You are screen sharing Stop Share

$f(n : \text{Int}) \rightarrow [\ \ ]$

$n = \emptyset \rightarrow [\,]$

$n = 1 \rightarrow$ ["0", "1"]

$n = 2 \rightarrow$ ["00", "01", "10", "11"]

$n = 3 \rightarrow$ ["000" "001" "010" "011" '100' '101' '110' '111']

2^n

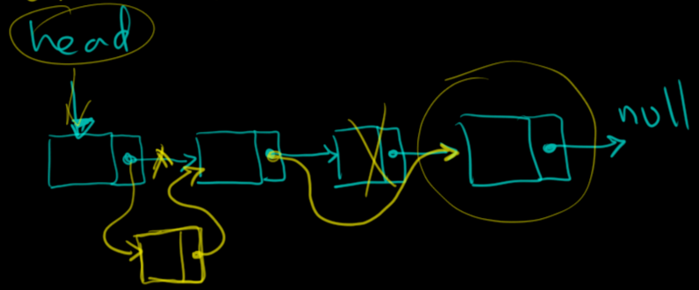
S_{n-1}

S_n

$S_\emptyset = [\,]$

$$\boxed{S_n = [\,'0' + S_{n-1},\ '1' + S_{n-1}\,]}$$

Linked list

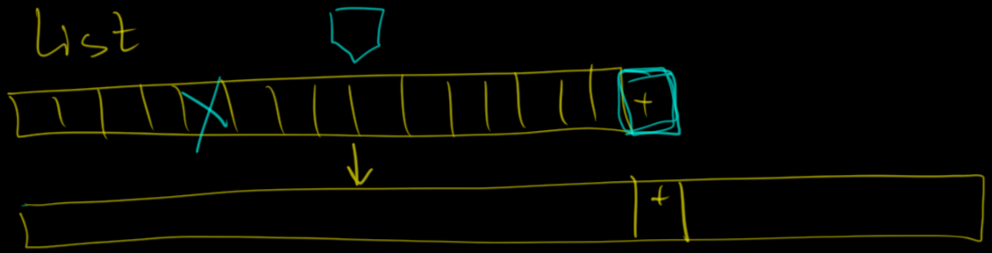


add
remove $\underline{O(1)}$

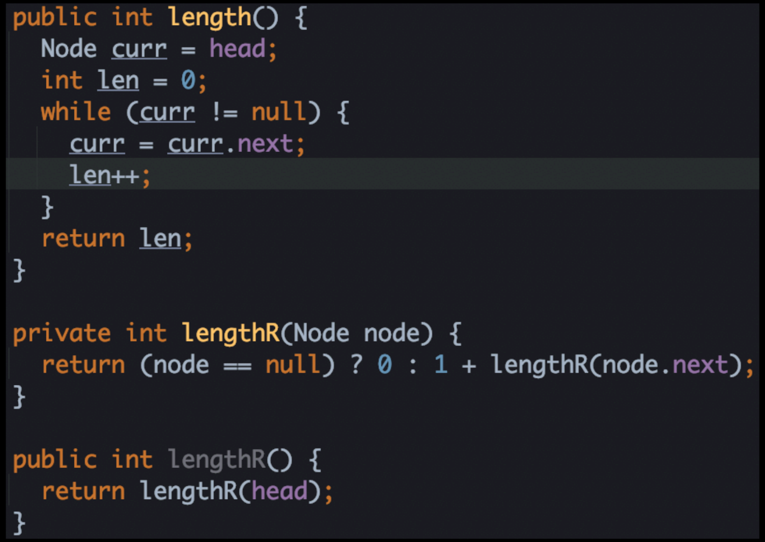
access $O(n)$

?

List



access $O(1)$
add
remove $O(n)$



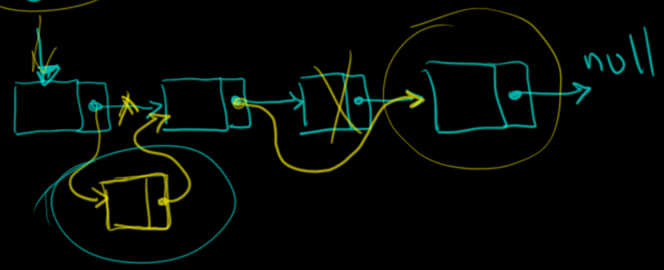
```java
public int length() {
  Node curr = head;
  int len = 0;
  while (curr != null) {
    curr = curr.next;
    len++;
  }
  return len;
}

private int lengthR(Node node) {
  return (node == null) ? 0 : 1 + lengthR(node.next);
}

public int lengthR() {
  return lengthR(head);
}
```

You are screen sharing Stop Share

#3

Linked List

head



curr =

1. $\boxed{x}$
2. $\boxed{x \mid \bullet} \rightarrow$ head
3. head = curr

replace(x, y)

addInto(x,

add After

add Before

add into the list

linked list ~~index~~

doesn't make any sence

x = 5

~~x = 6~~

y = 6

~~var~~

~~void~~

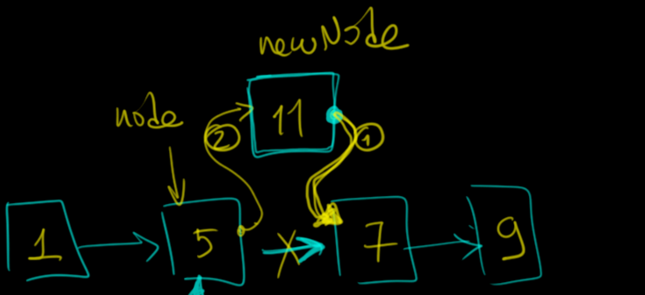
List

Array

$\boxed{\text{Iterable}}$

(i) List

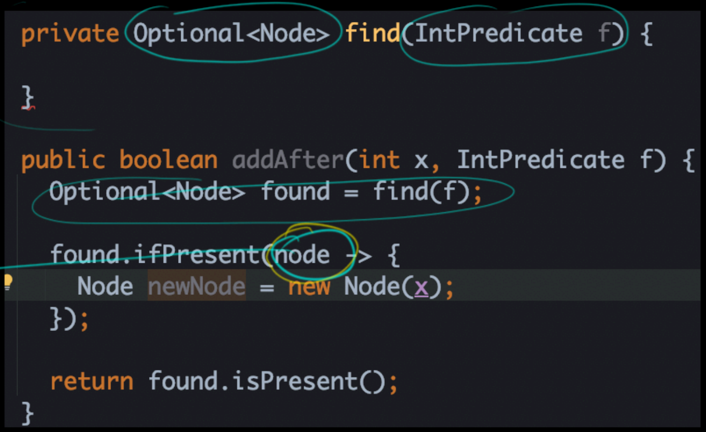
get (index) → A



① findNode (f) => Node

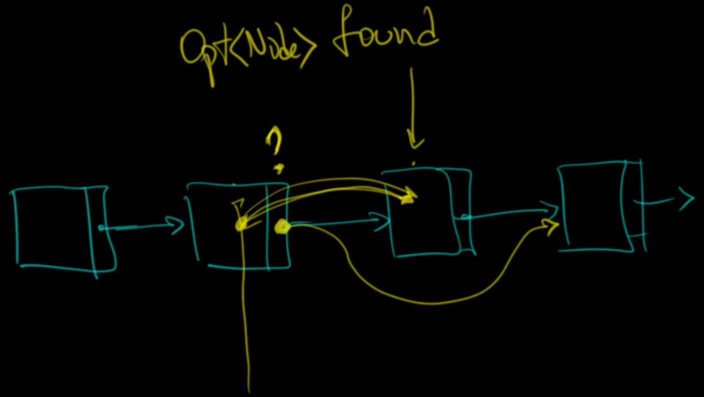
② create

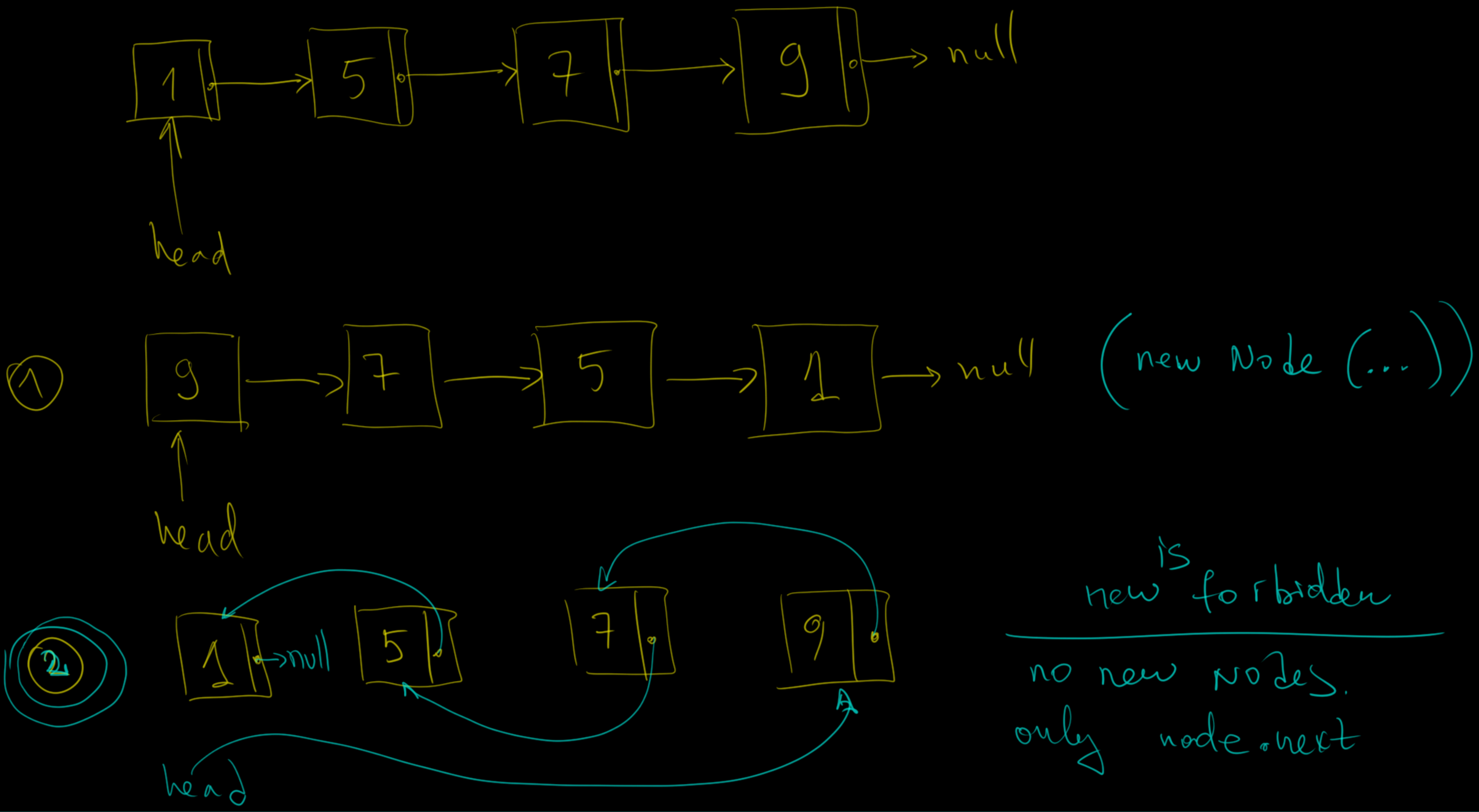
③ rewire links



```
private Optional<Node> find(IntPredicate f) {

}

public boolean addAfter(int x, IntPredicate f) {
  Optional<Node> found = find(f);

  found.ifPresent(node -> {
    Node newNode = new Node(x);
  });

  return found.isPresent();
}
```

remove (f : Int -> Bool)





new ~~new~~ is forbidden

no new nodes.
only node.next