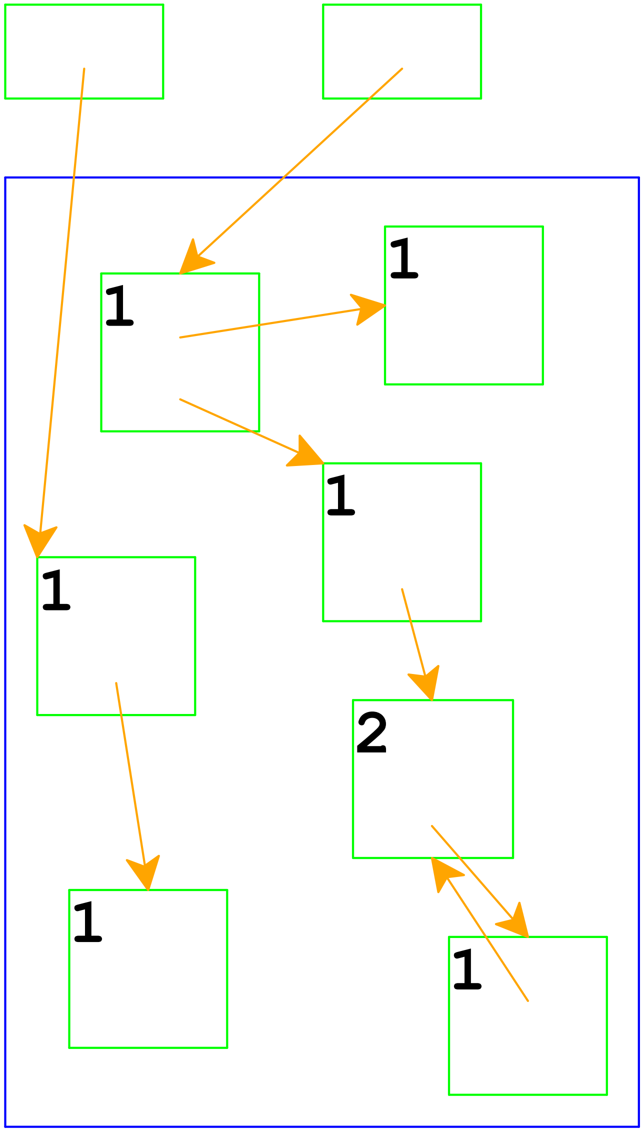
Reference Counting

Reference counting: a way to know whether a record has other users

- Attatch a count to every record, starting at 0
- When installing a pointer to a record, increment the record's count
- When replacing a pointer to a record, decrement the record's count
- When a count is decremented to 0, decrement counts for other records referenced by the record, then free the record

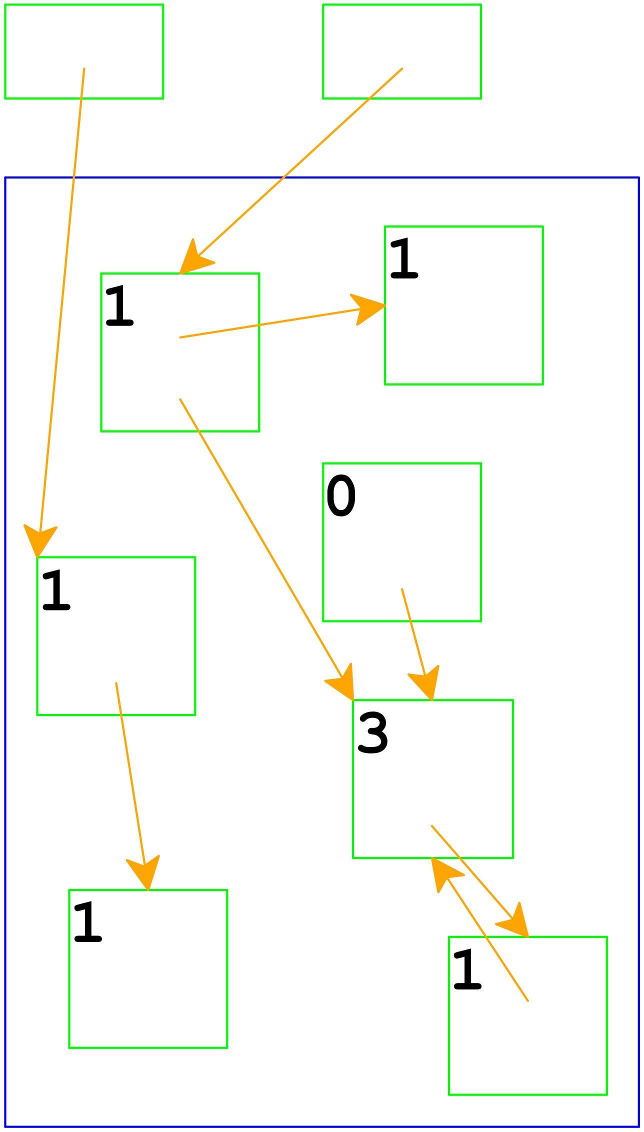
Reference Counting



Top boxes are the ***roots***, such as registers

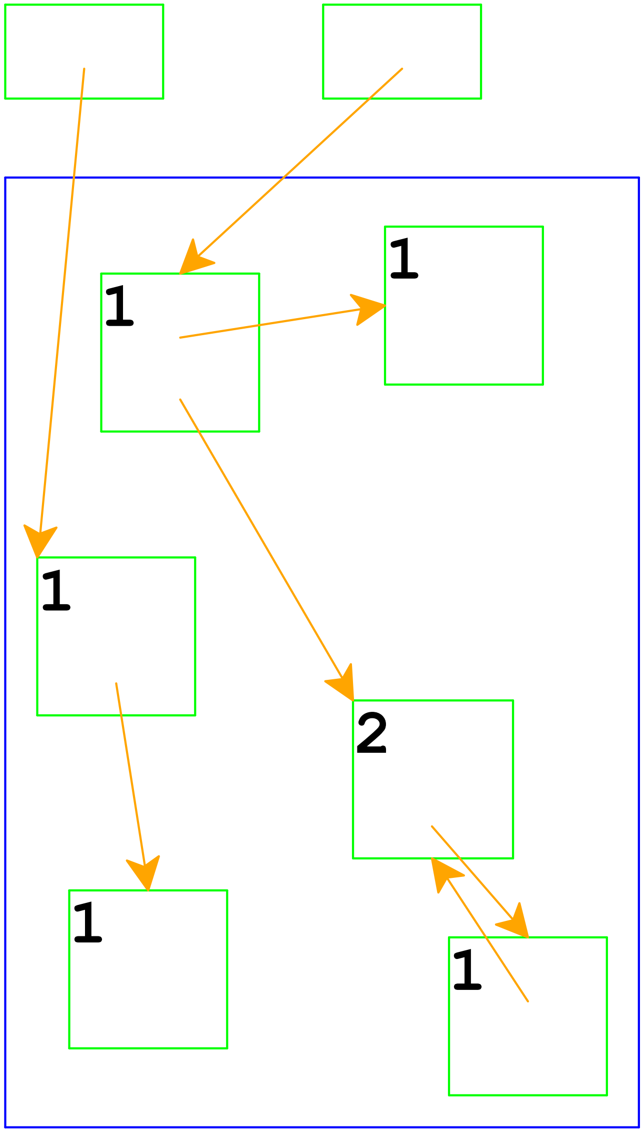
Boxes in the blue area are allocated

Reference Counting



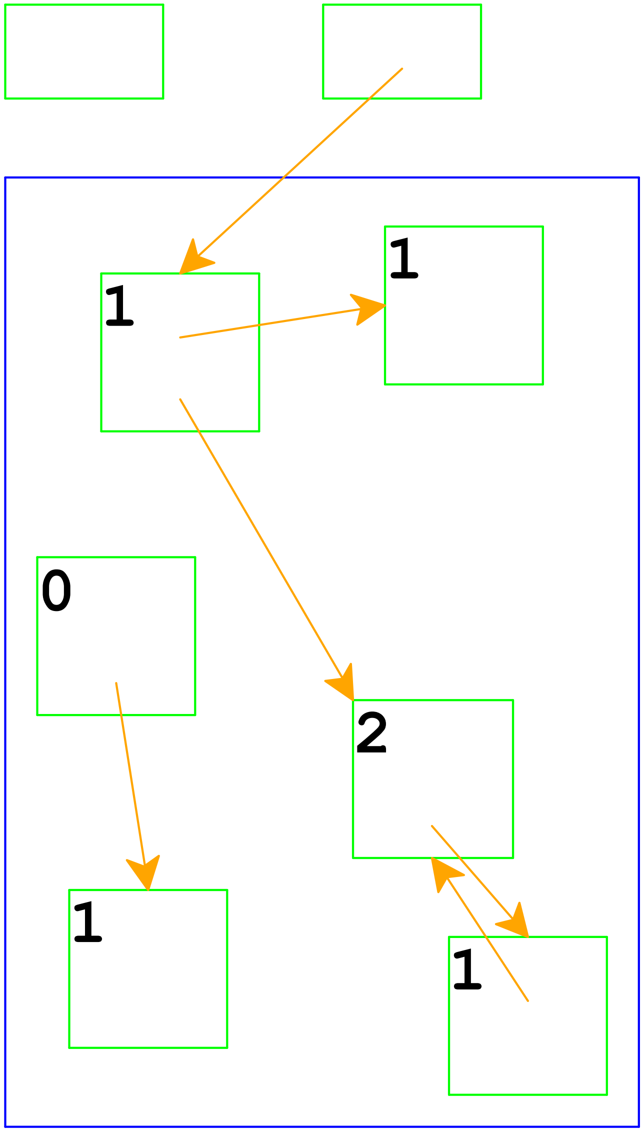
Adjust counts when a pointer is changed...

Reference Counting



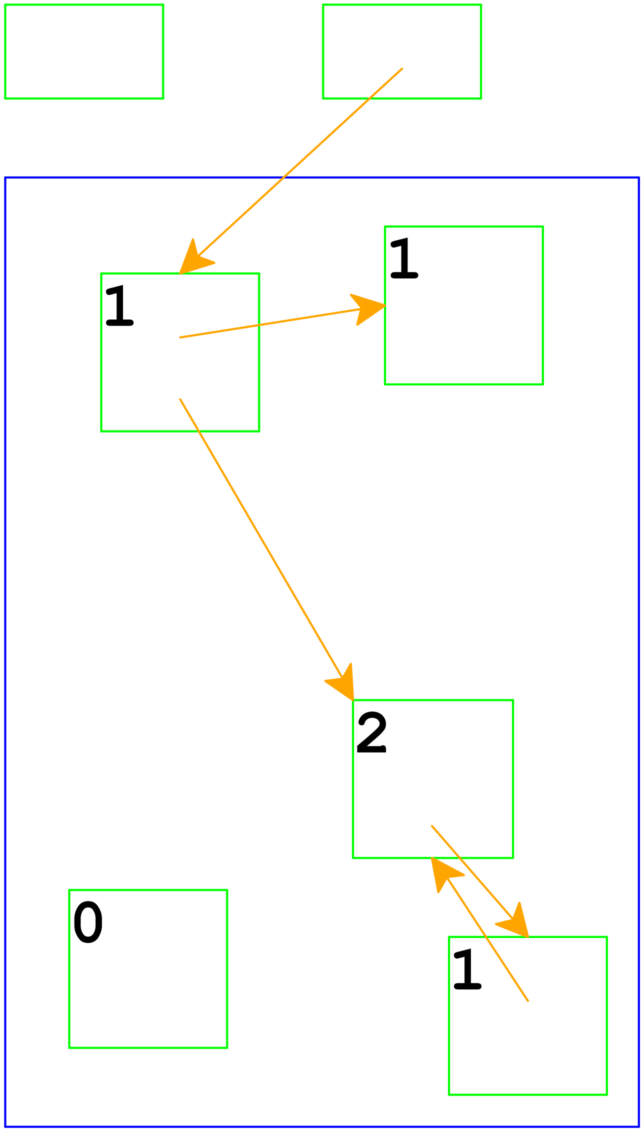
... freeing a record if its count goes to 0

Reference Counting



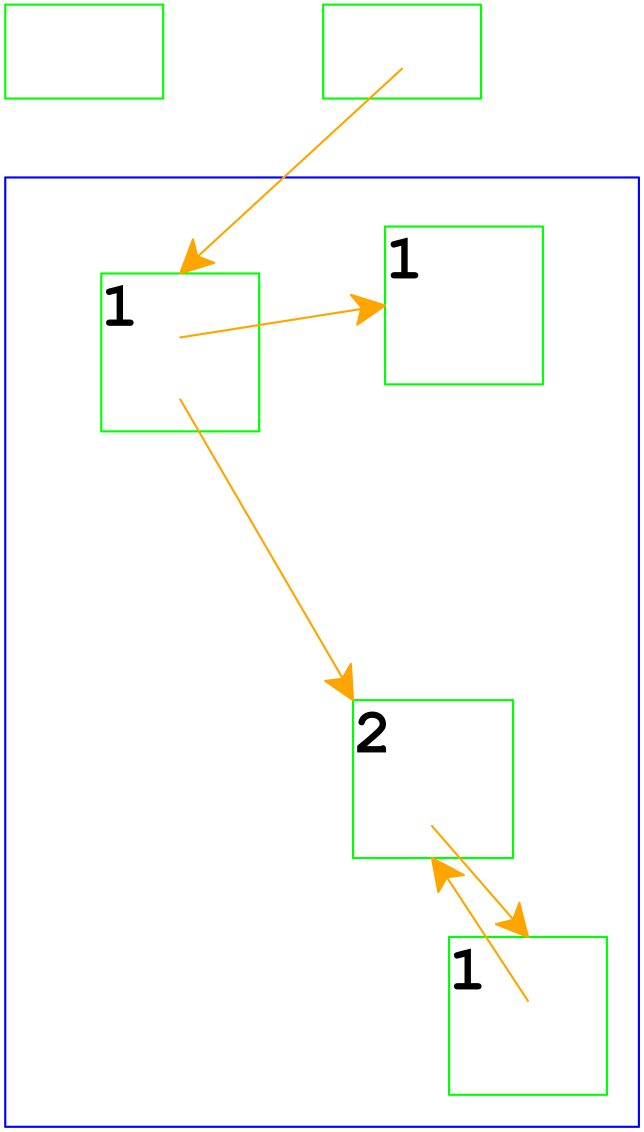
Same if the pointer is in a register

Reference Counting



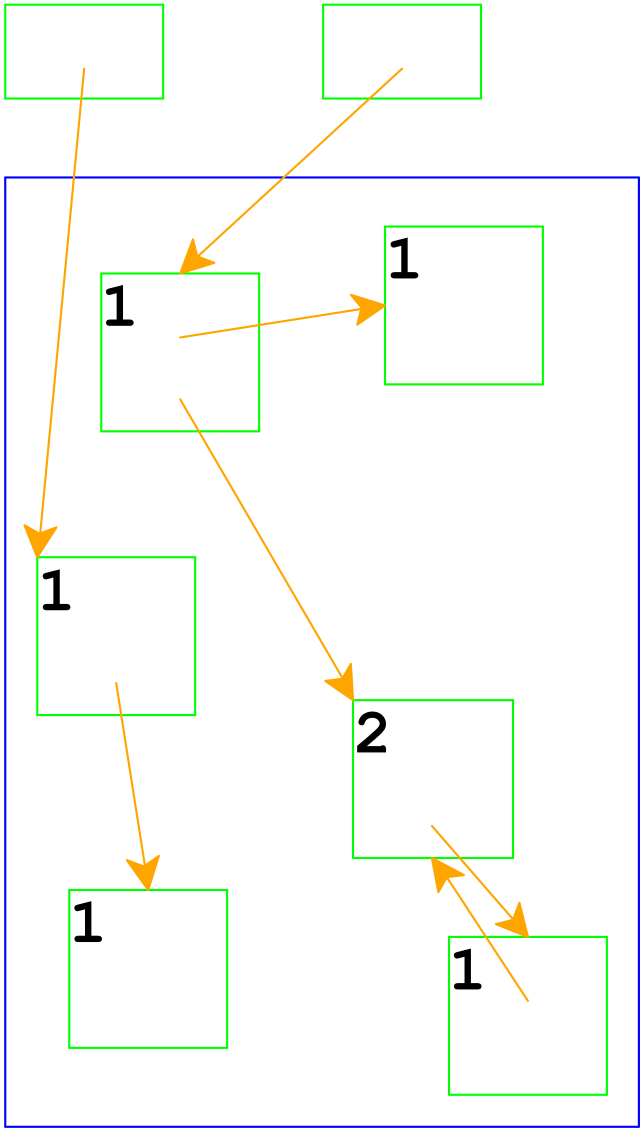
Adjust counts after frees, too...

Reference Counting



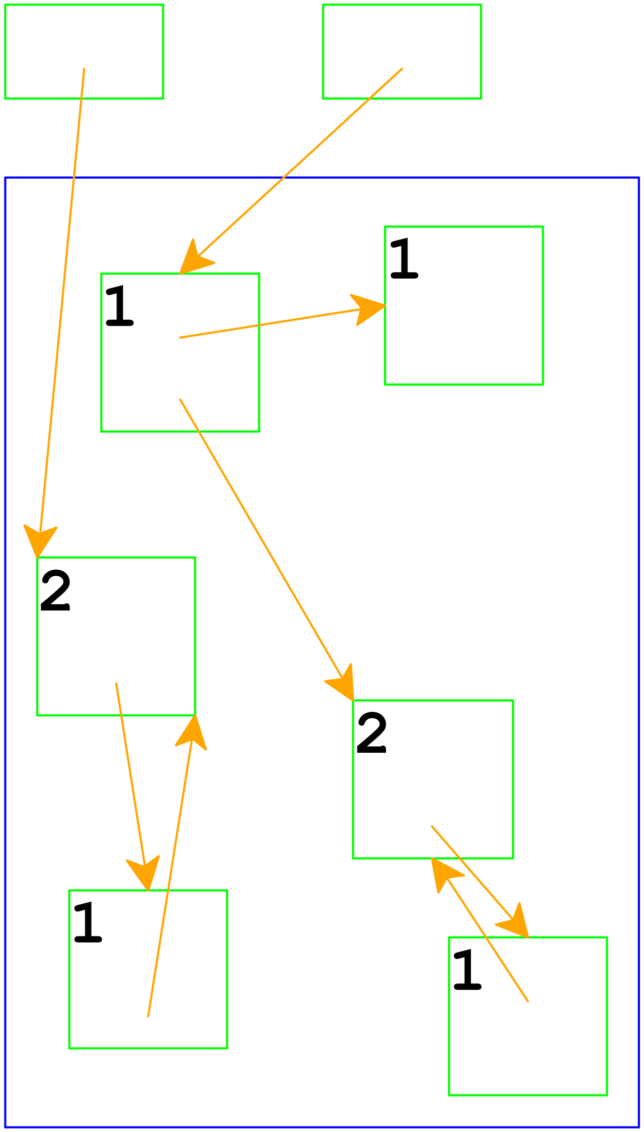
... which can trigger more frees

Reference Counting And Cycles



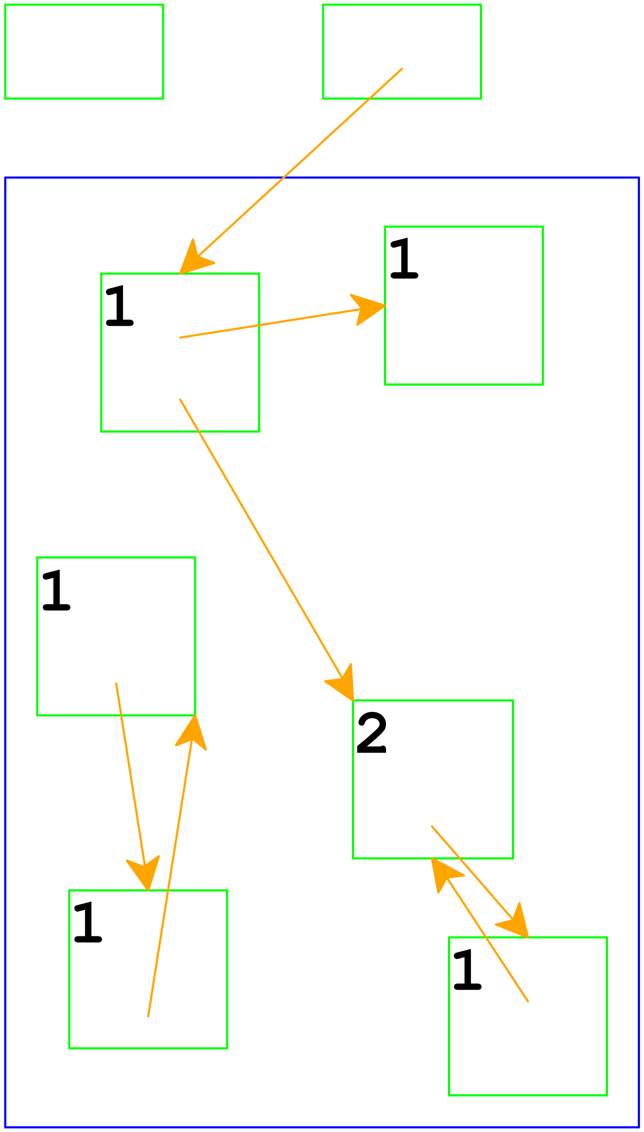
An assignment can create a cycle...

Reference Counting And Cycles



Adding a reference increments a count

Reference Counting And Cycles



Lower-left records are inaccessible, but not deallocated

In general, cycles break reference counting

Garbage Collection

Garbage collection: a way to know whether a record is *accessible*

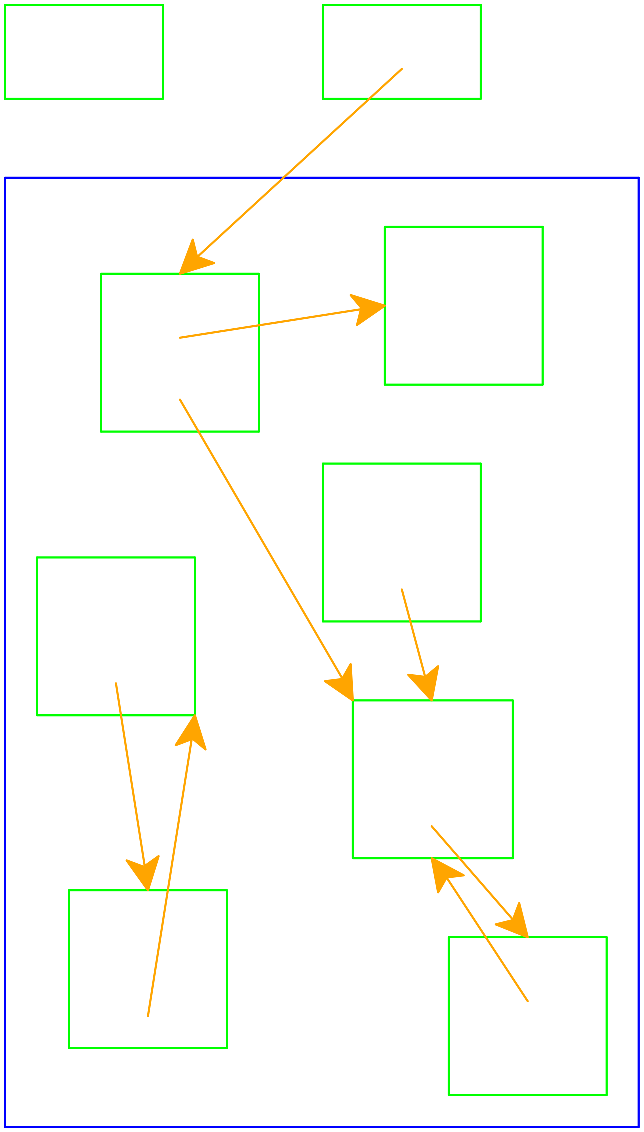
- A record referenced by a register is ***live***
- A record referenced by a live record is also live
- Works when a program only possibly uses live records (i.e., doesn't synthesize pointers from numbers)

- A garbage collector frees all records that are not live
- Allocate until we run out of memory, then run a garbage collector to get more space

Garbage Collection Algorithm

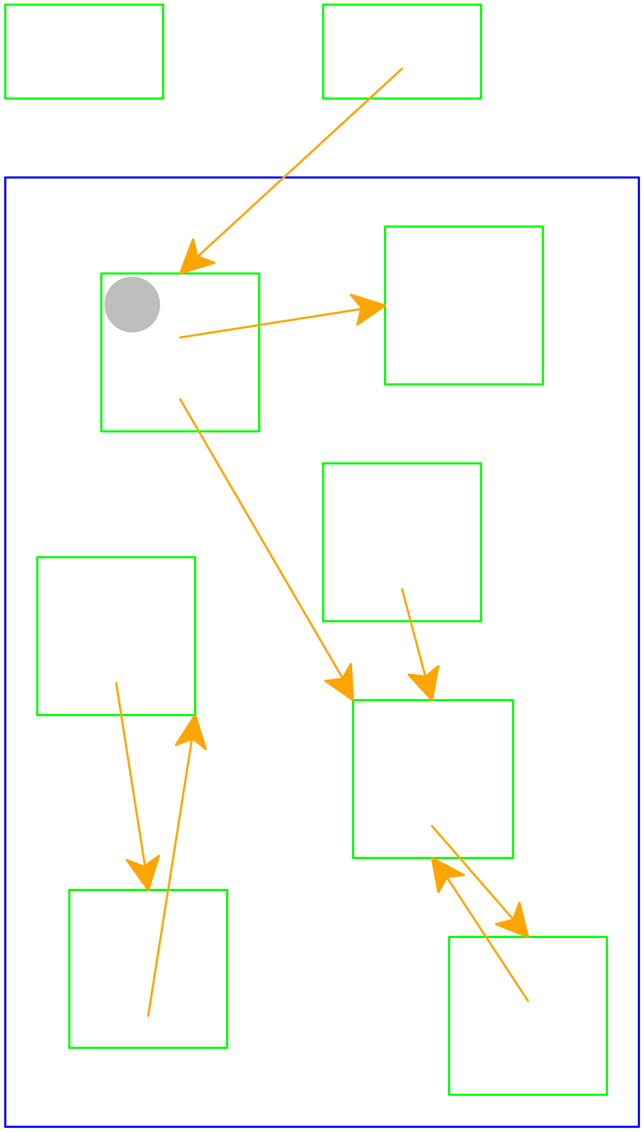
- Color all records ***white***
- Color records referenced by registers ***gray***
- Repeat until there are no gray records:
    - Pick a gray record, *r*
    - For each white record that *r* points to, make it gray
    - Color *r* ***black***
- Deallocate all white records

Garbage Collection



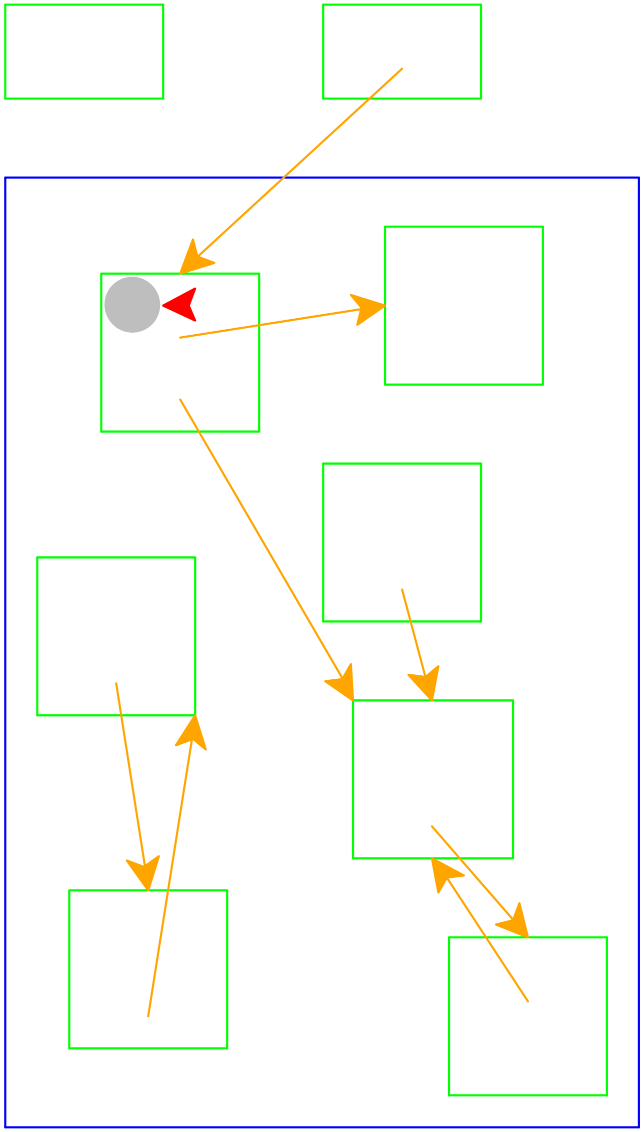
All records are marked white

Garbage Collection



Mark records referenced by registers as gray

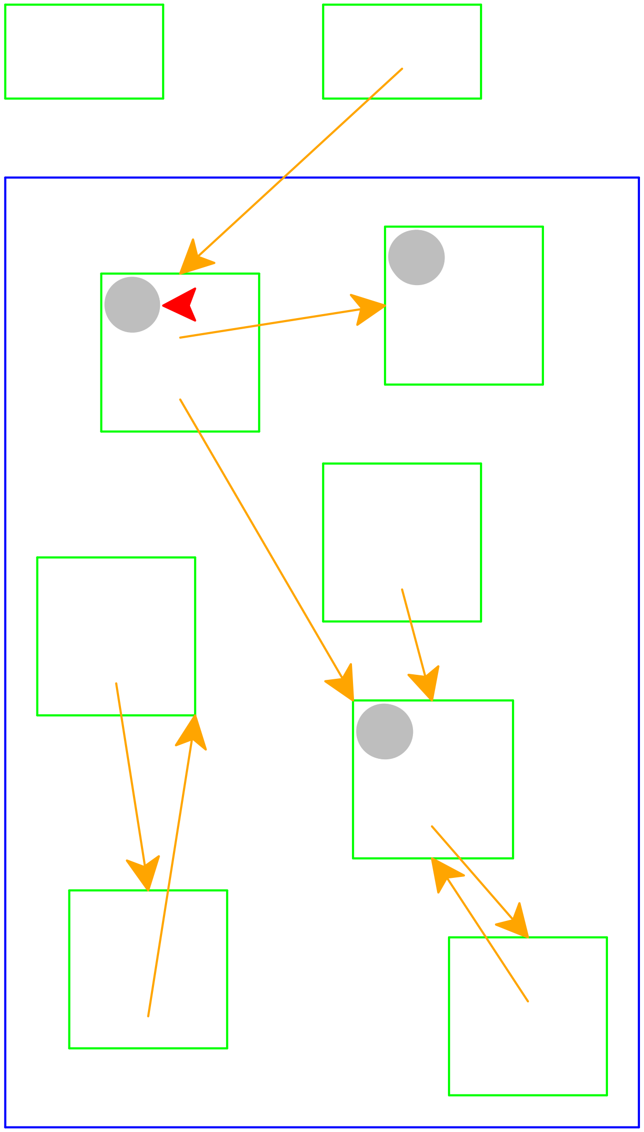
Garbage Collection



Need to pick a gray record

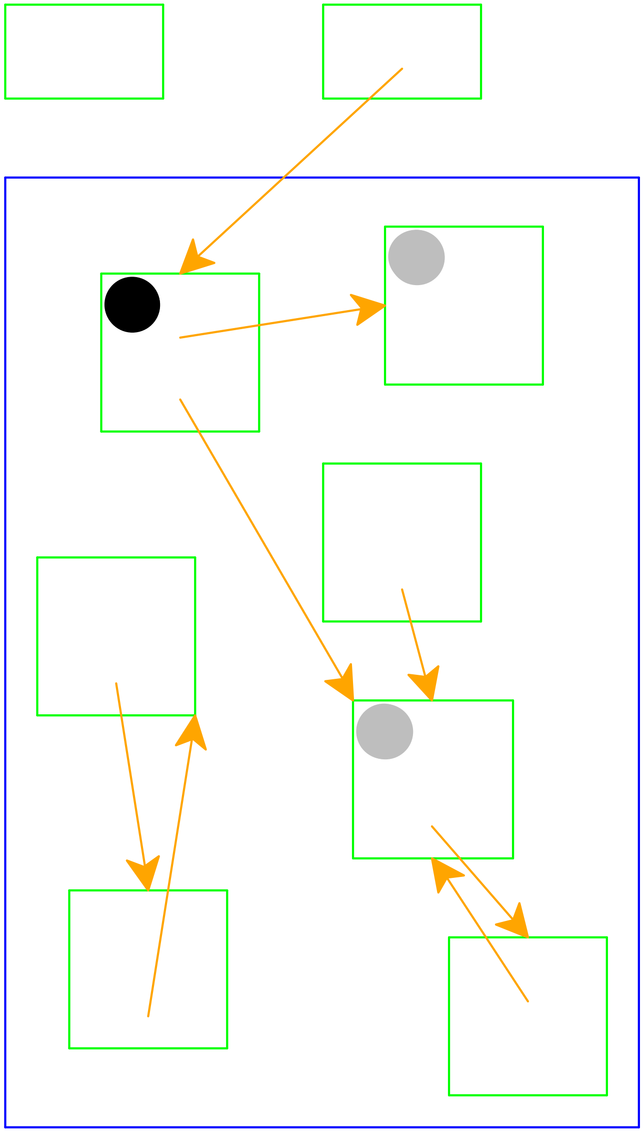
Red arrow indicates the chosen record

Garbage Collection



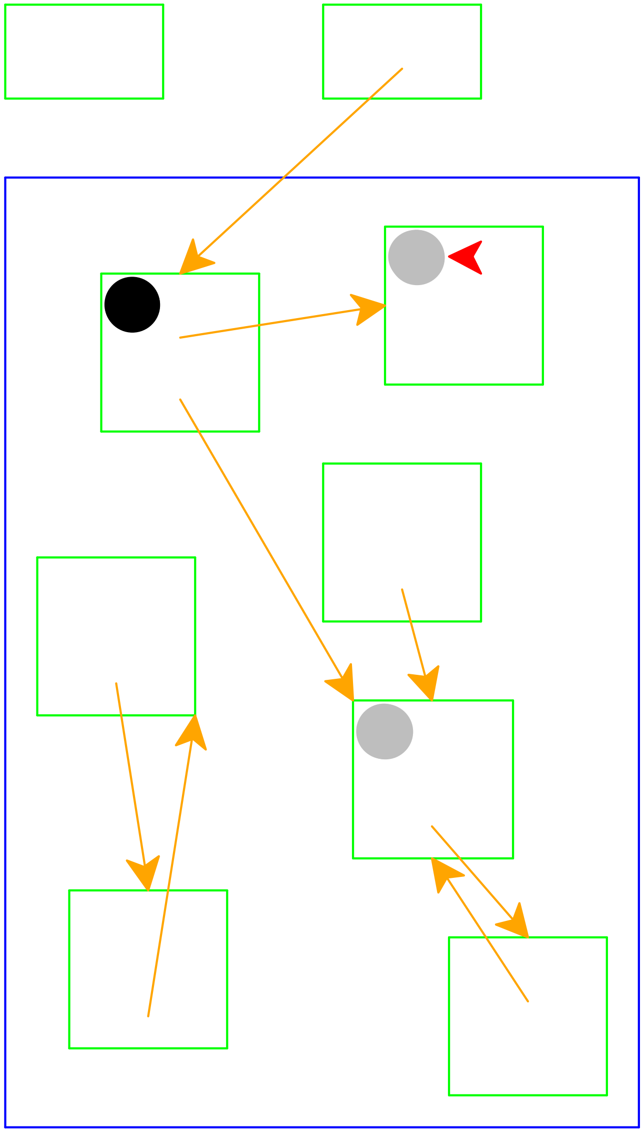
Mark white records referenced by chosen record as gray

Garbage Collection



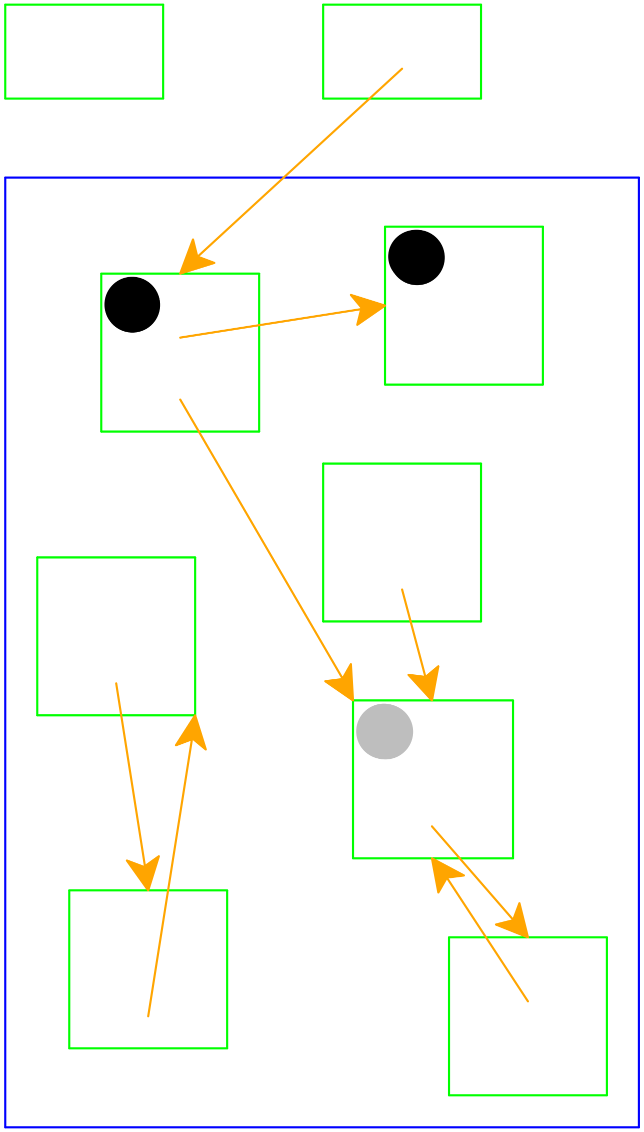
Mark chosen record black

Garbage Collection



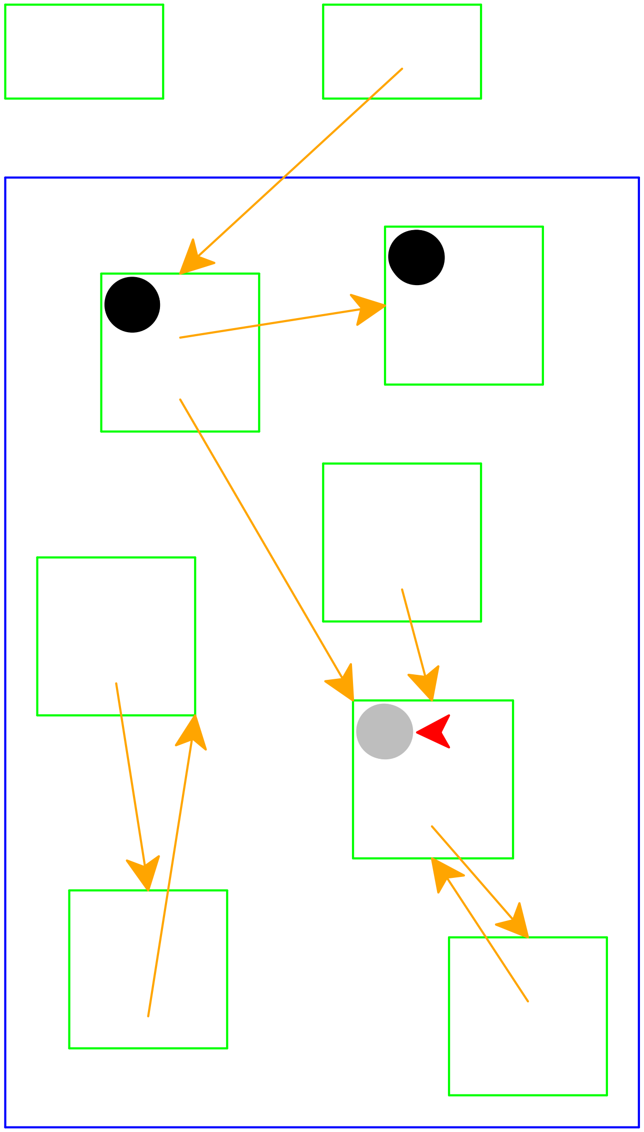
Start again: pick a gray record

Garbage Collection



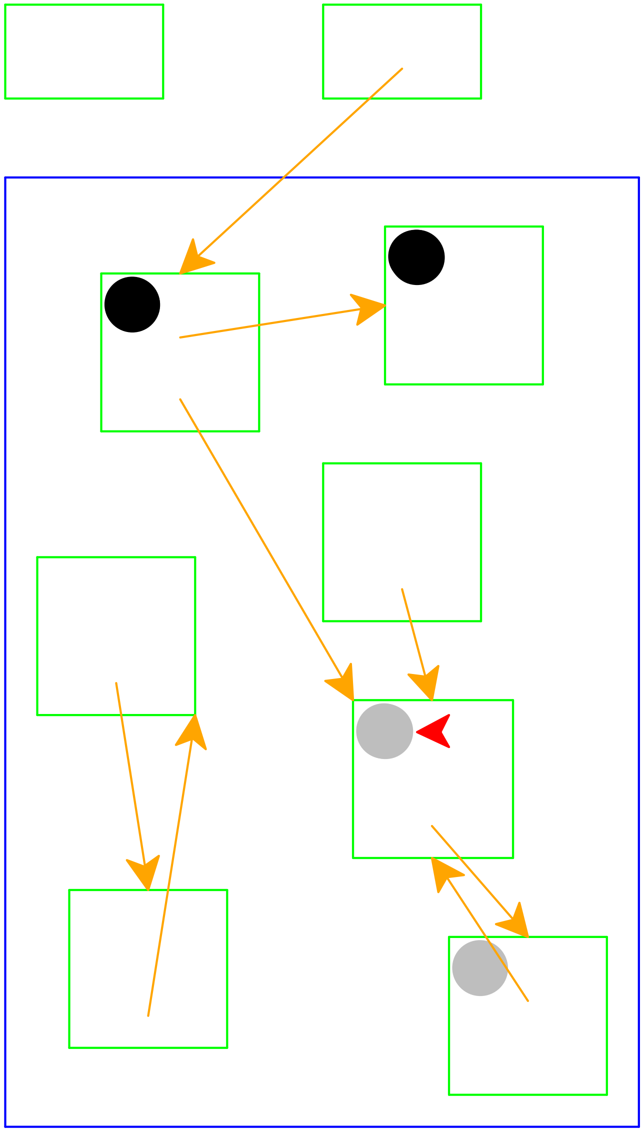
No referenced records; mark black

Garbage Collection



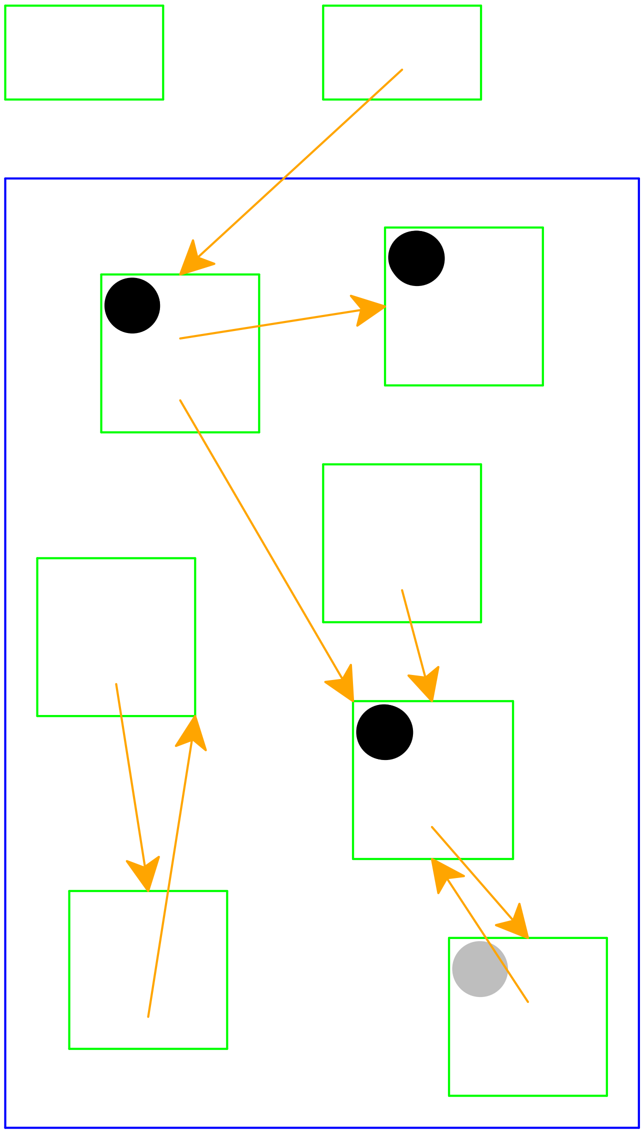
Start again: pick a gray record

Garbage Collection



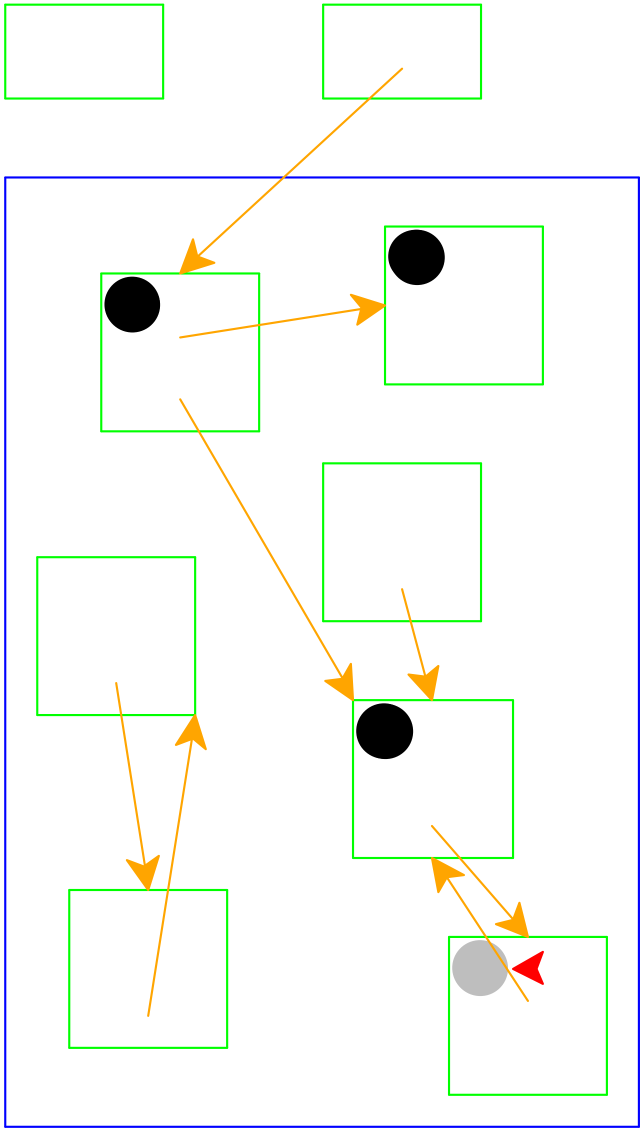
Mark white records referenced by chosen record as gray

Garbage Collection



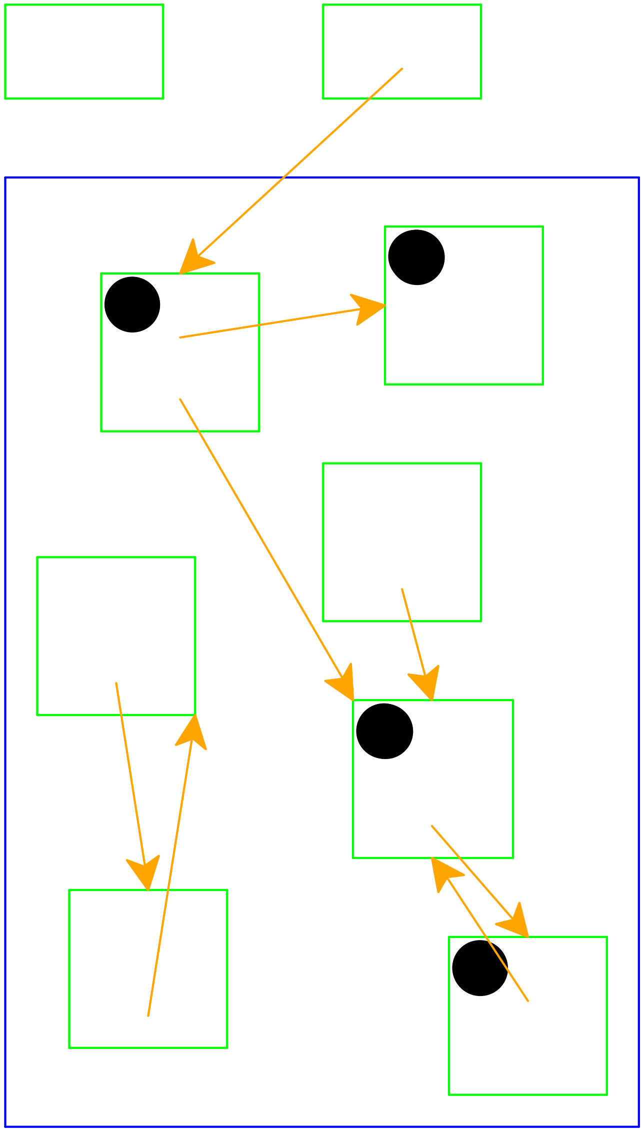
Mark chosen record black

Garbage Collection



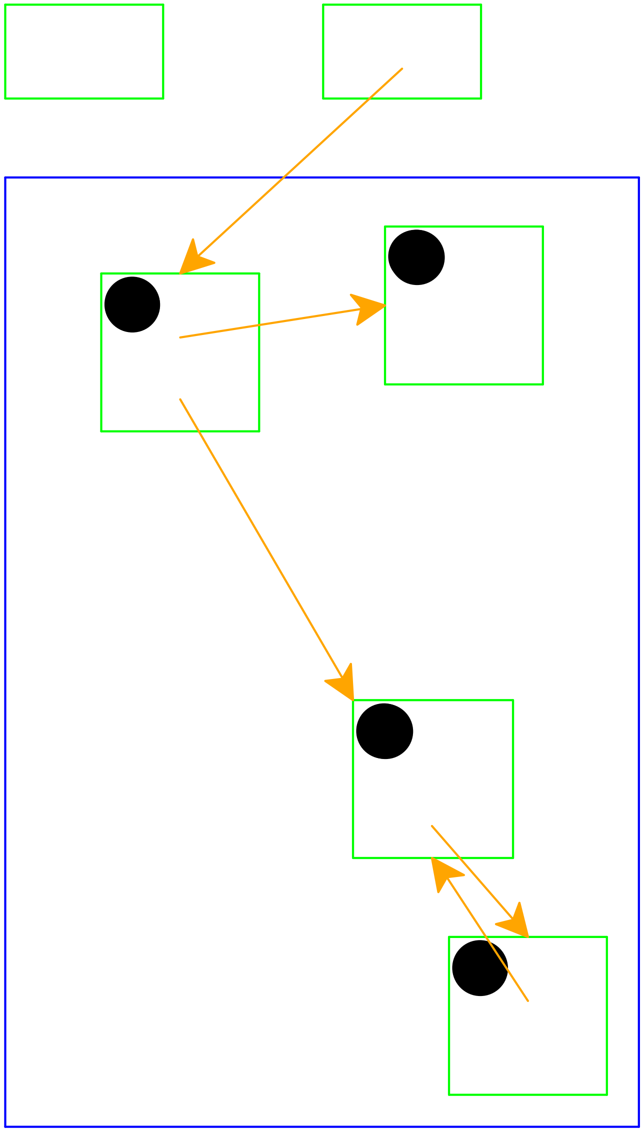
Start again: pick a gray record

Garbage Collection



No referenced white records; mark black

Garbage Collection



No more gray records; deallocate white records

Cycles ***do not*** break garbage collection

Two-Space Copying Collectors

A ***two-space*** copying collector compacts memory as it collects, making allocation easier.

Allocator:

- Partitions memory into ***to-space*** and ***from-space***
- Allocates only in ***to-space***

Collector:

- Starts by swapping ***to-space*** and ***from-space***
- Coloring gray ⇒ copy from ***from-space*** to ***to-space***
- Choosing a gray record ⇒ walk once though the new ***to-space***, update pointers

Copying Collectors

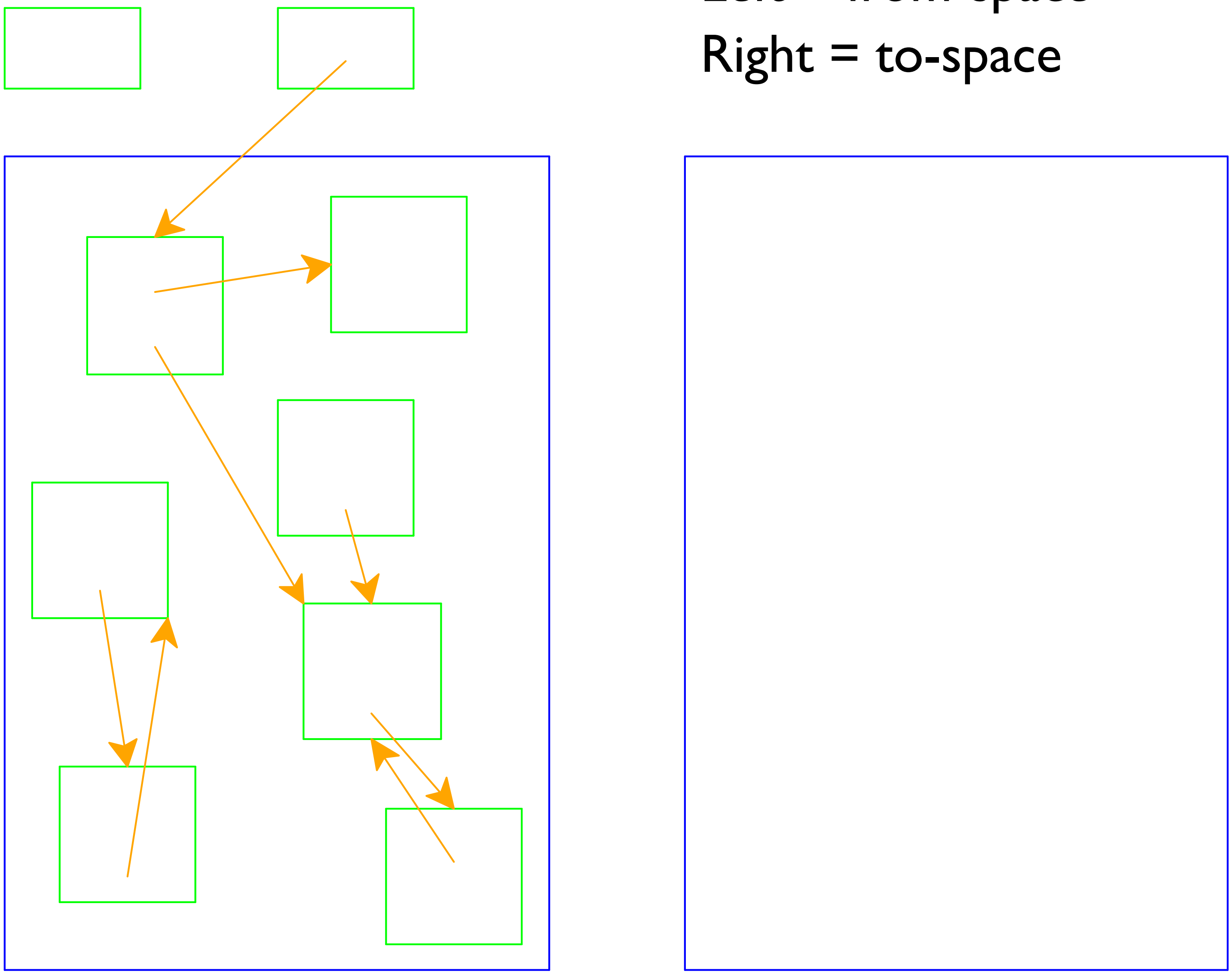
In some languages (not C), it’s ok to move allocated object

- Moving objects around gets rid of fragmentation
- Requires the ability to update any pointer that references a moved object

Two-Space Collection

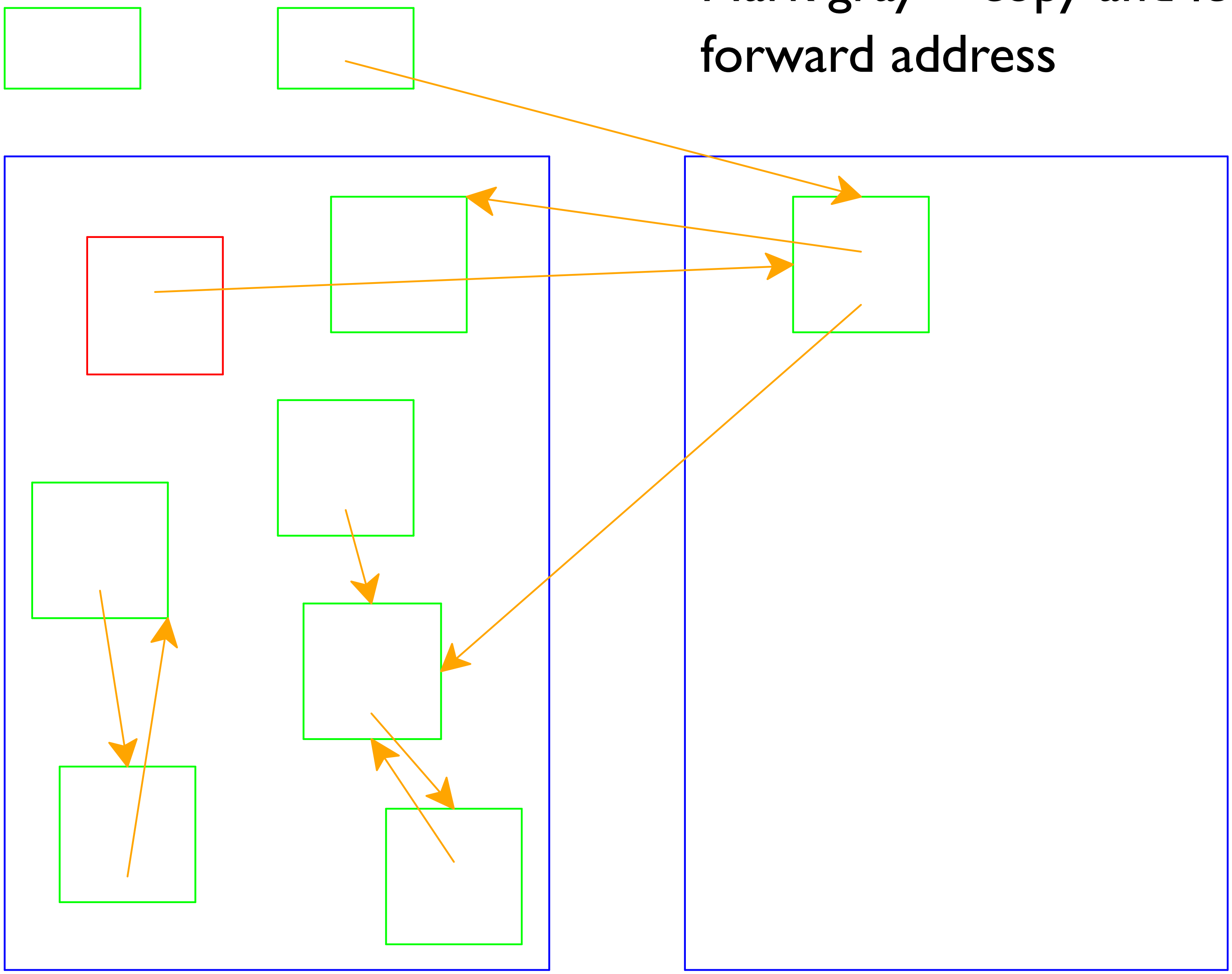
Left = from-space

Right = to-space



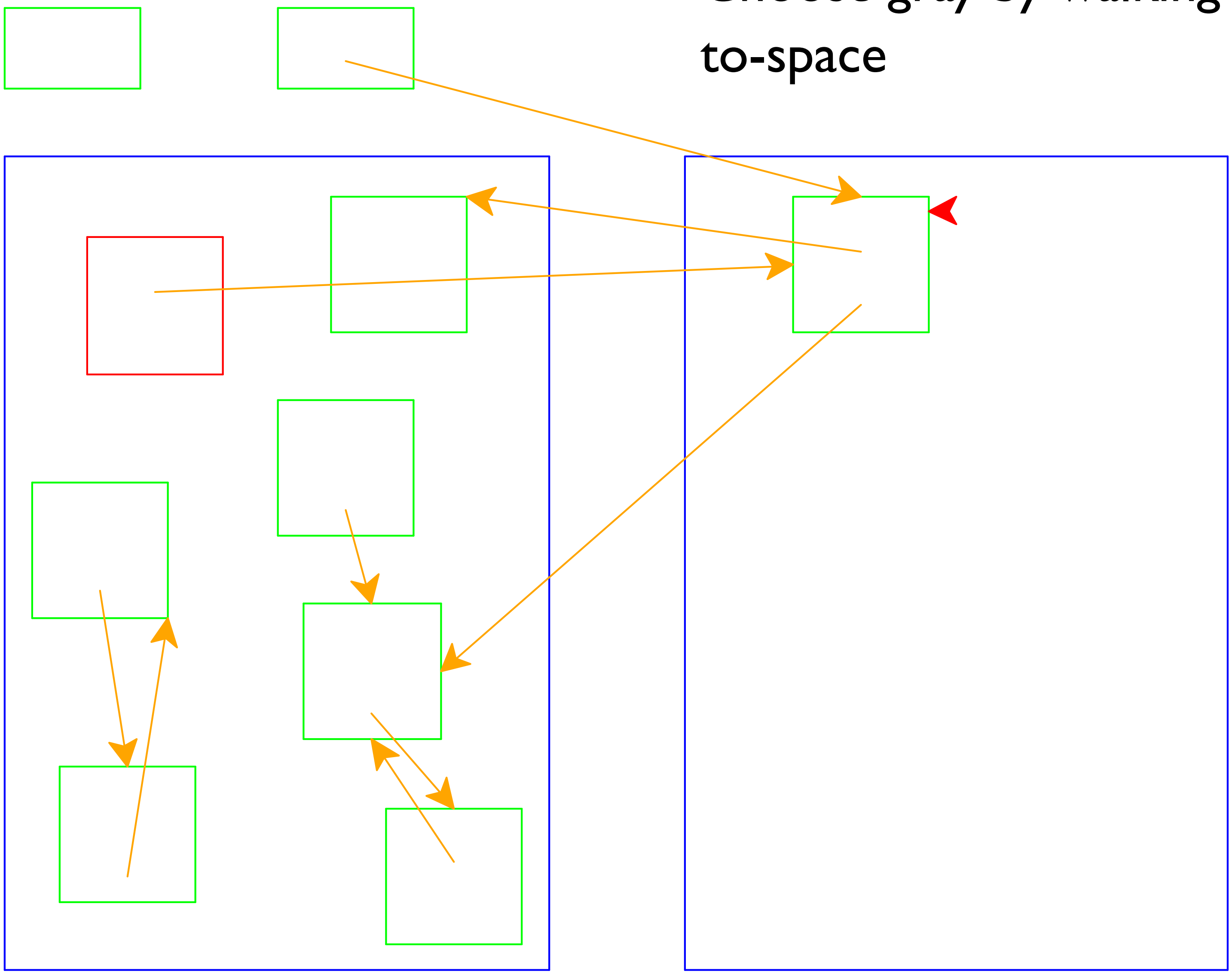
Two-Space Collection

Mark gray = copy and leave forward address



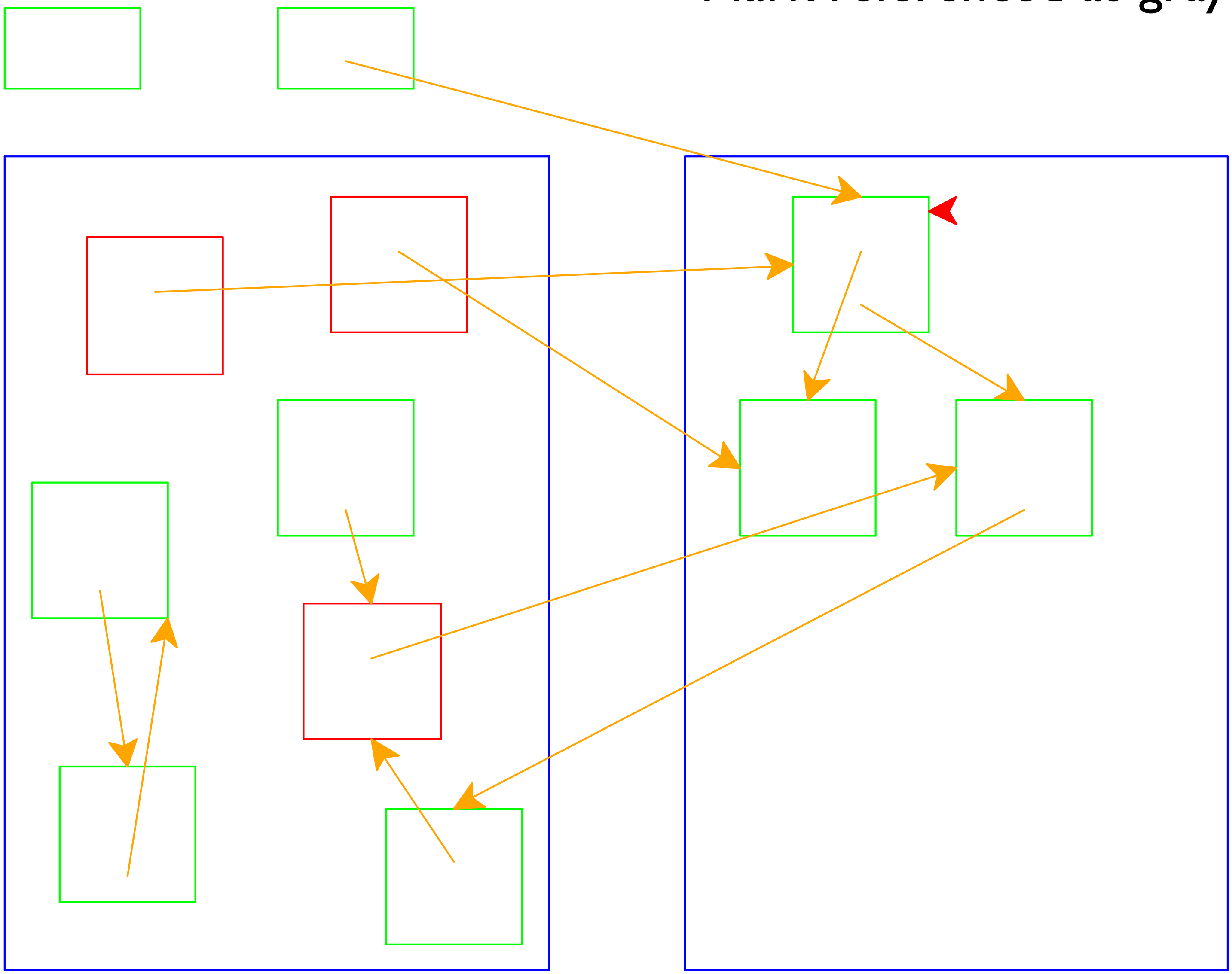
Two-Space Collection

Choose gray by walking through to-space



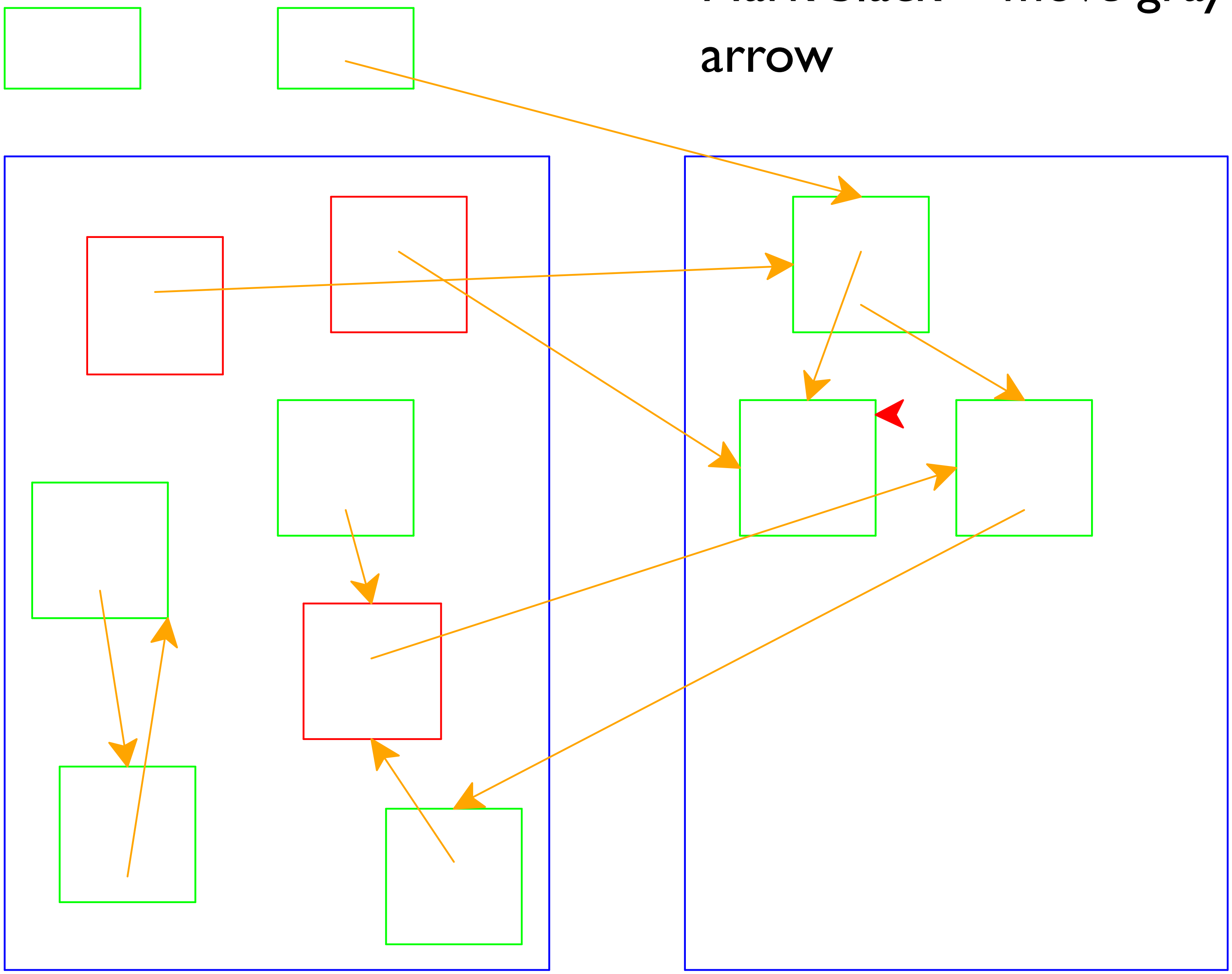
Two-Space Collection

Mark referenced as gray



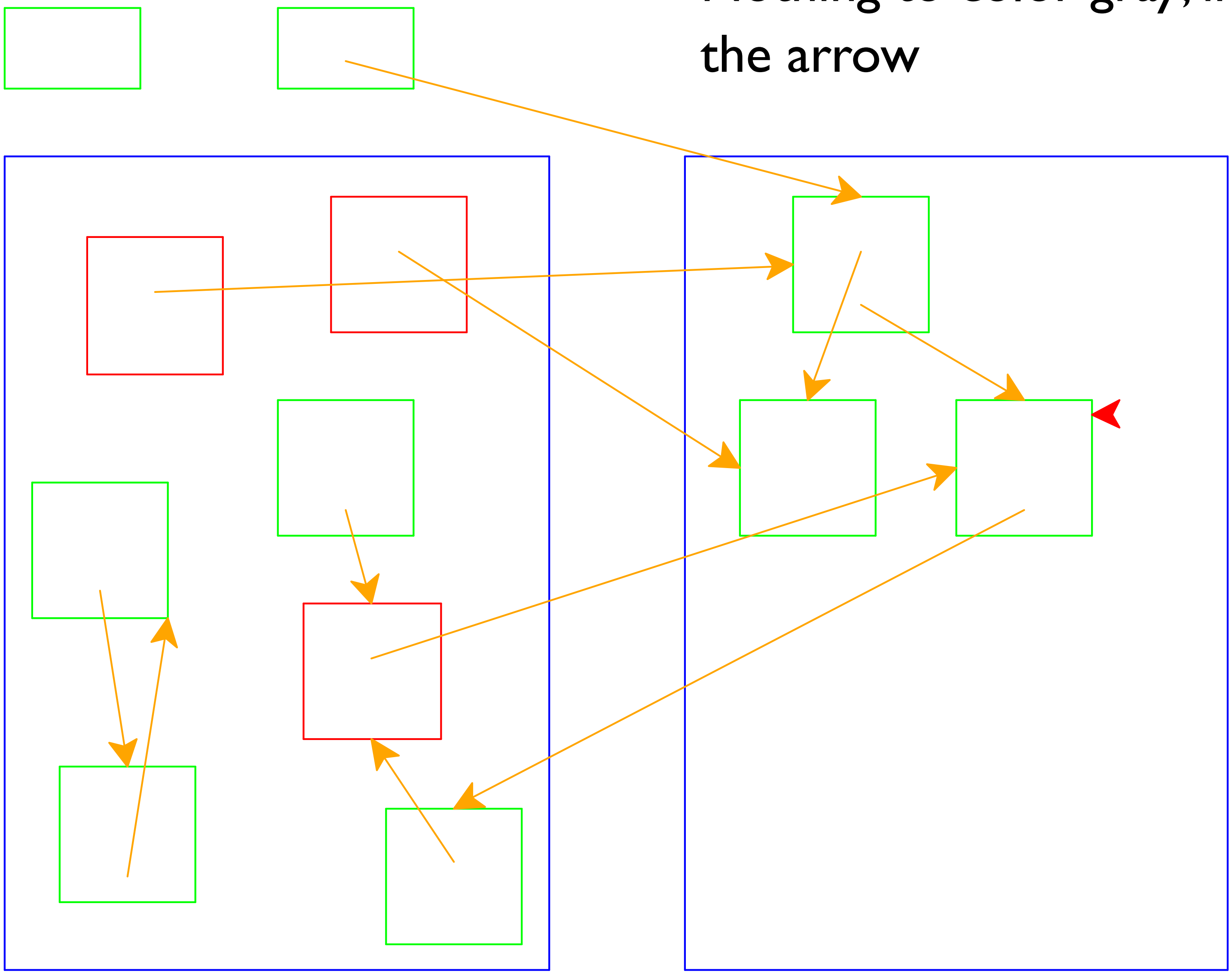
Two-Space Collection

Mark black = move gray-choosing arrow



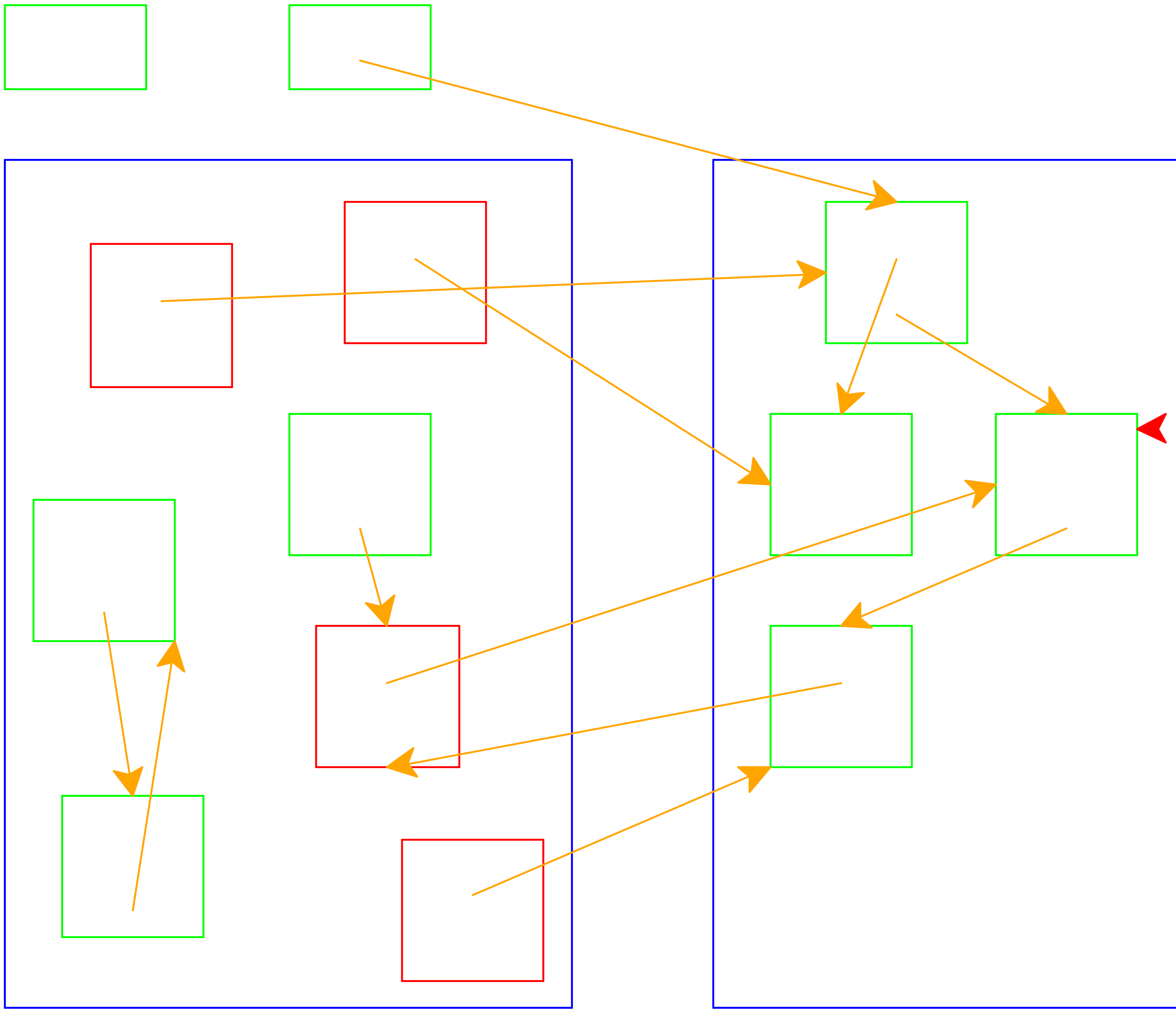
Two-Space Collection

Nothing to color gray; increment the arrow



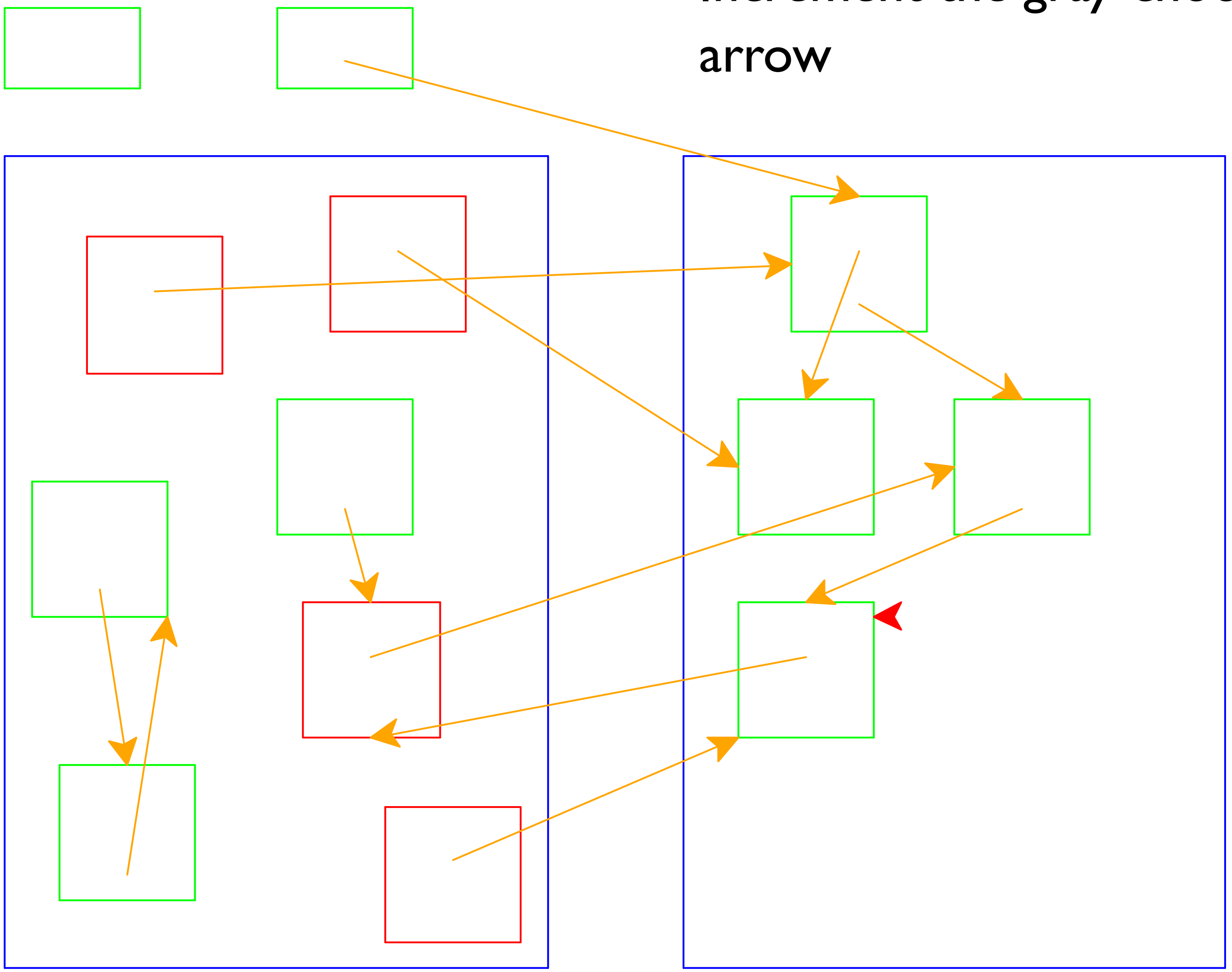
Two-Space Collection

Color referenced record gray



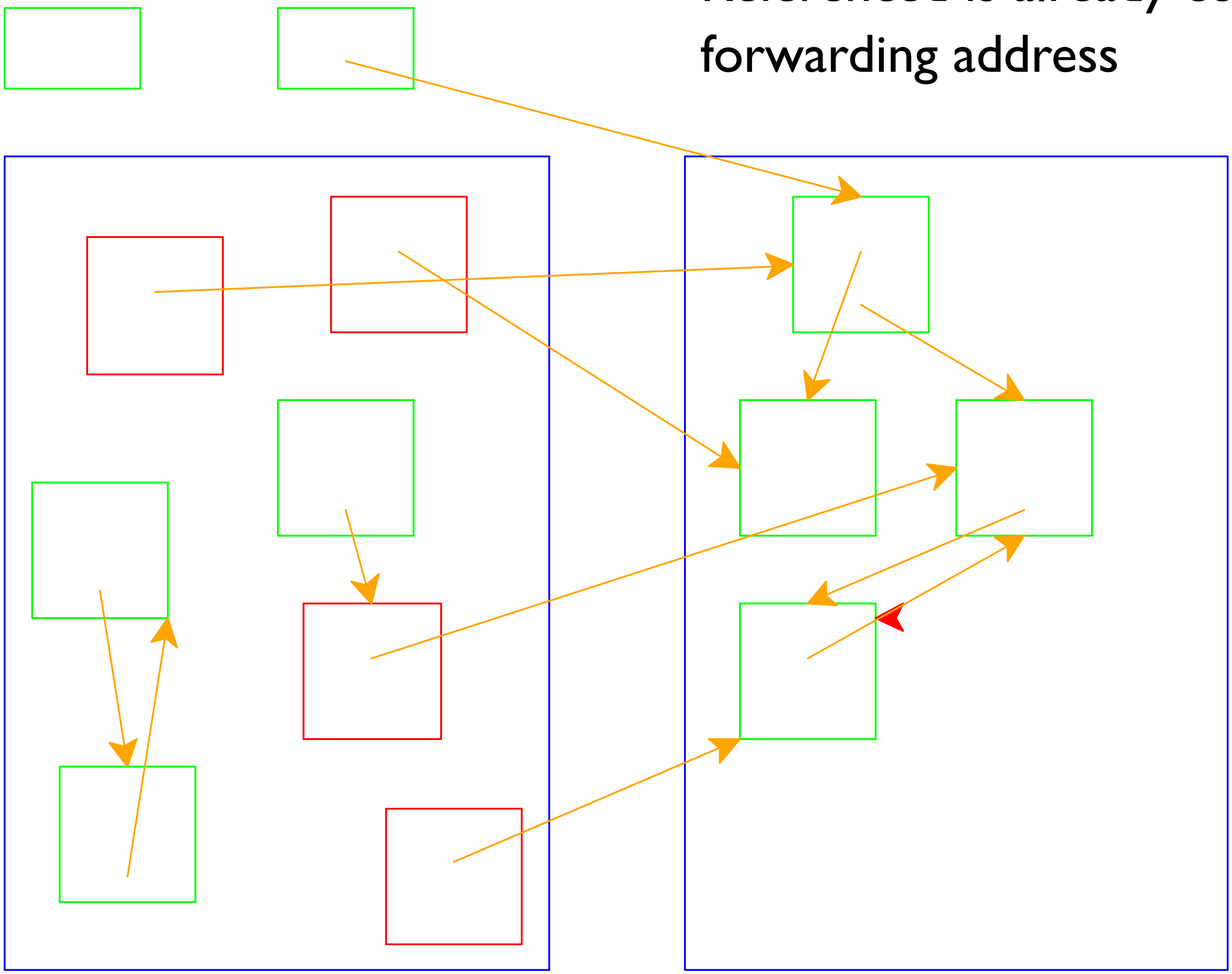
Two-Space Collection

Increment the gray-choosing arrow

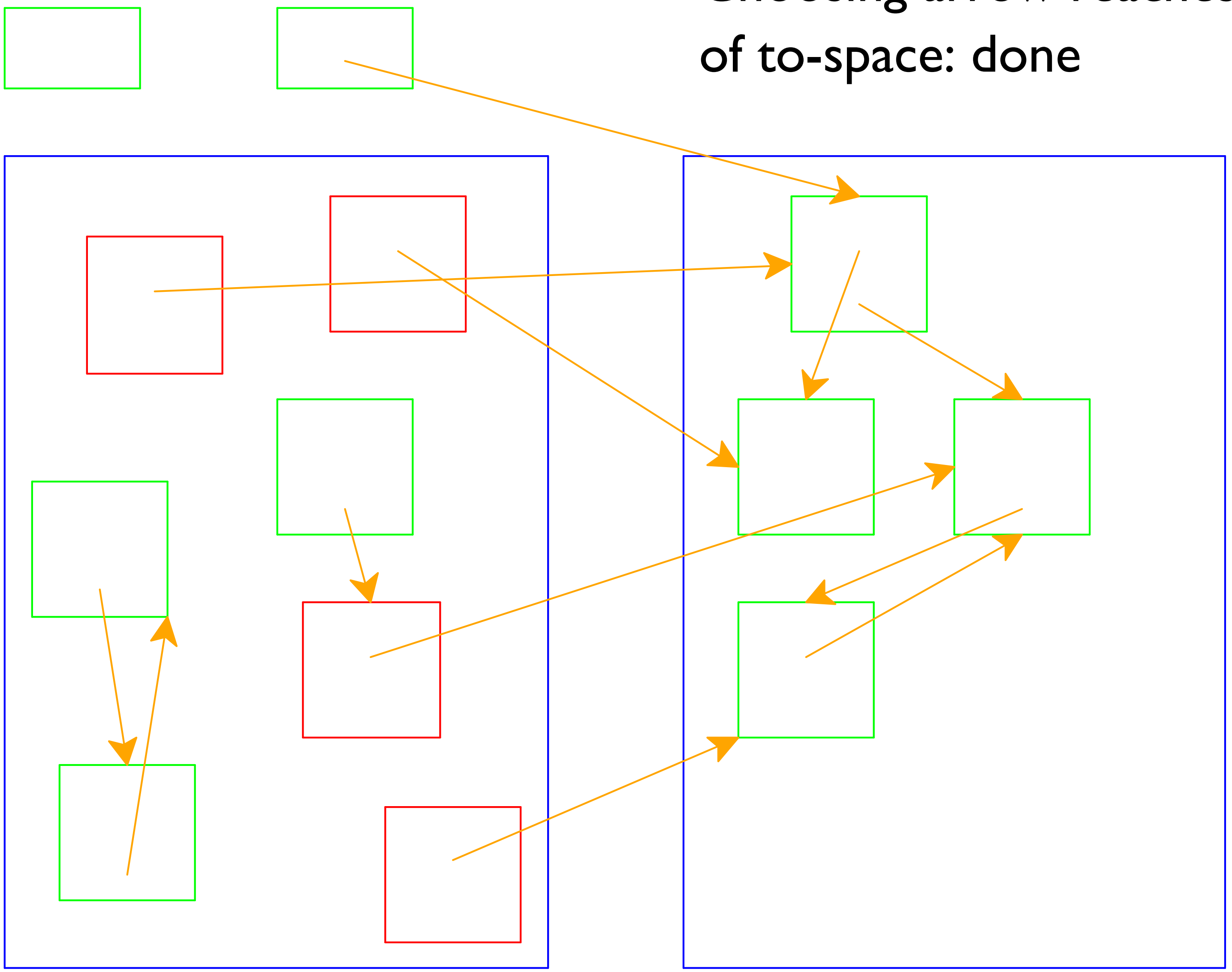


Two-Space Collection

Referenced is already copied, use forwarding address



Two-Space Collection



Choosing arrow reaches the end of to-space: done

Two-Space Collection

Right = from-space

Left = to-space