CSCI-344
Programming Language Concepts (Section 3)

Lecture 31

Copying Garbage Collection

Instructor: Hossein Hojjat

December 5, 2016

Where we are

Done:

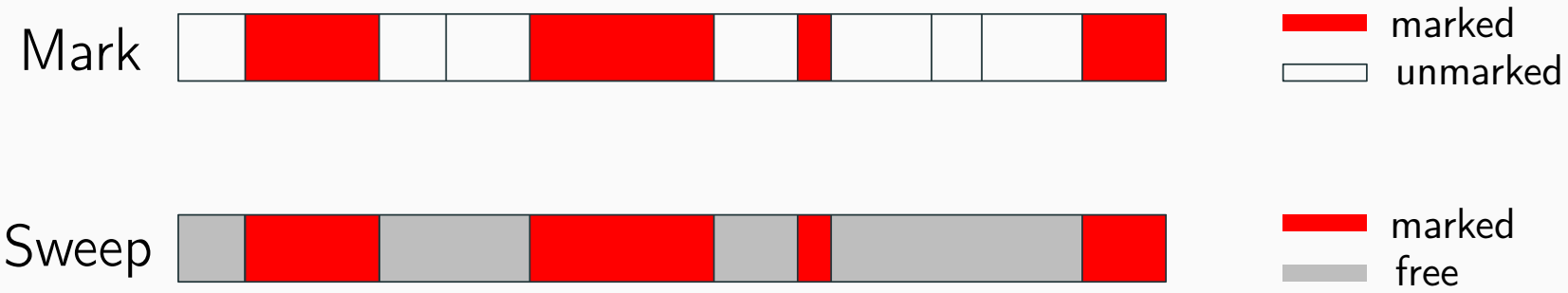
- Garbage Collection
  - Reference counting
  - Mark-and-Sweep Garbage Collection

This session:

- Copying Garbage Collection
- Quantitative aspects

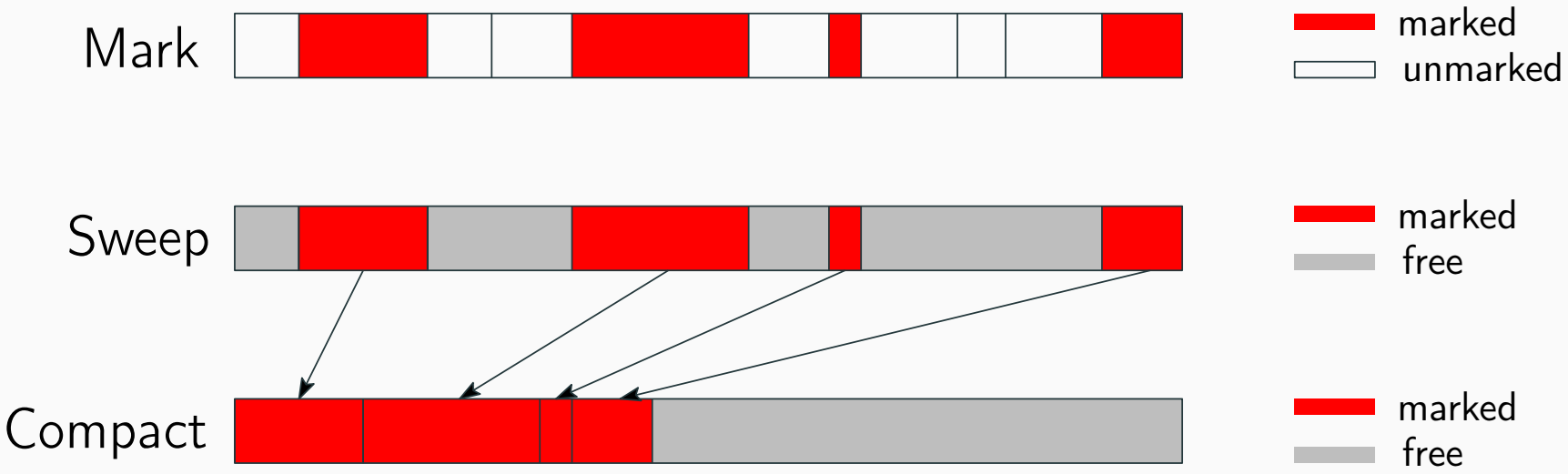
Mark-Compact Garbage Collection

- Mark-Sweep Disadvantage: Fragmentation
- Memory objects are not moved: space can become fragmented
- Makes is difficult to allocate large objects
  - although space is still available overall



Mark-Compact Garbage Collection

- Mark-Sweep Disadvantage: Fragmentation
- Memory objects are not moved: space can become fragmented
- Makes is difficult to allocate large objects
  - although space is still available overall
- Compaction moves live objects together to reclaim contiguous empty space



Mark-Compact Garbage Collection

Pros

- Solves the fragmentation problem
- Does not change the original ordering of the memory objects

Cons

- Several passes over data is required:

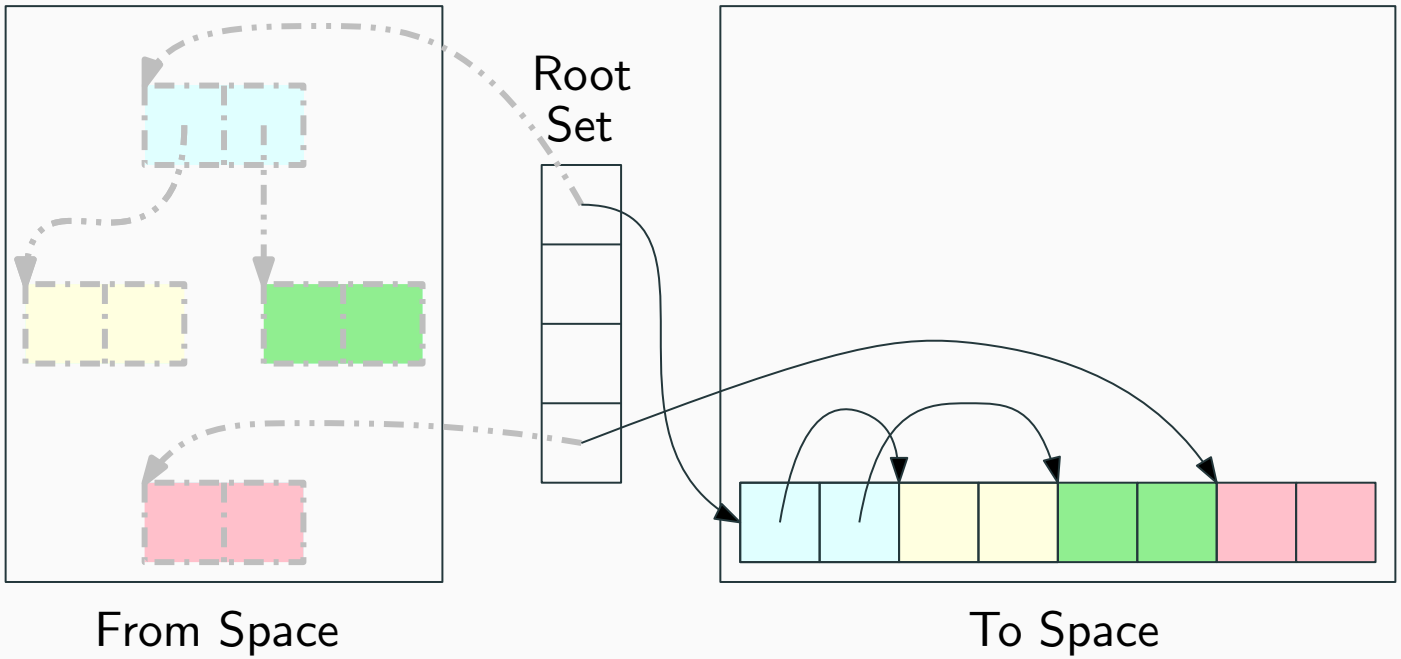
1. Marking phase
2. Compute the new locations that the objects will be moving to
3. Update pointers to refer to objects new locations
4. Actually move the objects

- Significantly slower than mark-sweep if a large portion of data needs compaction

Copying Garbage Collection

- Traverses the heap similar to marking phase
- Instead of marking objects it moves reachable objects
  - from current heap (“from”-space)
  - to new area (“to”-space)
- When all reachable objects have been moved, from-space and to-space switch roles
- Copying garbage collection trades space for time

Copying Garbage Collection



Allocation

Mark-Sweep Garbage Collection

- Maintains a segregated free list
- Free list is divided into several size classes
- Memory objects are allocated from the appropriate size class using a first-fit algorithm

Copying Garbage Collection

- Use bump pointer allocation (pointer increment)
- Allocator maintains a pointer to the start of available memory
- Increments pointer by the allocation size requested by the program
- Bump pointer allocation is much faster than free-list allocation

Cheney's Algorithm

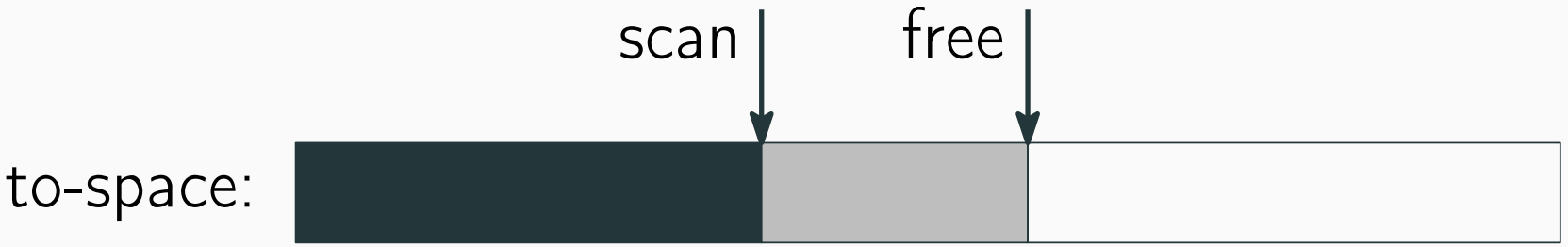
Copy all reachable objects with breadth-first traversal:

1) Copy objects that are reachable from the root to to-space
2) Scan to-space and fix pointers to from-space
   - Copy objects if not yet copied
   - When an object is copied, install a forwarding pointer on the old version of object
   - Update the pointer slot
3) Garbage collection completes when all objects in to-space scanned

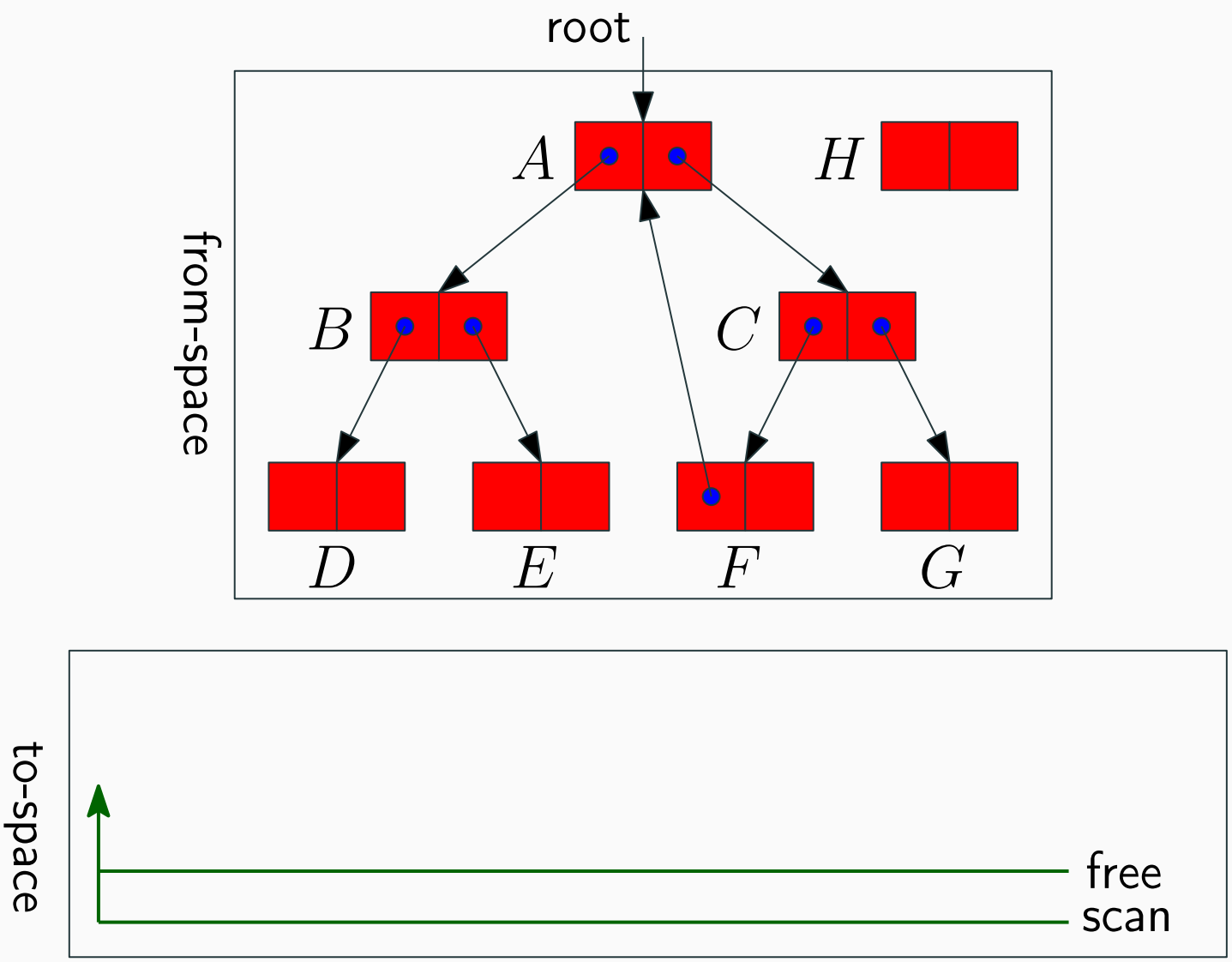
Cheney's Algorithm

A memory object has three different states:

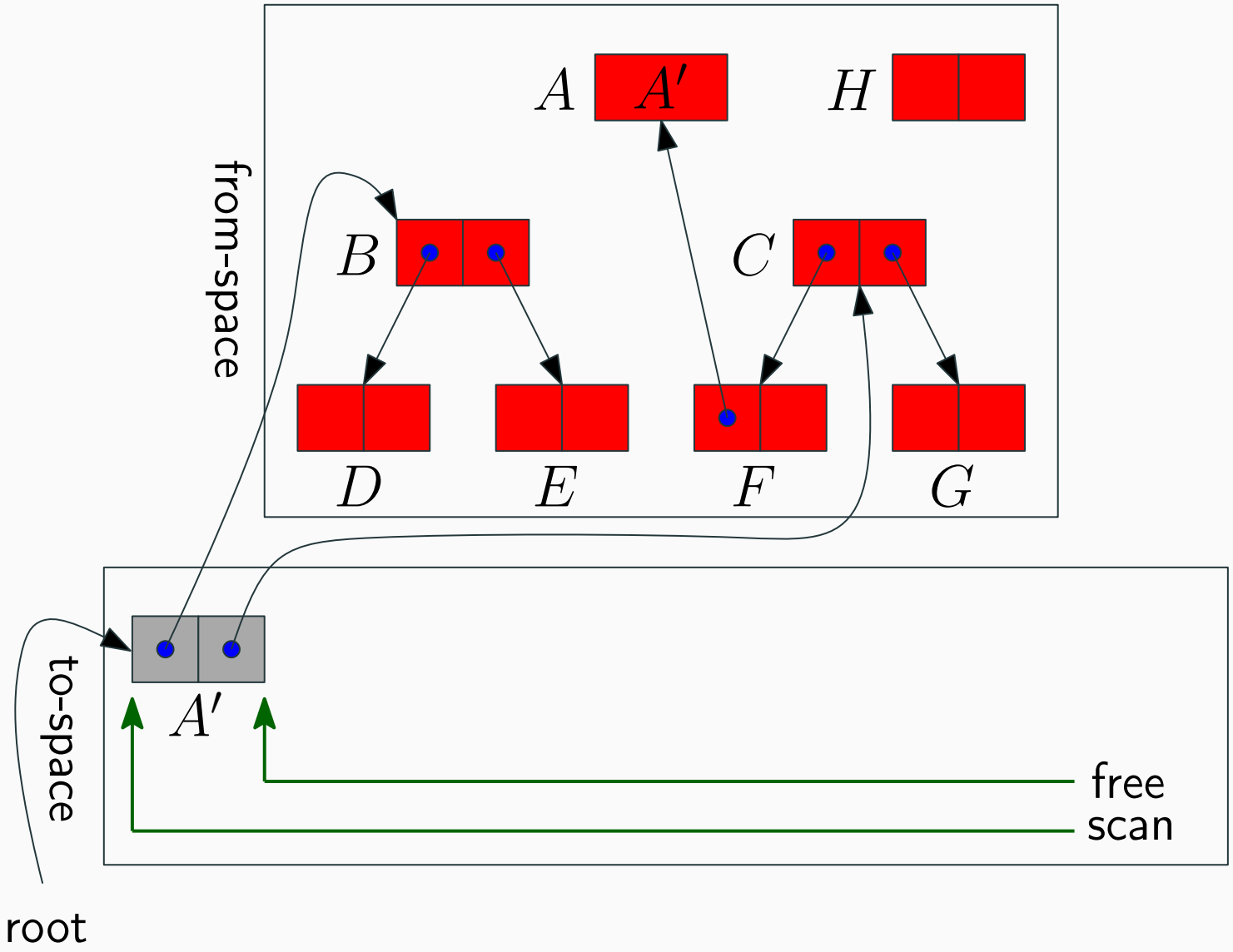
- **White:** still in from-space
- **Gray:** is copied to to-space but still references objects in the from-space
- **Black:** is in the to-space and only references objects in the to-space



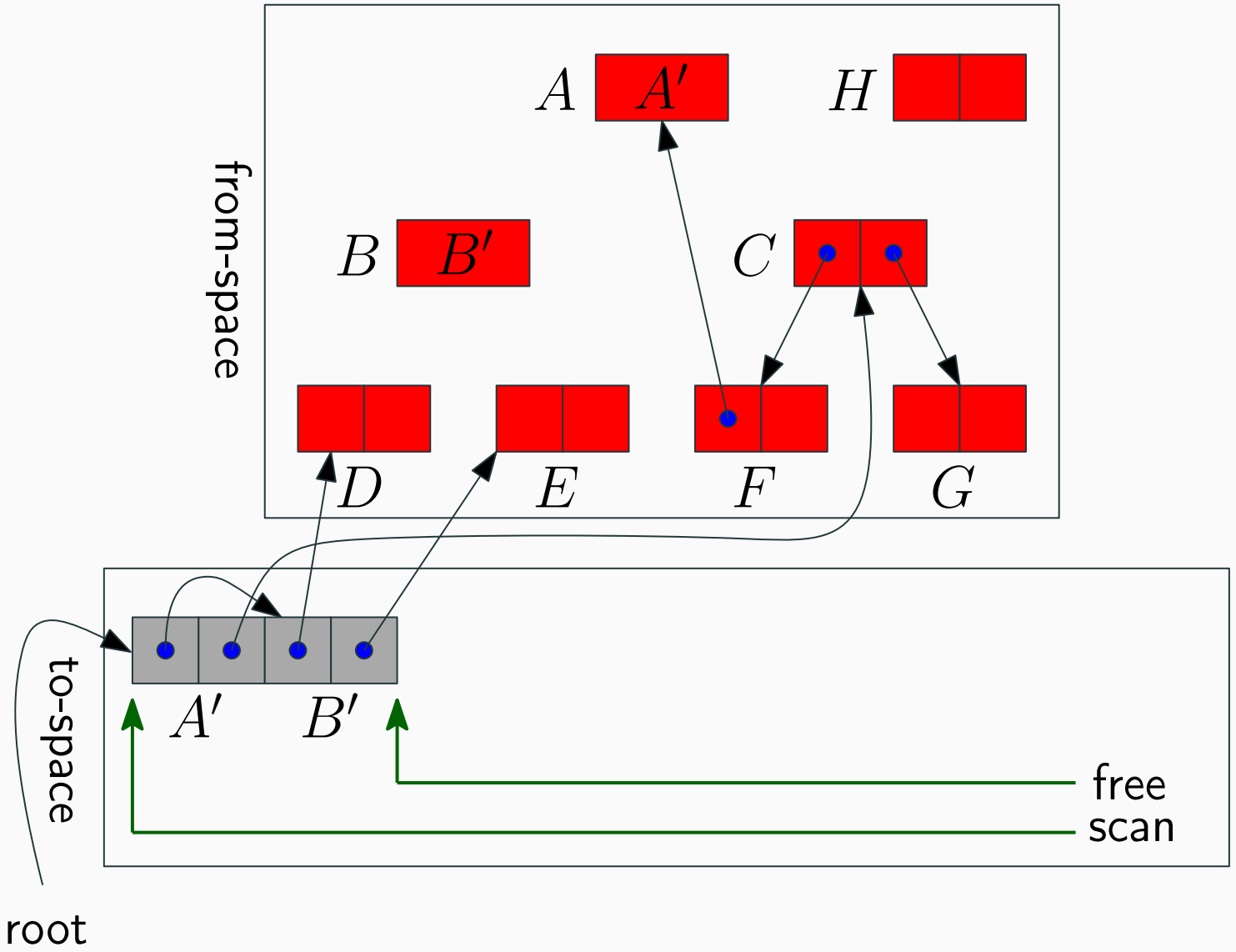
Cheney's Algorithm



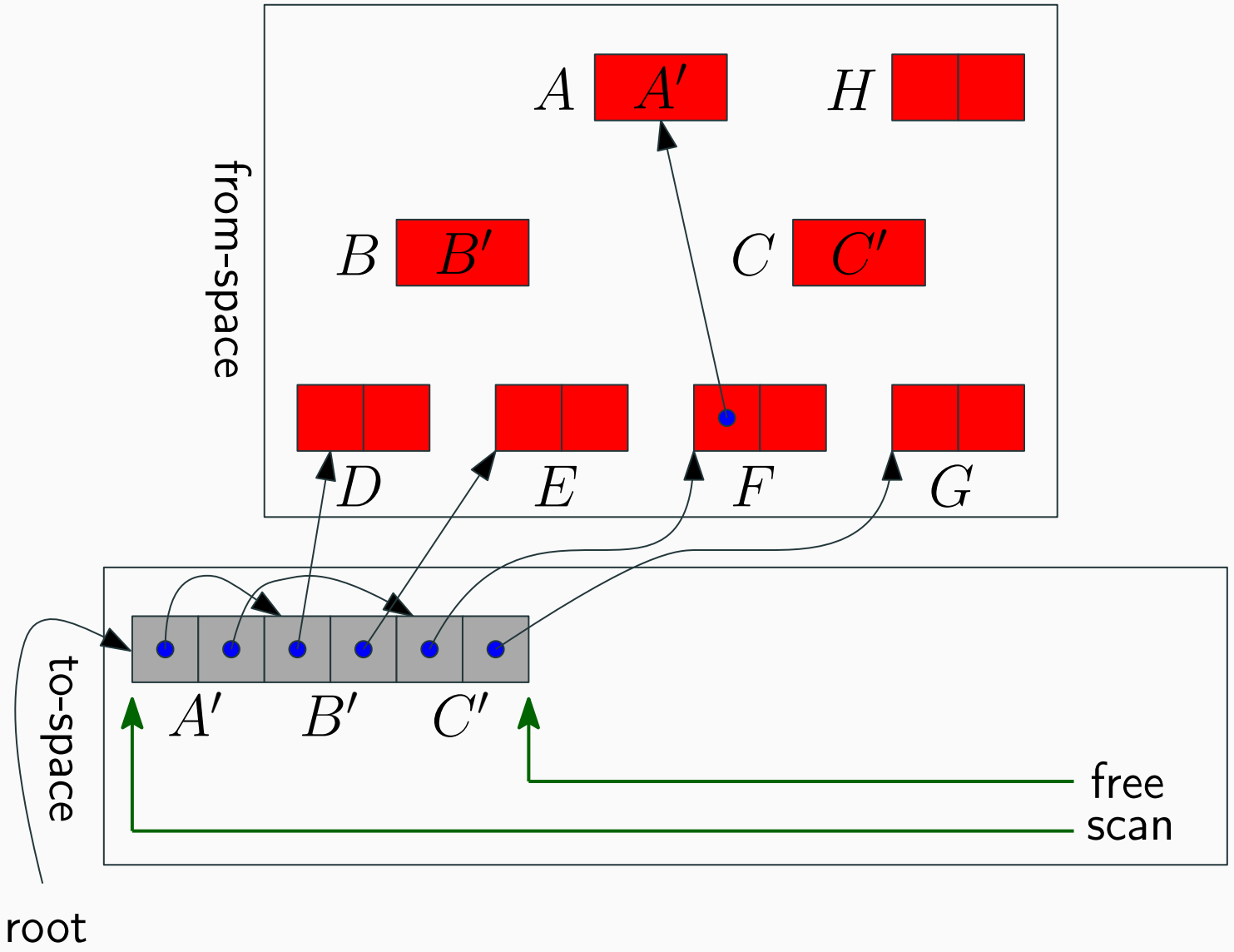
Cheney's Algorithm



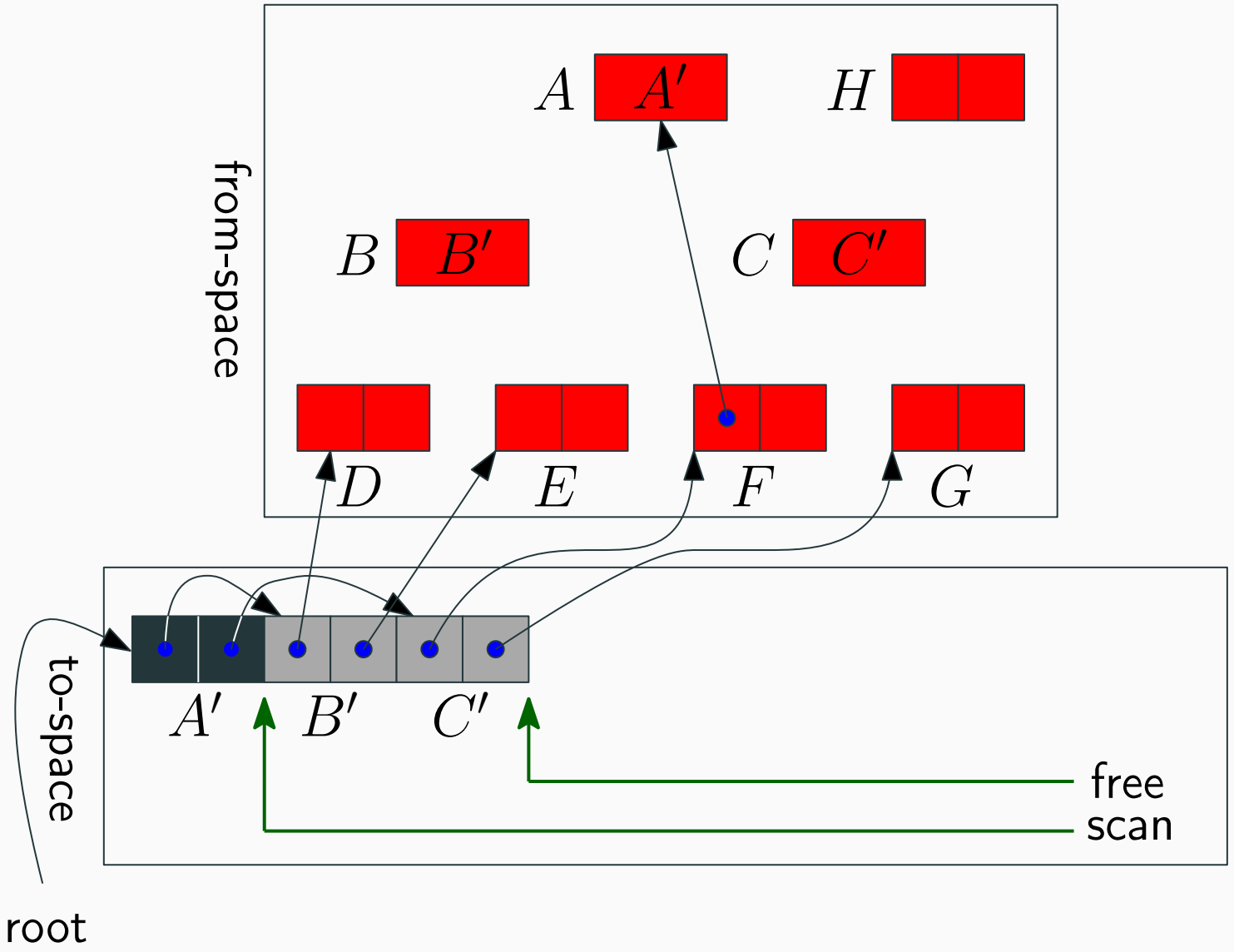
Cheney's Algorithm



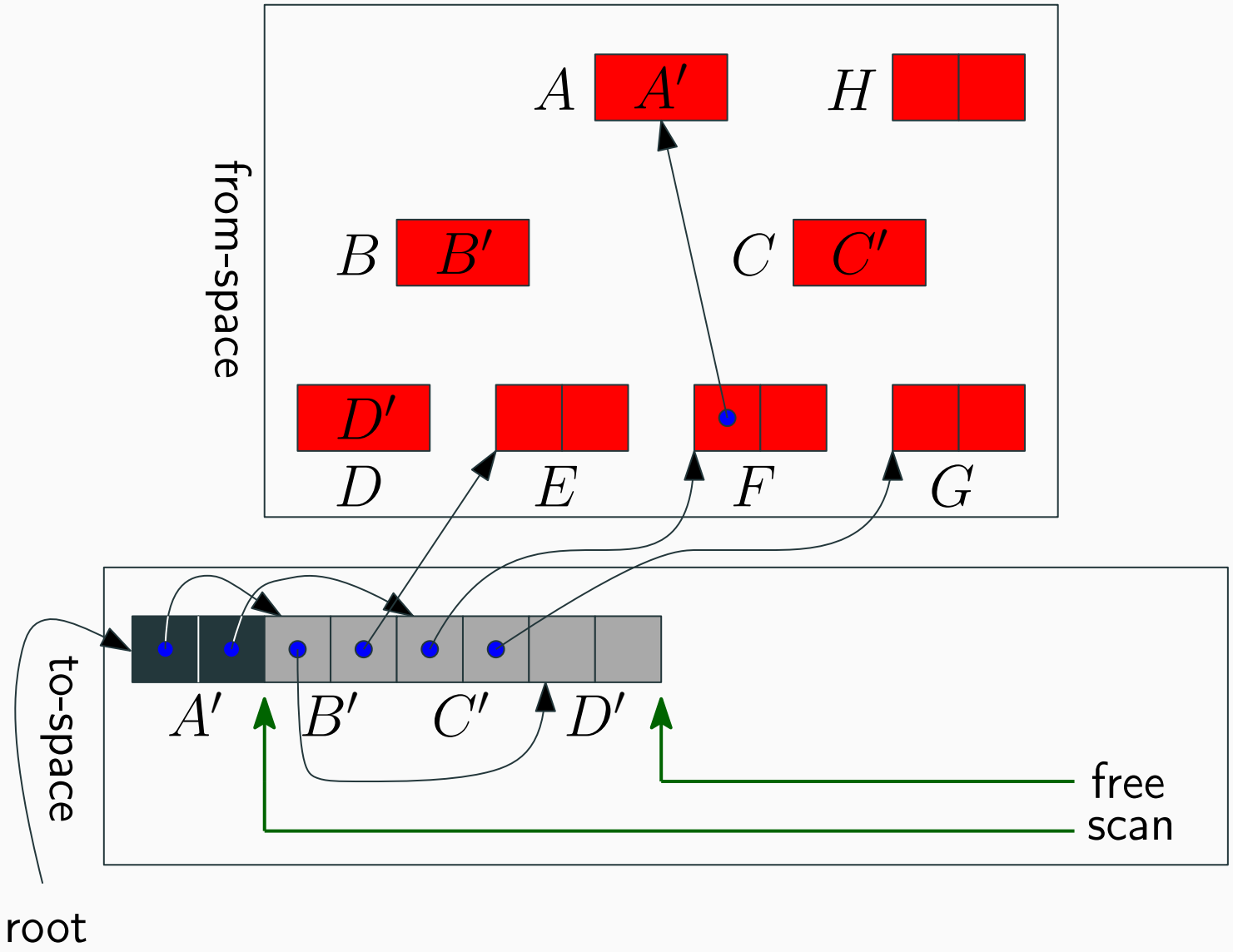
Cheney's Algorithm



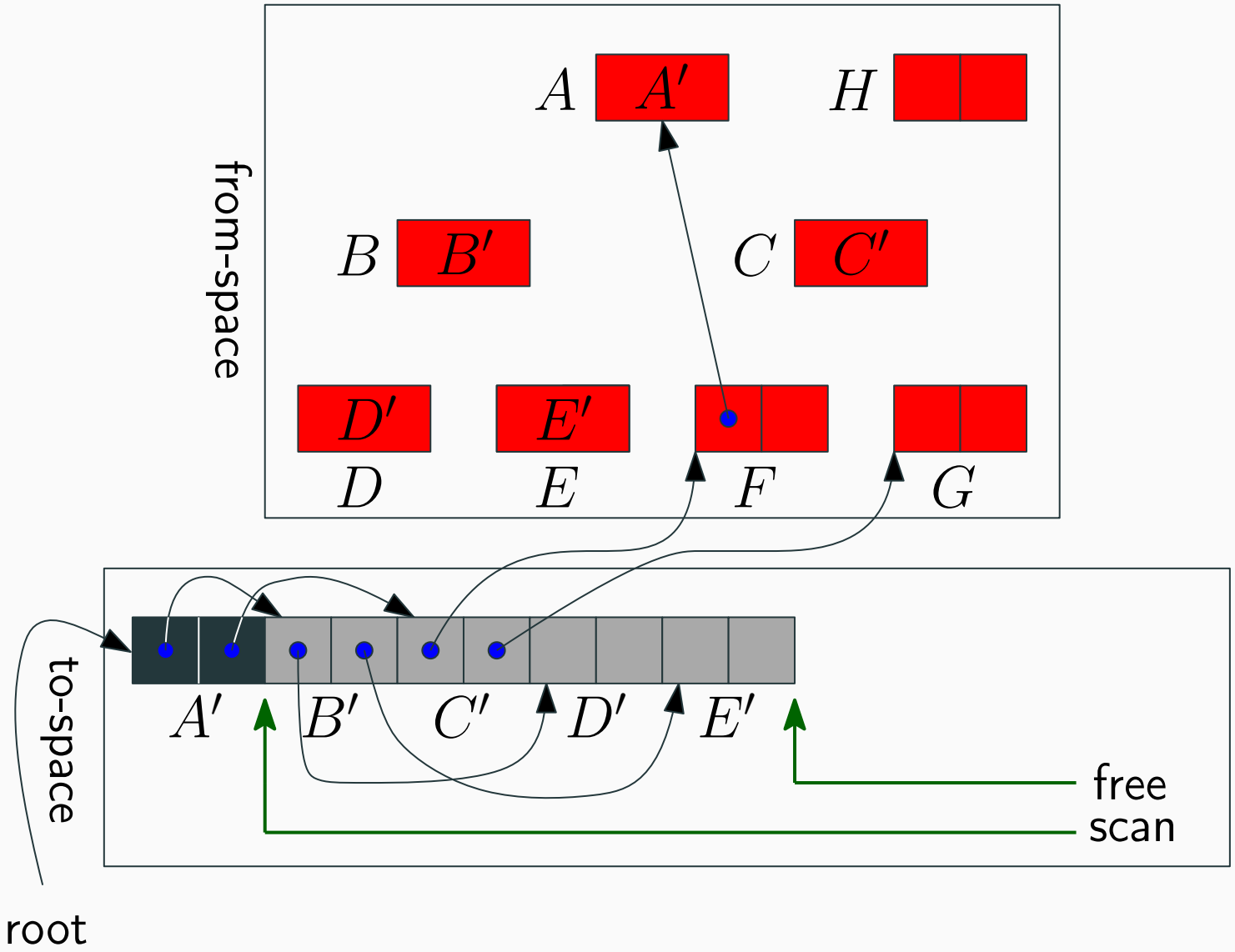
Cheney's Algorithm



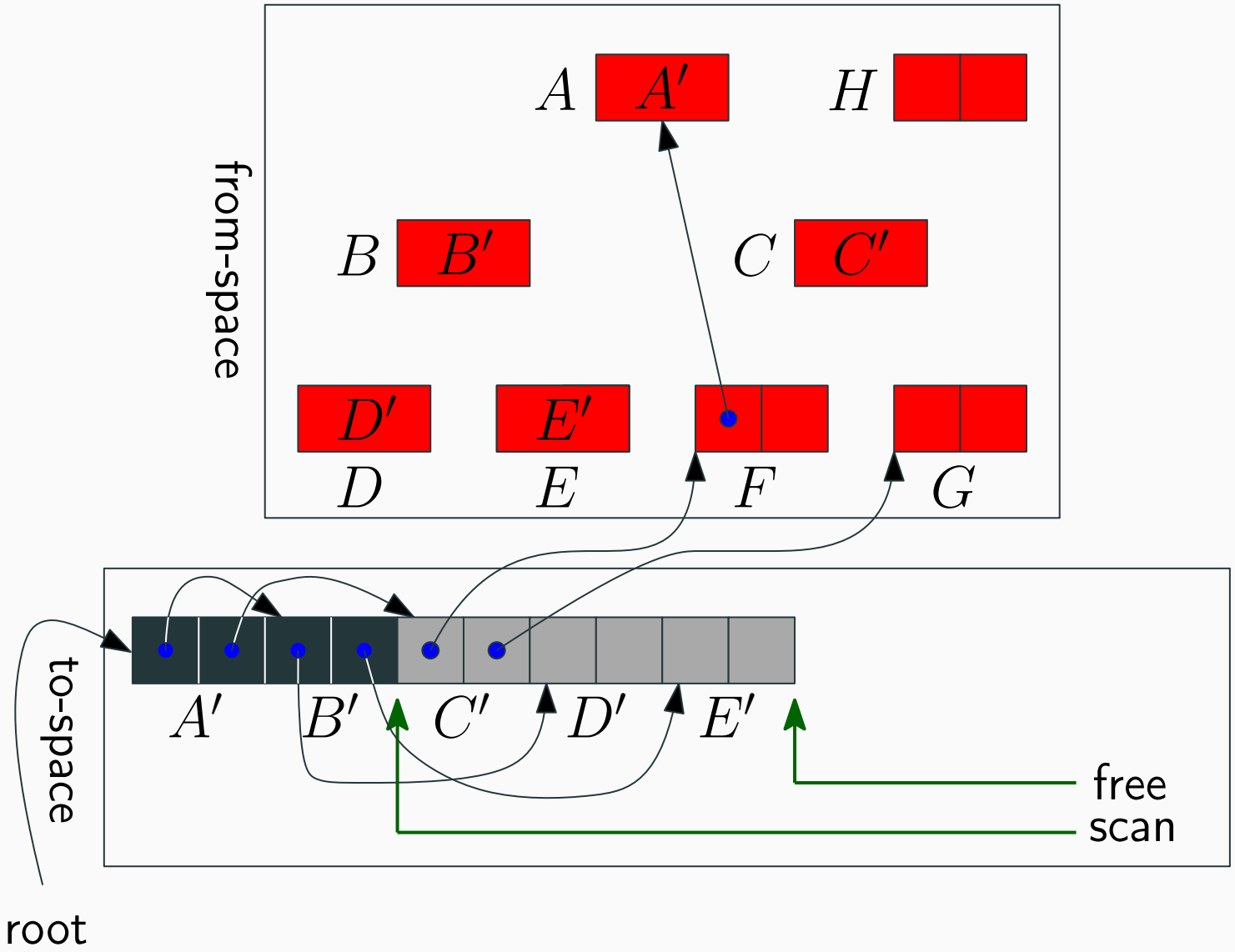
Cheney's Algorithm



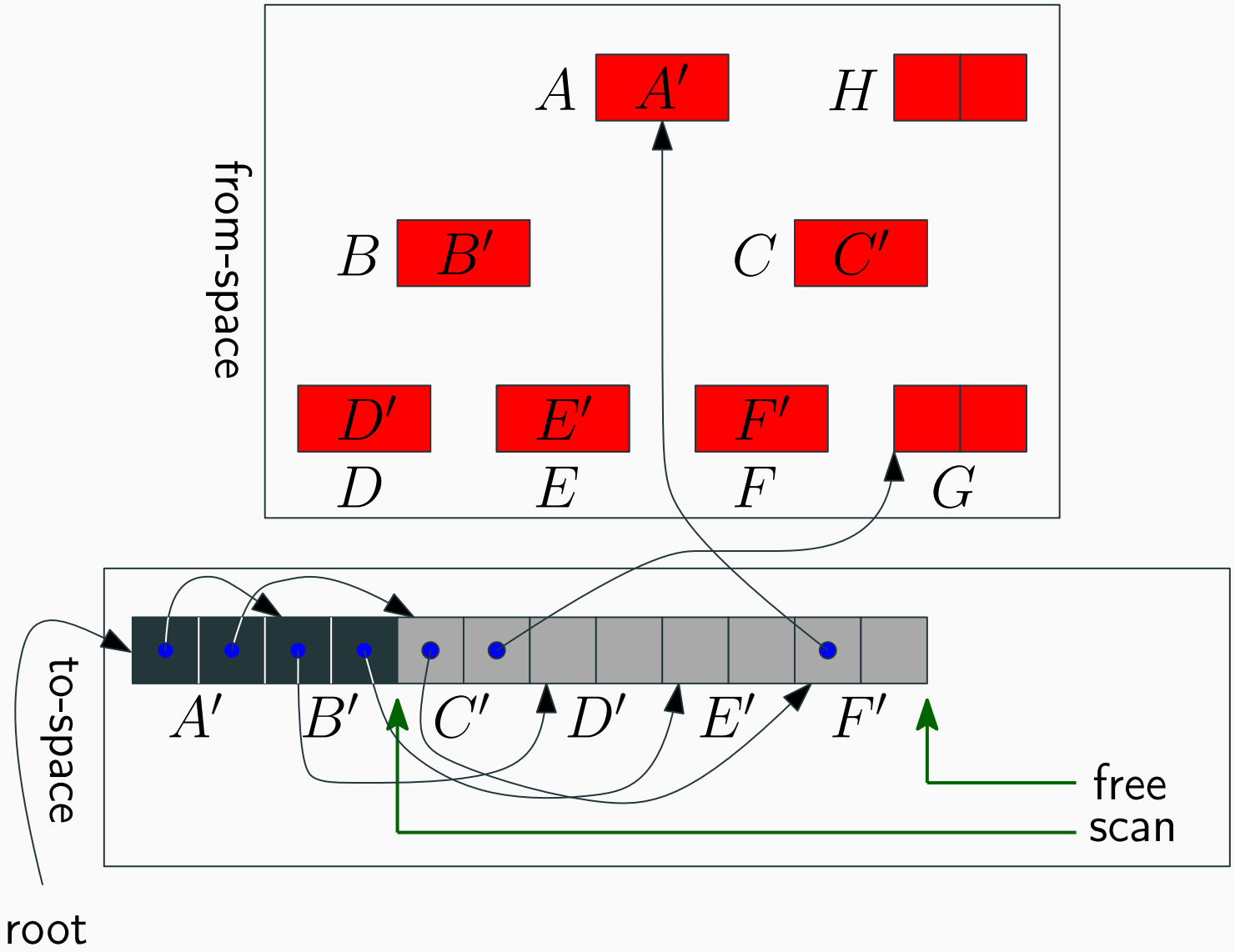
Cheney's Algorithm



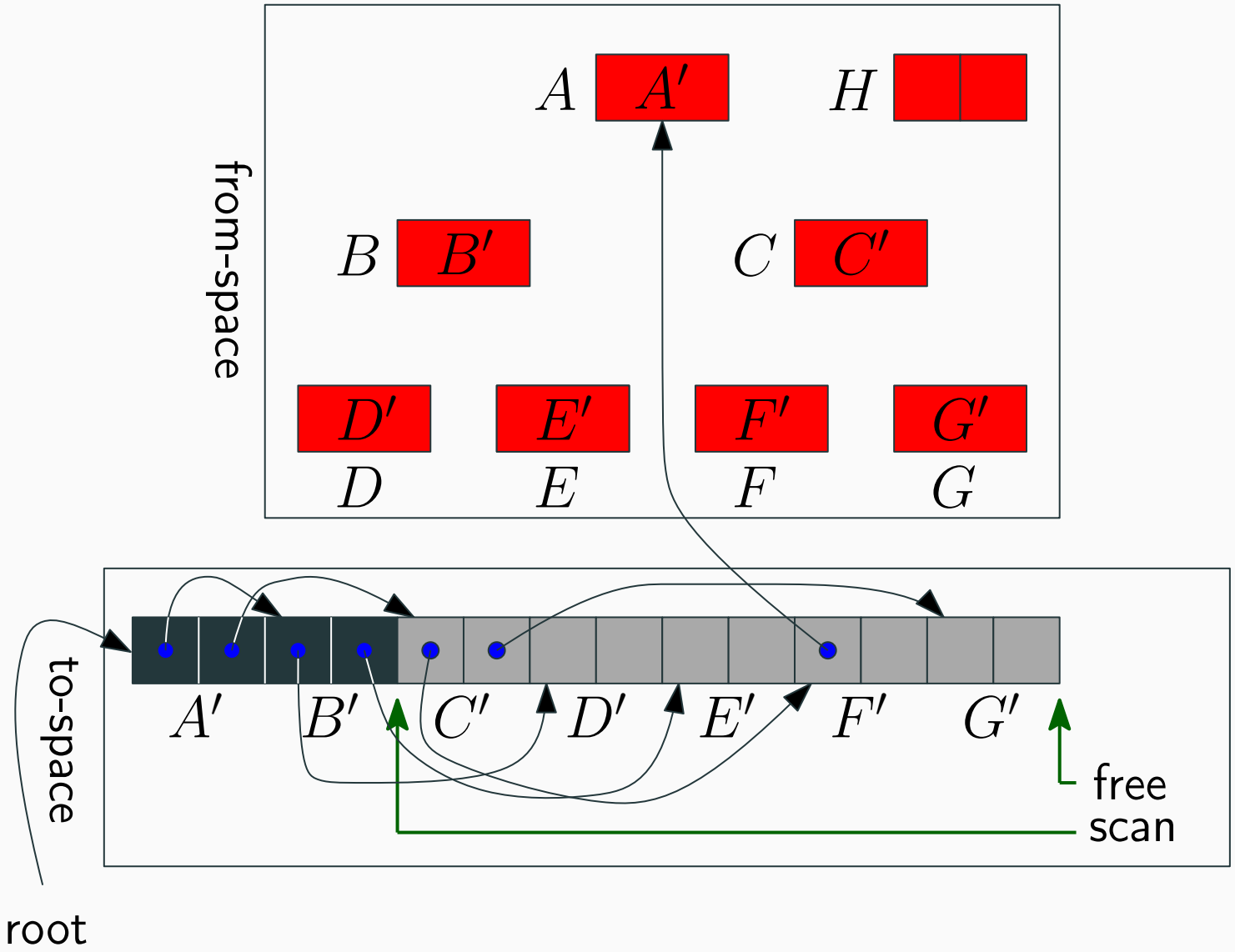
Cheney's Algorithm



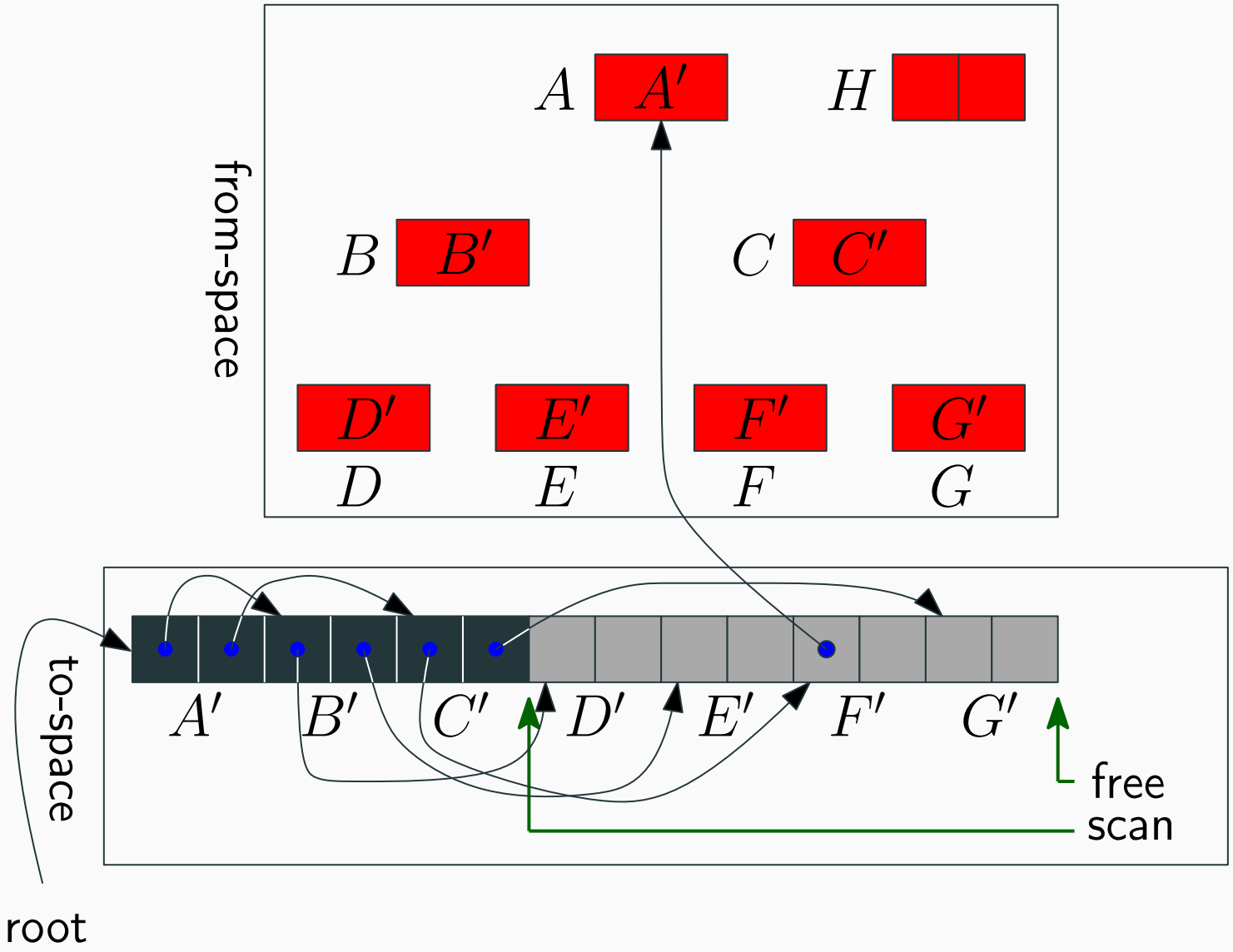
Cheney's Algorithm



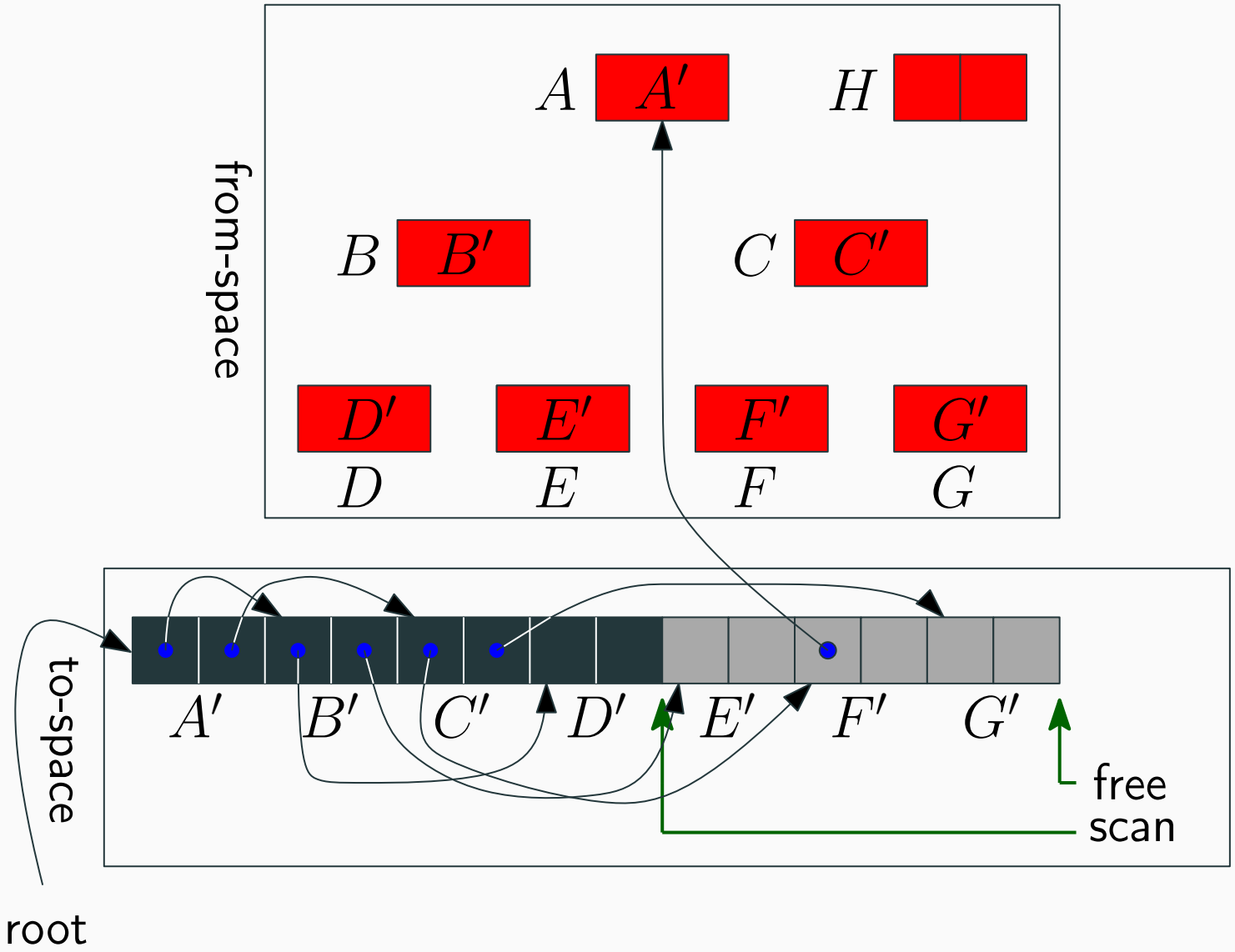
Cheney's Algorithm



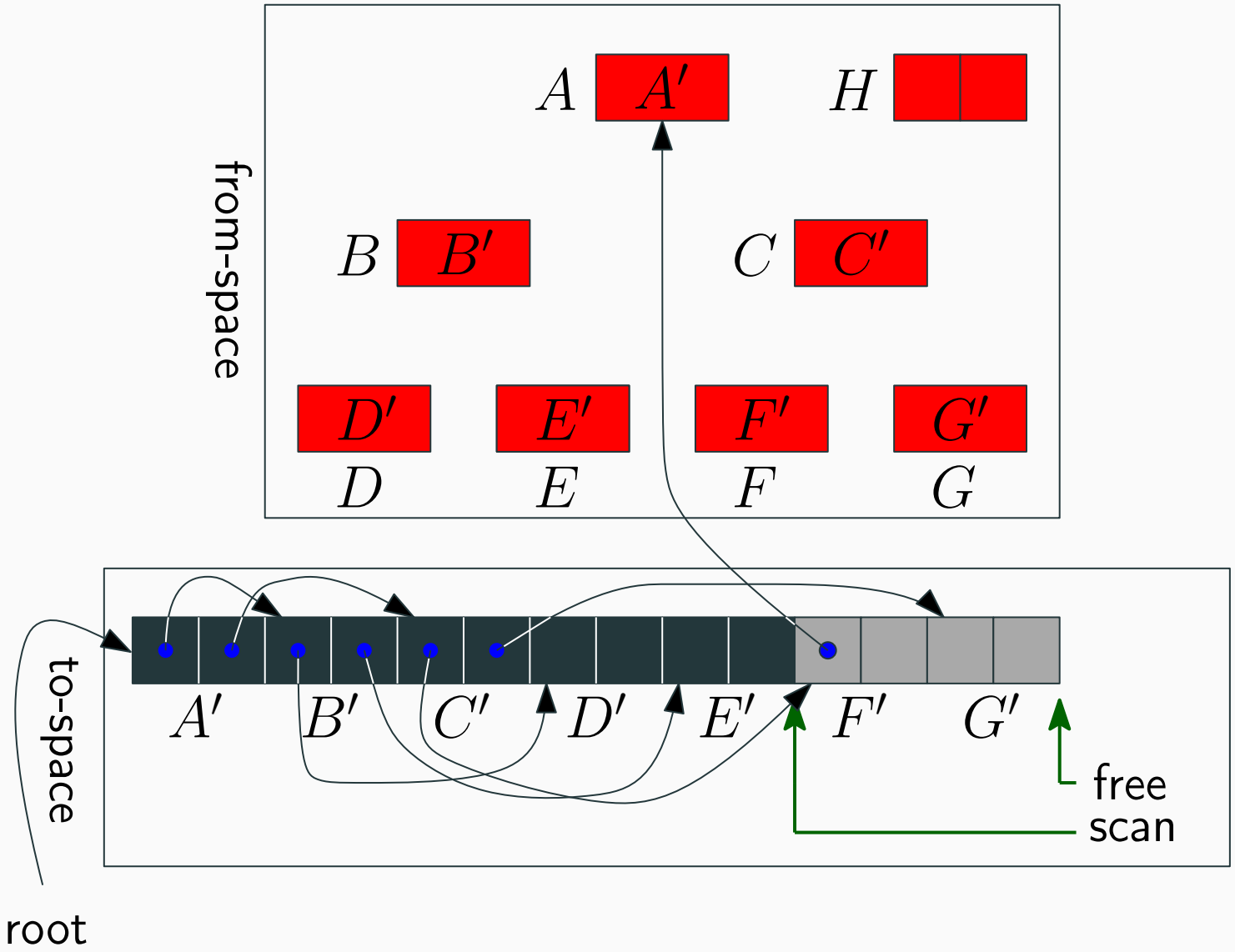
Cheney's Algorithm



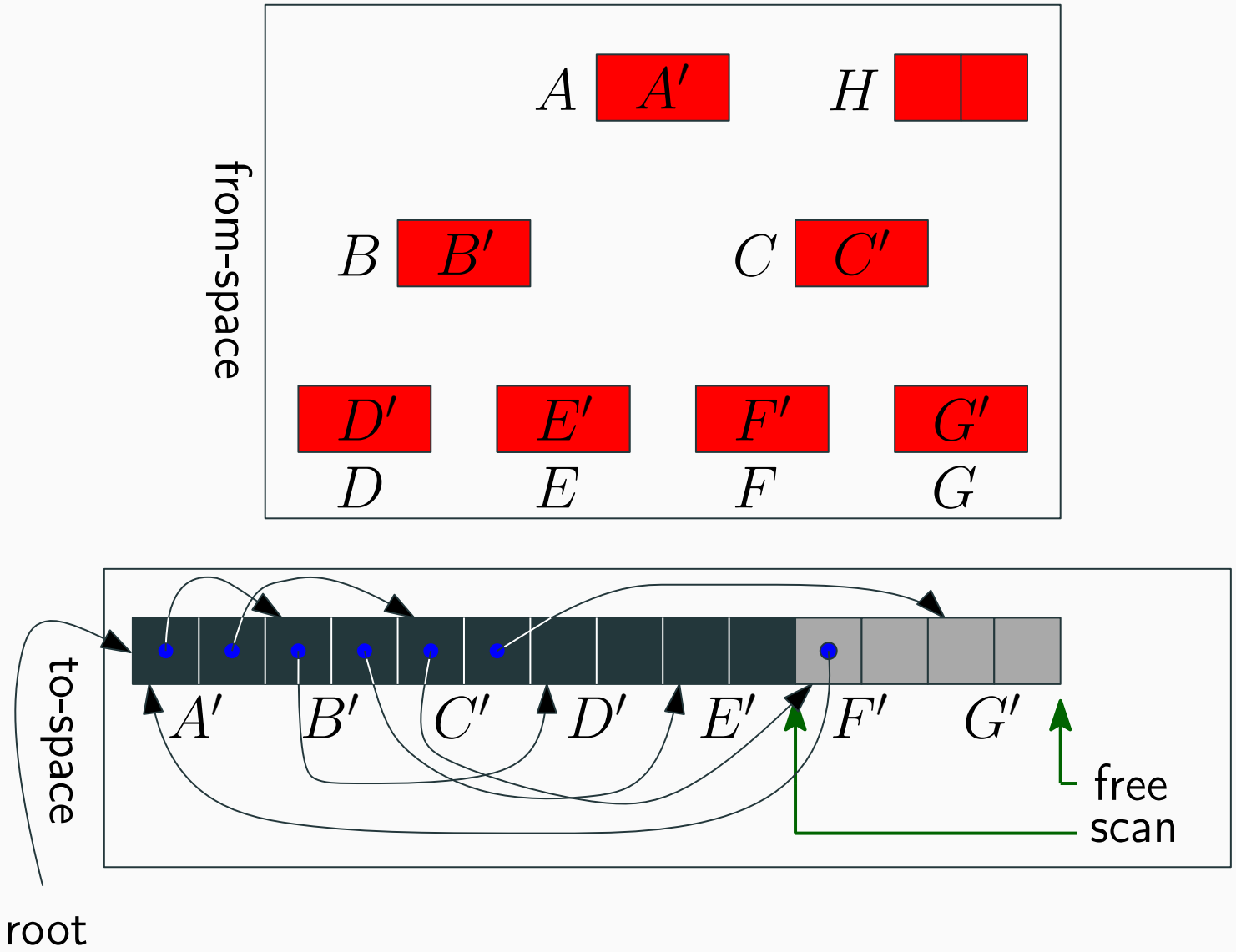
Cheney's Algorithm



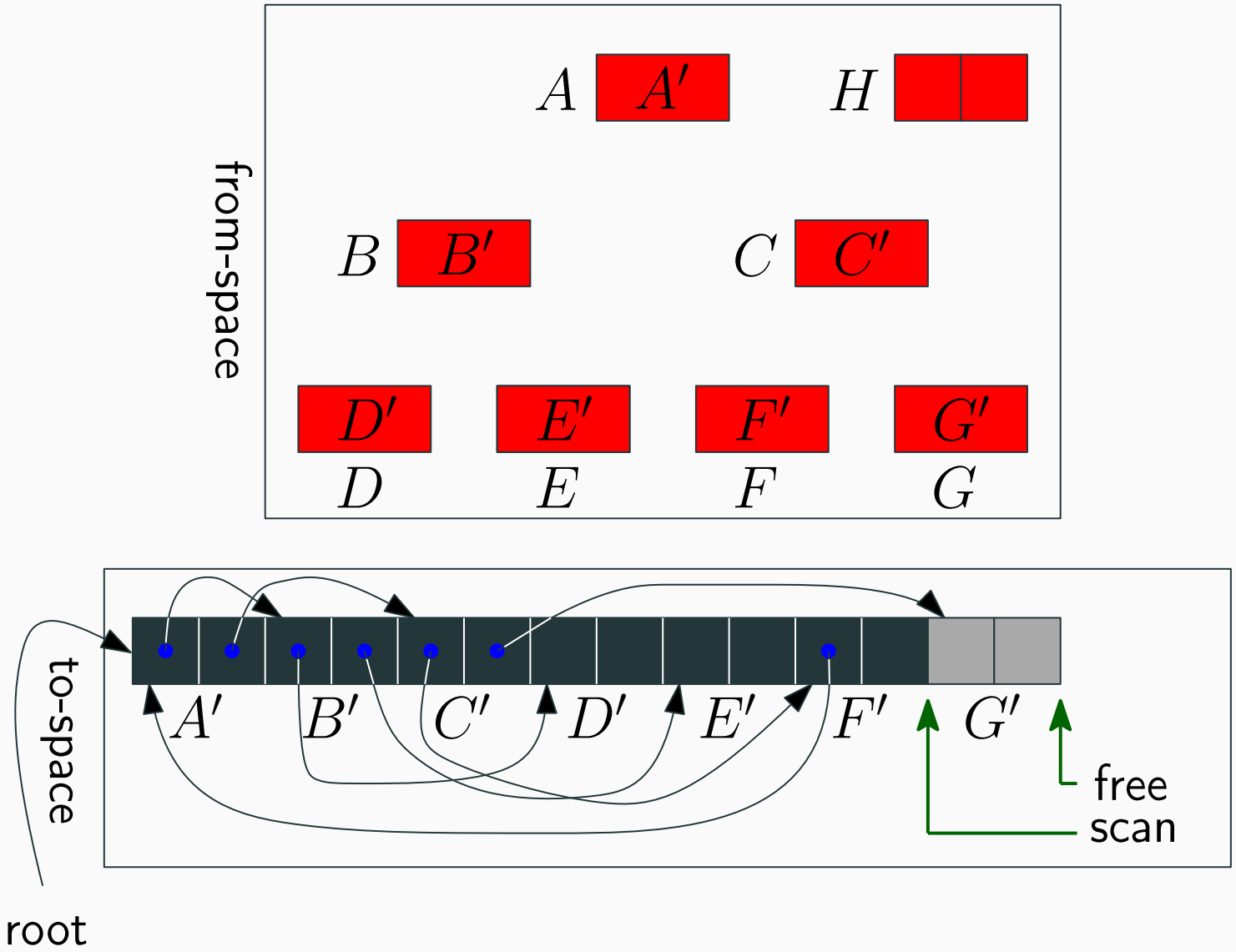
Cheney's Algorithm



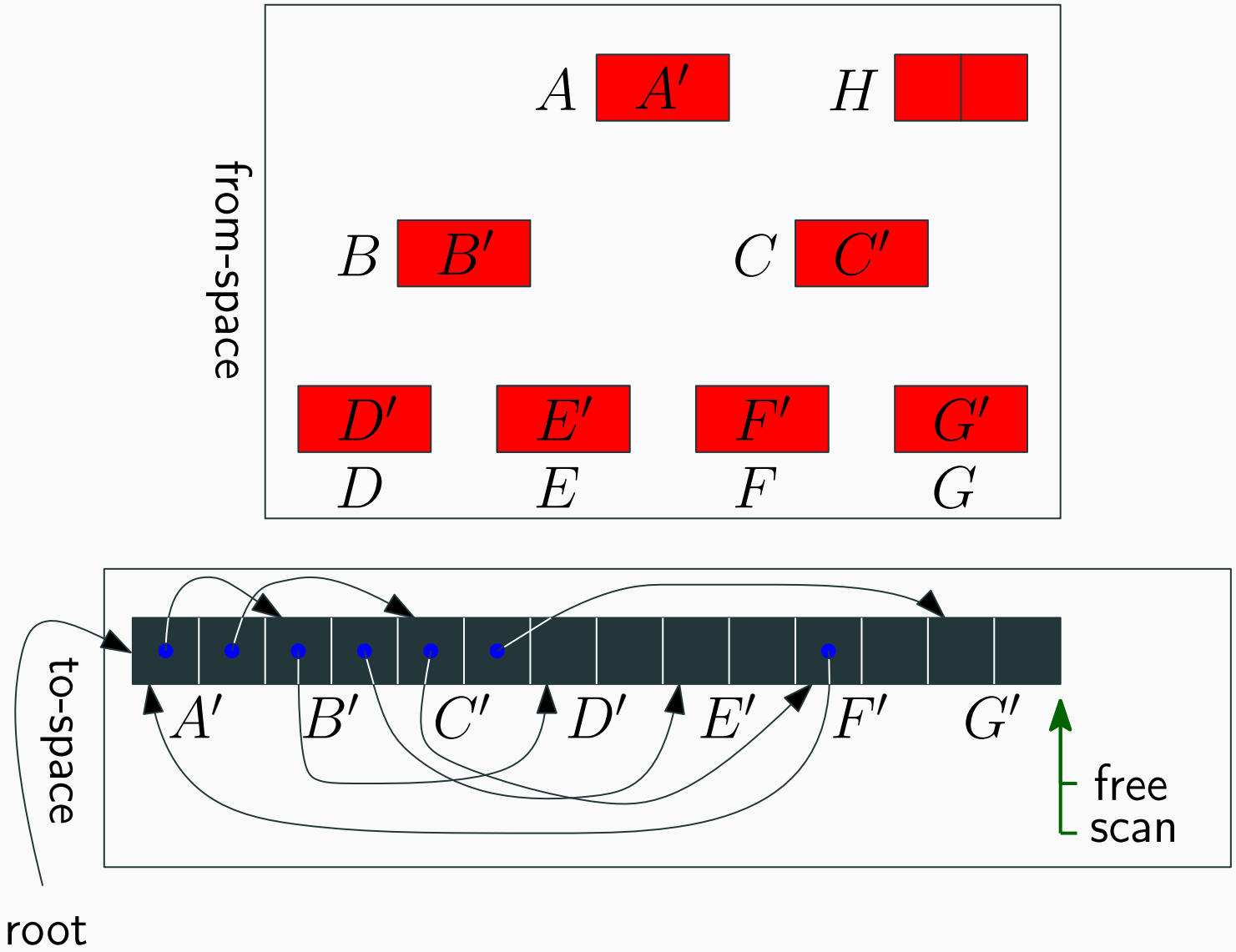
Cheney's Algorithm



Cheney's Algorithm



Cheney's Algorithm



Copying Garbage Collection

Pros

- Contiguous free memory space
- Object creation is cheap: bump allocation
- Only live objects are inspected

Cons

- Stop-the-world garbage collector
- Breath first traversal may mix locality patterns
- All pointers are updated
- Half of the heap is useless
- Objects with long life span are copied over and over again

Quantitative Aspects

- Amount of live data: L
- Size of the heap: H
- Ratio of heap size to live data: $\gamma = \frac{H}{L}$
- Live ratio $\frac{1}{\gamma}$: fraction of the heap occupied by live objects
- γ too small:
  - too few allocation between collections
  - overhead of collection can get very high

Comparison

	Ref. counting	Mark-Sweep	Mark-Compact	Copying
Defragmentation			✓	✓
Time Overhead	$O(\text{ptr manip})$	$O(H)$	$O(H) + O(L)$	$O(L)$
Space Required	L	L	L	$2 \times L$
Passes over memory	-	2	2-4	1

Correct Heap Size

- How should the heap grow?
- Start out with a fairly small heap, enlarge it in response to the growth of live data
- After a collection, if γ is smaller than some criterion, the collector increases H by asking the operating system for more memory
- Simple rule of thumb: maintain a roughly constant γ
- Mark-and-sweep collectors perform very well when $\gamma > 2$
- Copying collectors do better with $\gamma > 3$

Summary

This Lecture

- The last lecture of PLC!

Next Lecture

- There are no more lectures ☺