Partial Persistent Data-structure

Pointer Machine Model

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

Ephemeral Linked List

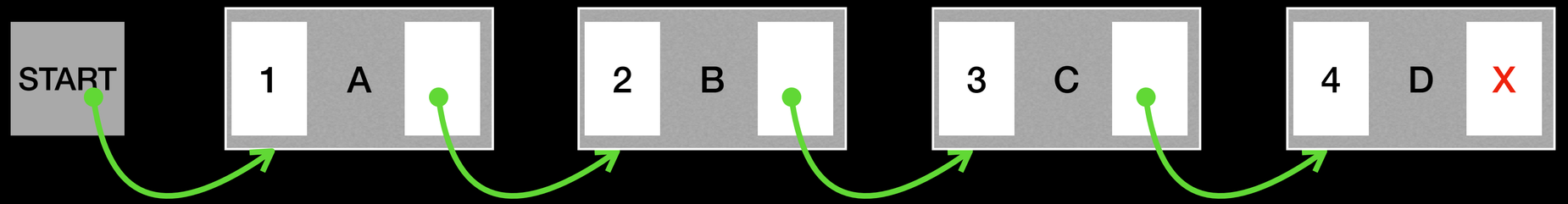
Operations:

- start = init() : To initiate linked list and “start” pointer holds the starting position
- add(x, y) : Add new node x after y
- remove(x) : Remove node x
- iterate_over_LL() : Iterate over the whole linked list
- update(f_i,x,val) : Update the i-th field in node x to new value ‘val’

Ephemeral Linkedlist

Node structure

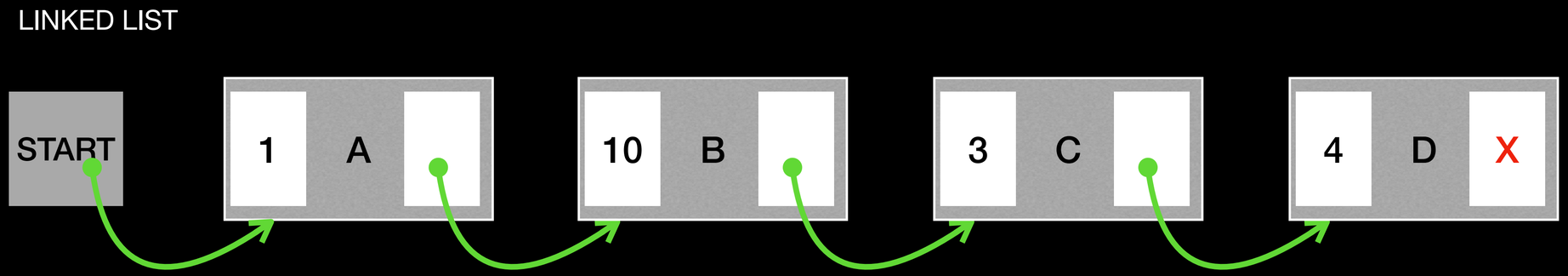
LINKED LIST



HERE IN-DEGREE OF A NODE IS <= 1 AND OUT-DEGREE OF A NODE IS <=1

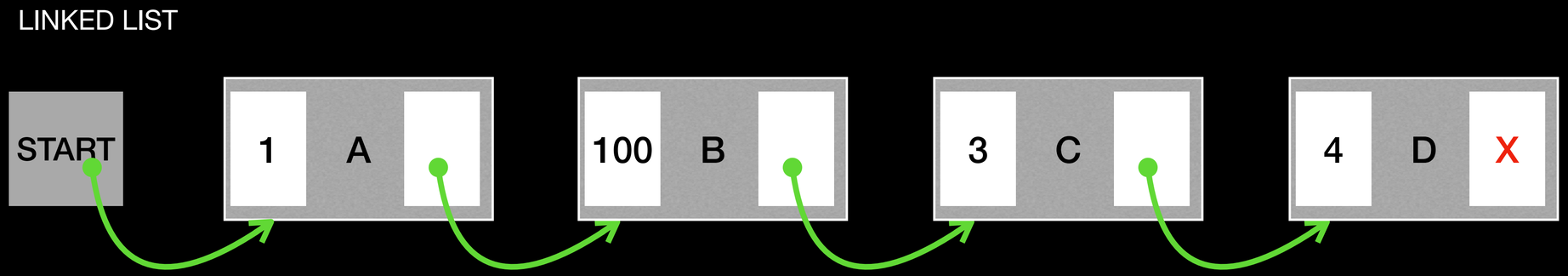
SOME OPERATIONS

update(f1,B,10)



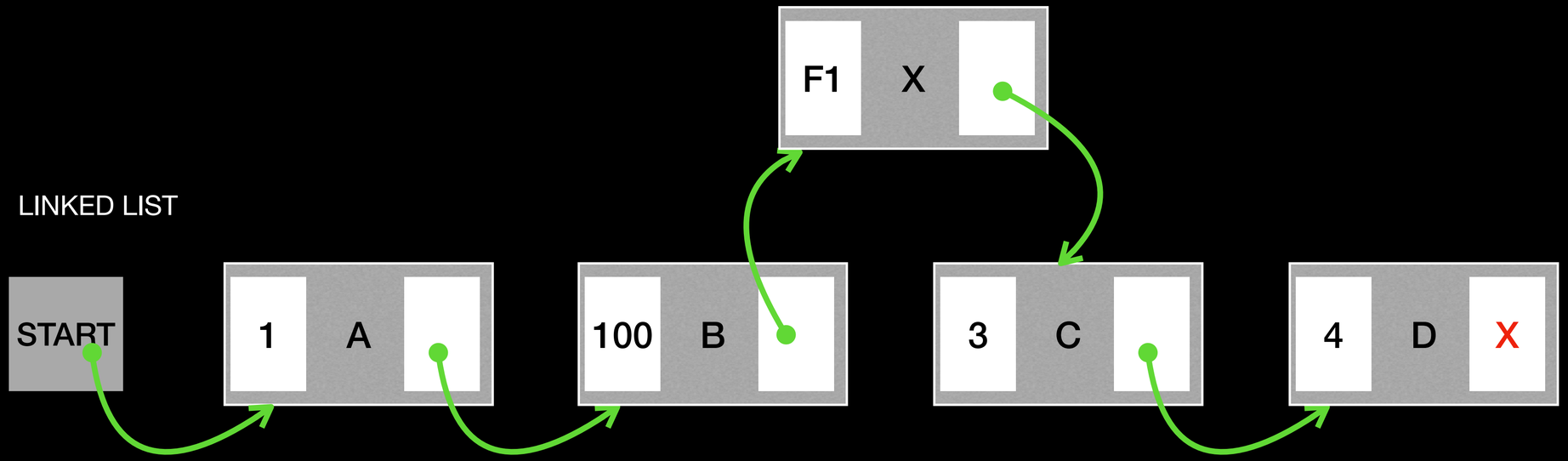
SOME OPERATIONS

update(f1,B,100)



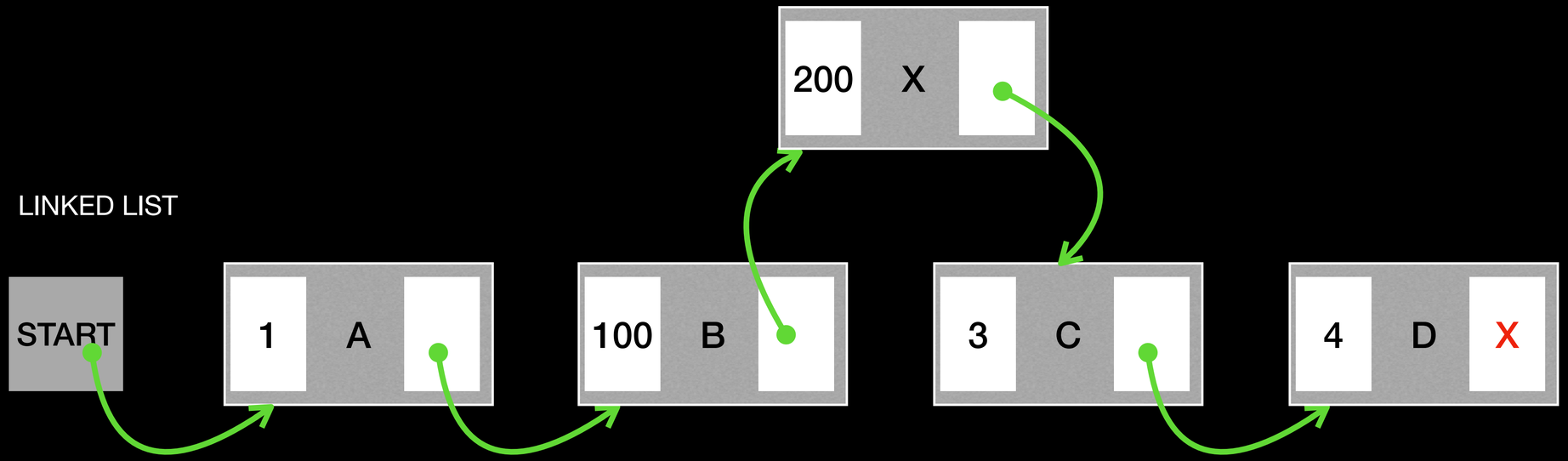
SOME OPERATIONS

add(X,B)



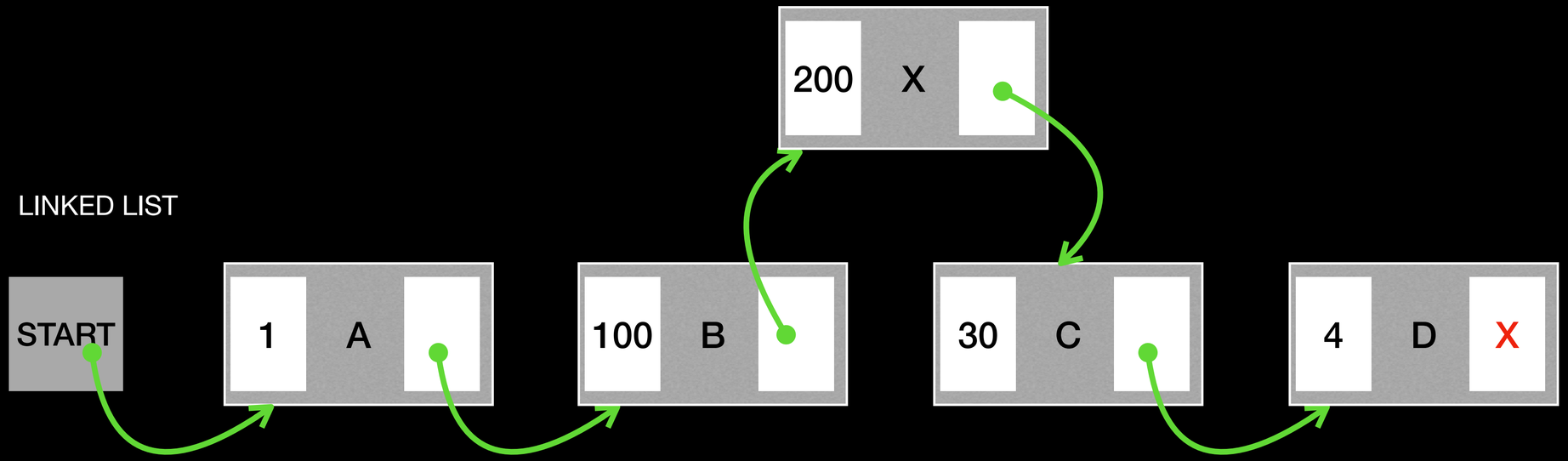
SOME OPERATIONS

update(f1,X,200)



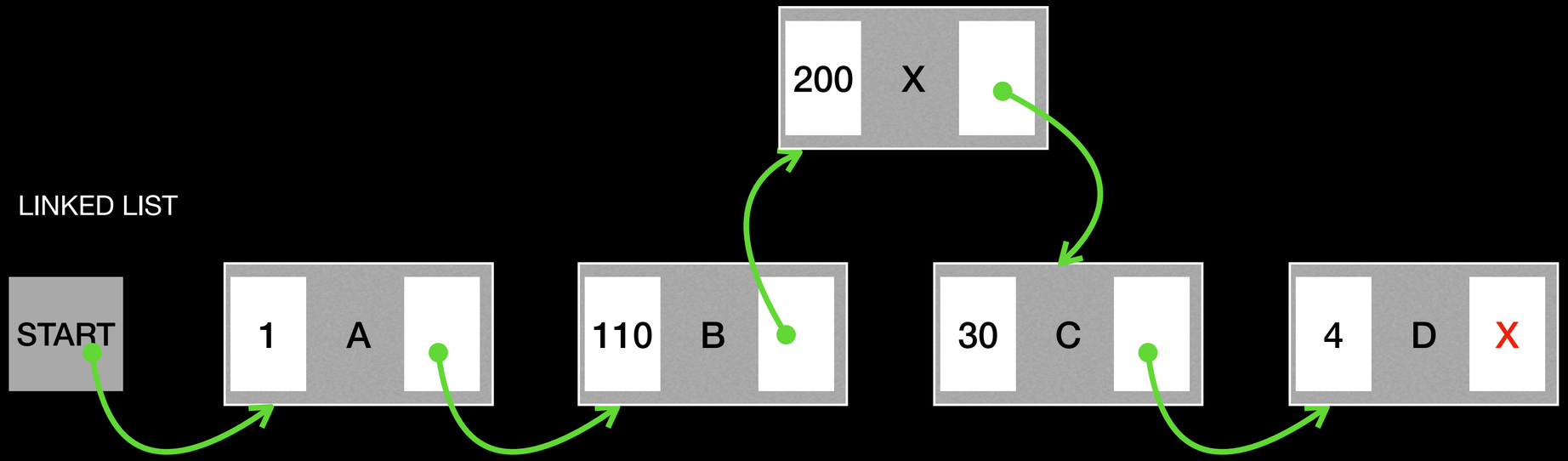
SOME OPERATIONS

update(f1,C,30)



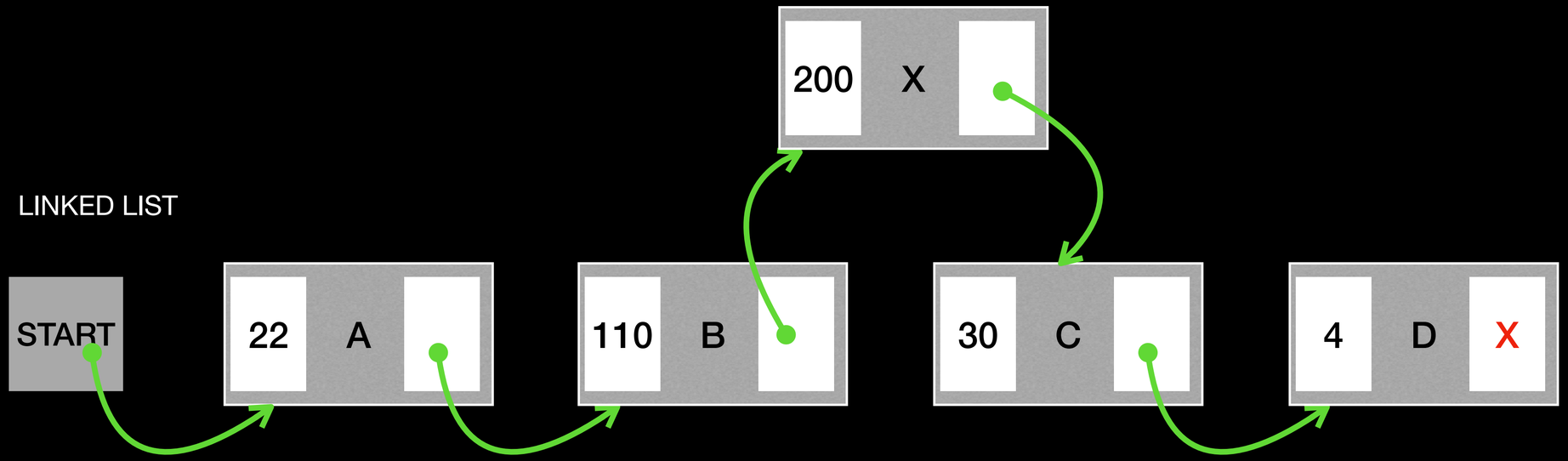
SOME OPERATIONS

update(f1,B,110)



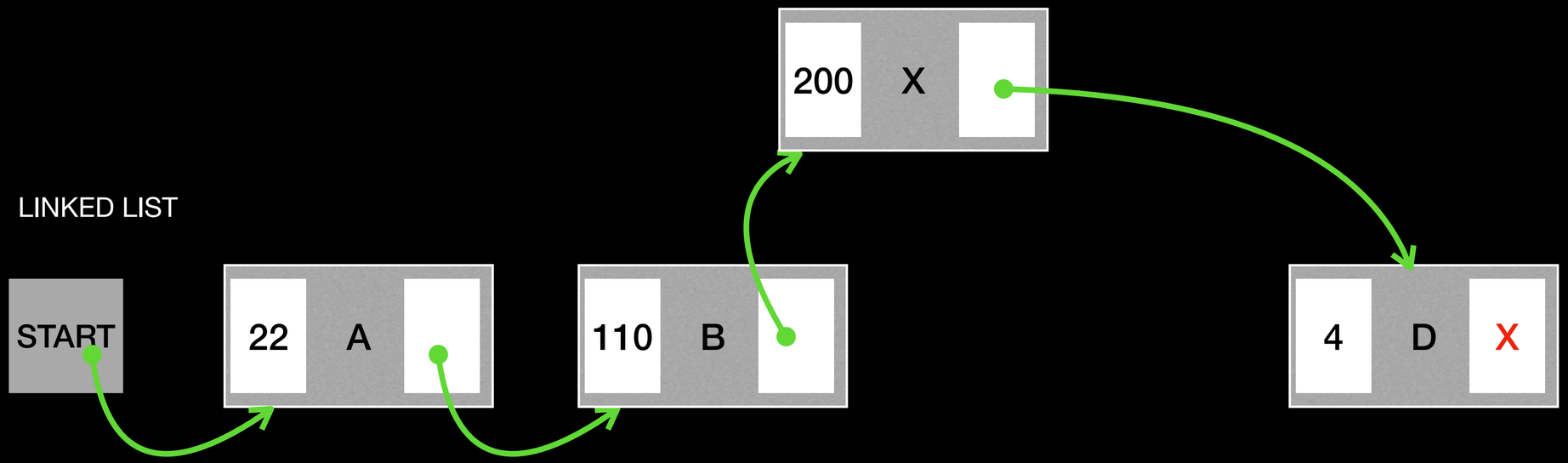
SOME OPERATIONS

update(f1,A,22)



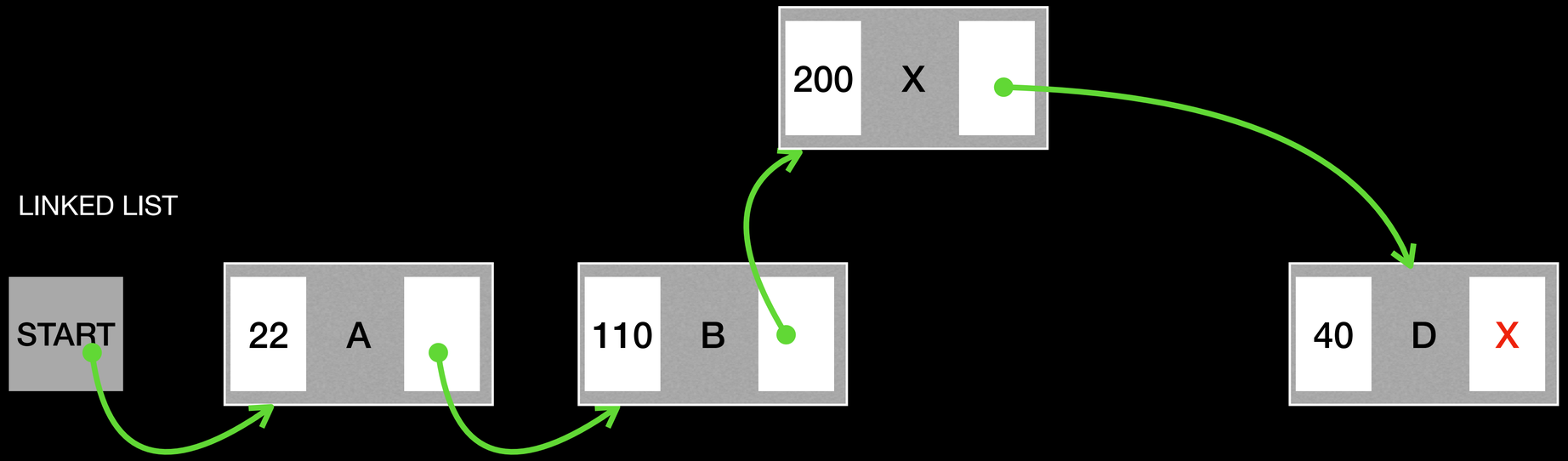
SOME OPERATIONS

remove(C)



SOME OPERATIONS

update(f1,D,40)



Partial Persistent Linked List

Operations:

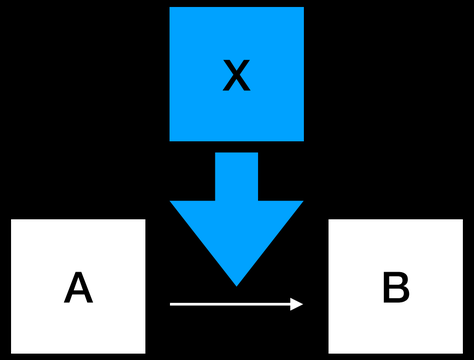
- start = init() : To initiate linked list and “start” pointer holds the starting position in v0
- create_node(x, a): Allocate a new node x with f1 = a and f2 = NULL, and set its default version to current version
- add(x, y, a) : Add new node x with value = a after y and update the version
- remove(x) : Remove node x and update the version
- iterate_over_LL(v) : Iterate over the whole linked list in version v
- update(f_i,x,val) : Update the i-th field in node x to new value ‘val’ and update the version

Interesting thing!

add(x,y) and remove(x) are not Elementary operations

add(X,C,123) consists of

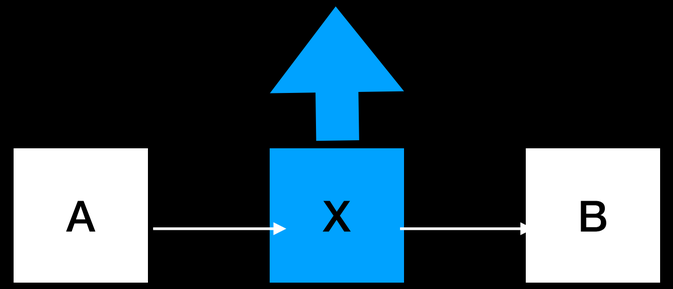
Create node X with value 123
Modify f2 of A to X : update(f2, A, X)
Modify f2 of X to B : update(f2, X, B)
Set BP of X to A: X.bp = A
Set BP of B to X: B.bp = X
~~Set f1 of X(optional)~~



remove(x) consists of

Modify F2 of Parent C (i.e., X) -> F2 of C (successor of C)

update(f2, A, B)
Set BP of B to A: B.bp = A
If all shared reference of X is removed Then free up the memory associated with X

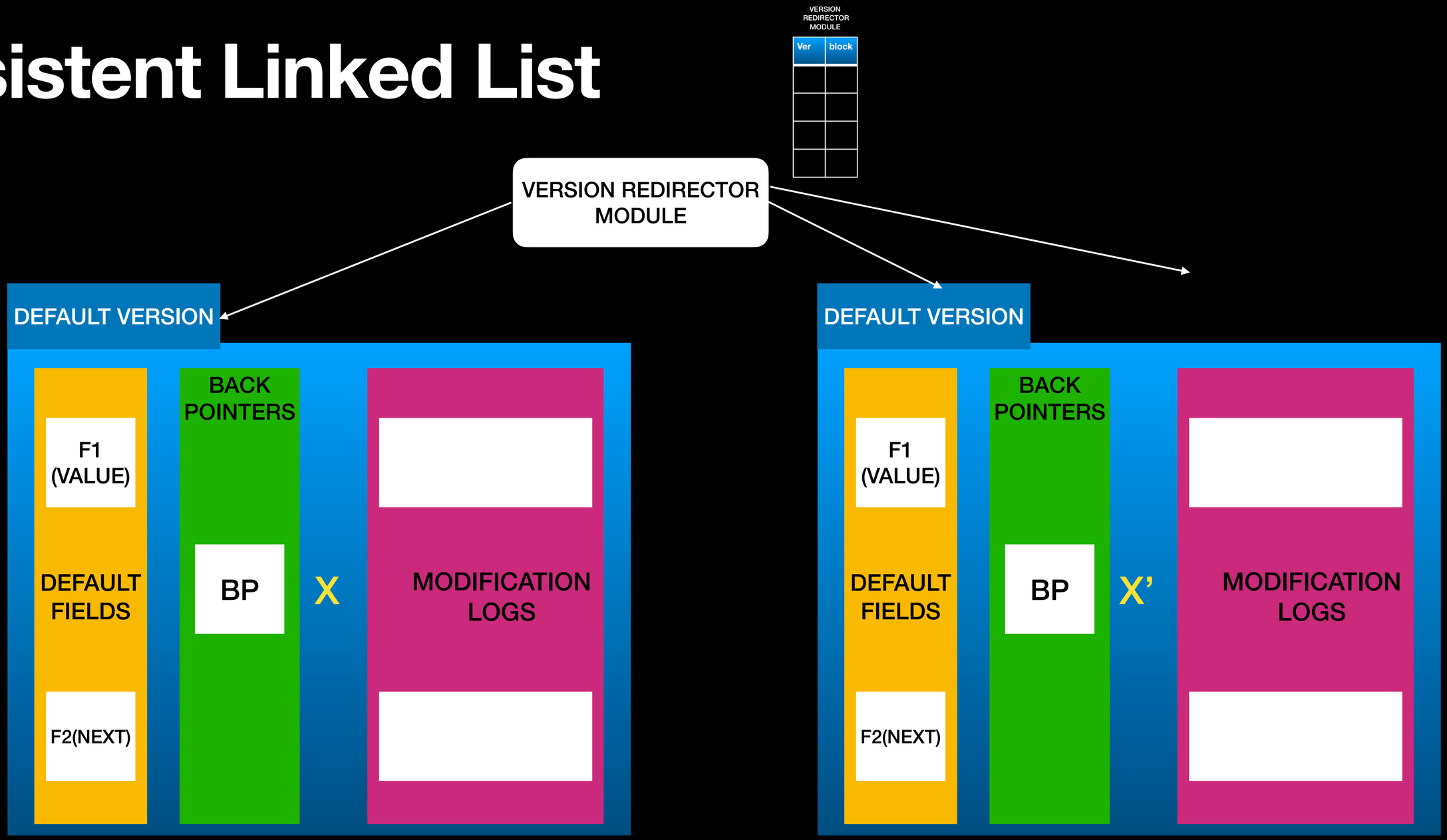


Elementary Operations:

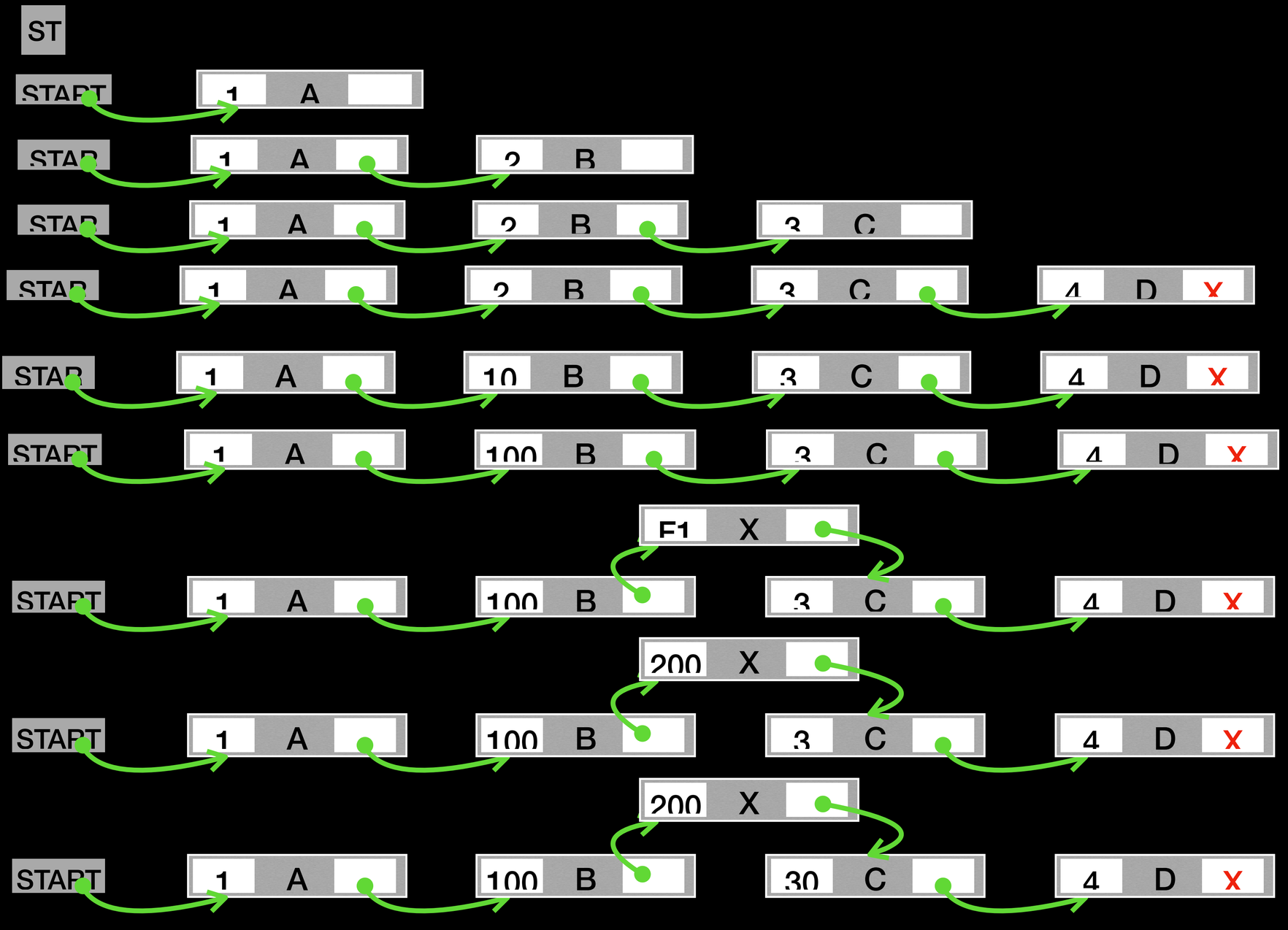
- start = init()
- create_node(x, a)
- iterate_over_LL(v)
- update(f_i,x,val)

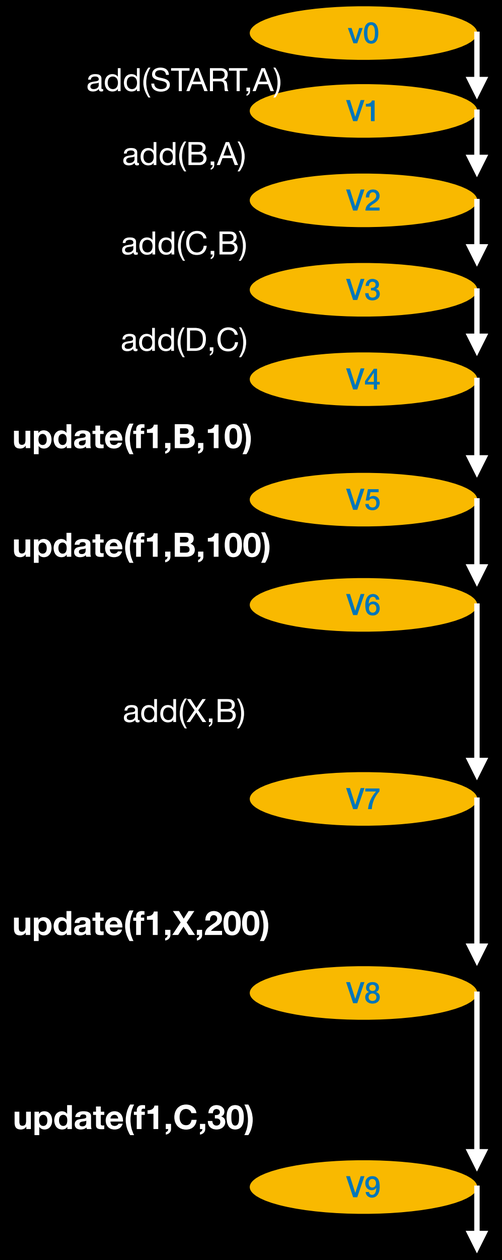
Persistent Linked List

Node

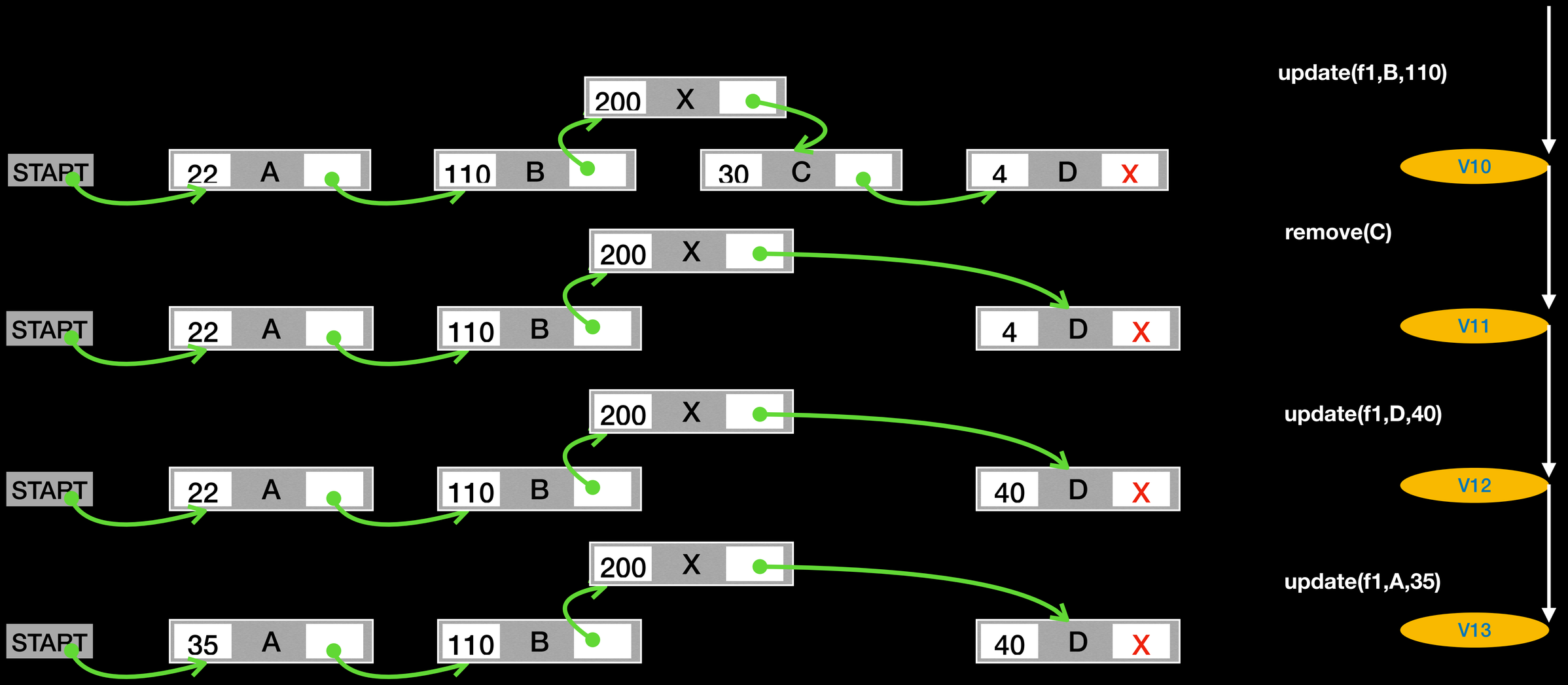


Ephemeral List and Corresponding Versions





Ephemeral List and Corresponding Versions (contd.)



Current Version: v0

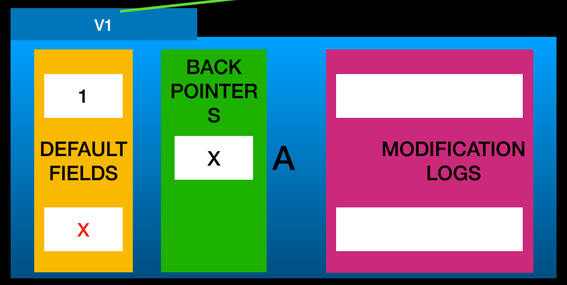
IMPLEMENTATION IN PERSISTENT POINTER MACHINE MODEL

Current Version: v1

START MODULE

V0

add(A,START,1)



VERSION REDIRECTOR MODULE

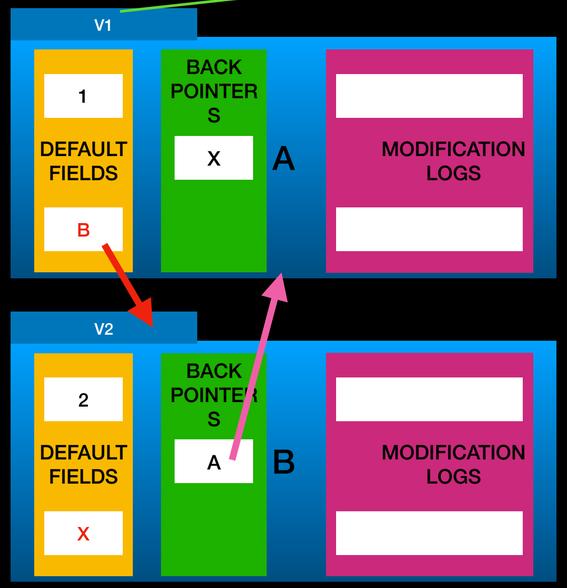
Ver	block
1	A

Current Version: v2

START MODULE

V0

add(B,A,2)



VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

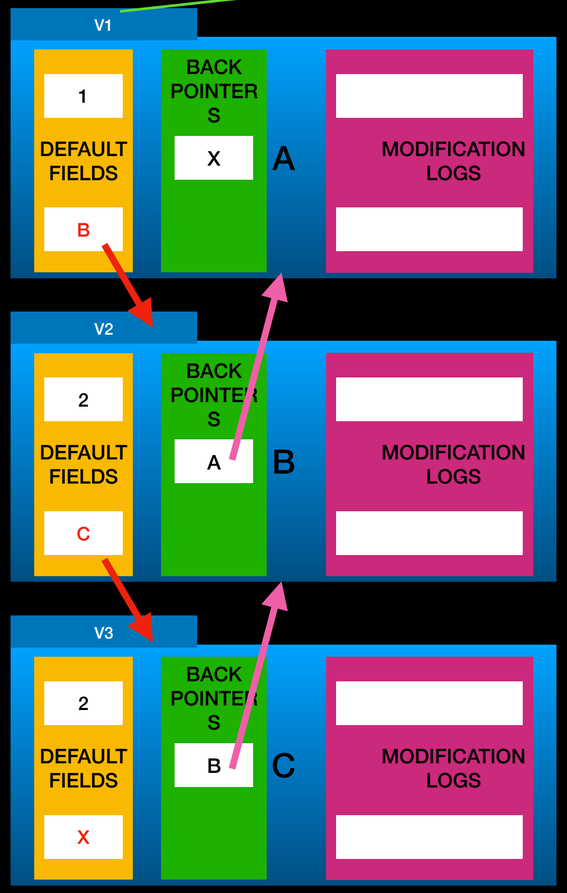
Ver	block
2	B

Current Version: v3

START MODULE

V0

add(C,B,3)



VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B

VERSION REDIRECTOR MODULE

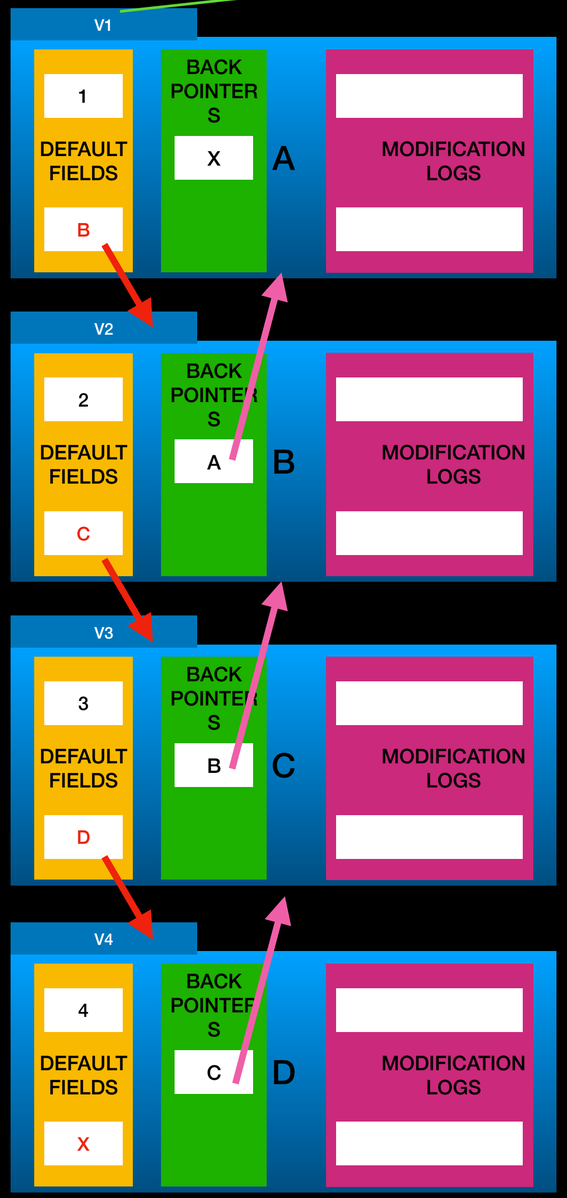
Ver	block
3	C

Current Version: v4

START MODULE

V0

add(D,C,4)



VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B

VERSION REDIRECTOR MODULE

Ver	block
3	C

VERSION REDIRECTOR MODULE

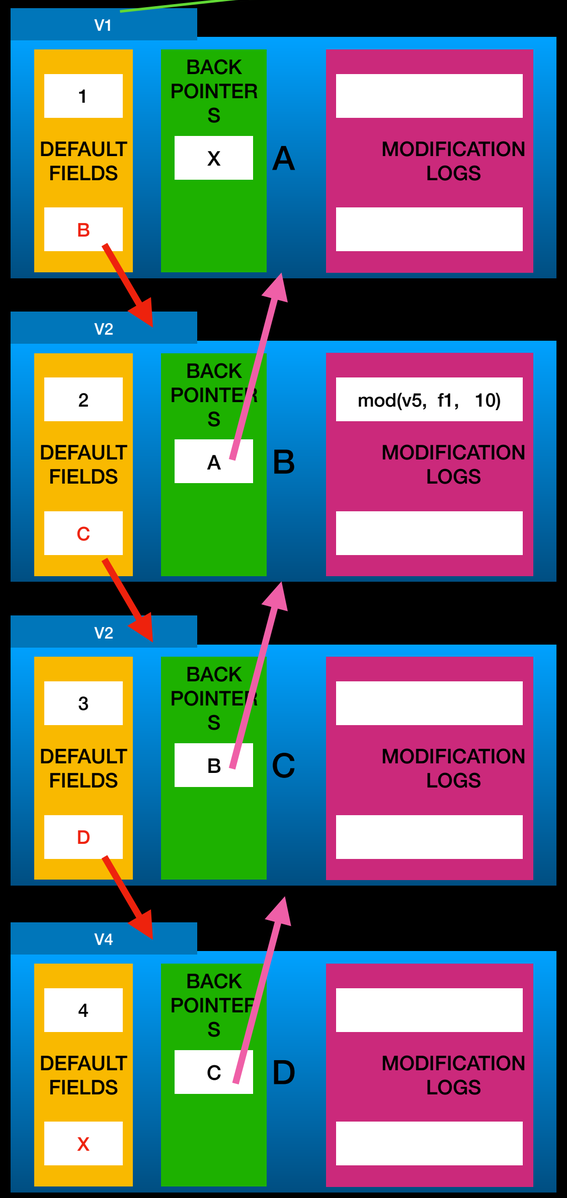
Ver	block

Current Version: v5

START MODULE

V0

update(f1,B,10)



VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B

VERSION REDIRECTOR MODULE

Ver	block
3	C

VERSION REDIRECTOR MODULE

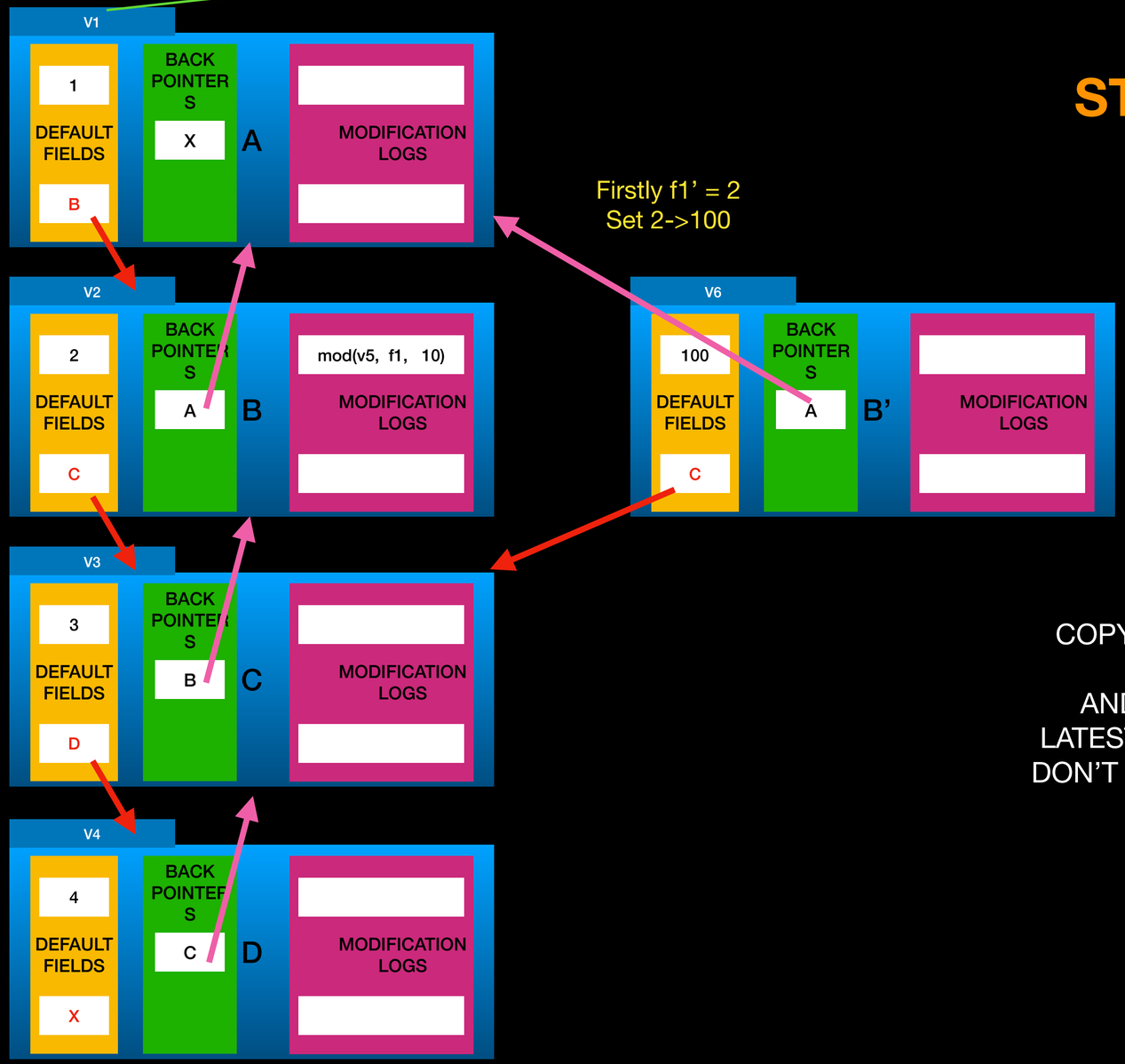
Ver	block
4	D

Current Version: v6

START MODULE

V0

update(f1,B,100)



Firstly f1' = 2
Set 2->100

STEP 1

COPY THE WHOLE NODE
AND KEEP THE LATEST VALUES AND
DON'T COPY MODLOG

VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B'

VERSION REDIRECTOR MODULE

Ver	block
3	C

VERSION REDIRECTOR MODULE

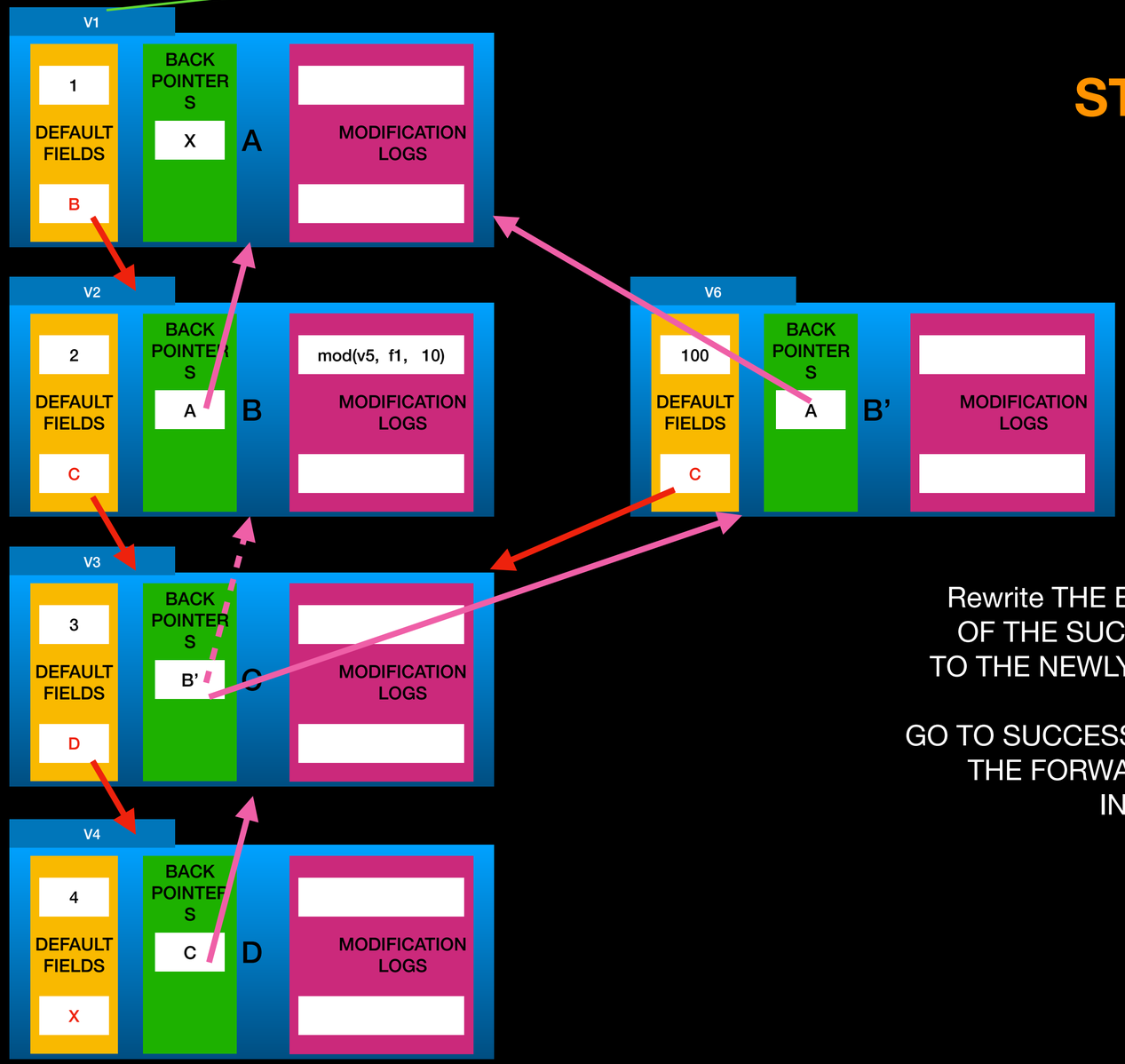
Ver	block
4	D

Current Version: v6

START MODULE

V0

update(f1,B,100)



STEP 2

Rewrite THE BACK POINTERS
OF THE SUCCESSOR NODE
TO THE NEWLY CREATED NODE

GO TO SUCCESSOR NODES USING
THE FORWARD POINTERS
IN O(1)

VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B'

VERSION REDIRECTOR MODULE

Ver	block
3	C

VERSION REDIRECTOR MODULE

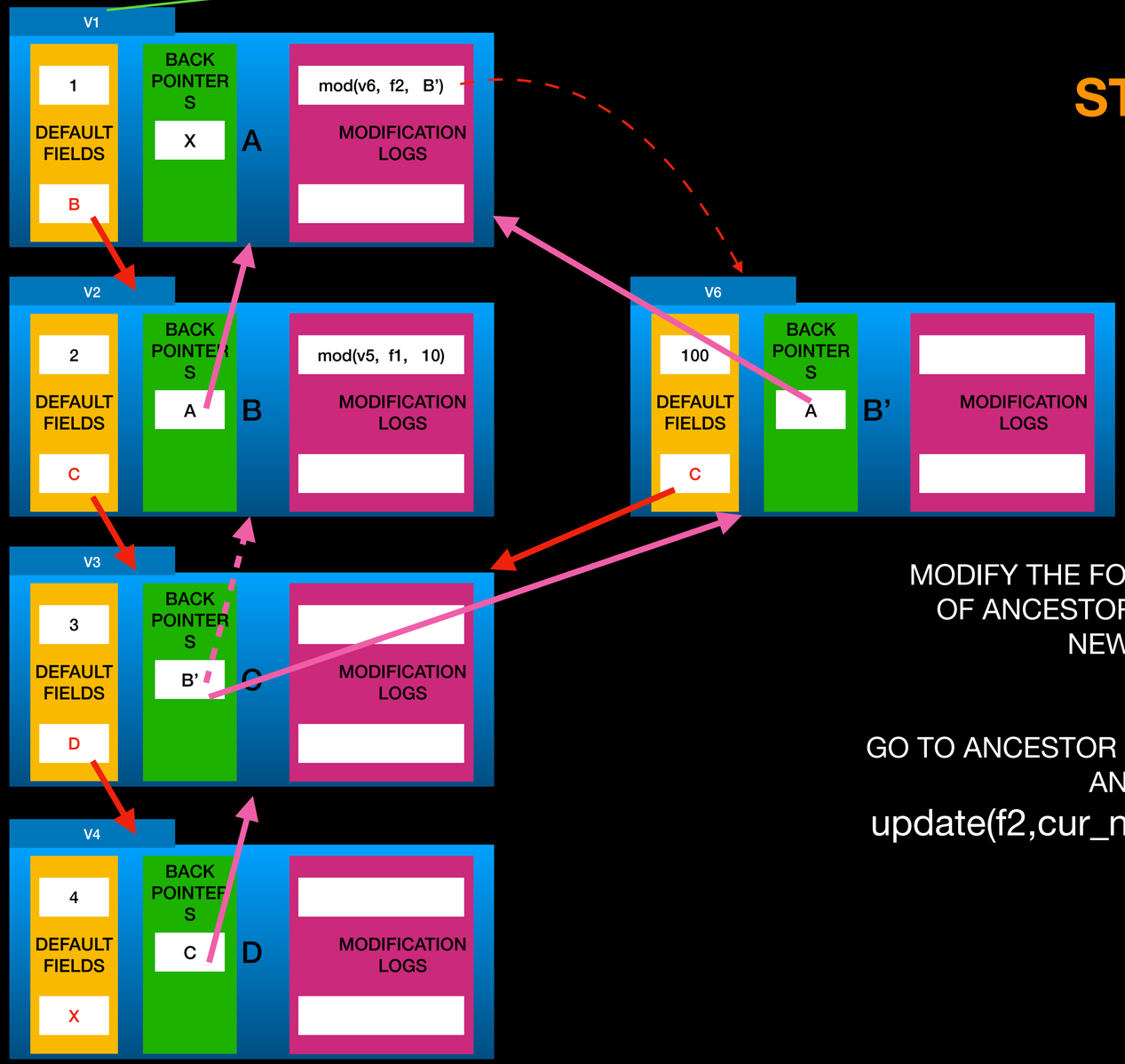
Ver	block
4	D

Current Version: v6

START MODULE

V0

update(f1,B,100)



STEP 3

MODIFY THE FORWARD POINTERS OF ANCESTOR NODES TO THE NEW NODE

GO TO ANCESTOR NODES RECURSIVELY AND DO

update(f2,cur_node,NEW_NODE)

VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B'

VERSION REDIRECTOR MODULE

Ver	block
3	C

VERSION REDIRECTOR MODULE

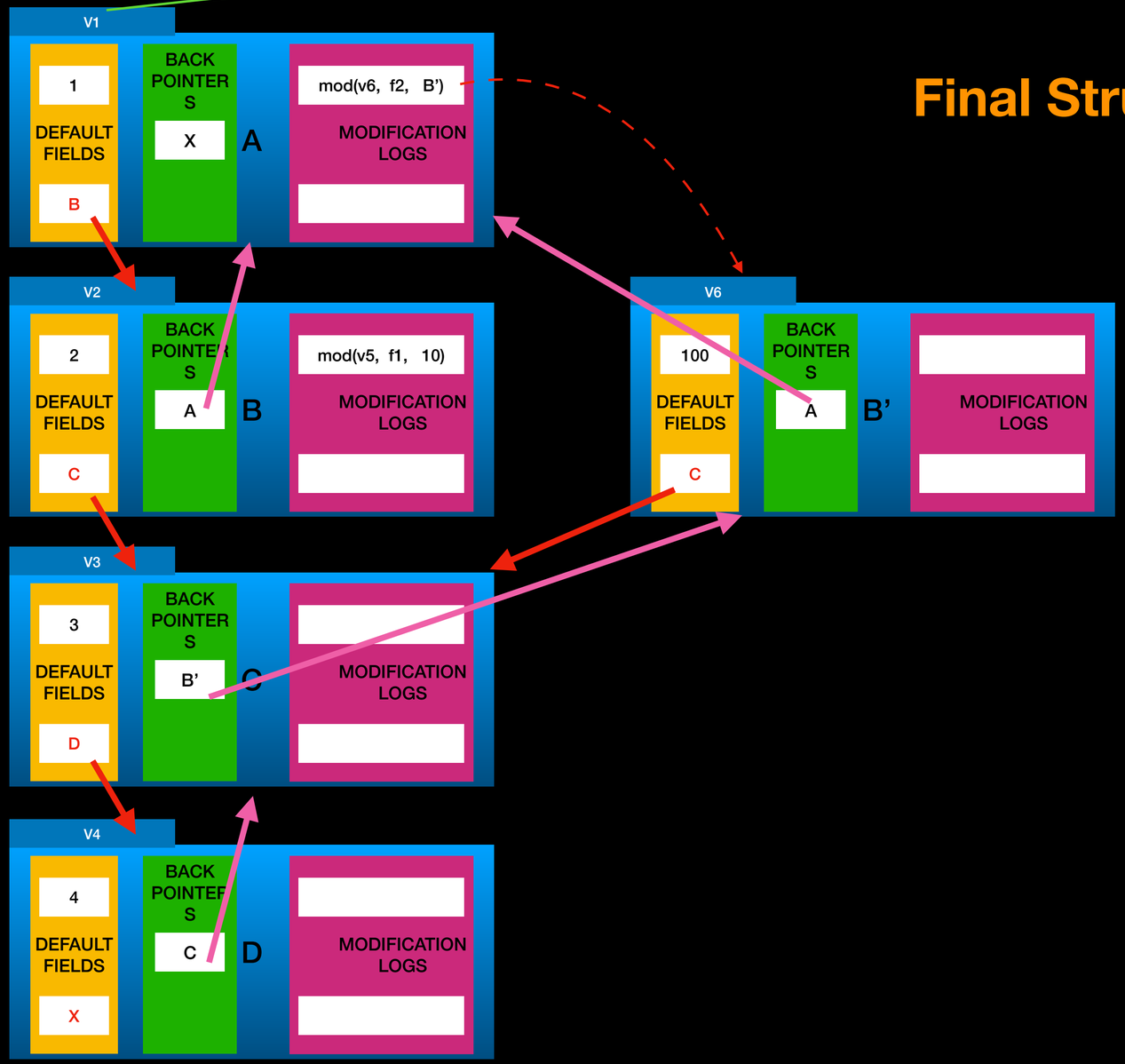
Ver	block
4	D

Current Version: v6

START MODULE

V0

update(f1,B,100)



Final Structure of v6

VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B'

VERSION REDIRECTOR MODULE

Ver	block
3	C

VERSION REDIRECTOR MODULE

Ver	block
4	D

Current Version: v7

START MODULE

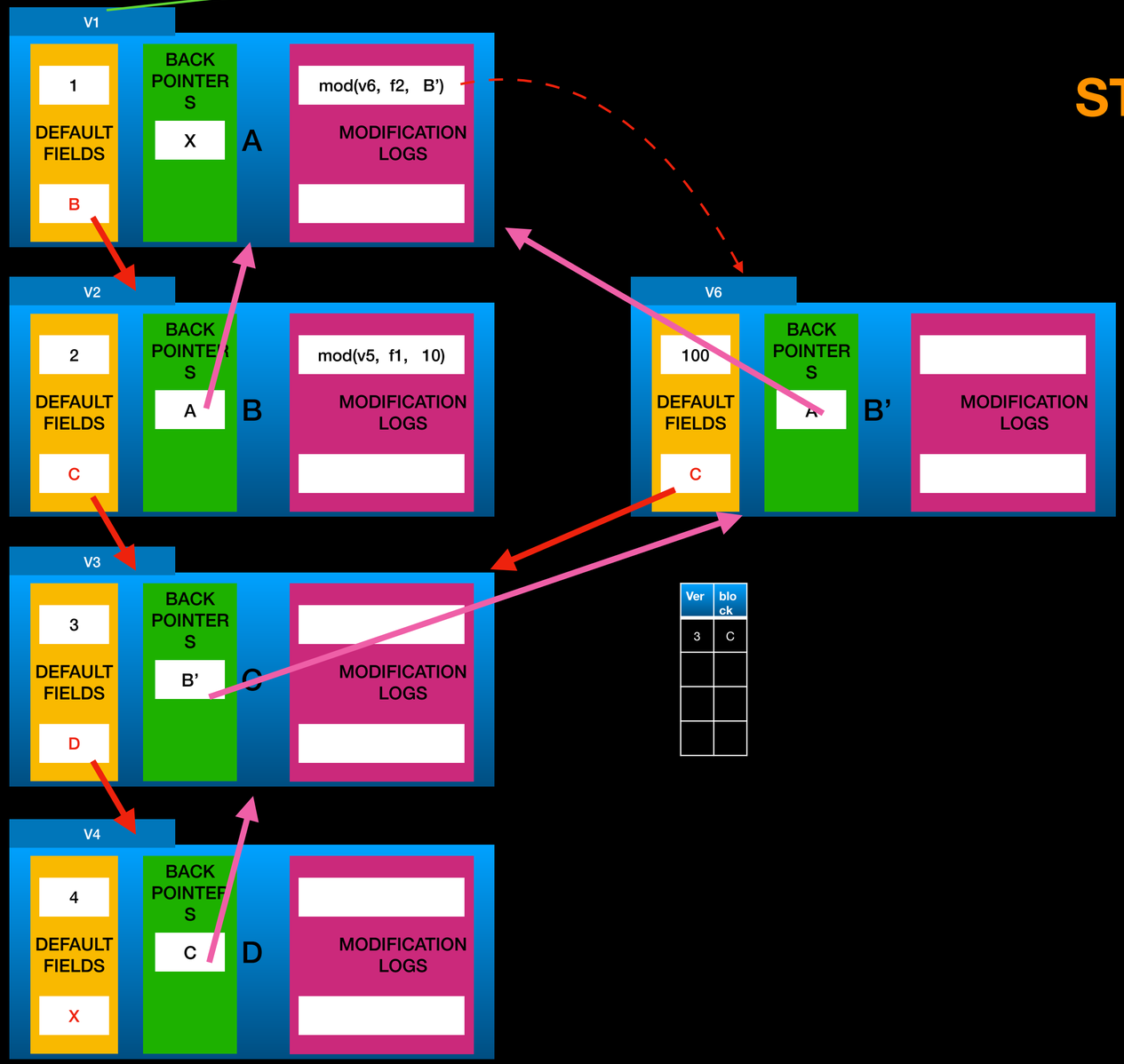
V0

add(X,B,_)

add(X,B,_) consists of

Create node X
Modify f2 of B’ to X
Modify f2 of X to C + Set BP of X to B’
Update BP of C to X
Set f1 of X

STEP 1



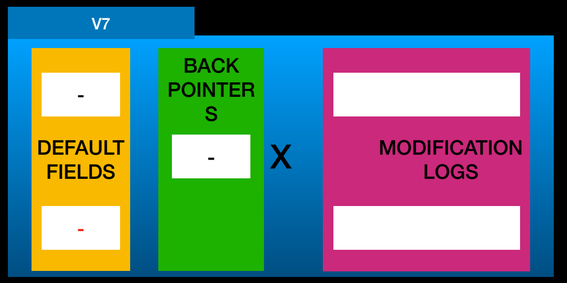
Ver	block
3	C

VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B’



VERSION REDIRECTOR MODULE

Ver	block
7	X

VERSION REDIRECTOR MODULE

Ver	block
4	D

Current Version: v7

START MODULE

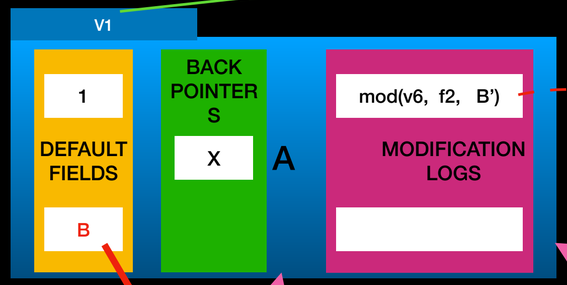
V0

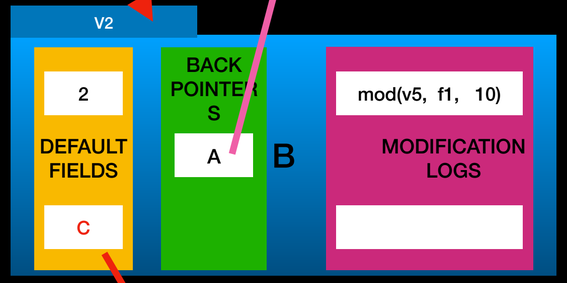
add(X,B,_)

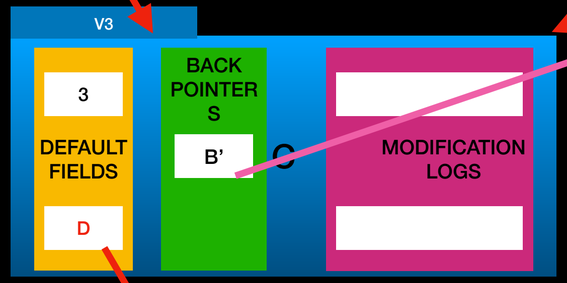
add(X,B,_) consists of

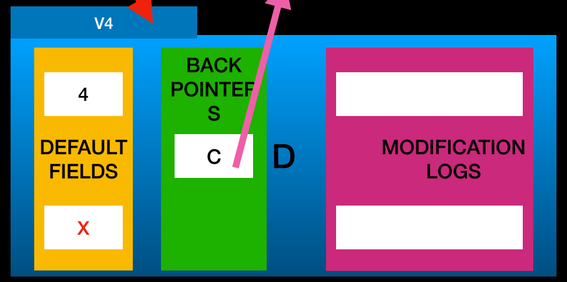
Create node X
Modify f2 of B’ to X
Modify f2 of X to C + Set BP of X to B’
Update BP of C to X
Set f1 of X

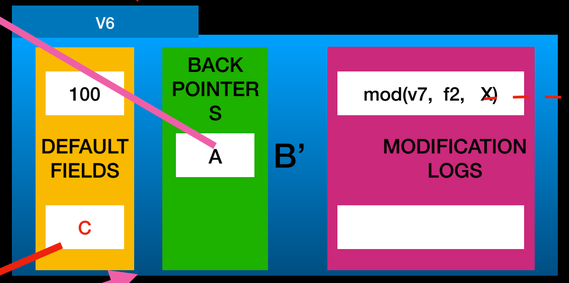
STEP 2



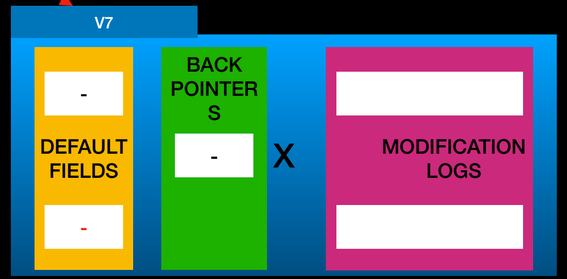








Ver	block
3	C



VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B’

VERSION REDIRECTOR MODULE

Ver	block
7	X

VERSION REDIRECTOR MODULE

Ver	block
4	D

Current Version: v7

START MODULE

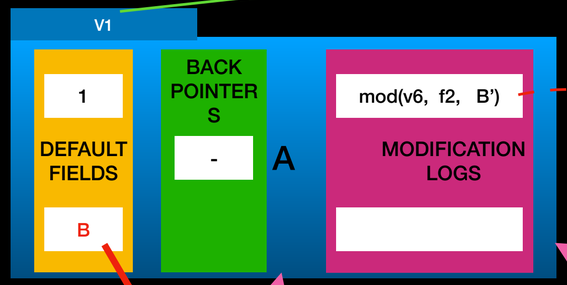
V0

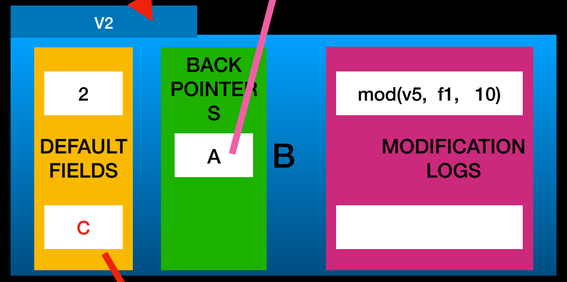
add(X,B,_)

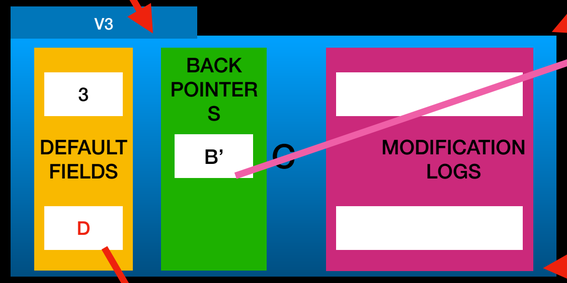
add(X,B,_) consists of

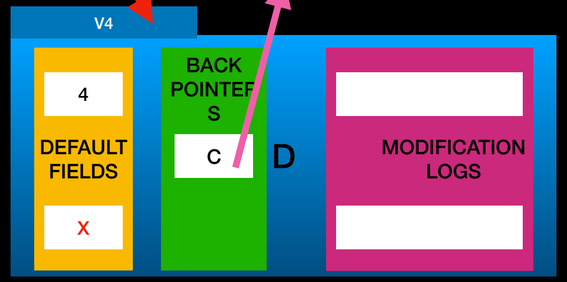
Create node X
Modify f2 of B’ to X
Modify f2 of X to C + Set BP of X to B’
Update BP of C to X
Set f1 of X

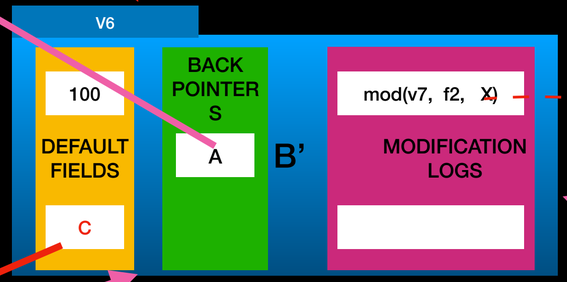
STEP 3

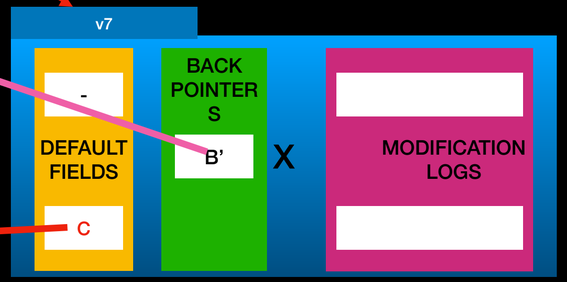












Ver	block
3	C

VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B’

VERSION REDIRECTOR MODULE

Ver	block
7	X

VERSION REDIRECTOR MODULE

Ver	block
4	D

Current Version: v7

START MODULE

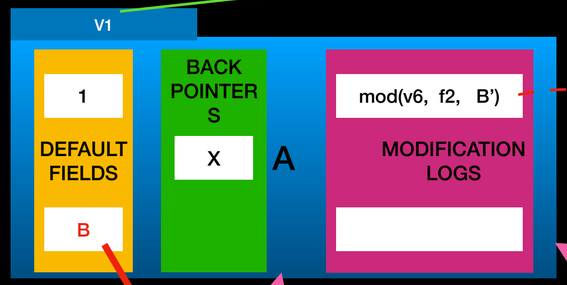
V0

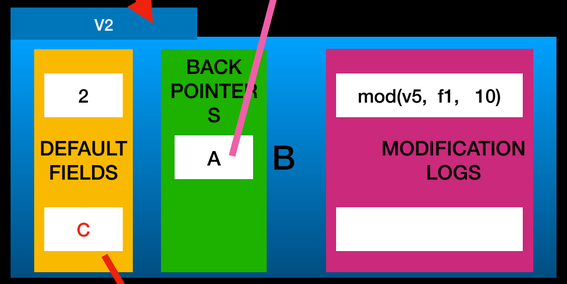
add(X,B,_)

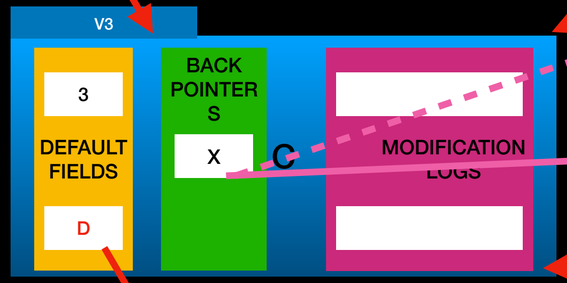
add(X,B,_) consists of

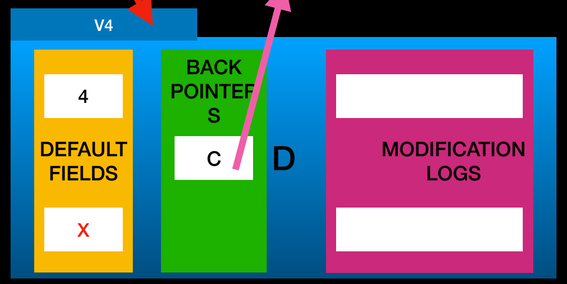
Create node X
Modify f2 of B’ to X
Modify f2 of X to C + Set BP of X to B’
Update BP of C to X
Set f1 of X

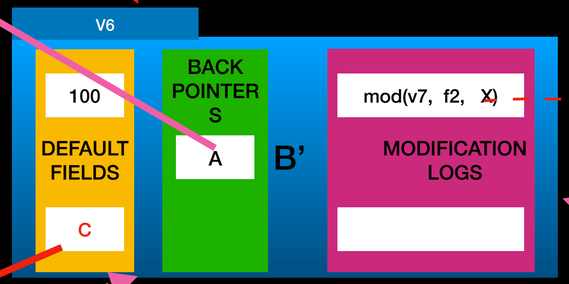
STEP 4



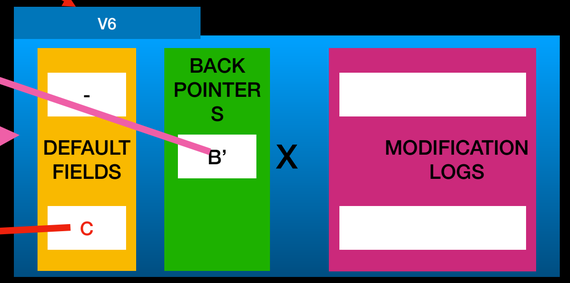








Ver	block
3	C



VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B’

VERSION REDIRECTOR MODULE

Ver	block
7	X

VERSION REDIRECTOR MODULE

Ver	block
4	D

Current Version: v7

START MODULE

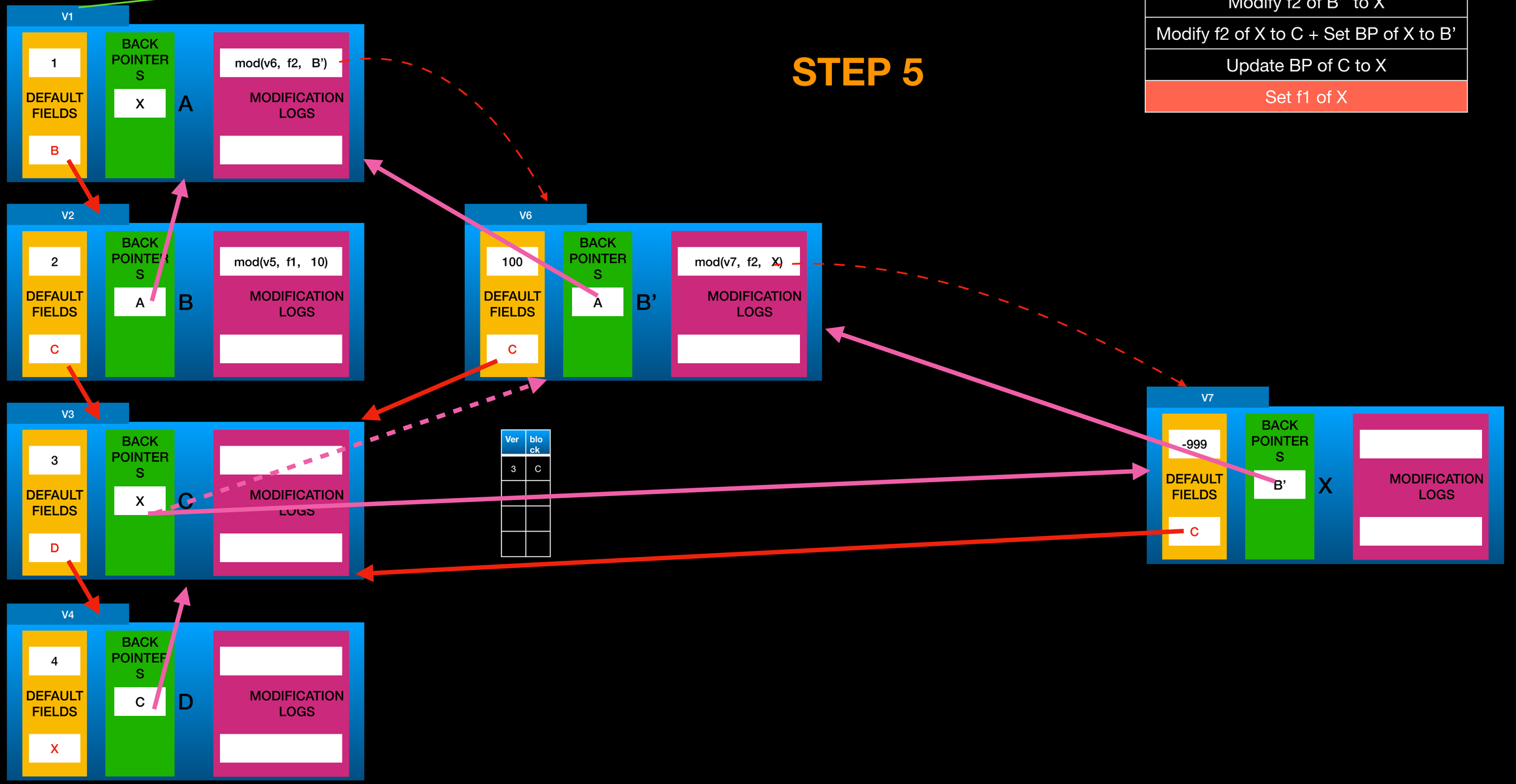
V0

add(X,B,_)

add(X,B,_) consists of

Create node X
Modify f2 of B’ to X
Modify f2 of X to C + Set BP of X to B’
Update BP of C to X
Set f1 of X

STEP 5



VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B’

VERSION REDIRECTOR MODULE

Ver	block
7	X

VERSION REDIRECTOR MODULE

Ver	block
4	D

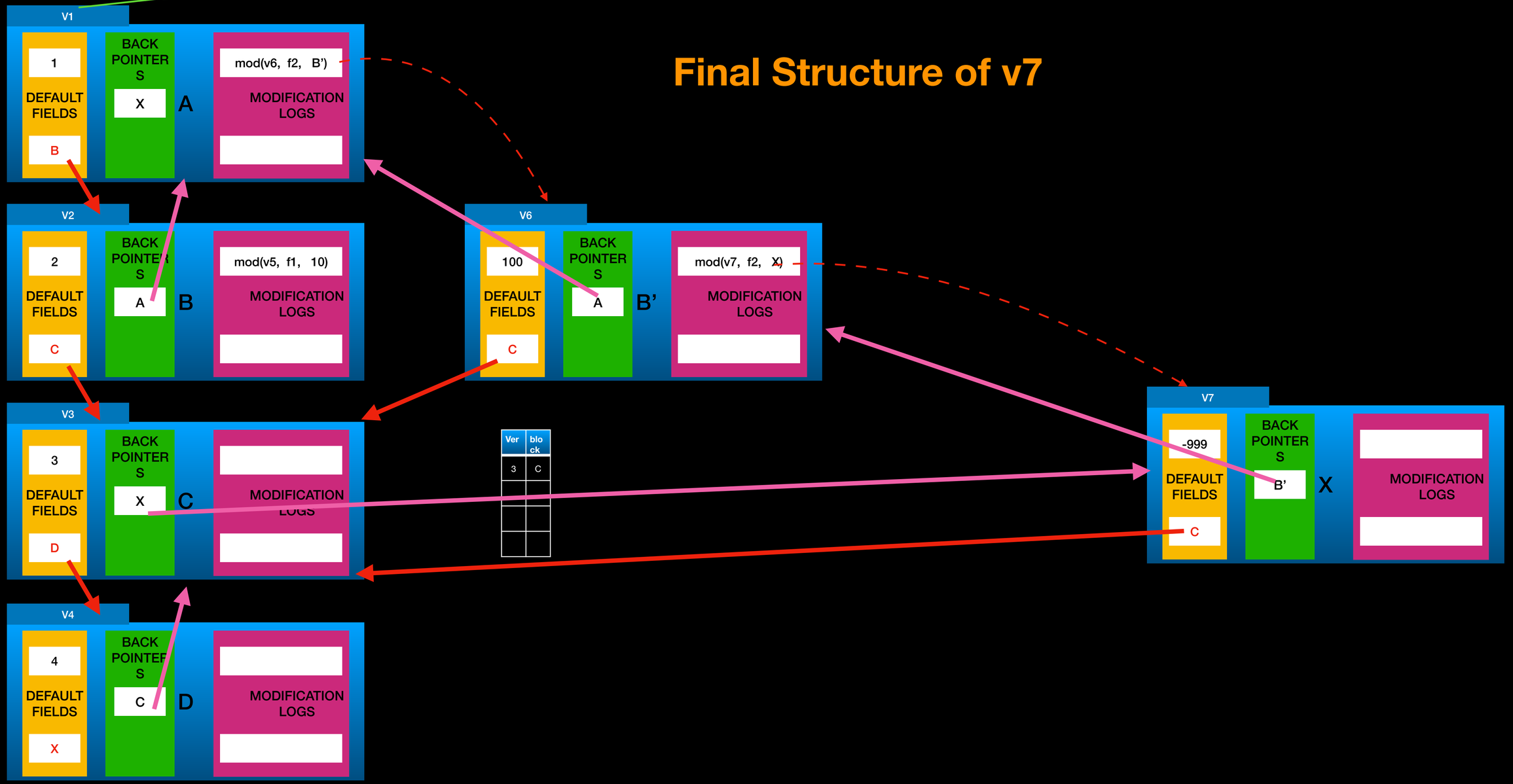
Current Version: v7

START MODULE

V0

add(X,B,_)

Final Structure of v7



Ver	block
3	C

VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B'

VERSION REDIRECTOR MODULE

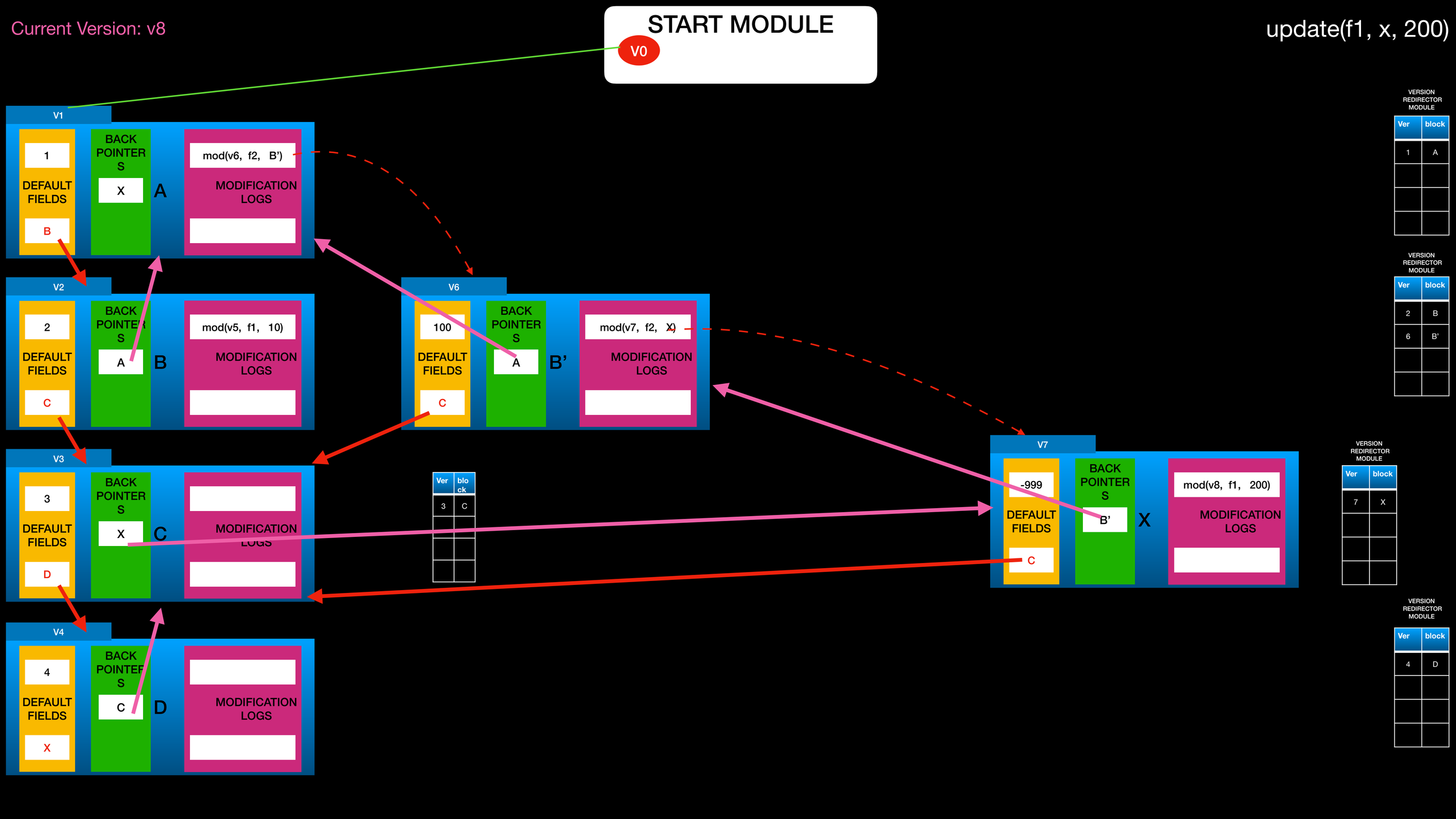
Ver	block
7	X

VERSION REDIRECTOR MODULE

Ver	block
4	D

Current Version: v8

update(f1, x, 200)

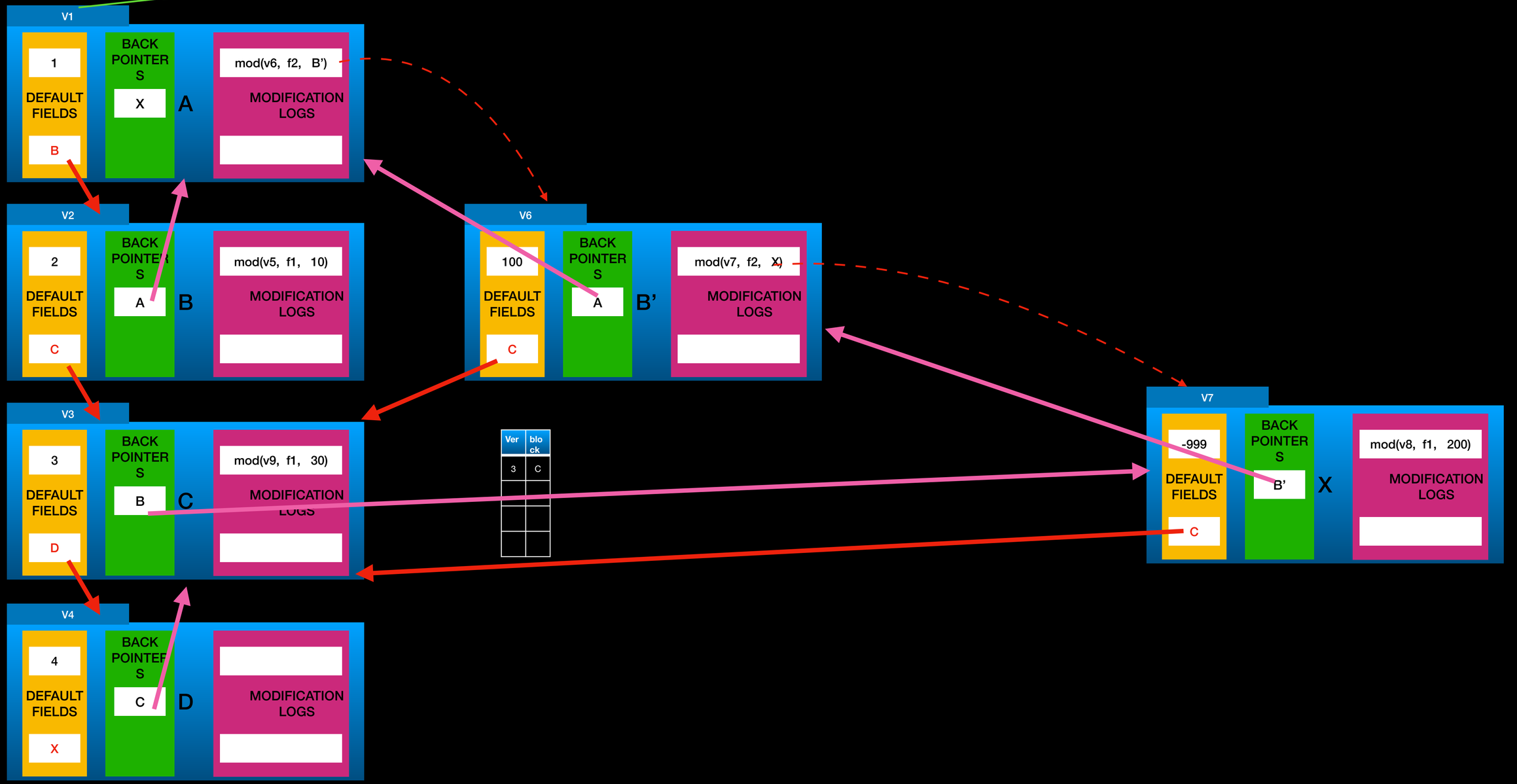


Current Version: v9

START MODULE

V0

update(f1, C,30)



Ver	block
3	C

VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B'

VERSION REDIRECTOR MODULE

Ver	block
7	X

VERSION REDIRECTOR MODULE

Ver	block
4	D

Current Version: v10

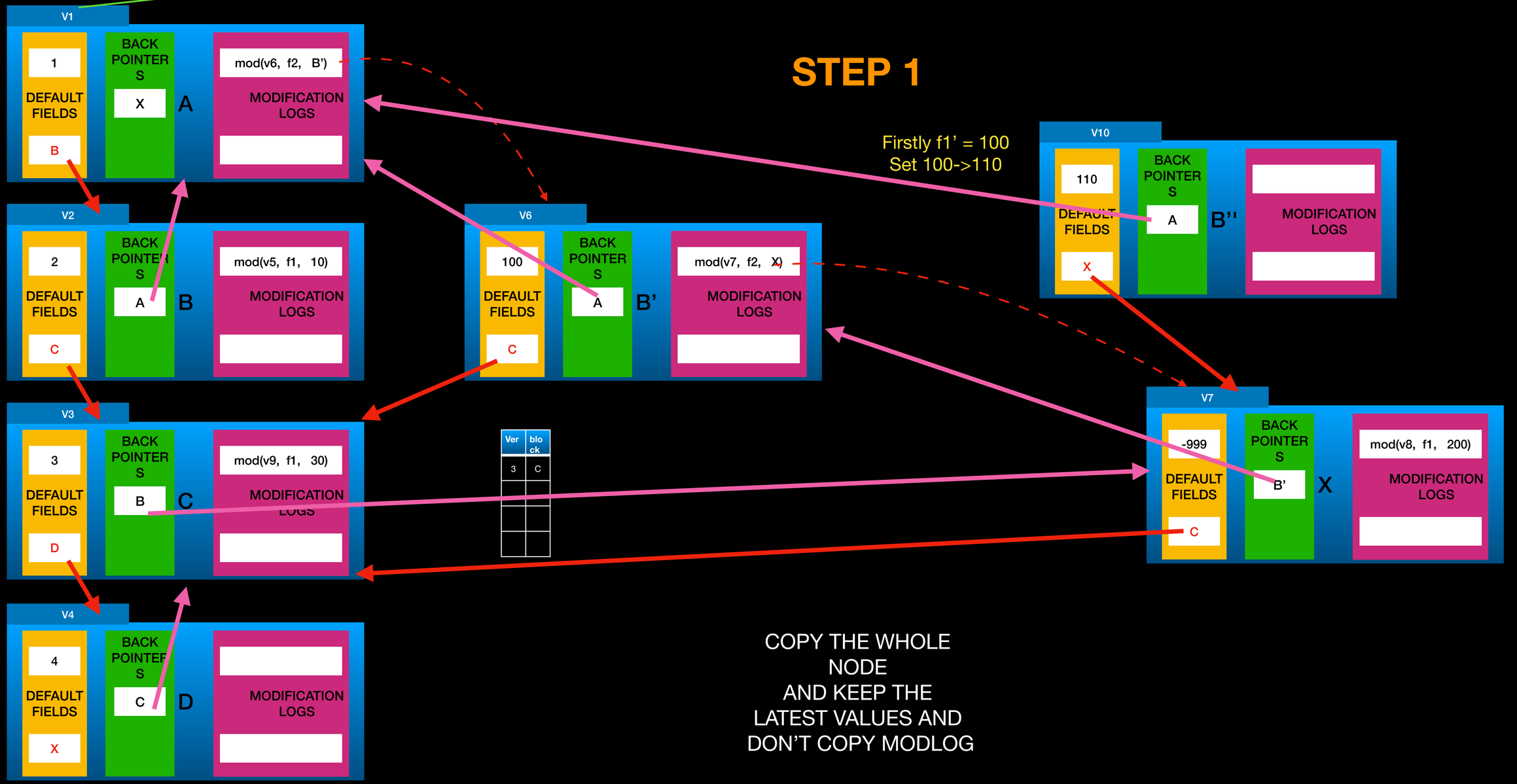
START MODULE

V0

update(f1, B,110)

STEP 1

Firstly f1' = 100
Set 100->110



Ver	block
3	C

VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B'
10	B''

VERSION REDIRECTOR MODULE

Ver	block
7	X

VERSION REDIRECTOR MODULE

Ver	block
4	D

COPY THE WHOLE NODE
AND KEEP THE LATEST VALUES AND
DON'T COPY MODLOG

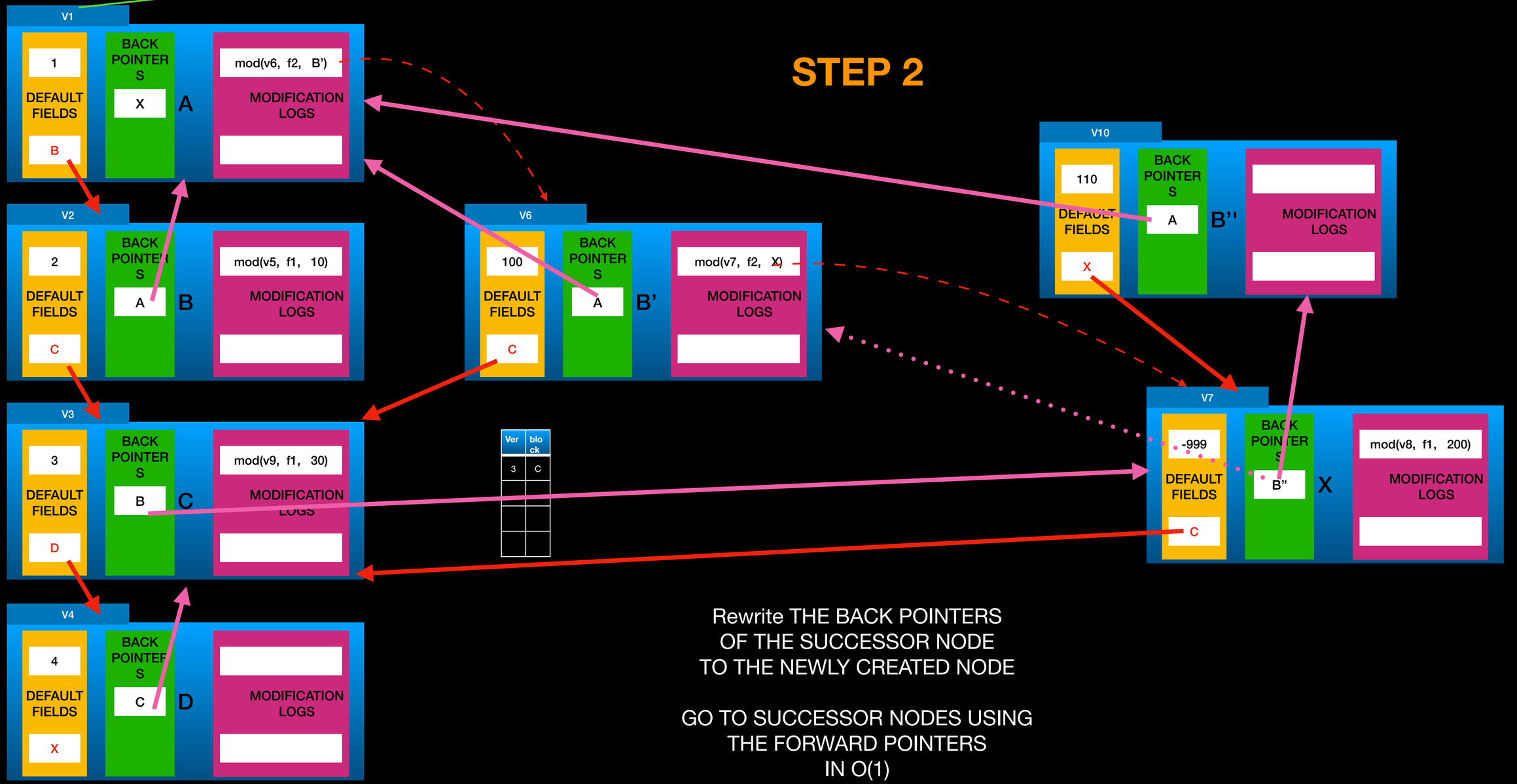
Current Version: v10

START MODULE

V0

update(f1, B,110)

STEP 2



Ver	block
3	C

VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B'
10	B"

VERSION REDIRECTOR MODULE

Ver	block
7	X

VERSION REDIRECTOR MODULE

Ver	block
4	D

Rewrite THE BACK POINTERS
OF THE SUCCESSOR NODE
TO THE NEWLY CREATED NODE

GO TO SUCCESSOR NODES USING
THE FORWARD POINTERS
IN O(1)

Current Version: v10

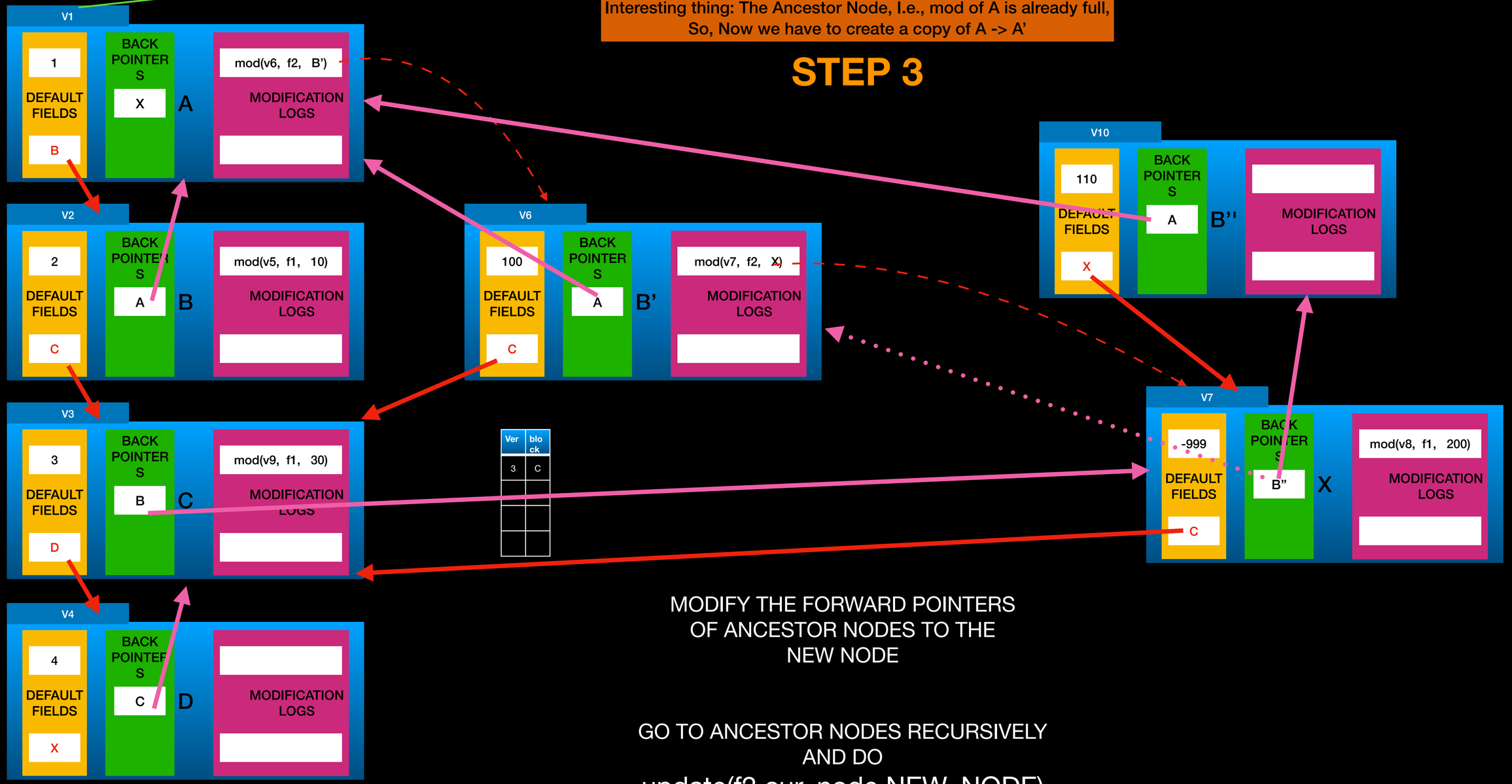
update(f1, B,110)

START MODULE

V0

Interesting thing: The Ancestor Node, I.e., mod of A is already full,
So, Now we have to create a copy of A -> A'

STEP 3



VERSION REDIRECTOR MODULE

Ver	block
1	A

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B'
10	B"

VERSION REDIRECTOR MODULE

Ver	block
7	X

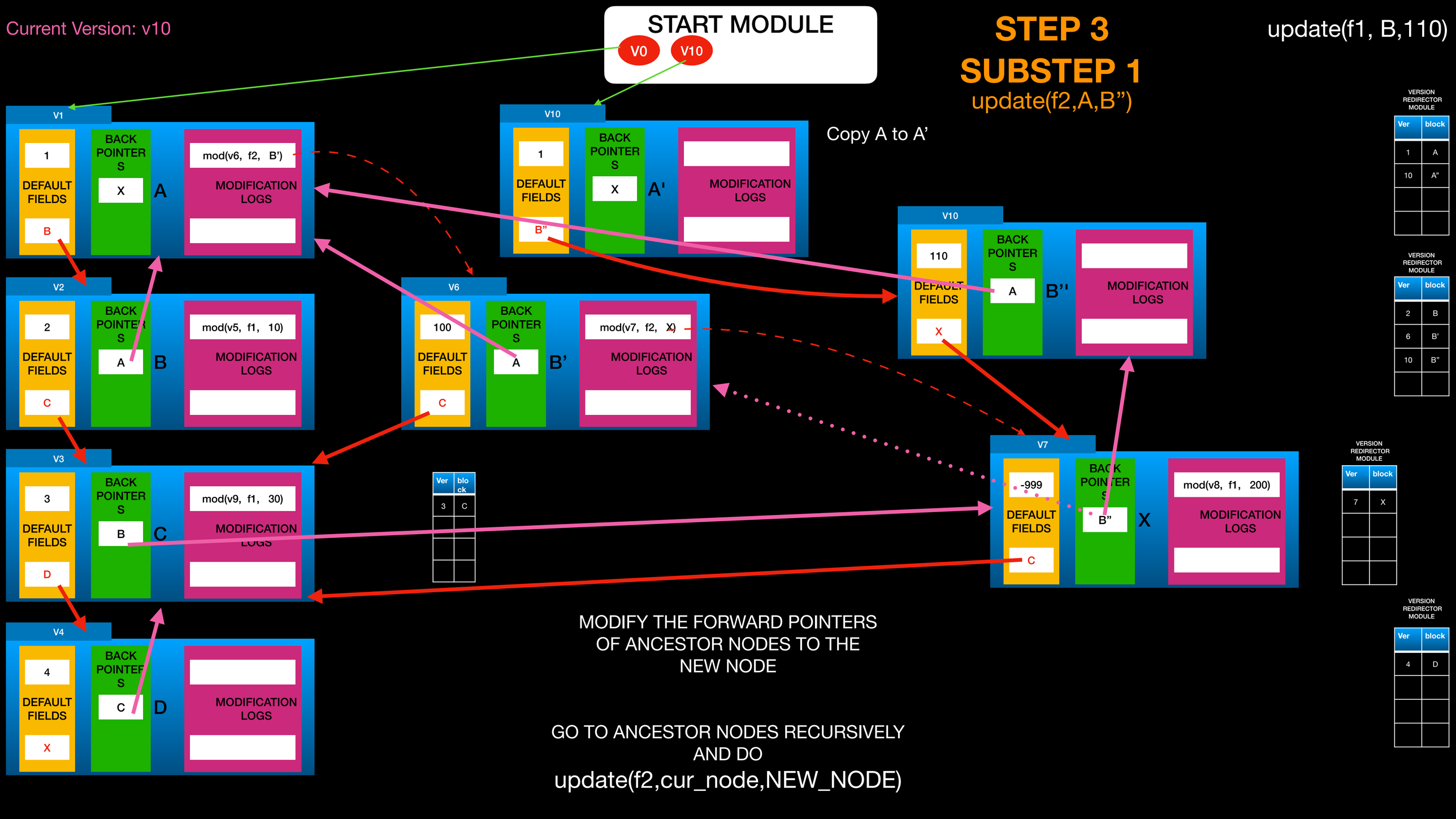
VERSION REDIRECTOR MODULE

Ver	block
4	D

MODIFY THE FORWARD POINTERS
OF ANCESTOR NODES TO THE
NEW NODE

GO TO ANCESTOR NODES RECURSIVELY
AND DO

update(f2,cur_node,NEW_NODE)


Current Version: v10
START MODULE
V0
V10
STEP 3
SUBSTEP 1
update(f2,A,B")
update(f1, B,110)
Copy A to A'
V1
1
DEFAULT FIELDS
B
BACK POINTERS
X
A
mod(v6, f2, B')
MODIFICATION LOGS
V10
1
DEFAULT FIELDS
B"
BACK POINTERS
X
A'
MODIFICATION LOGS
V10
110
DEFAULT FIELDS
X
BACK POINTERS
A
B''
MODIFICATION LOGS
V2
2
DEFAULT FIELDS
C
BACK POINTERS
A
B
mod(v5, f1, 10)
MODIFICATION LOGS
V6
100
DEFAULT FIELDS
C
BACK POINTERS
A
B'
mod(v7, f2, X)
MODIFICATION LOGS
V3
3
DEFAULT FIELDS
D
BACK POINTERS
B
C
mod(v9, f1, 30)
MODIFICATION LOGS
V7
-999
DEFAULT FIELDS
C
BACK POINTERS
B"
X
mod(v8, f1, 200)
MODIFICATION LOGS
V4
4
DEFAULT FIELDS
X
BACK POINTERS
C
D
MODIFICATION LOGS
Ver
block
3
C
VERSION REDIRECTOR MODULE
Ver
block
1
A
10
A"
VERSION REDIRECTOR MODULE
Ver
block
2
B
6
B'
10
B"
VERSION REDIRECTOR MODULE
Ver
block
7
X
VERSION REDIRECTOR MODULE
Ver
block
4
D
MODIFY THE FORWARD POINTERS OF ANCESTOR NODES TO THE NEW NODE
GO TO ANCESTOR NODES RECURSIVELY AND DO
update(f2,cur_node,NEW_NODE)

Current Version: v10

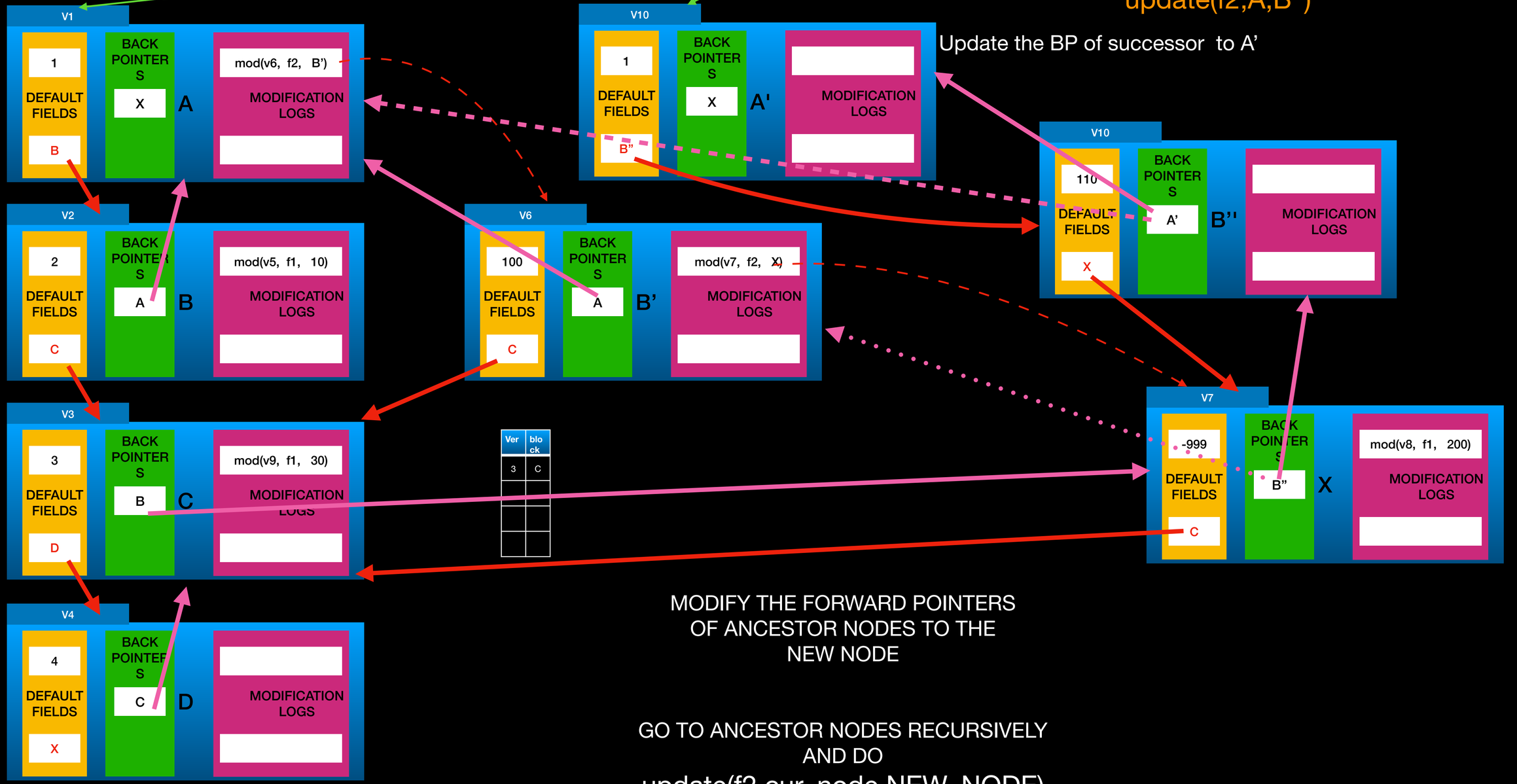
START MODULE

V0 V10

# STEP 3
# SUBSTEP 2
update(f2,A,B")

update(f1, B,110)

Update the BP of successor to A'



Ver	block
3	C

VERSION REDIRECTOR MODULE

Ver	block
1	A
10	A"

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B'
10	B"

VERSION REDIRECTOR MODULE

Ver	block
7	X

VERSION REDIRECTOR MODULE

Ver	block
4	D

MODIFY THE FORWARD POINTERS OF ANCESTOR NODES TO THE NEW NODE

GO TO ANCESTOR NODES RECURSIVELY AND DO

update(f2,cur_node,NEW_NODE)

Current Version: v10

START MODULE

V0 V10

STEP 3

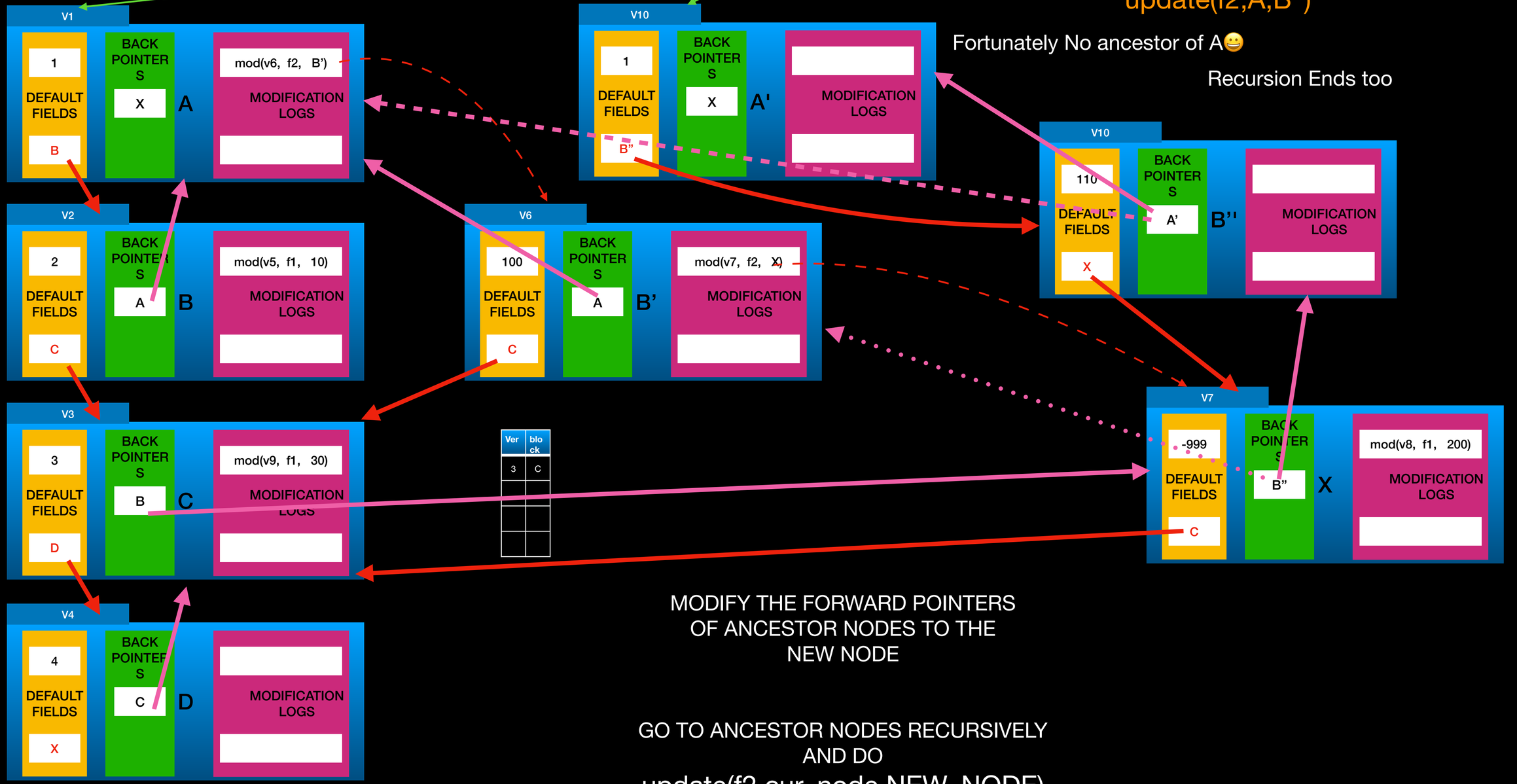
SUBSTEP 3

update(f2,A,B")

update(f1, B,110)

Fortunately No ancestor of A😀

Recursion Ends too



Ver	block
3	C

VERSION REDIRECTOR MODULE

Ver	block
1	A
10	A"

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B'
10	B"

VERSION REDIRECTOR MODULE

Ver	block
7	X

VERSION REDIRECTOR MODULE

Ver	block
4	D

MODIFY THE FORWARD POINTERS OF ANCESTOR NODES TO THE NEW NODE

GO TO ANCESTOR NODES RECURSIVELY AND DO

update(f2,cur_node,NEW_NODE)

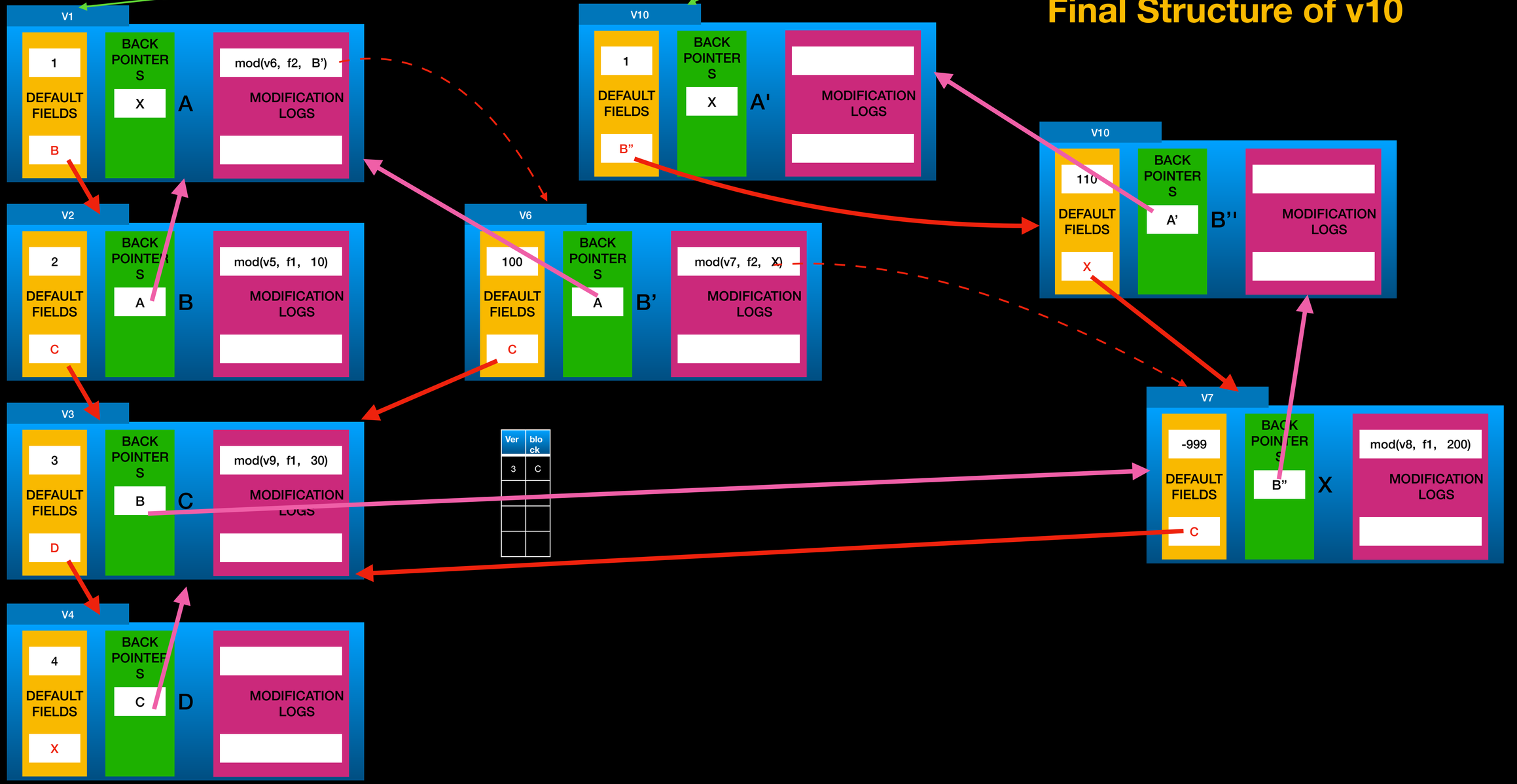
Current Version: v10

update(f1, B,110)

START MODULE

V0 V10

Final Structure of v10



Ver	block
3	C

VERSION REDIRECTOR MODULE

Ver	block
1	A
10	A"

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B'
10	B"

VERSION REDIRECTOR MODULE

Ver	block
7	X

VERSION REDIRECTOR MODULE

Ver	block
4	D

Current Version: v11

START MODULE

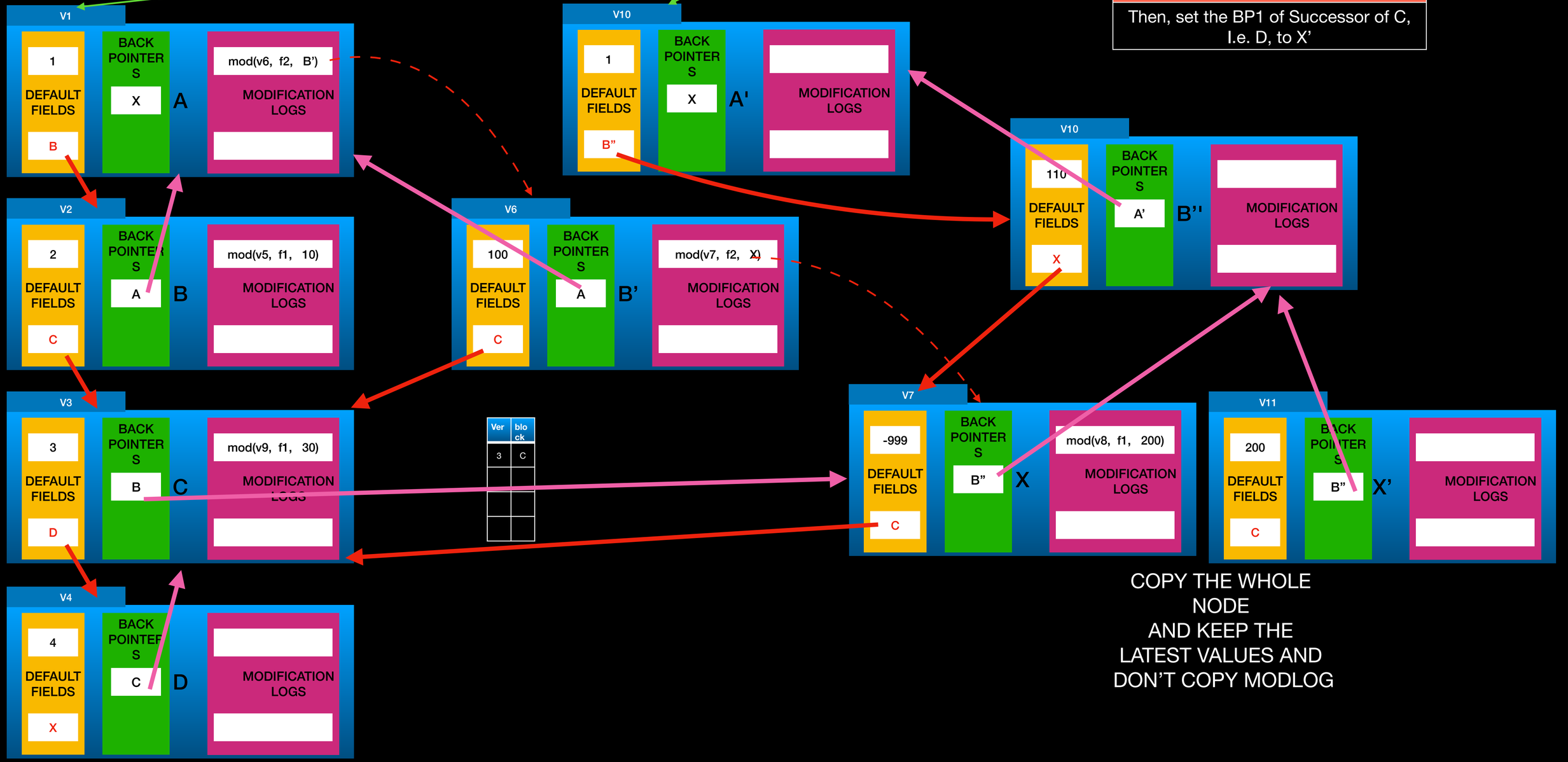
V0 V10

remove(C)

Remove C consists of

Modify F2 of Parent of C (i.e., X) -> F2 of C (successor of C)

Then, set the BP1 of Successor of C, I.e. D, to X’



Ver	block
3	C

VERSION REDIRECTOR MODULE

Ver	block
1	A
10	A”

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B’
10	B”

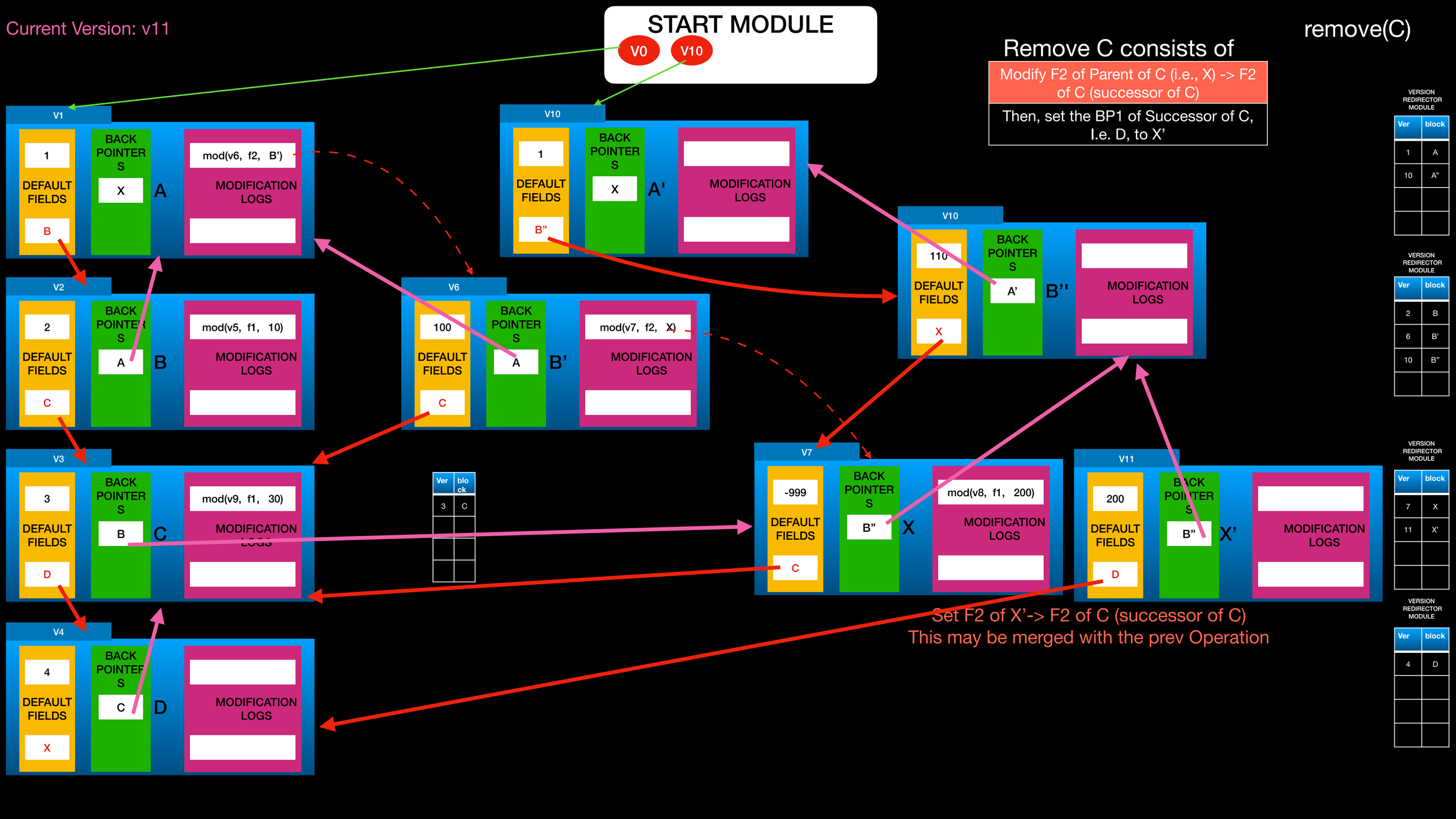
VERSION REDIRECTOR MODULE

Ver	block
7	X
11	X’

VERSION REDIRECTOR MODULE

Ver	block
4	D

COPY THE WHOLE NODE AND KEEP THE LATEST VALUES AND DON’T COPY MODLOG


Current Version: v11
START MODULE
V0
V10
remove(C)
Remove C consists of
Modify F2 of Parent of C (i.e., X) -> F2 of C (successor of C)
Then, set the BP1 of Successor of C, I.e. D, to X'
V1
1
DEFAULT FIELDS
B
BACK POINTERS
X
A
mod(v6, f2, B')
MODIFICATION LOGS
V10
1
DEFAULT FIELDS
B"
BACK POINTERS
X
A'
MODIFICATION LOGS
V10
110
DEFAULT FIELDS
X
BACK POINTERS
A'
B''
MODIFICATION LOGS
V2
2
DEFAULT FIELDS
C
BACK POINTERS
A
B
mod(v5, f1, 10)
MODIFICATION LOGS
V6
100
DEFAULT FIELDS
C
BACK POINTERS
A
B'
mod(v7, f2, X)
MODIFICATION LOGS
V3
3
DEFAULT FIELDS
D
BACK POINTERS
B
C
mod(v9, f1, 30)
MODIFICATION LOGS
Ver
block
3
C
V7
-999
DEFAULT FIELDS
C
BACK POINTERS
B"
X
mod(v8, f1, 200)
MODIFICATION LOGS
V11
200
DEFAULT FIELDS
D
BACK POINTERS
B"
X'
MODIFICATION LOGS
V4
4
DEFAULT FIELDS
X
BACK POINTERS
C
D
MODIFICATION LOGS
Set F2 of X'-> F2 of C (successor of C)
This may be merged with the prev Operation
VERSION REDIRECTOR MODULE
Ver
block
1
A
10
A"
VERSION REDIRECTOR MODULE
Ver
block
2
B
6
B'
10
B"
VERSION REDIRECTOR MODULE
Ver
block
7
X
11
X'
VERSION REDIRECTOR MODULE
Ver
block
4
D

Current Version: v11

START MODULE

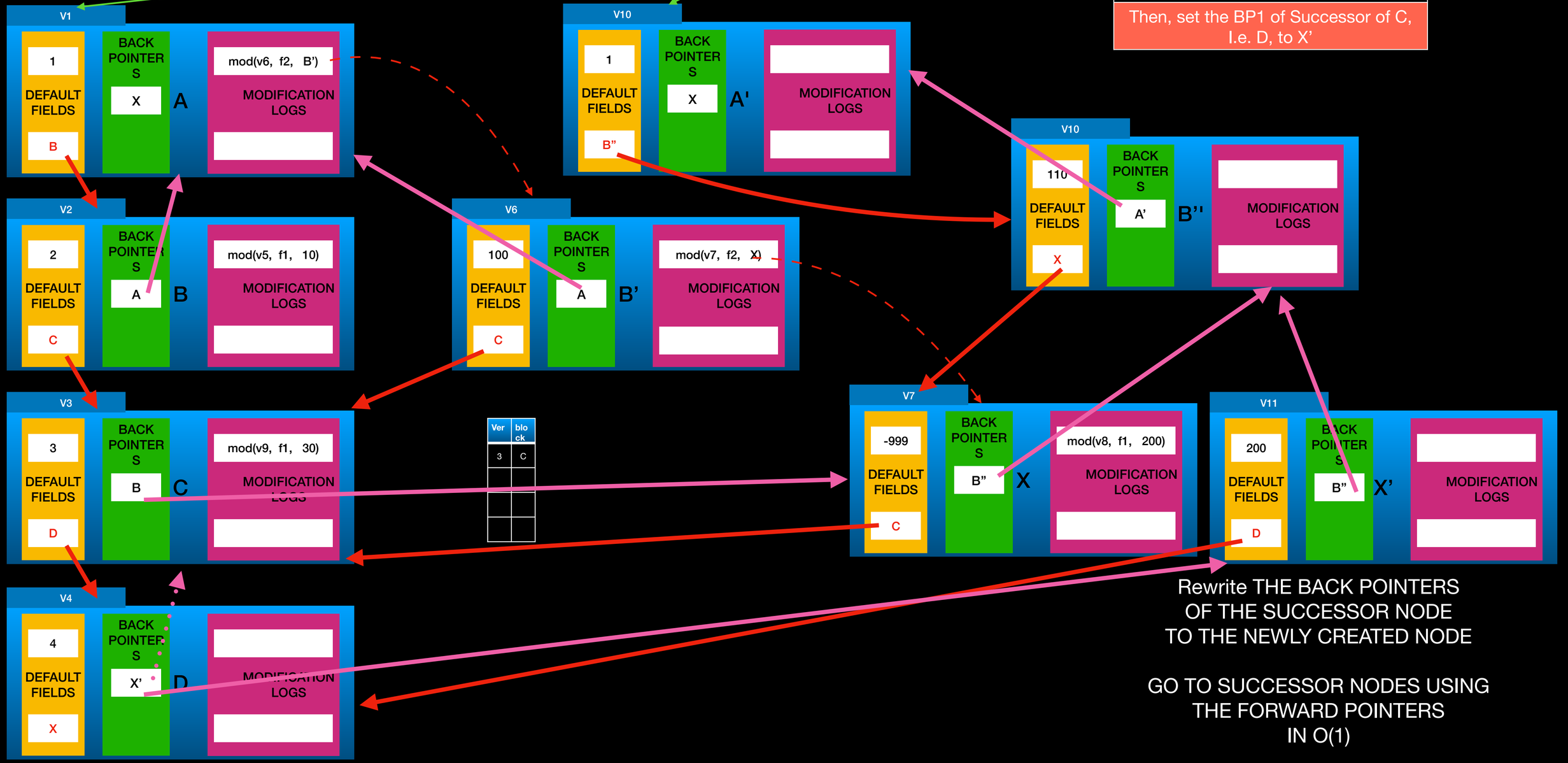
V0 V10

remove(C)

Remove C consists of

Modify F2 of Parent of C (i.e., X) -> F2 of C (successor of C)

Then, set the BP1 of Successor of C, I.e. D, to X'



Ver	block
3	C

VERSION REDIRECTOR MODULE

Ver	block
1	A
10	A"

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B'
10	B"

VERSION REDIRECTOR MODULE

Ver	block
7	X
11	X'

VERSION REDIRECTOR MODULE

Ver	block
4	D

Rewrite THE BACK POINTERS OF THE SUCCESSOR NODE TO THE NEWLY CREATED NODE

GO TO SUCCESSOR NODES USING THE FORWARD POINTERS IN O(1)

Current Version: v11

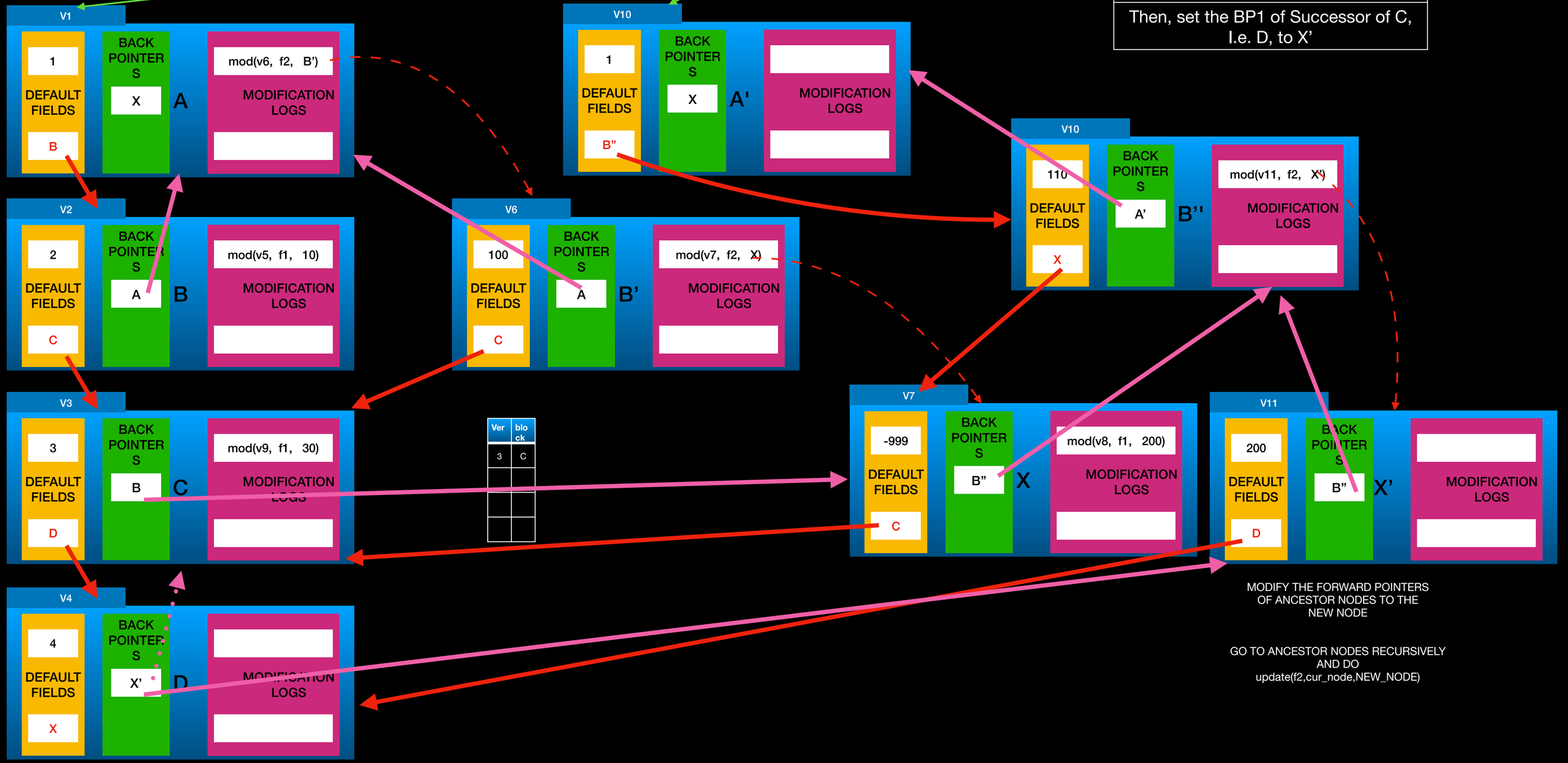
START MODULE

V0 V10

remove(C)

Remove C consists of

Modify F2 of Parent of C (i.e., X) -> F2 of C (successor of C)
Then, set the BP1 of Successor of C, I.e. D, to X’



Ver	block
3	C

MODIFY THE FORWARD POINTERS OF ANCESTOR NODES TO THE NEW NODE

GO TO ANCESTOR NODES RECURSIVELY AND DO update(f2,cur_node,NEW_NODE)

VERSION REDIRECTOR MODULE

Ver	block
1	A
10	A”

VERSION REDIRECTOR MODULE

Ver	block
2	B
6	B’
10	B”

VERSION REDIRECTOR MODULE

Ver	block
7	X
11	X’

VERSION REDIRECTOR MODULE

Ver	block
4	D

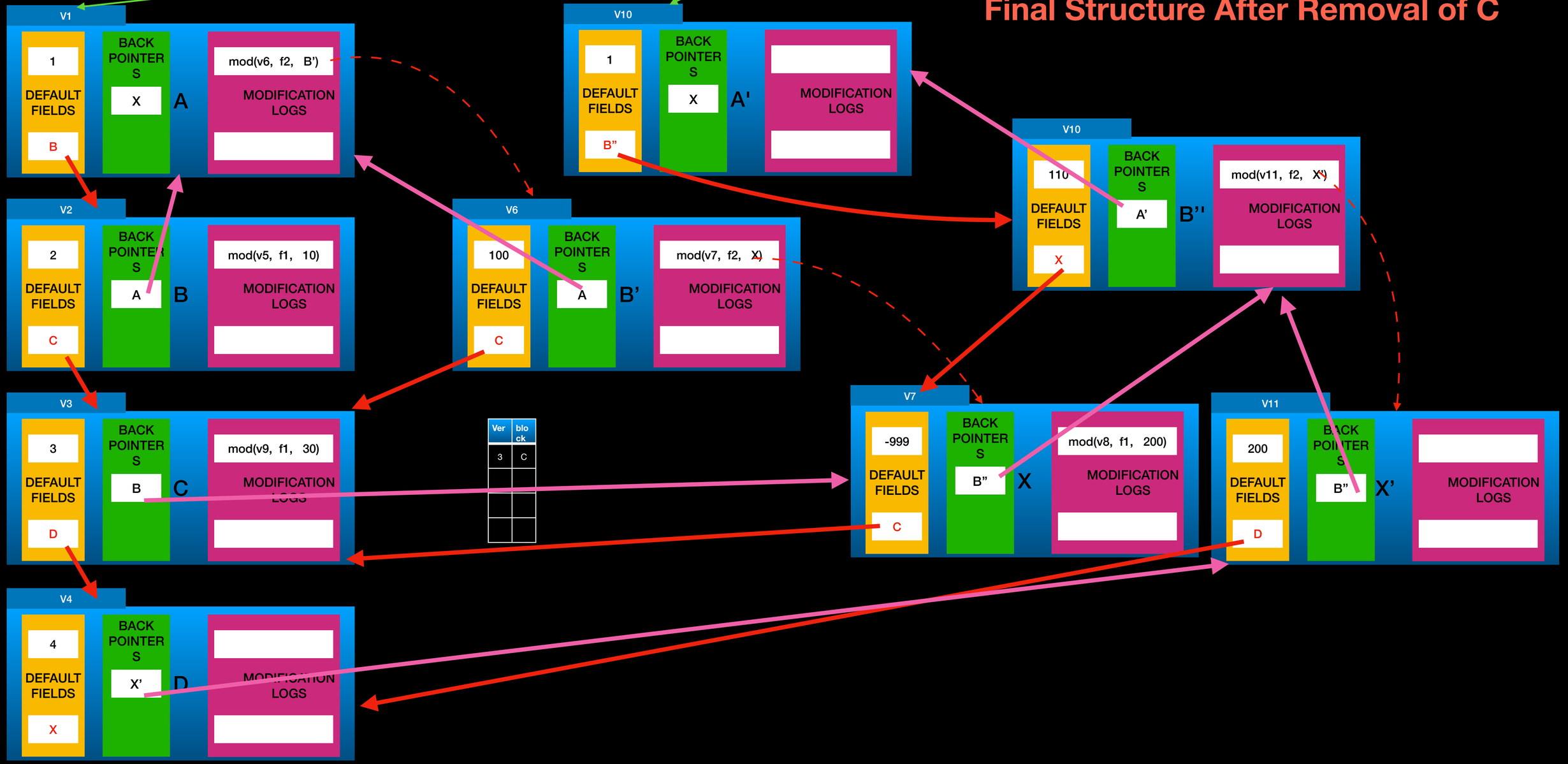
Current Version: v11

remove(C)

START MODULE

V0 V10

Final Structure After Removal of C



VERSION REDIRECTOR MODULE

Ver	block
1	A
10	A"

VERSION REDIRECTOR MODULE

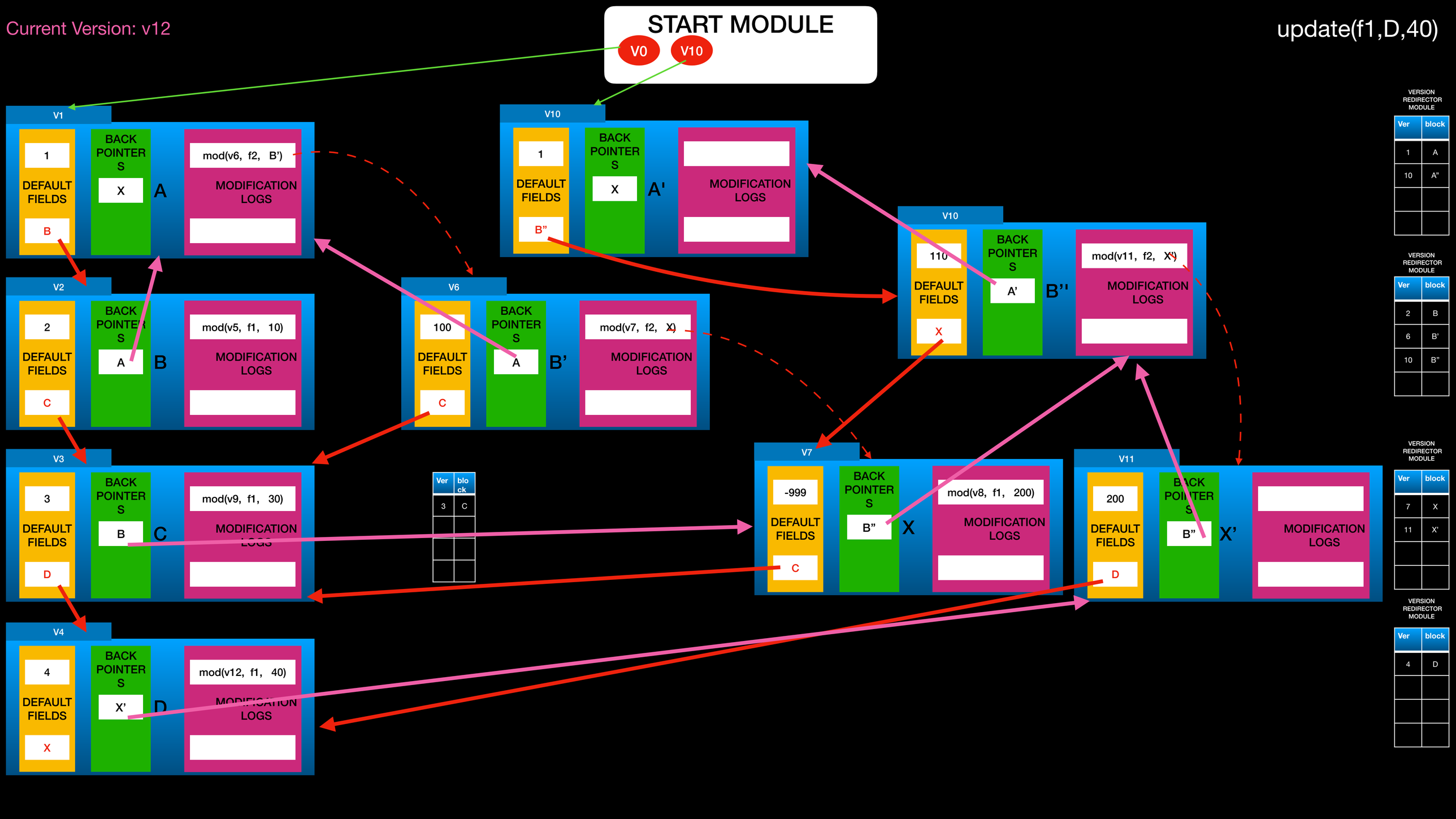
Ver	block
2	B
6	B'
10	B"

VERSION REDIRECTOR MODULE

Ver	block
7	X
11	X'

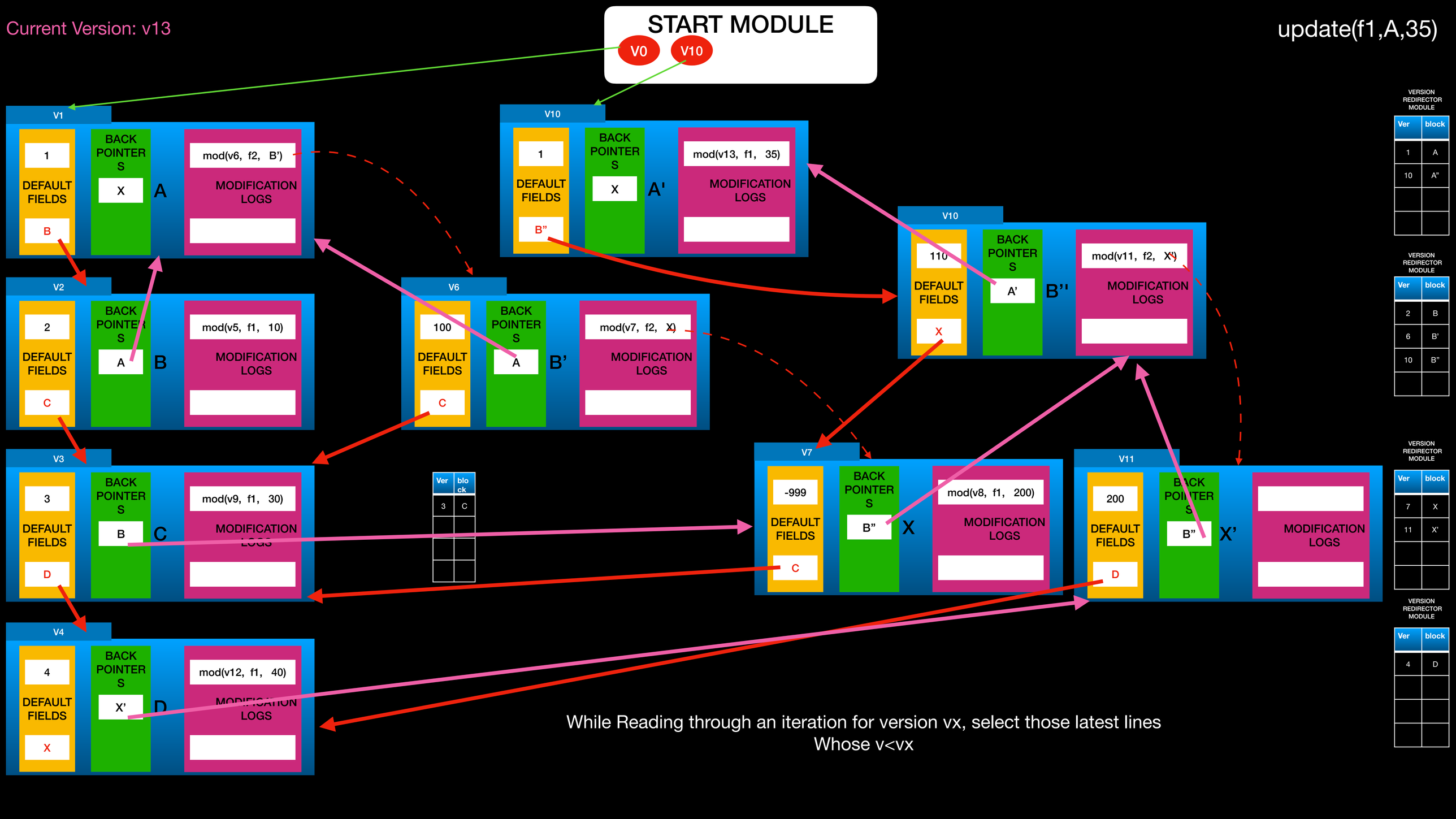
VERSION REDIRECTOR MODULE

Ver	block
4	D


Current Version: v12
START MODULE
V0
V10
update(f1,D,40)
V1
1
DEFAULT FIELDS
B
BACK POINTERS
X
A
mod(v6, f2, B')
MODIFICATION LOGS
V10
1
DEFAULT FIELDS
B"
BACK POINTERS
X
A'
MODIFICATION LOGS
V10
110
DEFAULT FIELDS
X
BACK POINTERS
A'
B''
mod(v11, f2, X')
MODIFICATION LOGS
V2
2
DEFAULT FIELDS
C
BACK POINTERS
A
B
mod(v5, f1, 10)
MODIFICATION LOGS
V6
100
DEFAULT FIELDS
C
BACK POINTERS
A
B'
mod(v7, f2, X)
MODIFICATION LOGS
V3
3
DEFAULT FIELDS
D
BACK POINTERS
B
C
mod(v9, f1, 30)
MODIFICATION LOGS
Ver
block
3
C
V7
-999
DEFAULT FIELDS
C
BACK POINTERS
B"
X
mod(v8, f1, 200)
MODIFICATION LOGS
V11
200
DEFAULT FIELDS
D
BACK POINTERS
B"
X'
MODIFICATION LOGS
V4
4
DEFAULT FIELDS
X
BACK POINTERS
X'
D
mod(v12, f1, 40)
MODIFICATION LOGS
VERSION REDIRECTOR MODULE
Ver
block
1
A
10
A"
VERSION REDIRECTOR MODULE
Ver
block
2
B
6
B'
10
B"
VERSION REDIRECTOR MODULE
Ver
block
7
X
11
X'
VERSION REDIRECTOR MODULE
Ver
block
4
D

Current Version: v13

update(f1,A,35)



While Reading through an iteration for version vx, select those latest lines
Whose v<vx

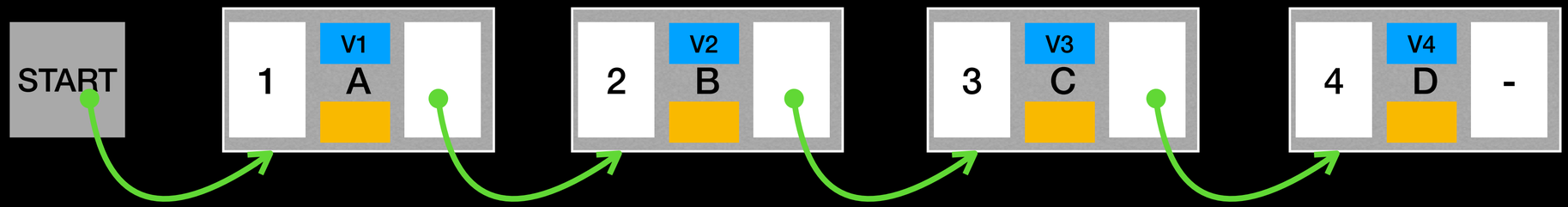
iterate_LL_at_v(vx)

To print the list at v_x

- Start from start module
- Choose those lines whose version **IS JUST** LESS THAN OR EQUAL TO v_x (as, suppose if we are traversing for v5 , either v0,v1,v2,v3, v4 or v5 can be on that path, lines>v5 can’t be on that).
- The word “JUST LESS” is written because if a NODE has two version lines in that, e.g. v2 and v4 and we are searching for v5, then we should prefer v4 line over v2.

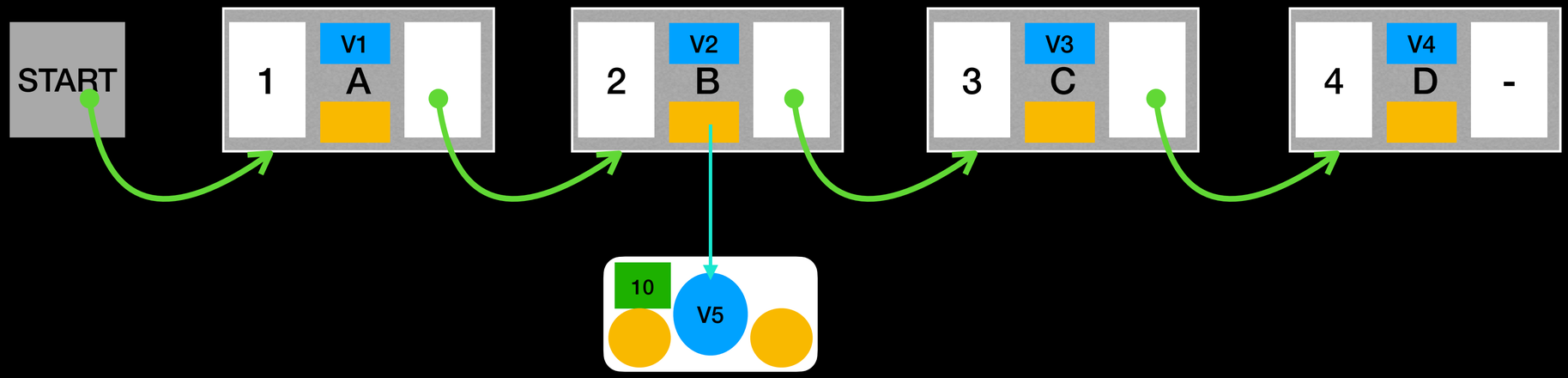
Why such approach is better ? 🥲

Normal Implementation (using balanced BST [may use just LL])



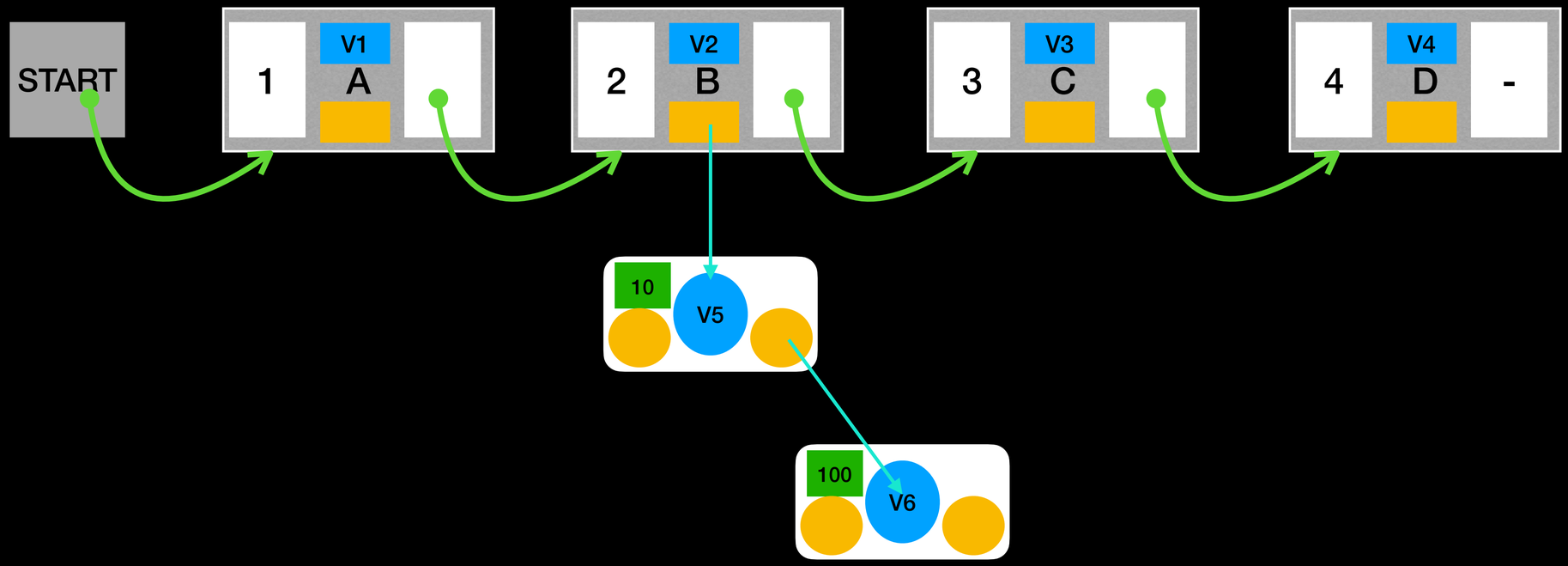
Why such approach is better ? 🥲

Normal Implementation (using balanced BST)



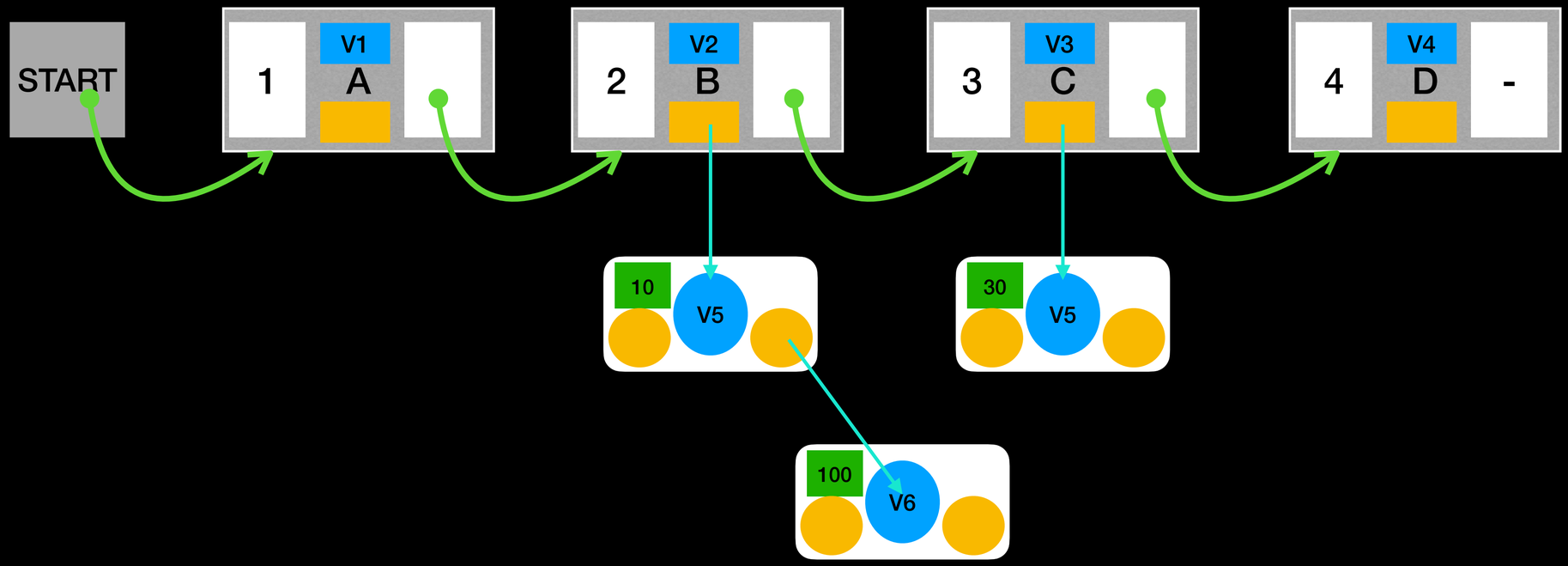
Why such approach is better ? 🥲

Normal Implementation (using balanced BST)



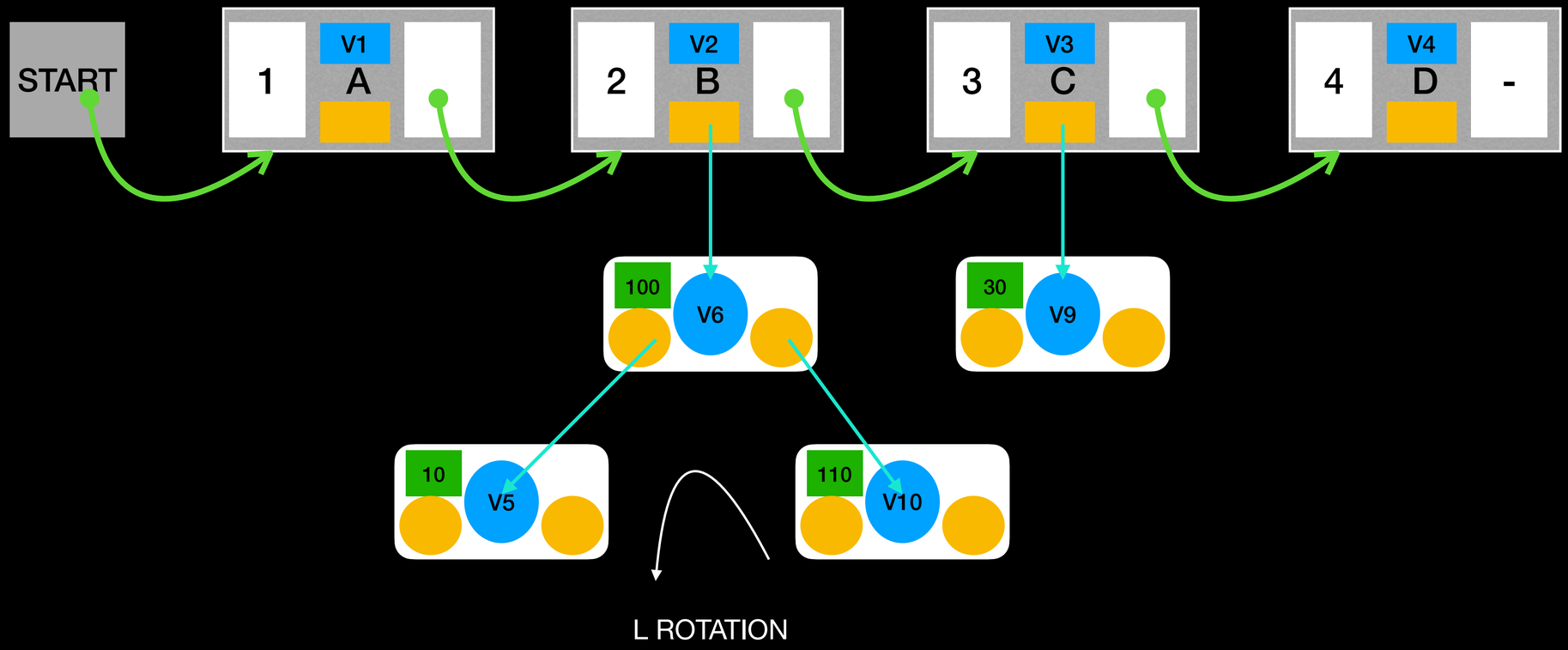
Why such approach is better ? 🥲

Normal Implementation (using balanced BST)



Why such approach is better ? 🥲

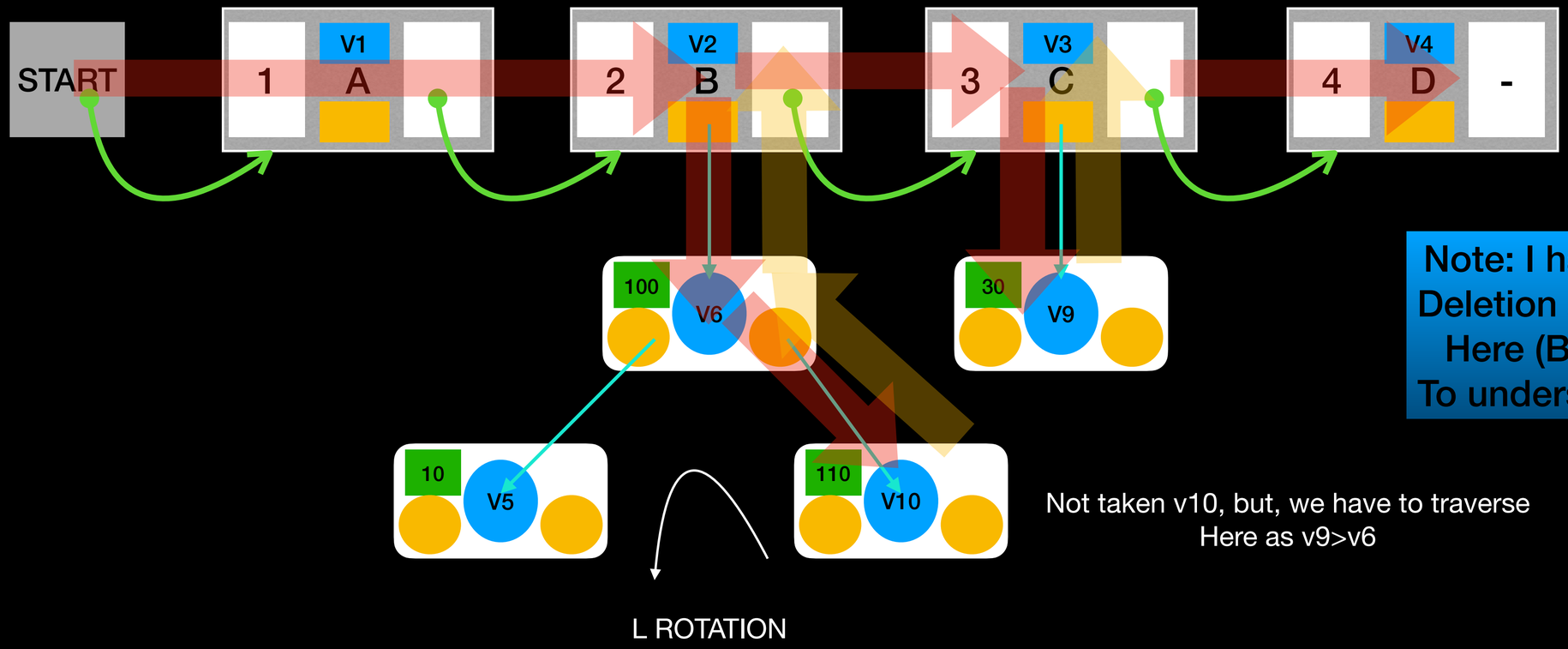
Normal Implementation (using balanced BST)



Why such approach is better ? 🥲

Normal Implementation (using balanced BST) (Reading v9)

Output: 1 -> 100 -> 30 -> 4



Not taken v10, but, we have to traverse
Here as v9>v6

Note: I have not considered the Deletion and Insertion operation Here (But, this is not relevant To understand the performance)

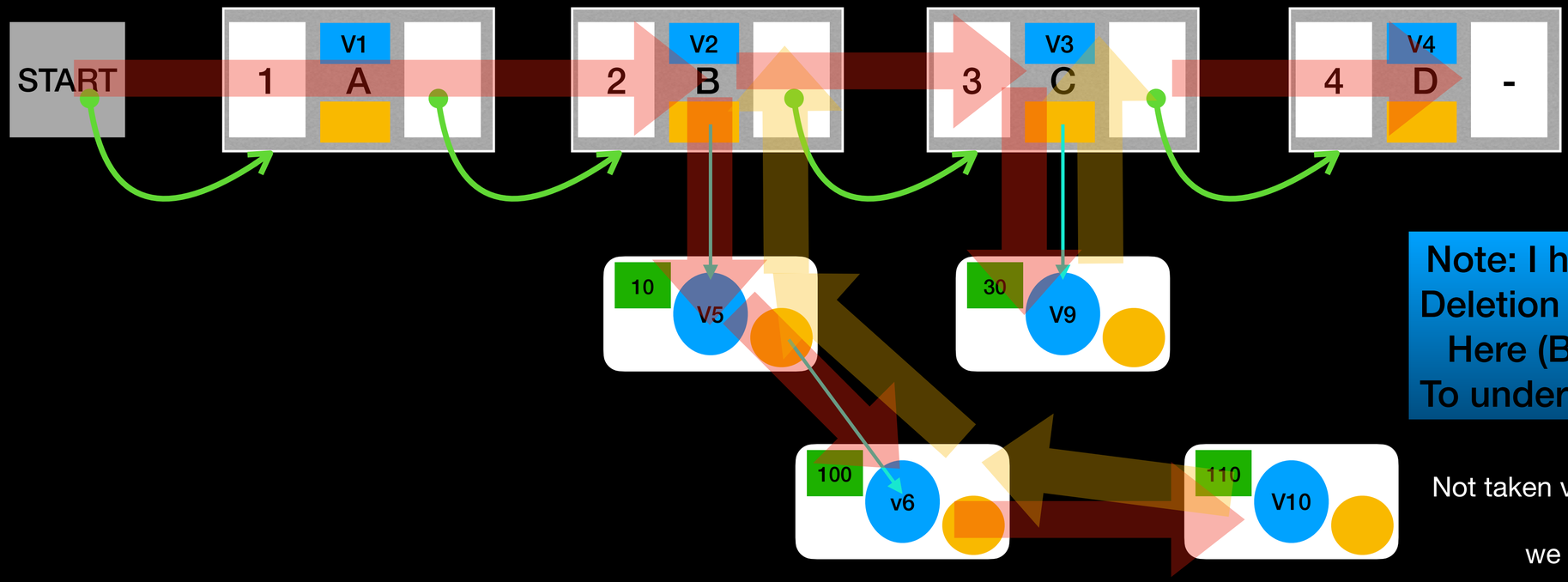
So, the time complexity of reading list v9 … n * O(log m)
n is length of LL and m is is #mods per node

Also the creation of the AVL of version to node Takes O(log m) each time

Why such approach is better ? 🥲

Normal Implementation (Using List) (Reading v9)

Output: 1 -> 100 -> 30 -> 4



Note: I have not considered the Deletion and Insertion operation Here (But, this is not relevant To understand the performance)

Not taken v10, but, we have to traverse Here as v9>v6, we take v6 from this line

So, the time complexity of reading list v9 … n * O(m)
n is length of LL and m is is #mods per node

The creation of the List of version to node Takes O(1) each time

Why such approach is better ? 🥲

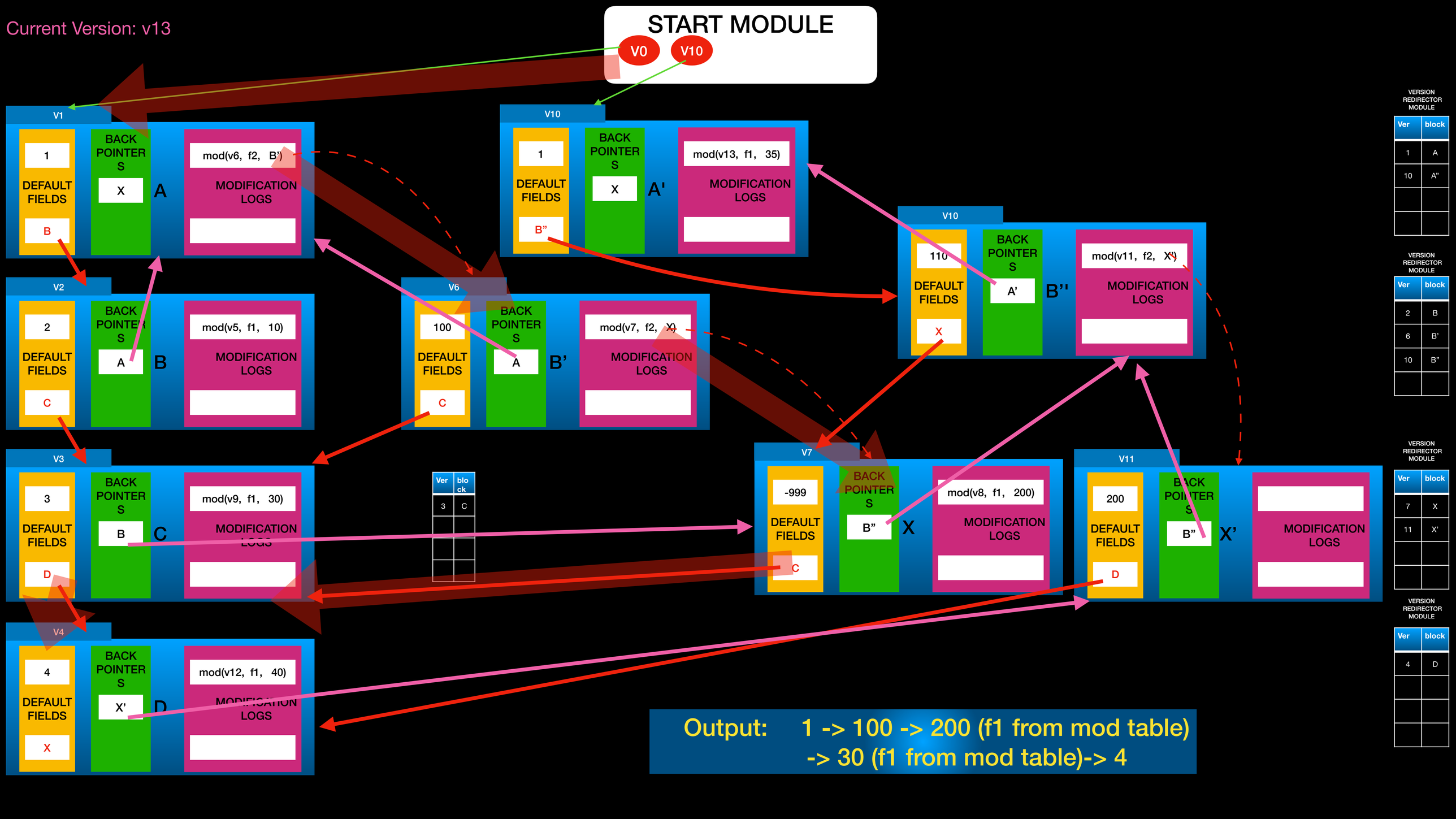
Pointer Machine (Reading v9)

We are taking O(2k.n)->O(n) time for iterating over the Linked List
Here k is the in-degree (Here 1) of the DS

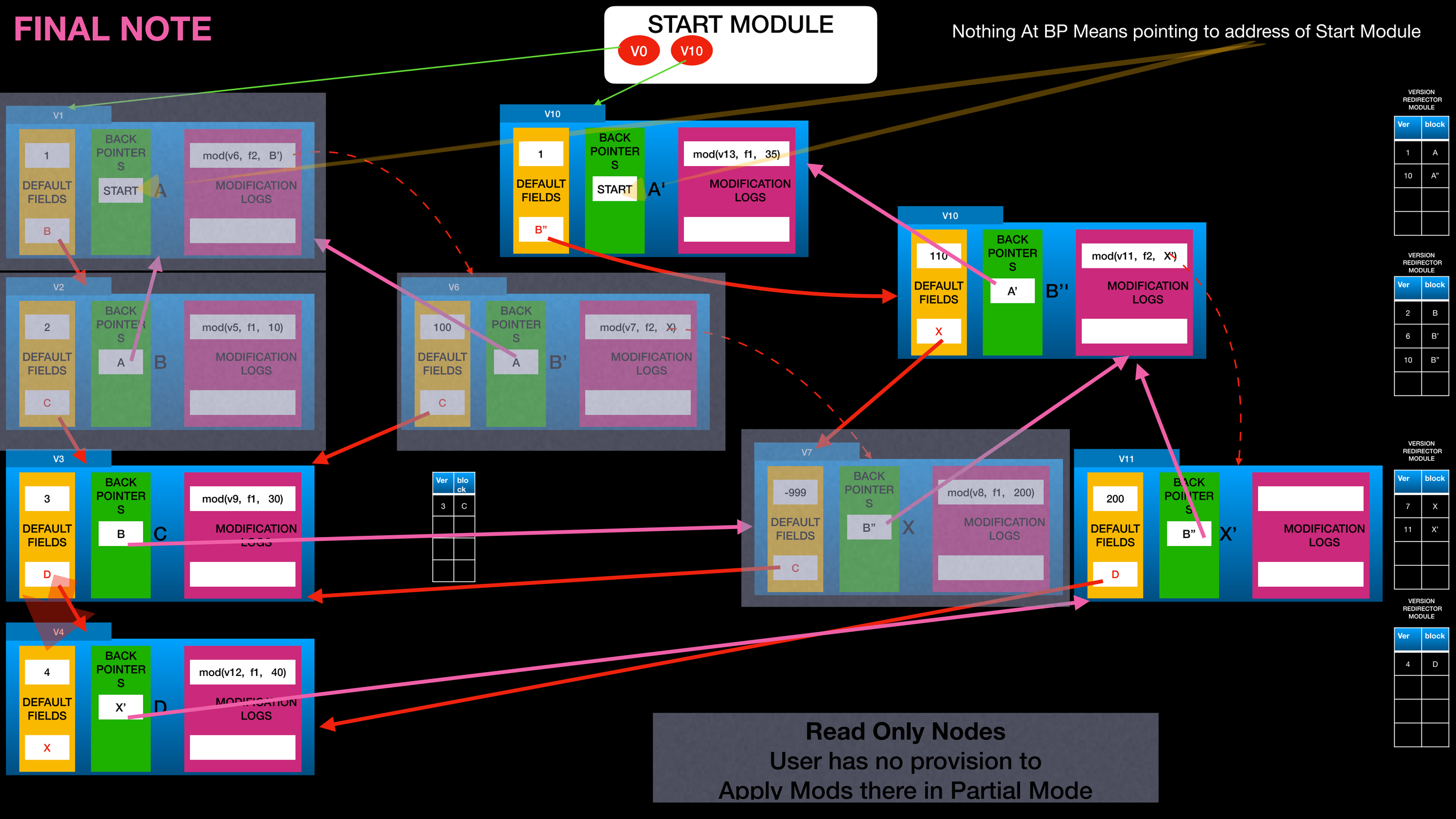
We are taking O(2) **amortised time** for modifying/adding each node to the Linked List
Here k is the in-degree (Here 1) of the DS

Simulation is shown at next page …

Current Version: v13



FINAL NOTE


START MODULE
V0
V10
Nothing At BP Means pointing to address of Start Module
V1
1
DEFAULT FIELDS
B
BACK POINTERS
START
A
mod(v6, f2, B')
MODIFICATION LOGS
V10
1
DEFAULT FIELDS
B"
BACK POINTERS
START
A'
mod(v13, f1, 35)
MODIFICATION LOGS
V10
110
DEFAULT FIELDS
X
BACK POINTERS
A'
B''
mod(v11, f2, X')
MODIFICATION LOGS
V2
2
DEFAULT FIELDS
C
BACK POINTERS
A
B
mod(v5, f1, 10)
MODIFICATION LOGS
V6
100
DEFAULT FIELDS
C
BACK POINTERS
A
B'
mod(v7, f2, X)
MODIFICATION LOGS
V3
3
DEFAULT FIELDS
D
BACK POINTERS
B
C
mod(v9, f1, 30)
MODIFICATION LOGS
V7
-999
DEFAULT FIELDS
C
BACK POINTERS
B"
X
mod(v8, f1, 200)
MODIFICATION LOGS
V11
200
DEFAULT FIELDS
D
BACK POINTERS
B"
X'
MODIFICATION LOGS
V4
4
DEFAULT FIELDS
X
BACK POINTERS
X'
D
mod(v12, f1, 40)
MODIFICATION LOGS
Ver
block
3
C
VERSION REDIRECTOR MODULE
Ver
block
1
A
10
A"
VERSION REDIRECTOR MODULE
Ver
block
2
B
6
B'
10
B"
VERSION REDIRECTOR MODULE
Ver
block
7
X
11
X'
VERSION REDIRECTOR MODULE
Ver
block
4
D
Read Only Nodes
User has no provision to
Apply Mods there in Partial Mode