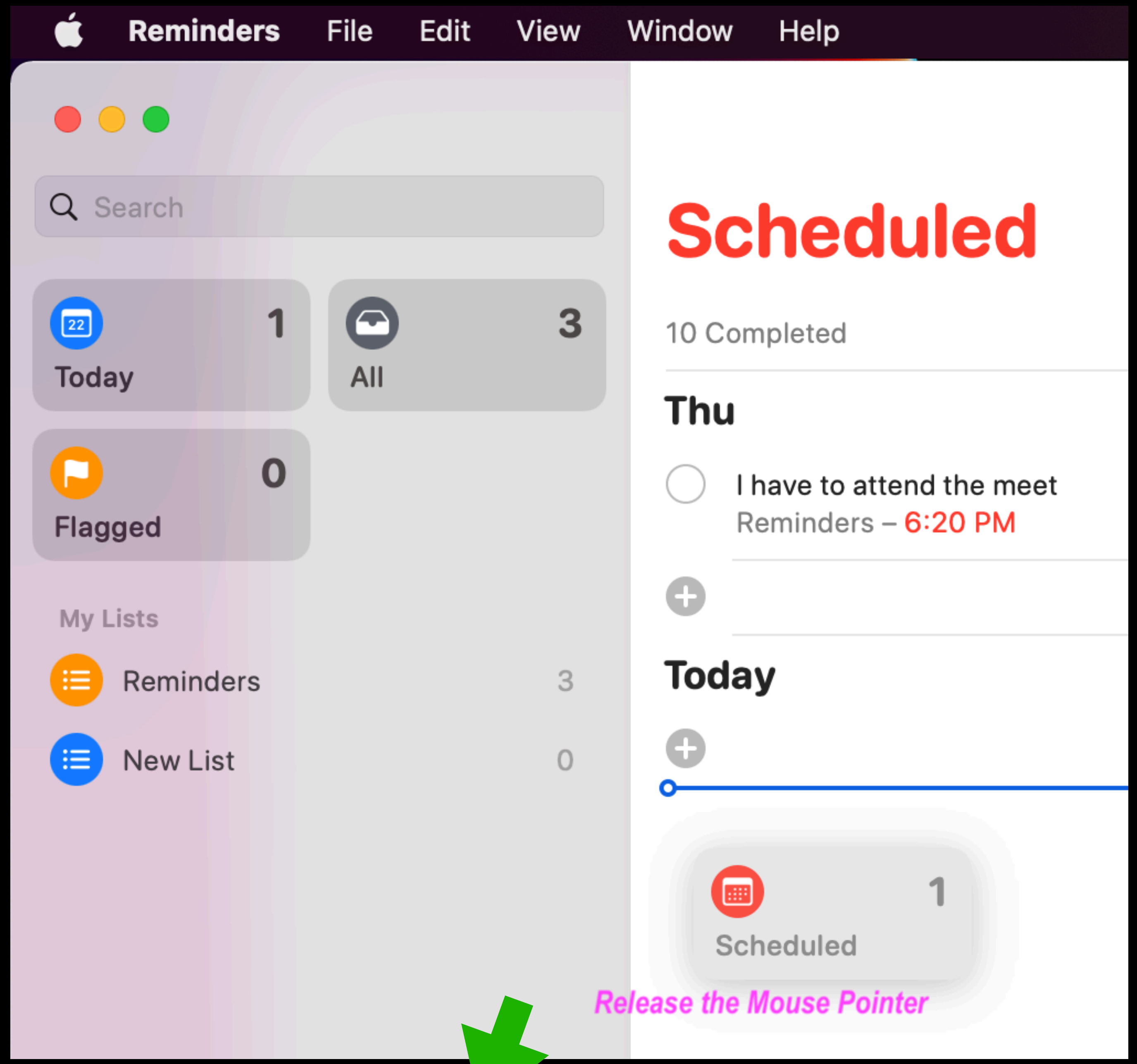
Scheduled

10 Completed

Thu

○ I have to attend the meet
Reminders – 6:20 PM

Today



Scheduled

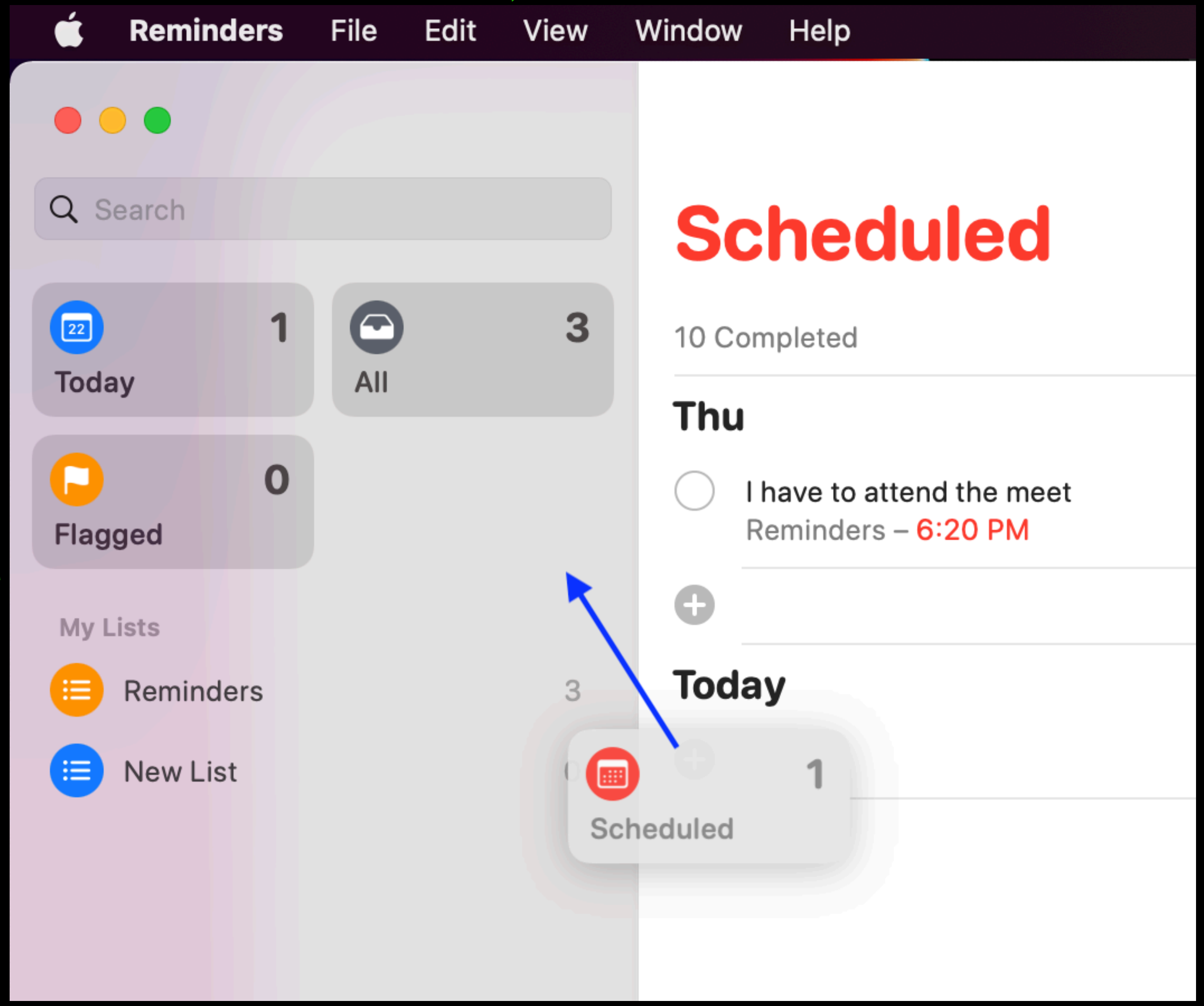
10 Completed

Thu

○ I have to attend the meet
Reminders – 6:20 PM

Today

Release the Mouse Pointer



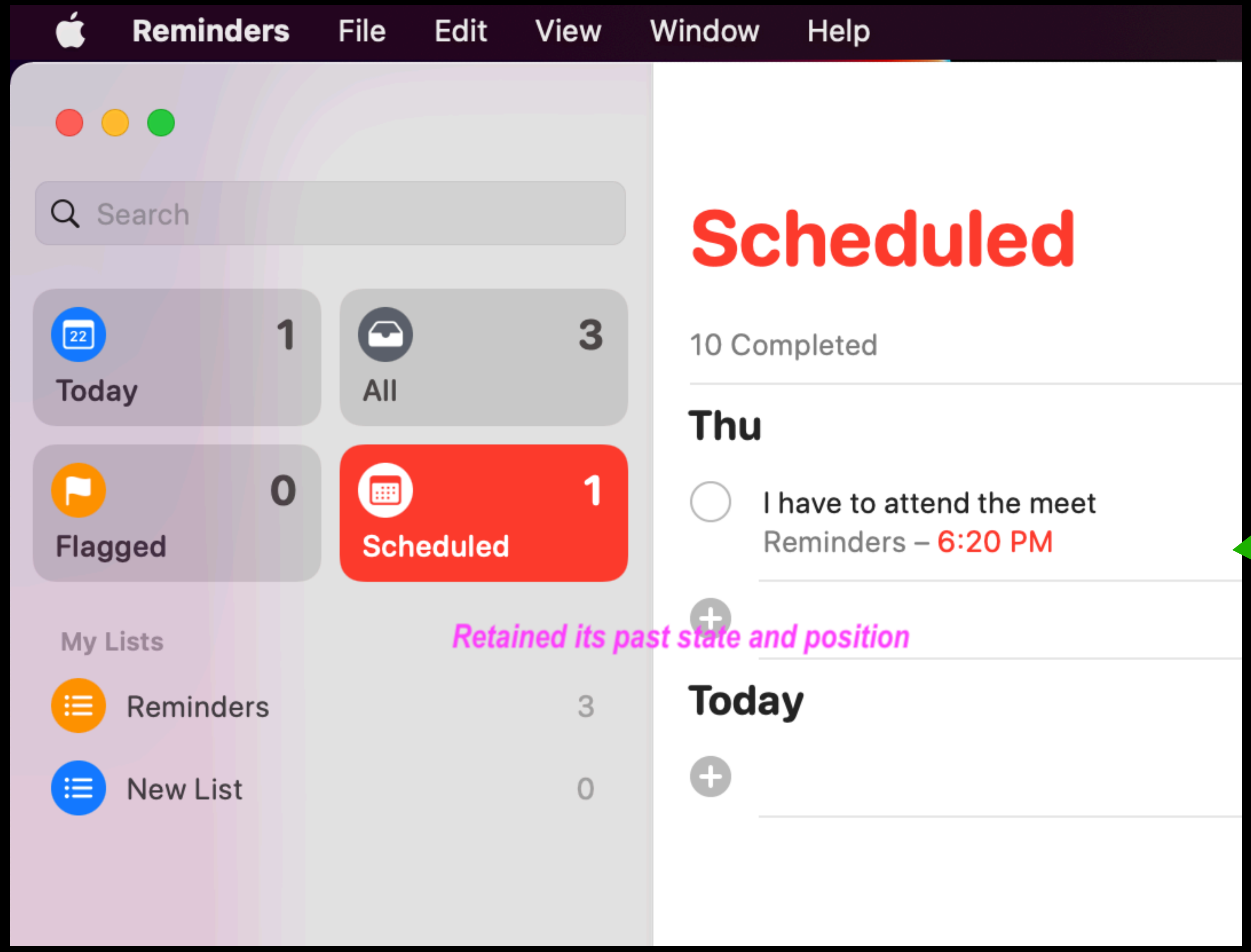
Scheduled

10 Completed

Thu

○ I have to attend the meet
Reminders – 6:20 PM

Today



Scheduled

10 Completed

Thu

○ I have to attend the meet
Reminders – 6:20 PM

Retained its past state and position

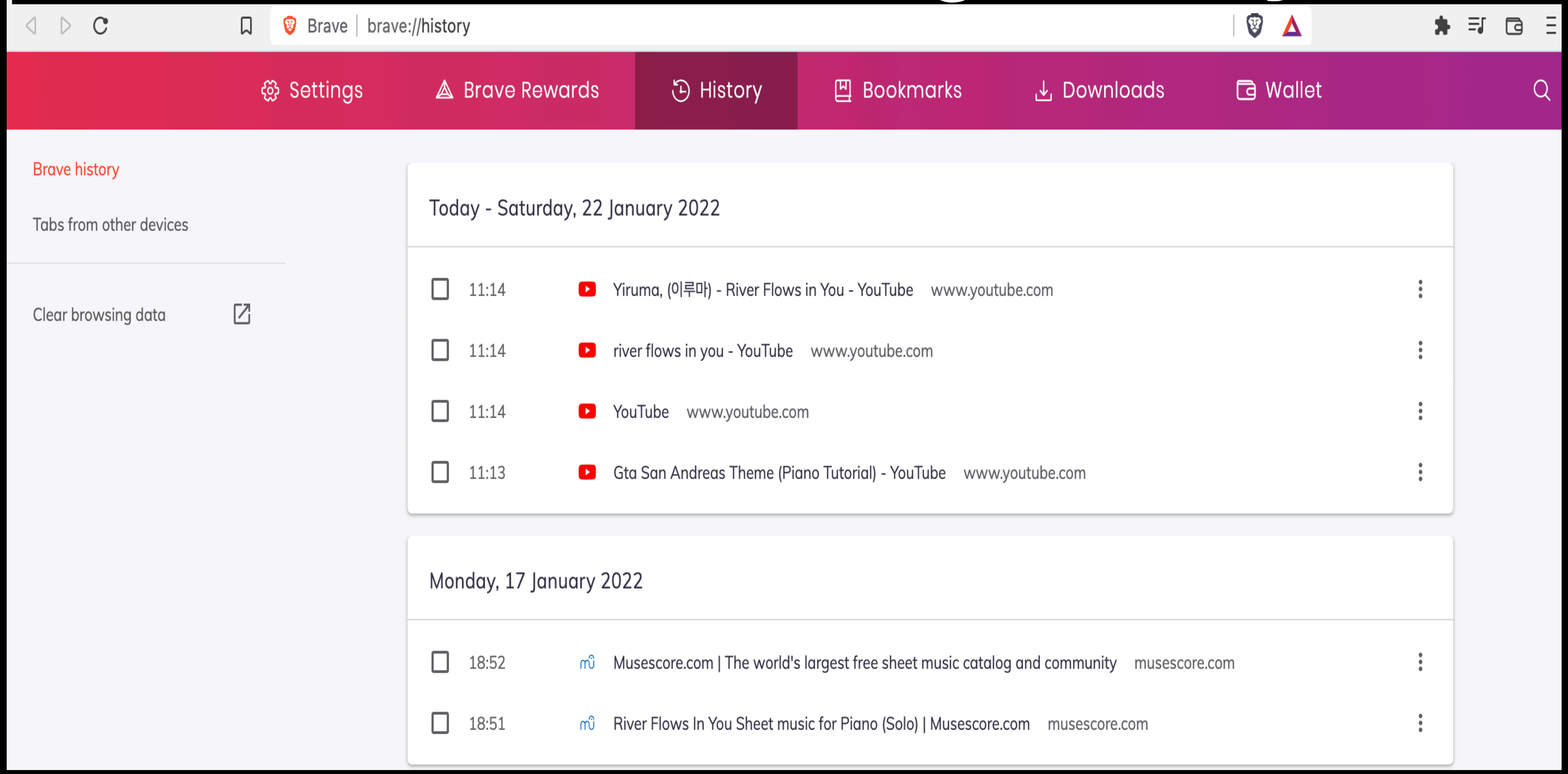
Today

If we try to drag a Card into some invalid space holding the mouse pointer(reached current time), then suddenly we release the mouse, then, the Card went back to the previous state(Old State)

This thing can be done optimally using Persistance

Source: A Seminar
Persistent Data Structures – Ivan Vergiliev – PartialConf 2017
https://www.youtube.com/watch?v=UcczyTC0wmU

5. Undo And Browsing History:



Brave | brave://history

Settings | Brave Rewards | History | Bookmarks | Downloads | Wallet

Brave history

Tabs from other devices

Clear browsing data

Today - Saturday, 22 January 2022

□ 11:14 Yiruma, (이루마) - River Flows in You - YouTube www.youtube.com

□ 11:14 river flows in you - YouTube www.youtube.com

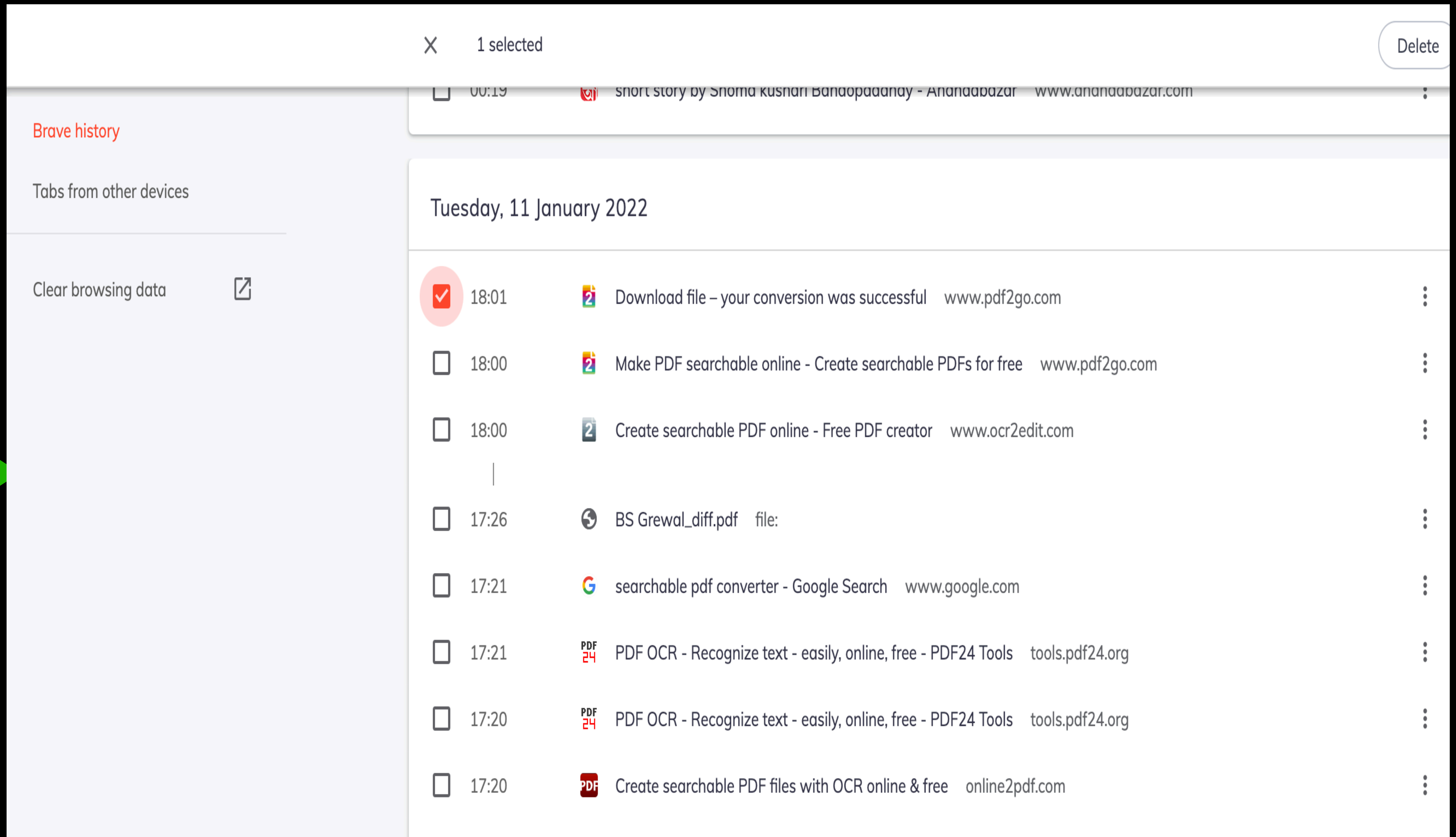
□ 11:14 YouTube www.youtube.com

□ 11:13 Gta San Andreas Theme (Piano Tutorial) - YouTube www.youtube.com

Monday, 17 January 2022

□ 18:52 Musescore.com | The world's largest free sheet music catalog and community musescore.com

□ 18:51 River Flows In You Sheet music for Piano (Solo) | Musescore.com musescore.com



X 1 selected Delete

□ 00:19 short story by Shoma Kushari Bandopadhay - Anandabazar www.anandabazar.com

Brave history

Tabs from other devices

Clear browsing data

Tuesday, 11 January 2022

☑ 18:01 Download file – your conversion was successful www.pdf2go.com

□ 18:00 Make PDF searchable online - Create searchable PDFs for free www.pdf2go.com

□ 18:00 Create searchable PDF online - Free PDF creator www.ocr2edit.com

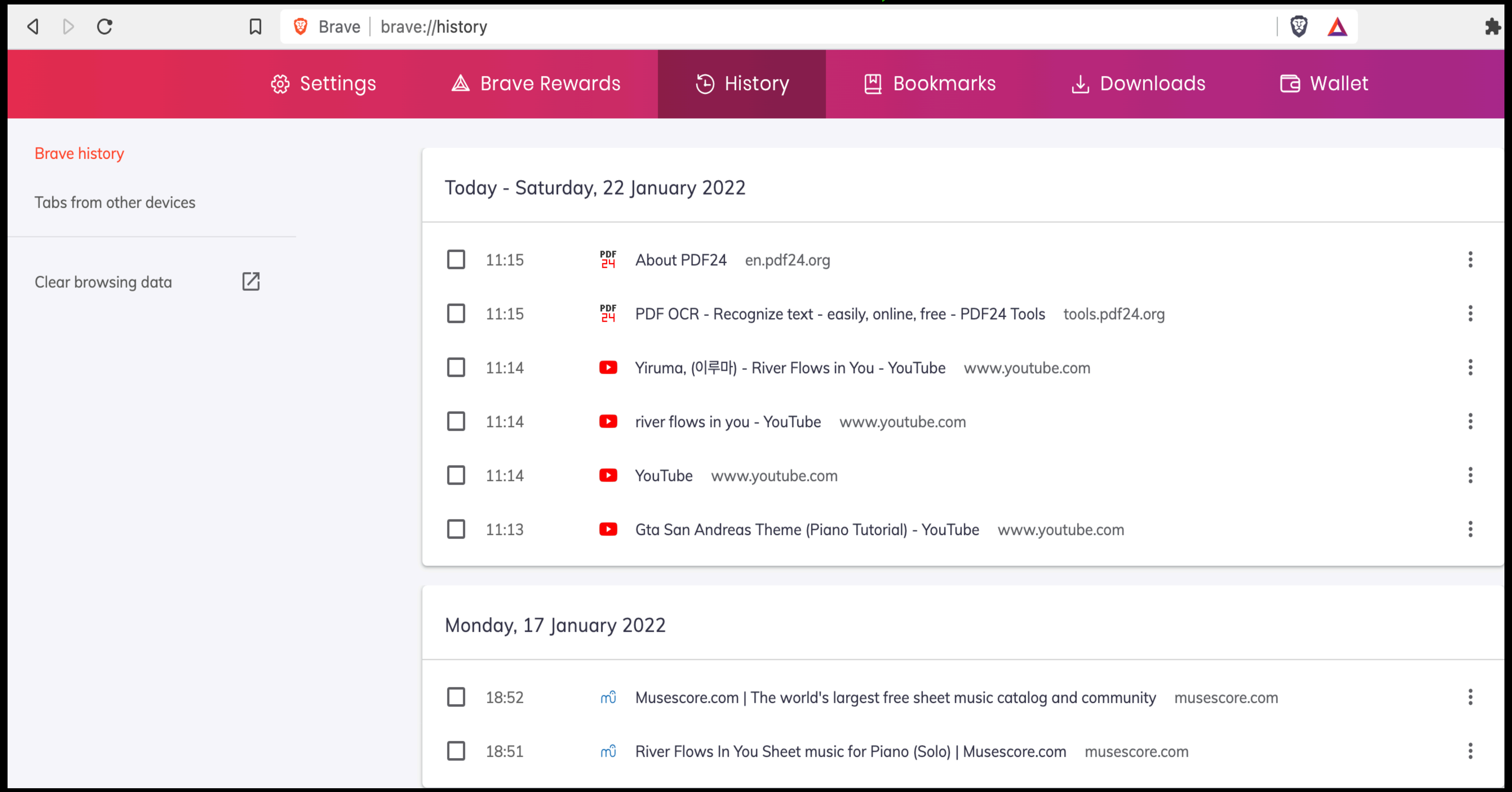
□ 17:26 BS Grewal_diff.pdf file:

□ 17:21 searchable pdf converter - Google Search www.google.com

□ 17:21 PDF OCR - Recognize text - easily, online, free - PDF24 Tools tools.pdf24.org

□ 17:20 PDF OCR - Recognize text - easily, online, free - PDF24 Tools tools.pdf24.org

□ 17:20 Create searchable PDF files with OCR online & free online2pdf.com



Brave | brave://history

Settings | Brave Rewards | History | Bookmarks | Downloads | Wallet

Brave history

Tabs from other devices

Clear browsing data

Today - Saturday, 22 January 2022

□ 11:15 About PDF24 en.pdf24.org

□ 11:15 PDF OCR - Recognize text - easily, online, free - PDF24 Tools tools.pdf24.org

□ 11:14 Yiruma, (이루마) - River Flows in You - YouTube www.youtube.com

□ 11:14 river flows in you - YouTube www.youtube.com

□ 11:14 YouTube www.youtube.com

□ 11:13 Gta San Andreas Theme (Piano Tutorial) - YouTube www.youtube.com

Monday, 17 January 2022

□ 18:52 Musescore.com | The world's largest free sheet music catalog and community musescore.com

□ 18:51 River Flows In You Sheet music for Piano (Solo) | Musescore.com musescore.com

We often to go back to the past history, and, proceed from the past in current time

This thing can be done optimally using Persistance

Source: A Seminar

Persistent Data Structures – Ivan Vergiliev – PartialConf 2017

https://www.youtube.com/watch?v=UcczyTC0wmU

Thank you …