An efficient machine-independent procedure for garbage collection in various list structures

Schorr & Waite, CACM, 1967
Presented by Nick Sumner
31 Jan 2012

Background

3 main techniques at this point

- Manual memory management

Background

3 main techniques at this point

- Manual memory management
- Reference counting

Background

3 main techniques at this point

- Manual memory management
- Reference counting
  - Can't handle cycles
  - Onerous bookkeeping

Background

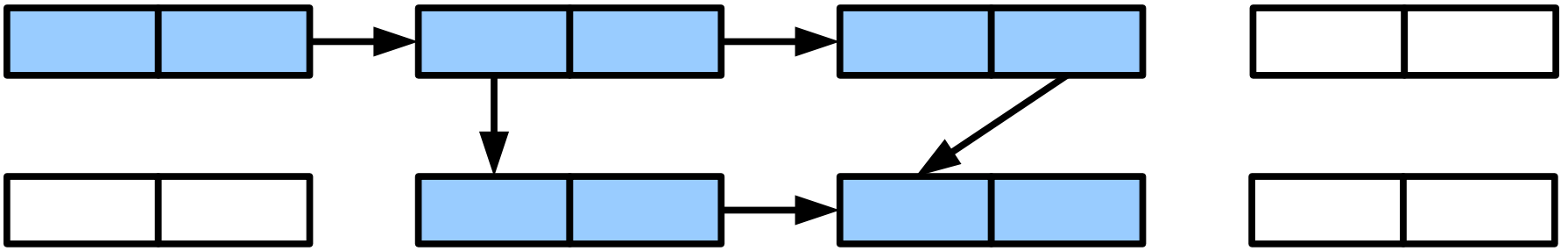
3 main techniques at this point

- Manual memory management
- Reference counting
- Tracing GC (mark and sweep)

Background

3 main techniques at this point

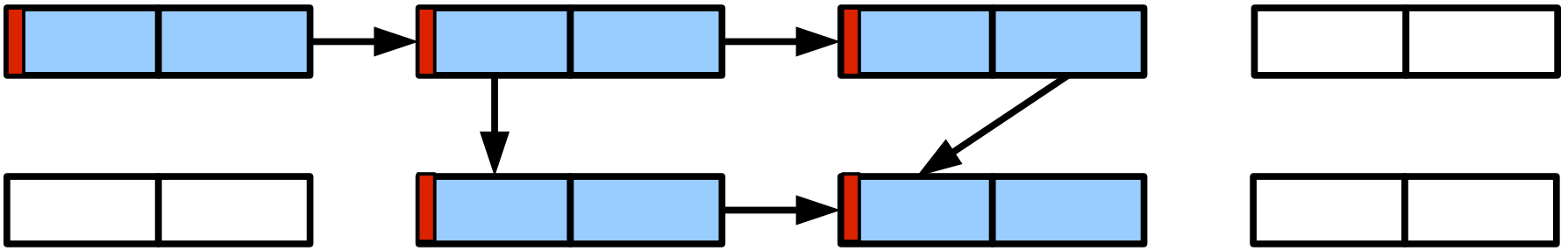
- Manual memory management
- Reference counting
- Tracing GC (mark and sweep)



Background

3 main techniques at this point

- Manual memory management
- Reference counting
- Tracing GC (mark and sweep)



Background

3 main techniques at this point

- Manual memory management
- Reference counting
- Tracing GC (mark and sweep)
  - Traversing the graph uses substantial space
  - List elements must smaller than 1 word

Background

3 main techniques at this point

- Manual memory management
- Reference counting
- Tracing GC (mark and sweep)

Improve mark and sweep GC

Background

3 main techniques at this point

- Manual memory management
- Reference counting
- Tracing GC (mark and sweep)

Improve mark and sweep GC

- No extra space to traverse
- Heterogeneous objects handled

Improving Mark and Sweep

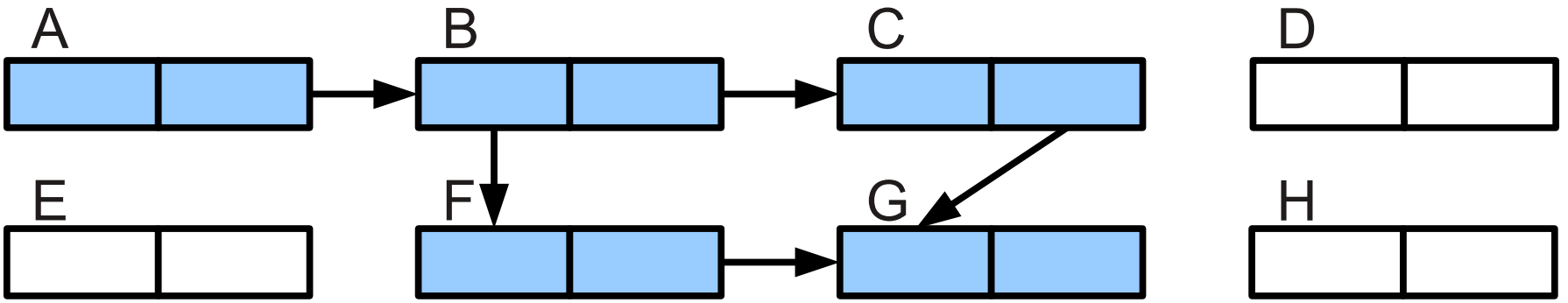
How can we reduce the space taken by M&S?

- Where does the space go?
- Maintaining current traversal state

Improving Mark and Sweep

How can we reduce the space taken by M&S?

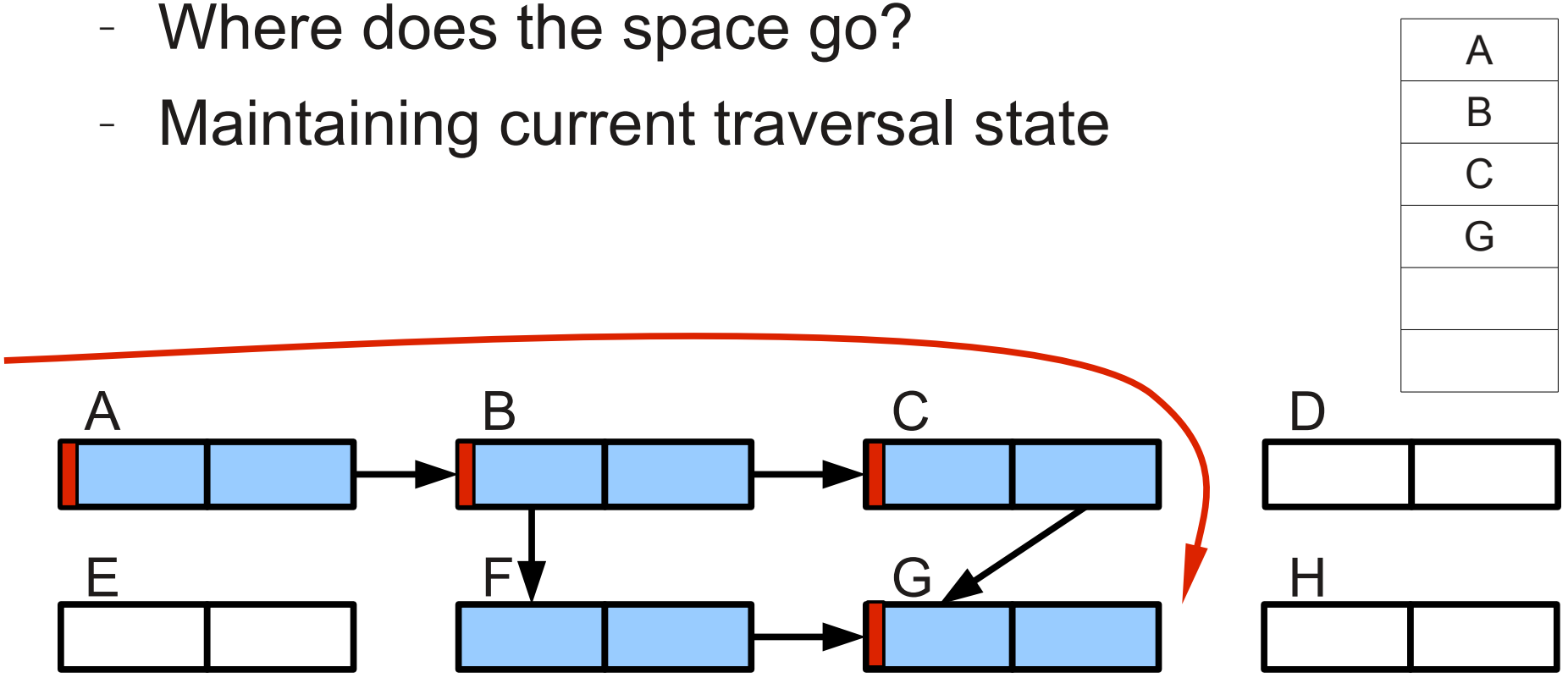
- Where does the space go?
- Maintaining current traversal state



Improving Mark and Sweep

How can we reduce the space taken by M&S?

- Where does the space go?
- Maintaining current traversal state



Improving Mark and Sweep

How can we reduce the space taken by M&S?

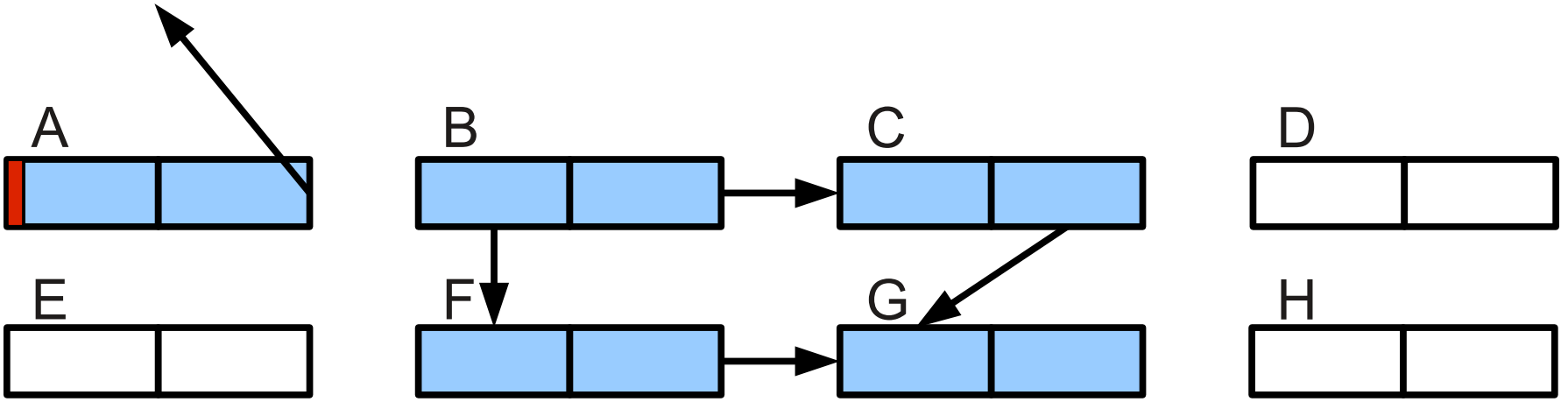
- Where does the space go?
- Maintaining current traversal state

Idea:

- Maintain the current traversal state by modifying the graph pointers!

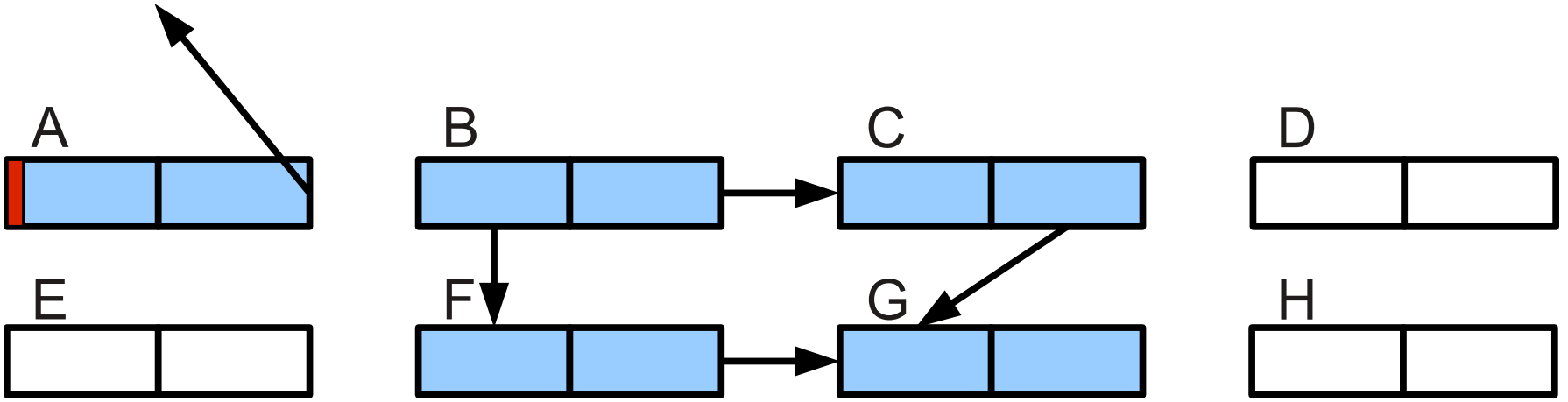
Better intuition from David Gries' lecture notes, 2006
Published, 1979

Improving Mark and Sweep

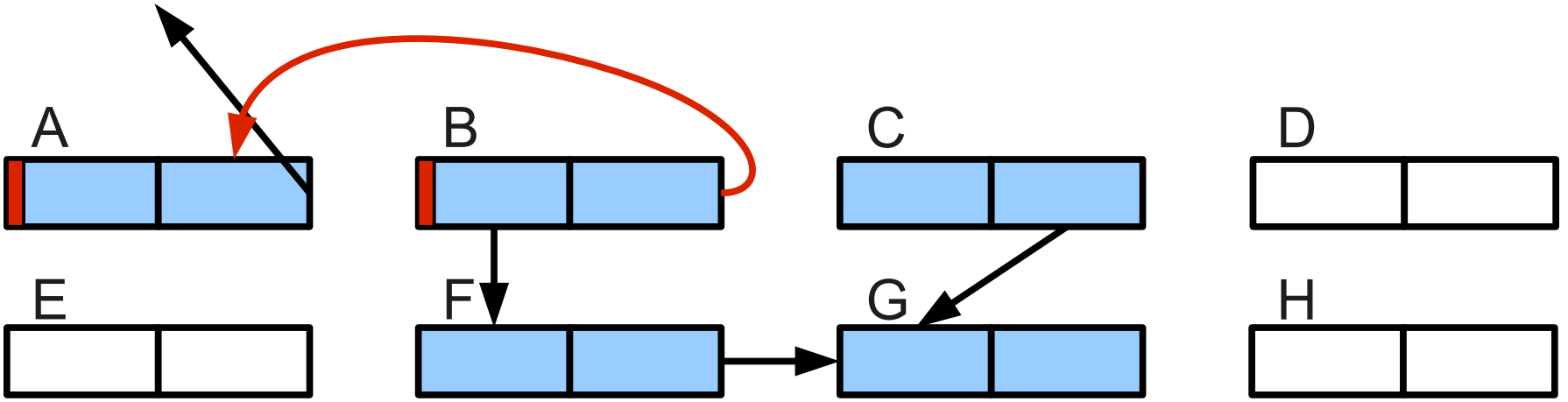


Improving Mark and Sweep

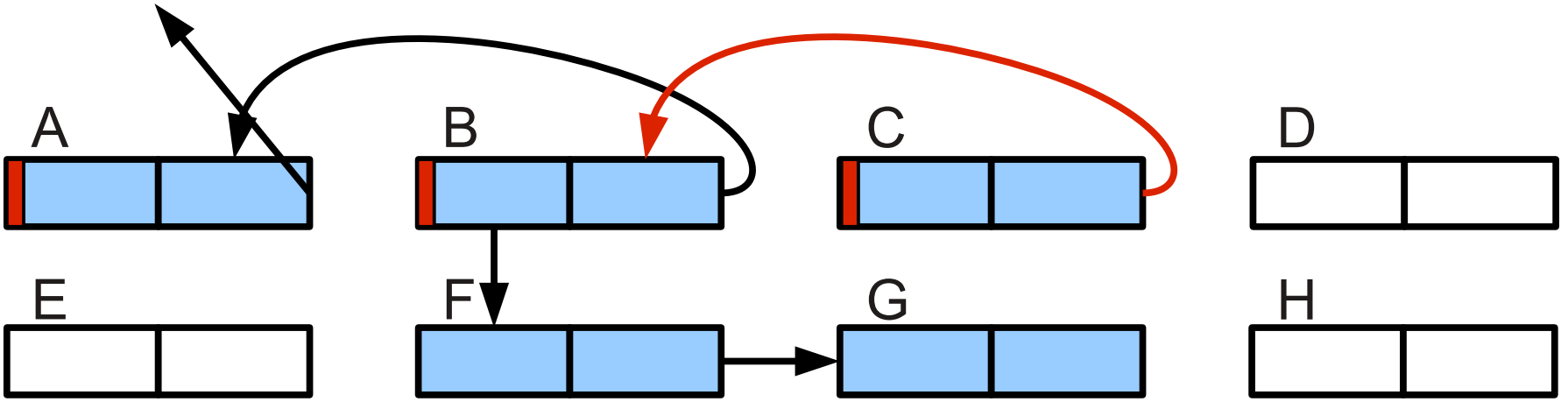
How can we get to B?



Improving Mark and Sweep

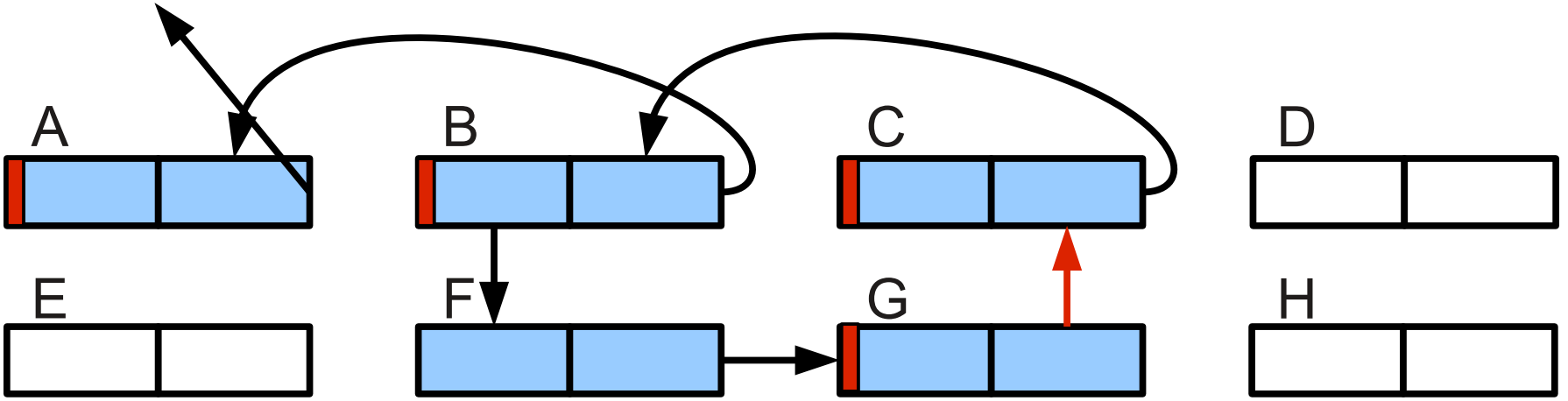


Improving Mark and Sweep

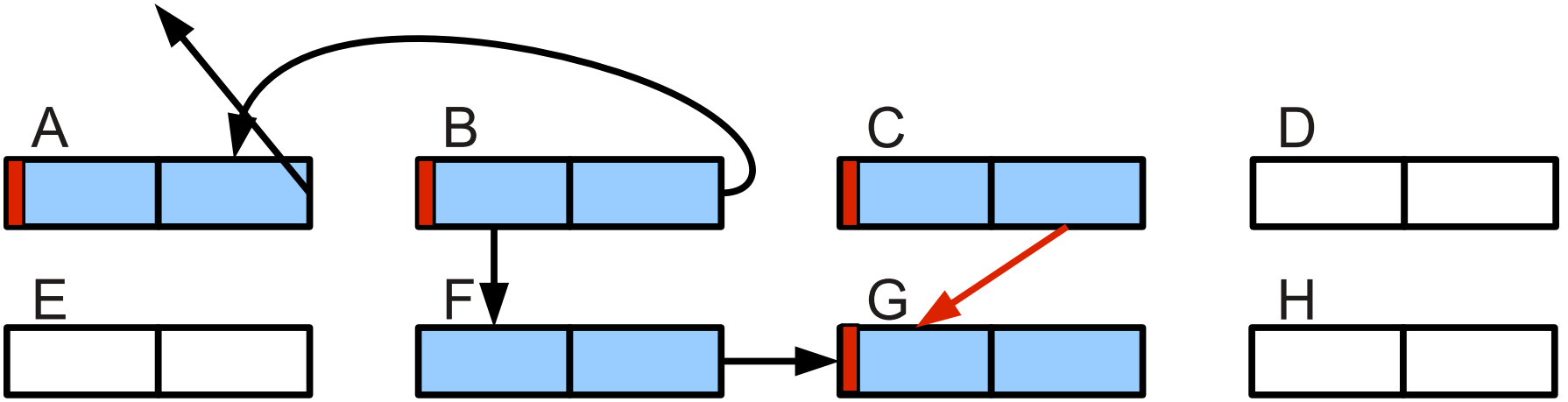


Improving Mark and Sweep

We've reached the end!
Return along pointers.

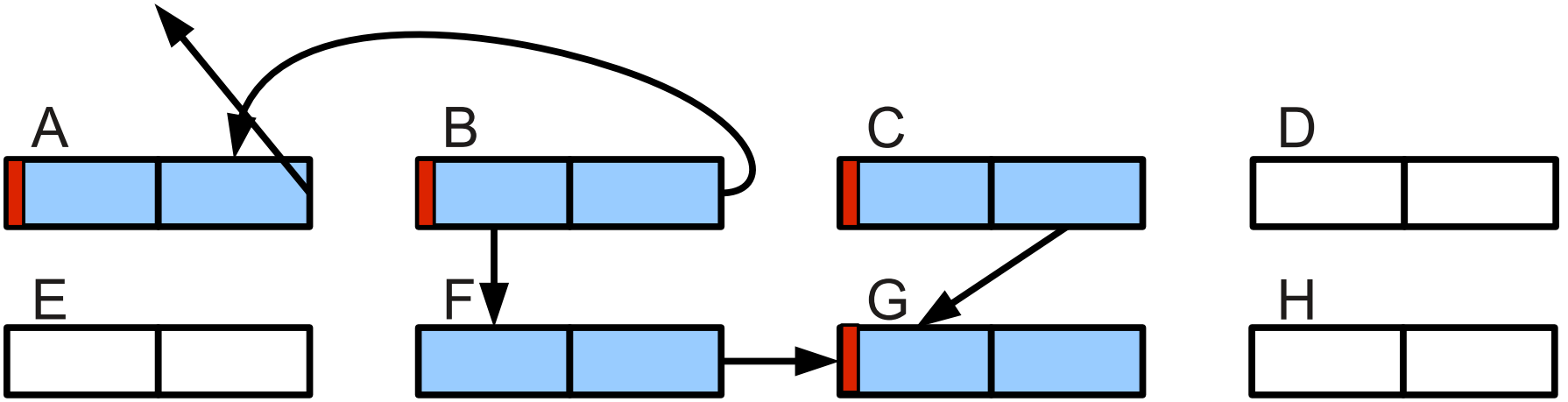


Improving Mark and Sweep

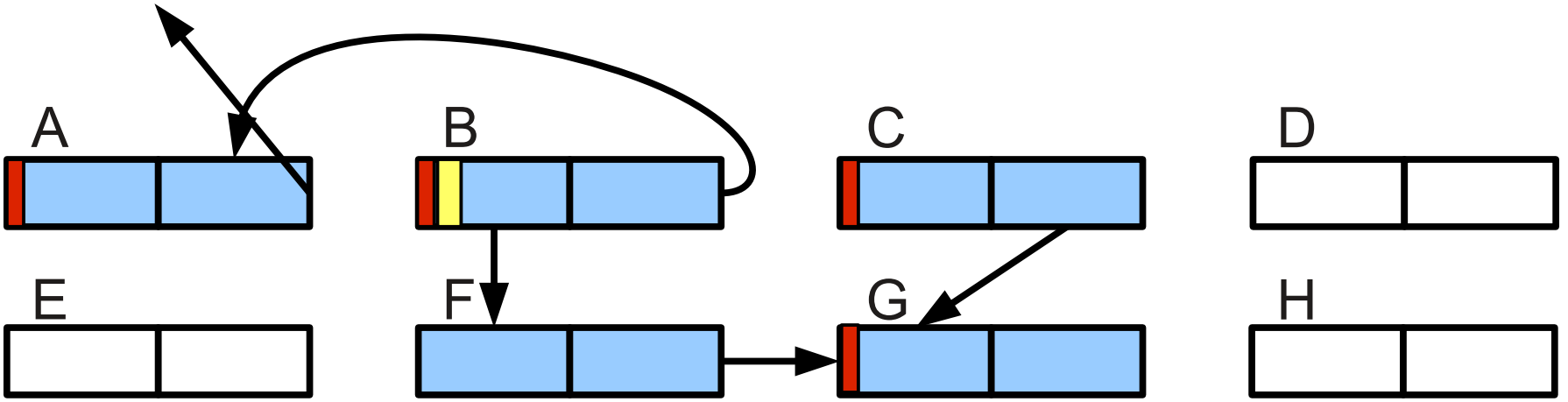


Improving Mark and Sweep

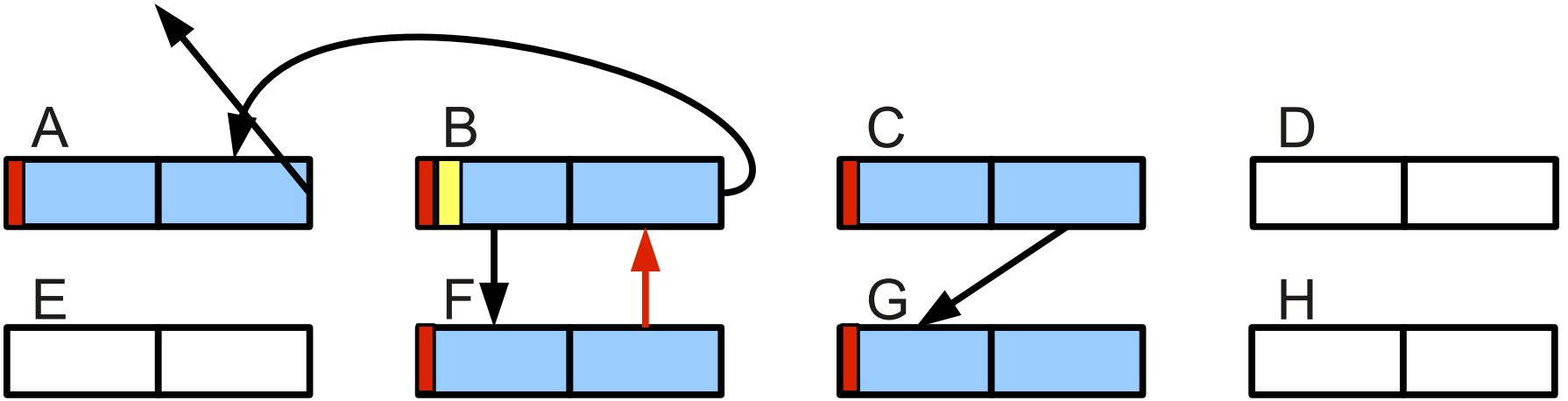
B holds a sublist.
We have to traverse it now!



Improving Mark and Sweep

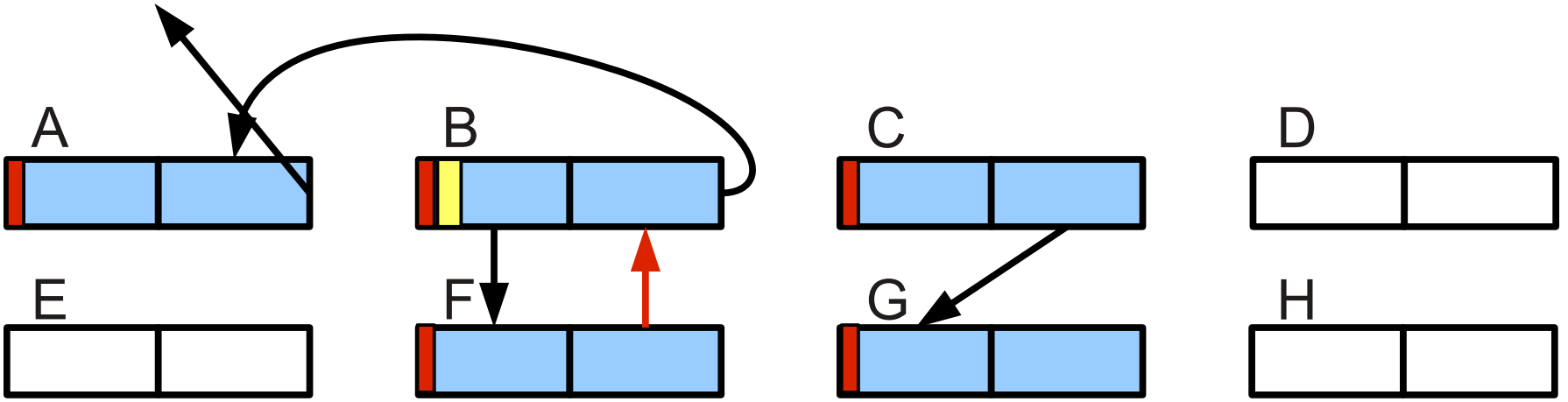


Improving Mark and Sweep

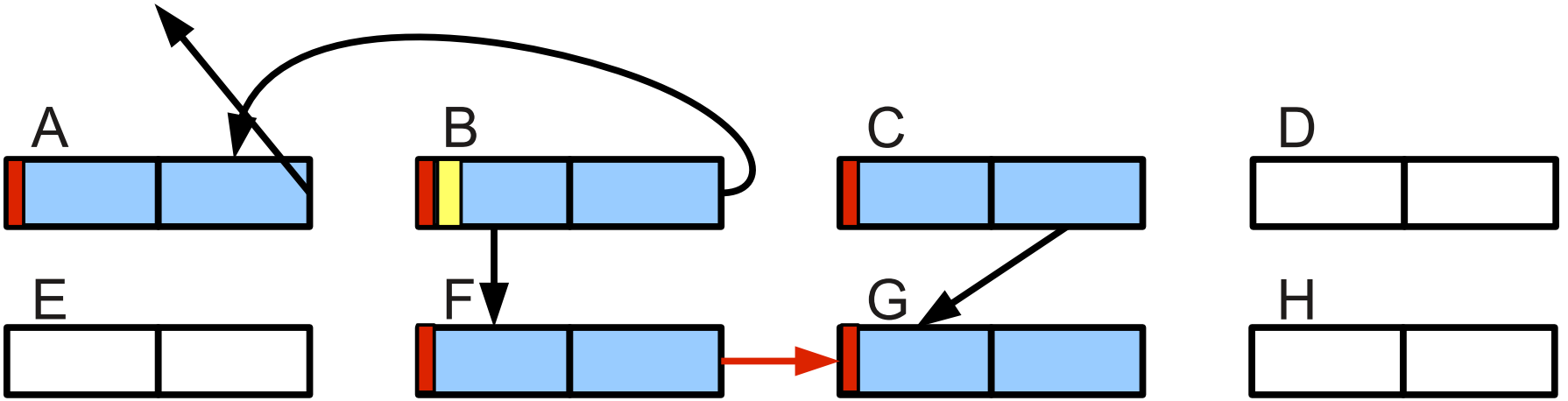


Improving Mark and Sweep

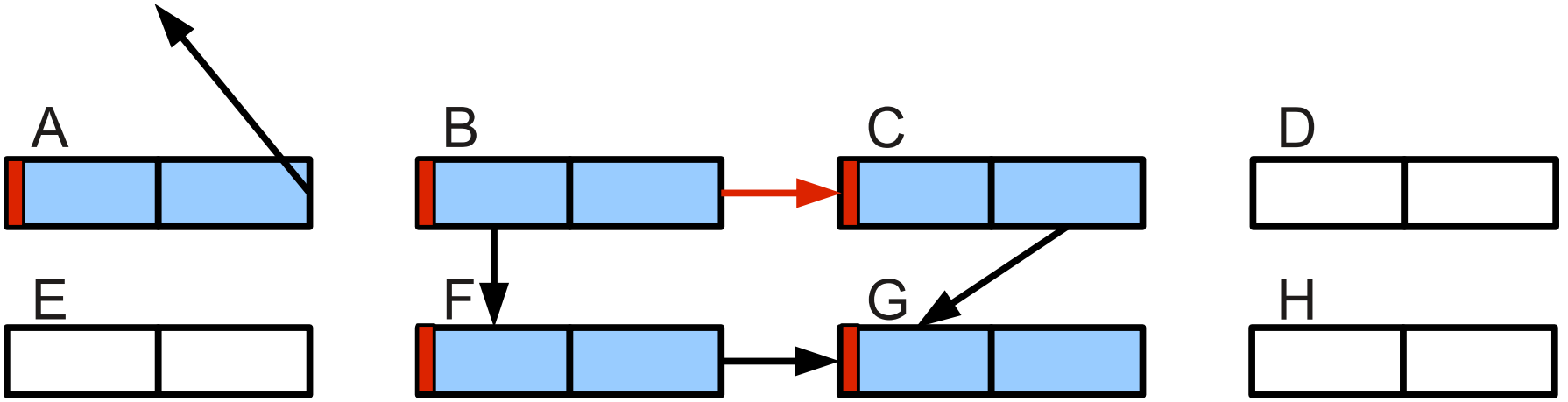
G has already been marked, so return.



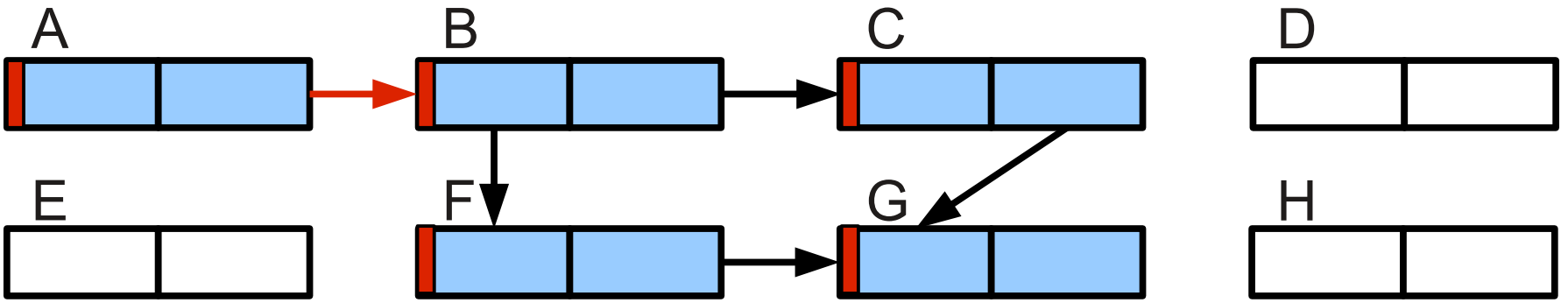
Improving Mark and Sweep



Improving Mark and Sweep

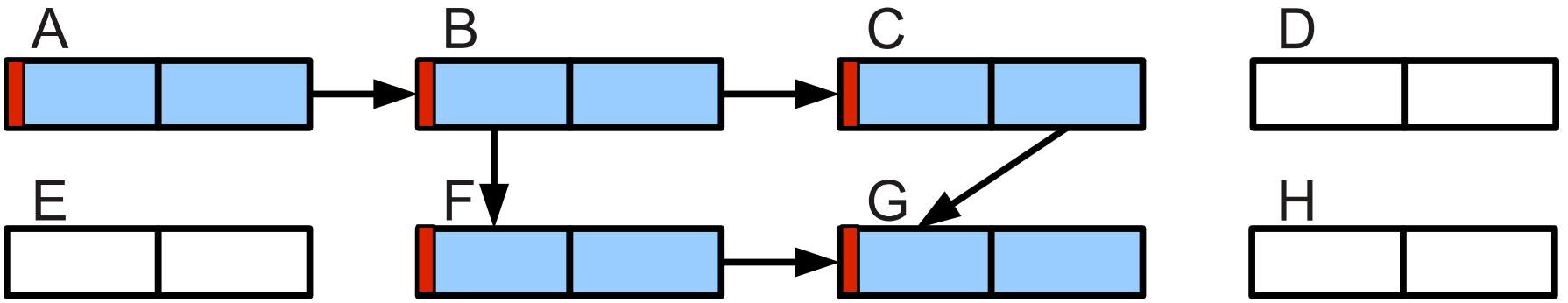


Improving Mark and Sweep



Improving Mark and Sweep

- Reverse pointers to implicitly store the 'stack' in the graph itself!



Evaluation

- Run on complete binary tree (20,000 nodes)
  - Schorr & Waite: 1.85 seconds
  - Wilkes: 2.75 seconds
  - DFS, stack size 48: 0.448 seconds

- Throughput? Real world scenarios?

Heterogeneous Lists

- Multiple words for an atom?
  - Use another prefix bit to identify
  - Store the number of words in the first word

- Full word atoms? (negative numbers)
  - Treat like multi-word atoms...

Questions

- How general is it, really?
  - Wikipedia: IBM 7094
  - Are all of the mark bits necessary?

Questions

- How general is it, really?
  - Wikipedia: IBM 7094
  - Are all of the mark bits necessary?
- What if you have more than 2 outgoing edges per node?
  - Implications in other designs?

Questions

- How general is it, really?
  - Wikipedia: IBM 7094
  - Are all of the mark bits necessary?
- What if you have more than 2 outgoing edges per node?
  - Implications in other designs?
- How appropriate is the evaluation?

Myths and realities: The performance impact of garbage collection (3.0-3.1)

Blackburn, Cheng, & McKinley
SIGMETRICS, 2004

Background

- Implemented several MM *policies*
  - allocation scheme + collection scheme
- Allocation
  - Contiguous (bump pointers)
  - Free list (size segregated)
- Collection
  - Tracing
  - Reference counting
- Whole heap v. Generational

Contrasting GC Counterparts

	Original	Generational
SemiSpace (copying)	Contiguous + tracing Use two regions, allocate in R1 Copy to R2 when full. O(Live objects)	
MarkSweep	Free list + tracing Mark and *lazily collect* on allocation O(Live objects)	
RefCount	Free list + reference counting Deferred (*coalesced*) counting Trial deletion for cycles O(Dead objects) + mutator load	

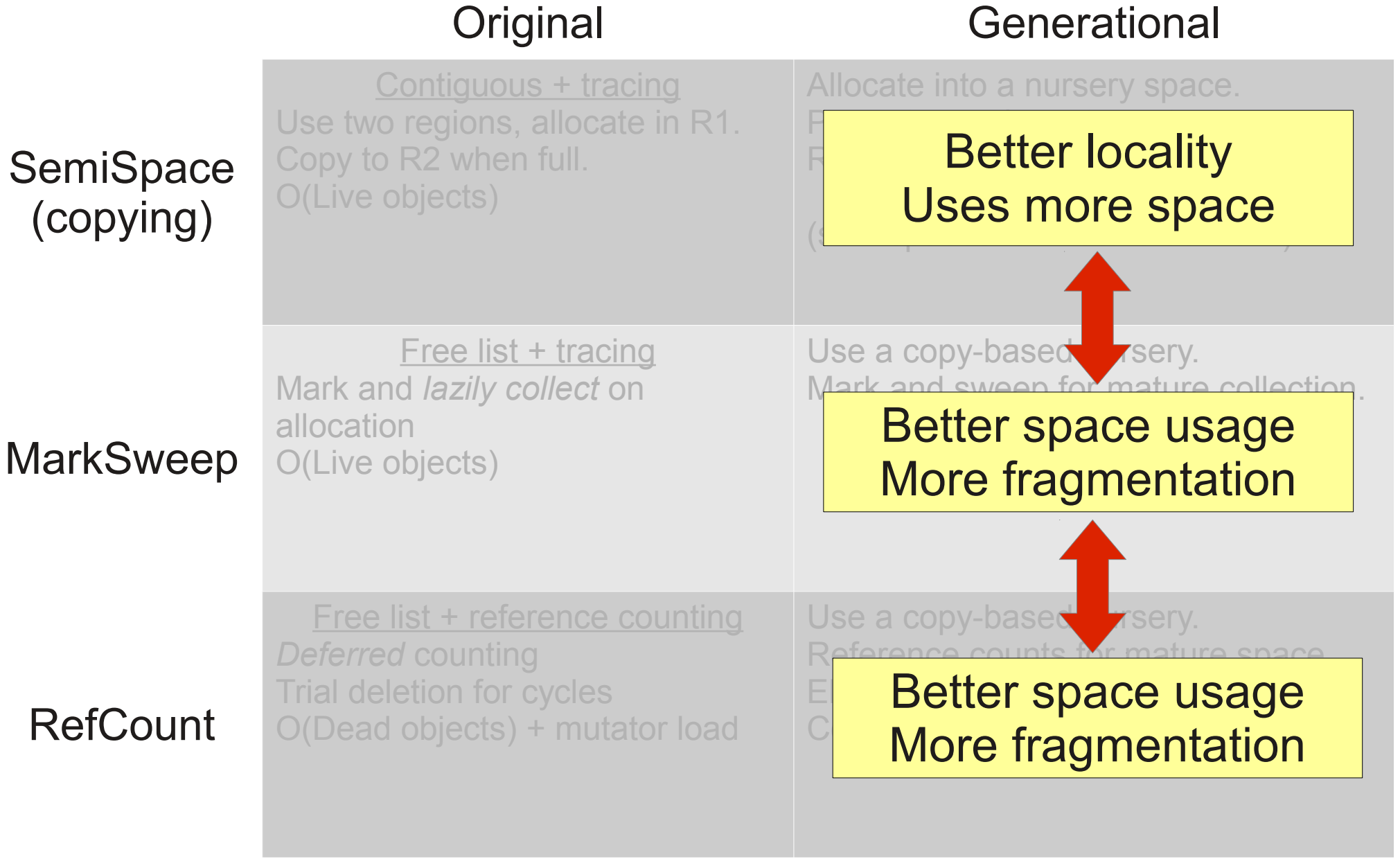
Contrasting GC Counterparts

	Original	Generational
SemiSpace (copying)	Contiguous + tracing Use two regions, allocate in R1 Copy to R2 when full. O(Live objects)	Repeatedly copies long lived objects. Doubles required space
MarkSweep	Free list + tracing Mark and *lazily collect* on allocation O(Live objects)	Repeatedly traverses long lived objects.
RefCount	Free list + reference counting Deferred (*coalesced*) counting Trial deletion for cycles O(Dead objects?) + mutator load	High mutator load.

Contrasting GC Counterparts

	Original	Generational
SemiSpace (copying)	Contiguous + tracing Use two regions, allocate in R1. Copy to R2 when full. O(Live objects)	Allocate into a nursery space. Promote survivors to mature space. Reclaim mature only as necessary. (semispaces for mature as well?)
MarkSweep	Free list + tracing Mark and *lazily collect* on allocation O(Live objects)	Use a copy-based nursery. Mark and sweep for mature collection.
RefCount	Free list + reference counting Deferred (*coalesced*) counting Trial deletion for cycles O(Dead objects) + mutator load	Use a copy-based nursery. Reference counts for mature space. Eliminates counting for young objects. Collects entire mature space.

Contrasting GC Counterparts



Questions

- What are the trade-offs between free lists & contiguous allocation?
  - Are they preserved when generational GC is used?

Some questions inspired by remainder of paper

Questions

- What are the trade-offs between free lists & contiguous allocation?
  - Are they preserved when generational GC is used?
- With the same nursery, GenCopy and GenMS are similar. What might this mean?

Some questions inspired by remainder of paper

Questions

- What are the trade-offs between free lists & contiguous allocation?
  - Are they preserved when generational GC is used?
- With the same nursery, GenCopy and GenMS are similar. What might this mean?
- What might the pathological cases for the different techniques be?
  - Are they common? Reducible?

Some questions inspired by remainder of paper