Heaps and Garbage

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2016

Mark-And-Sweep

1 Mark Bit (MB) for each object, initially 0

A: number of alive objects; N: all objects on heap

Pass 1 (Mark) – O(A)

- Traverse graph, set MB to 1 for visited node

Pass 2 (Sweep) – O(N)

- Traverse entire heap, collect objects whose MB=0; otherwise, set MB to 0 (prepare for the next collection)

Mark-and-Compact

A: number of alive objects; N: all objects on heap

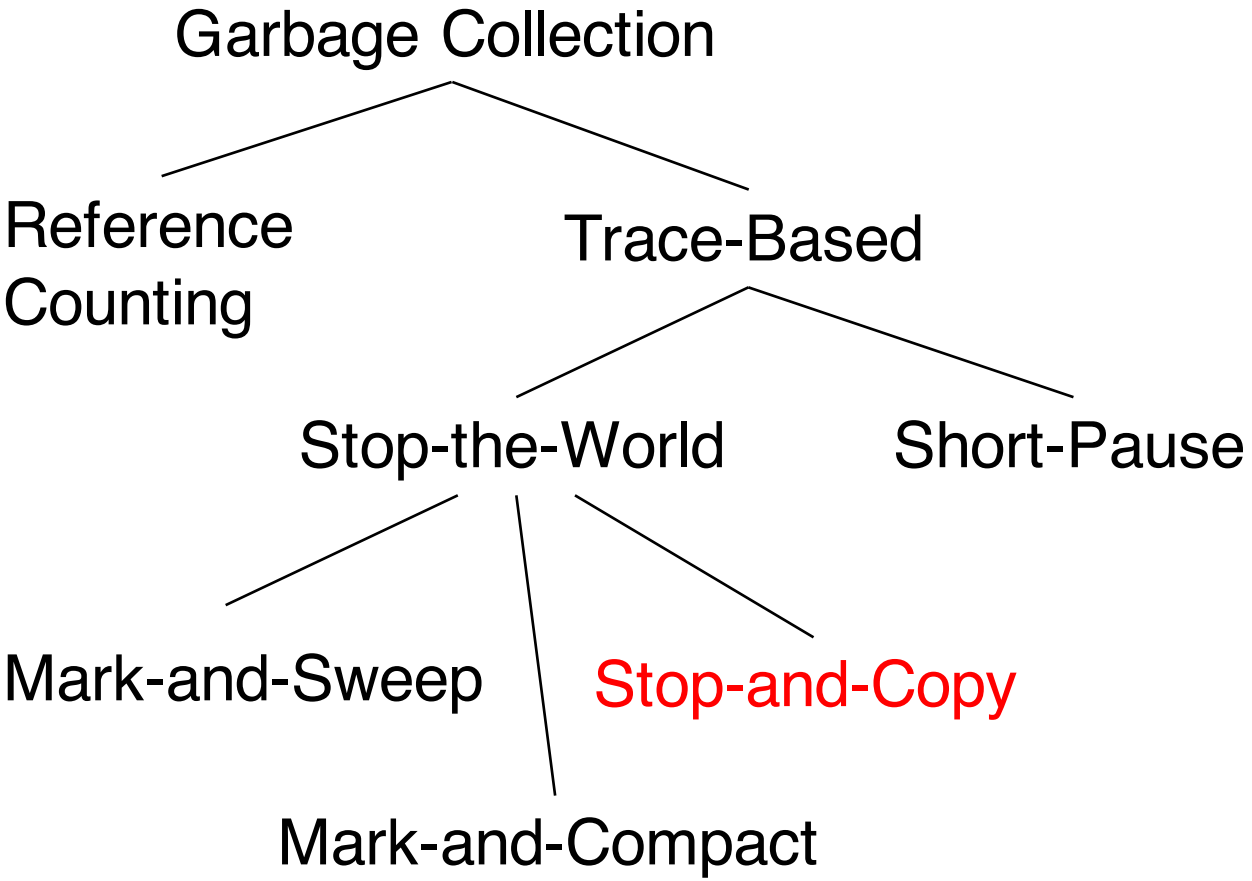
Pass 1 (Mark) – O(A)

- same as mark-and-sweep

Pass 2 (Compact) – O(N)

- Compute new locations for objects
- Move new objects to new location

Taxonomy


Garbage Collection
Reference Counting
Trace-Based
Stop-the-World
Short-Pause
Mark-and-Sweep
Stop-and-Copy
Mark-and-Compact

Stop-and-Copy (Copying Collector)

Time-Space Tradeoff:

- 2 Heaps: allocate space in one, copy to second when the other is full (half of the heap is unused)
- Only one pass over live objects is needed (much faster than previous algorithms)

MB bit is not needed

Stop-and-Copy

Triggered when heap in use is full, interrupt program

For each visited object

1. Copy it to the other heap (no fragmentation)
2. For the object in the old heap, keep a *forwarding pointer* to the new address
3. For each object it points to:

   a) if visited: update links (follow forwarding pointer)

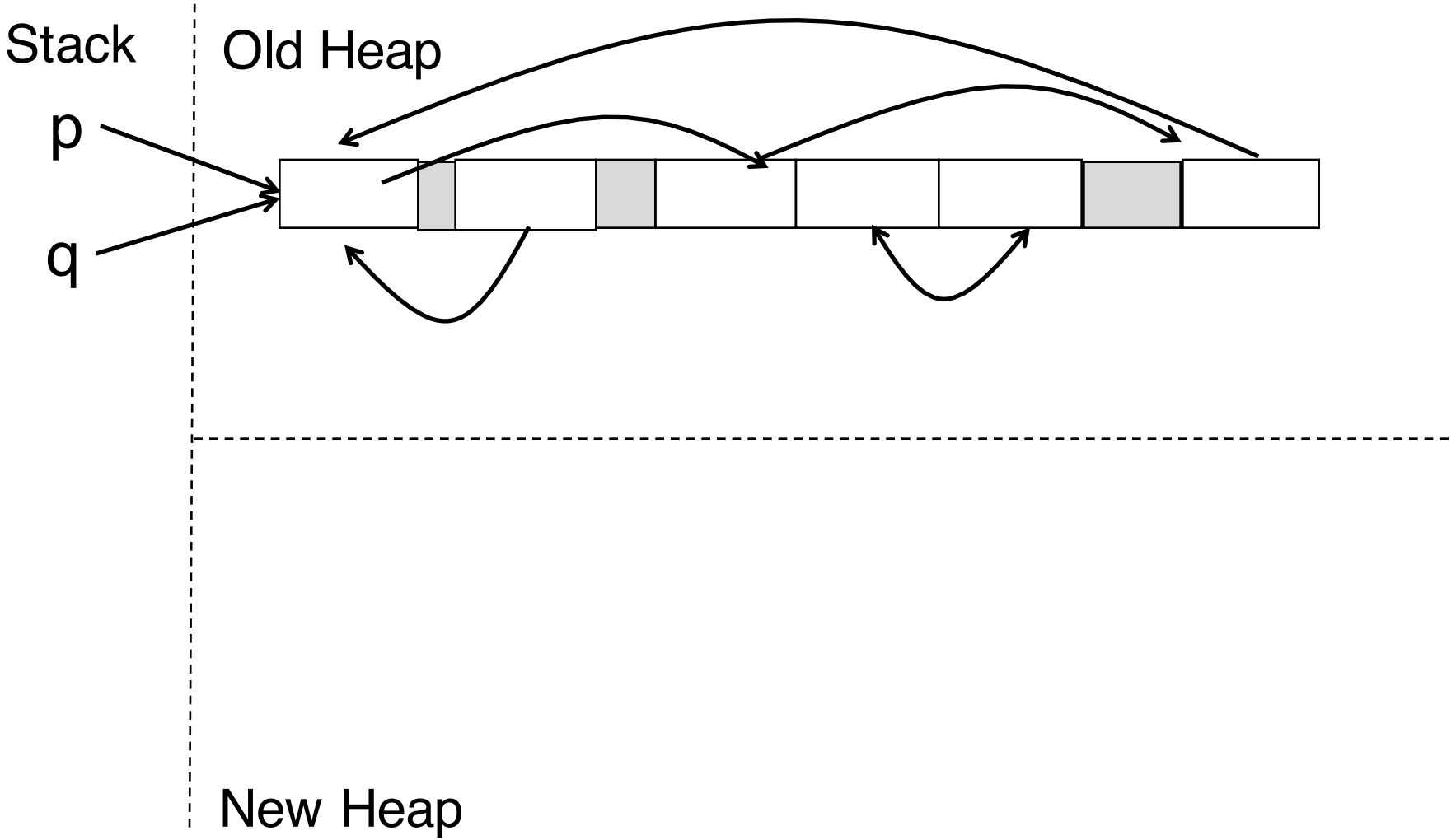
   b) otherwise, (recursively) visit object and update link

Retarget root references

Switch heaps

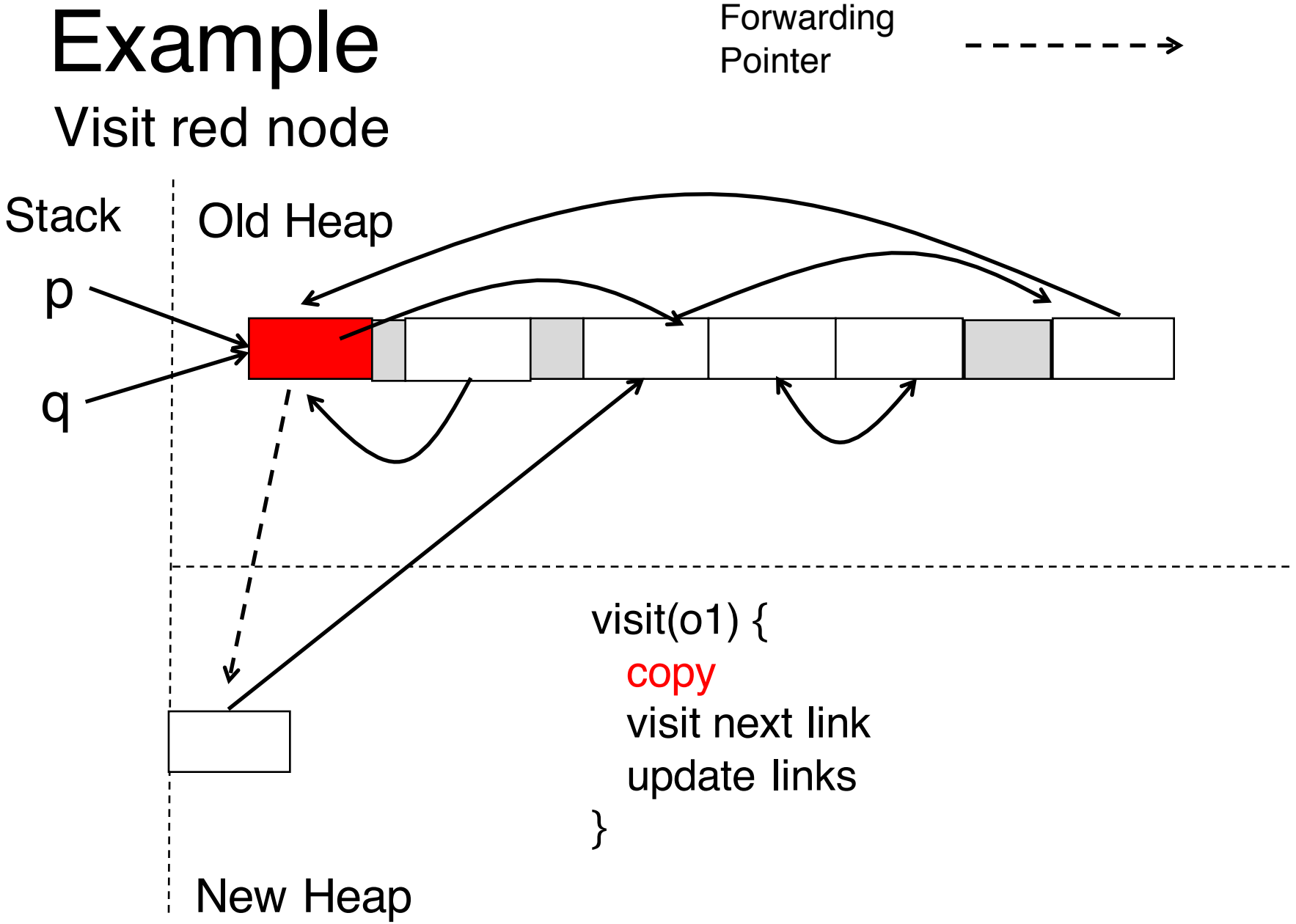
Example

Initial state



Example

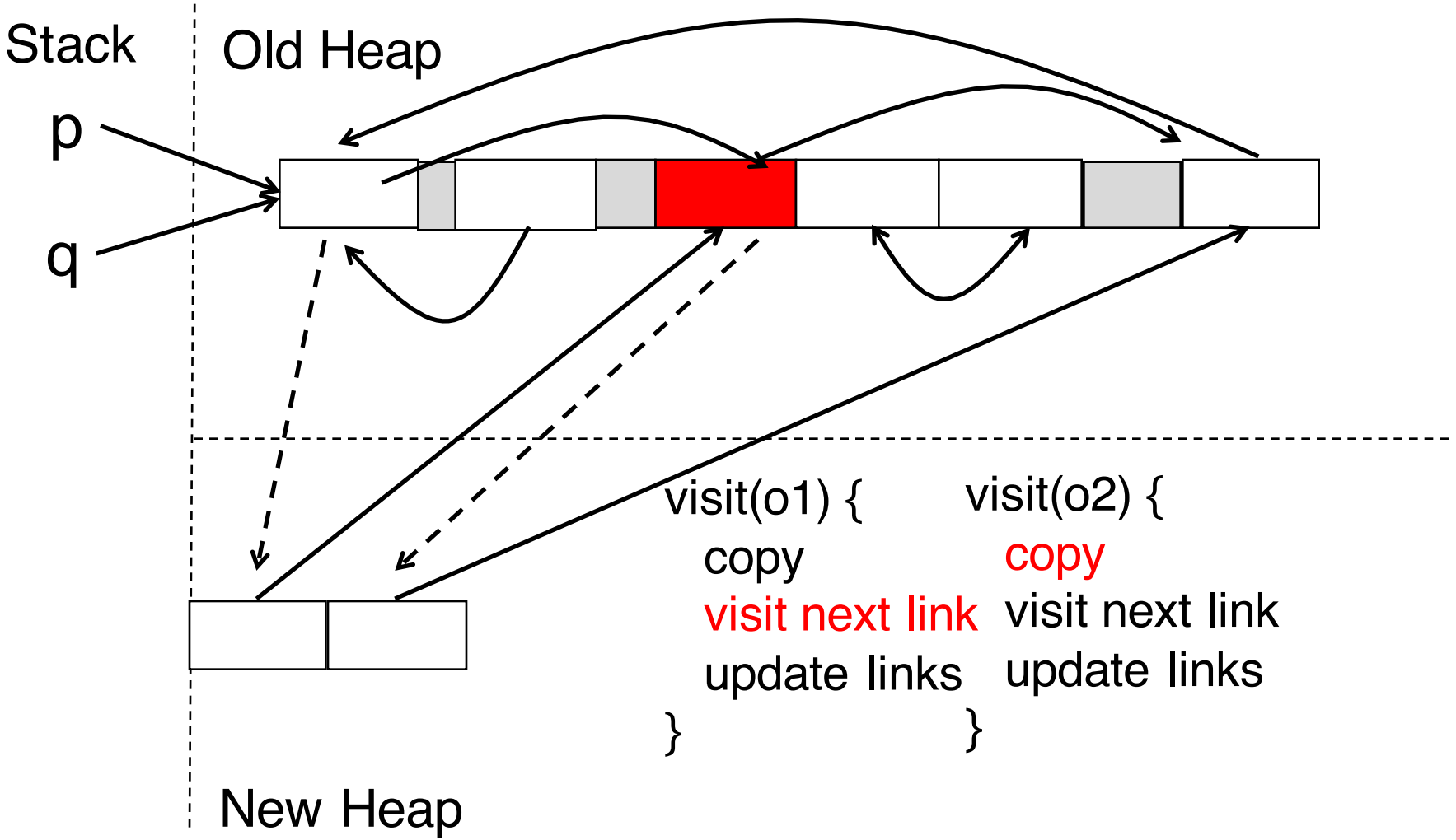
Visit red node



Example

Forwarding Pointer

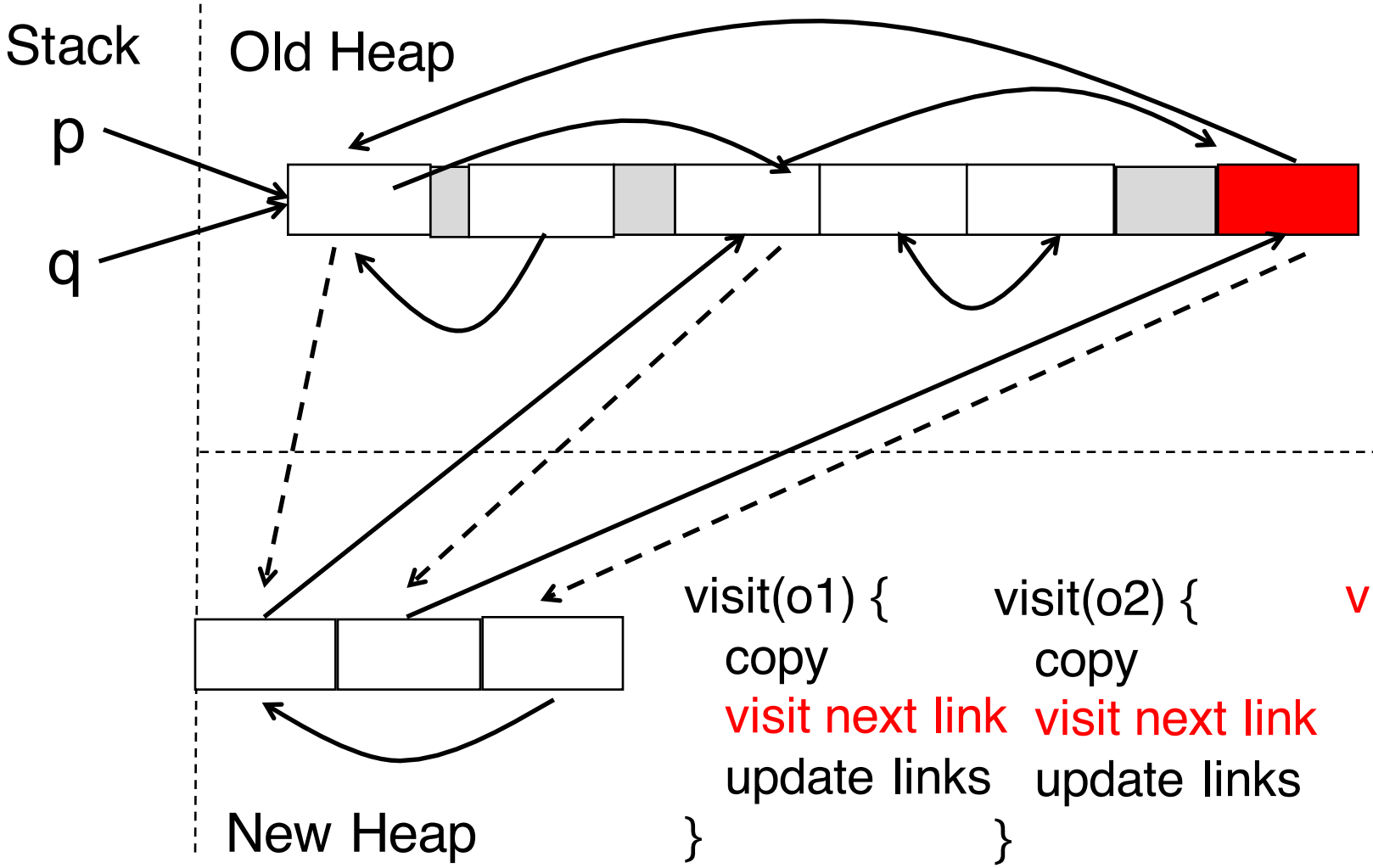
Visit red node (following link from new heap)



Example

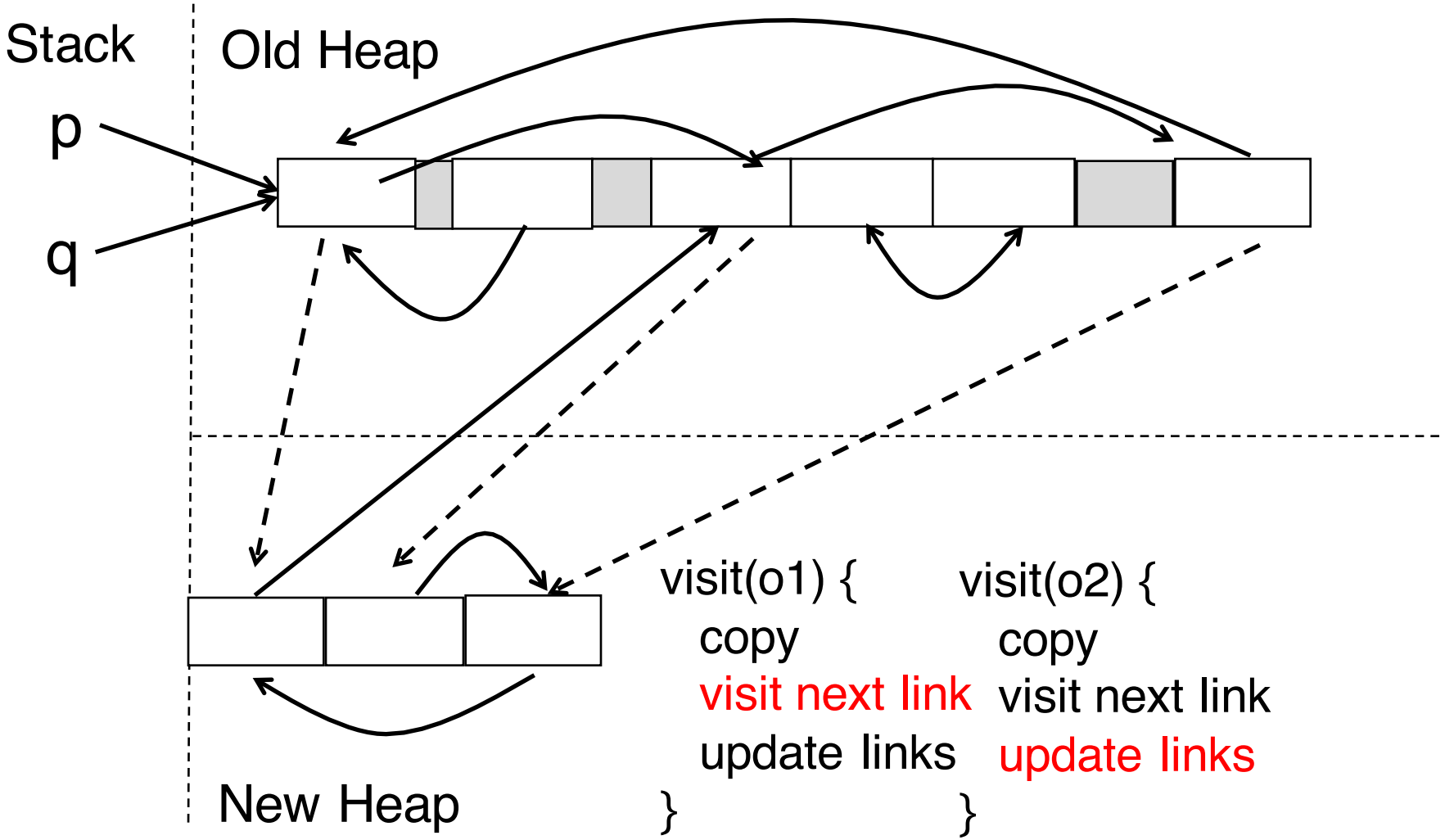
Forwarding Pointer

Copy red node (internal link is updated)



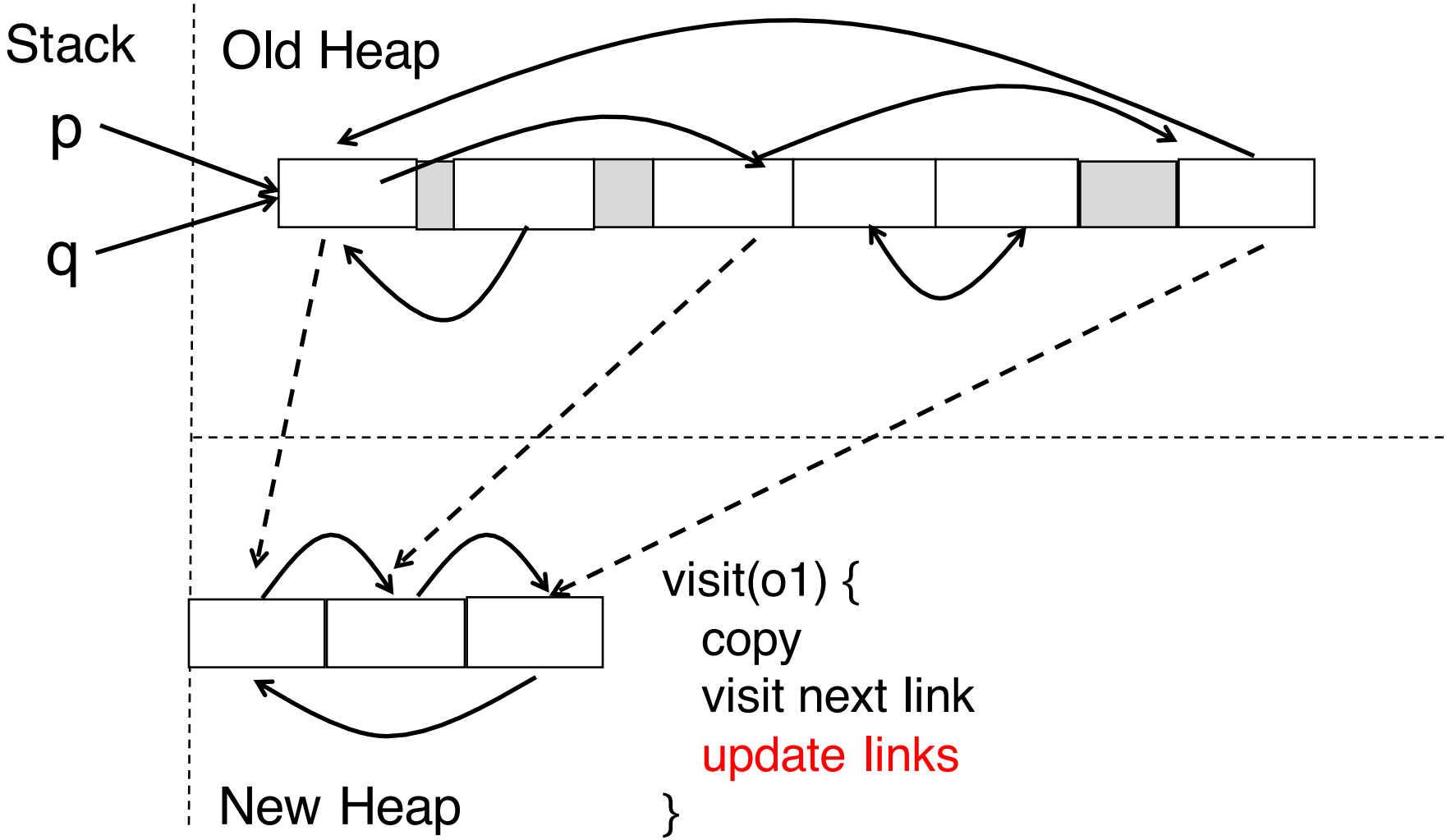
Example

Update links



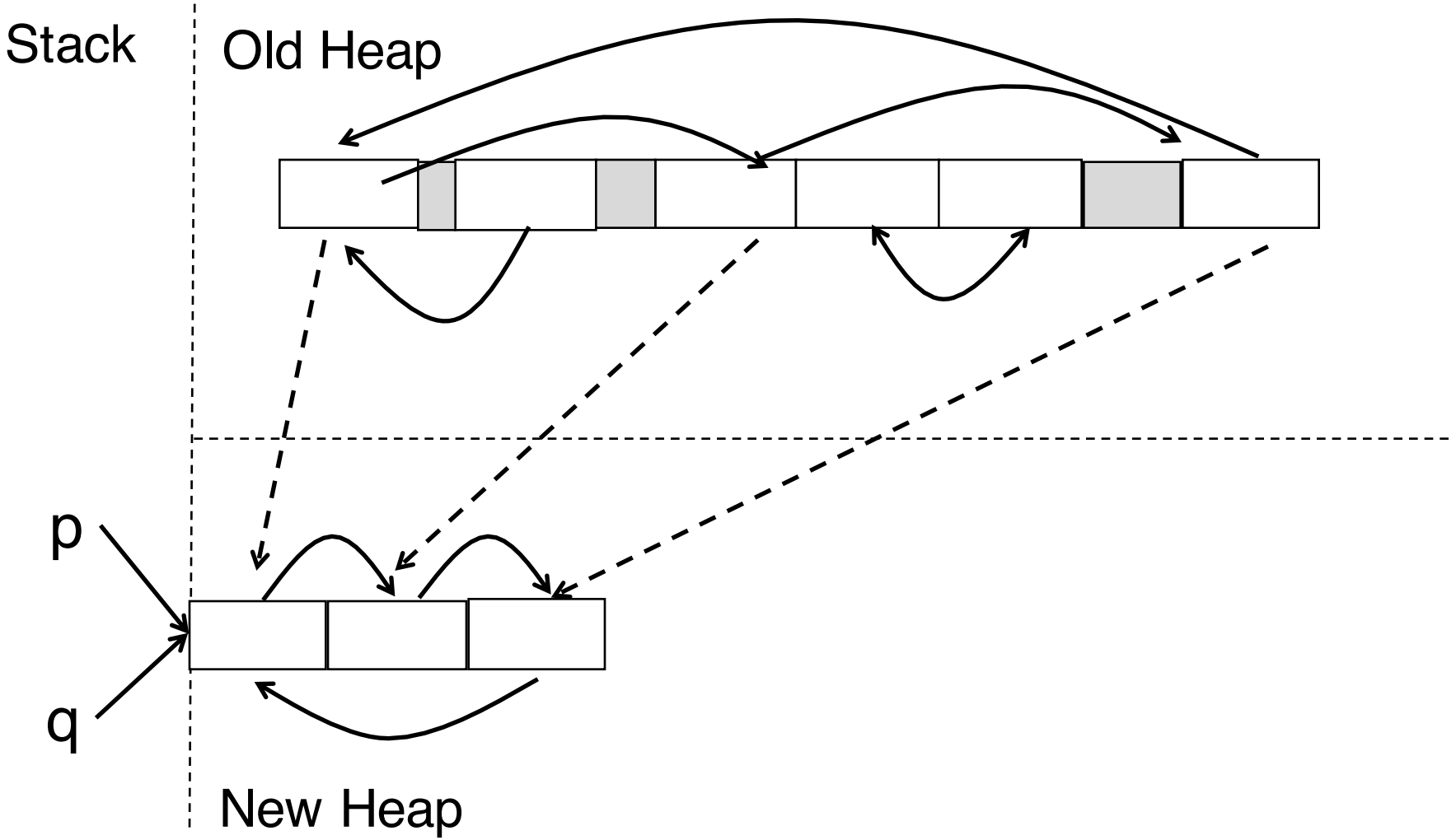
Example

Update links



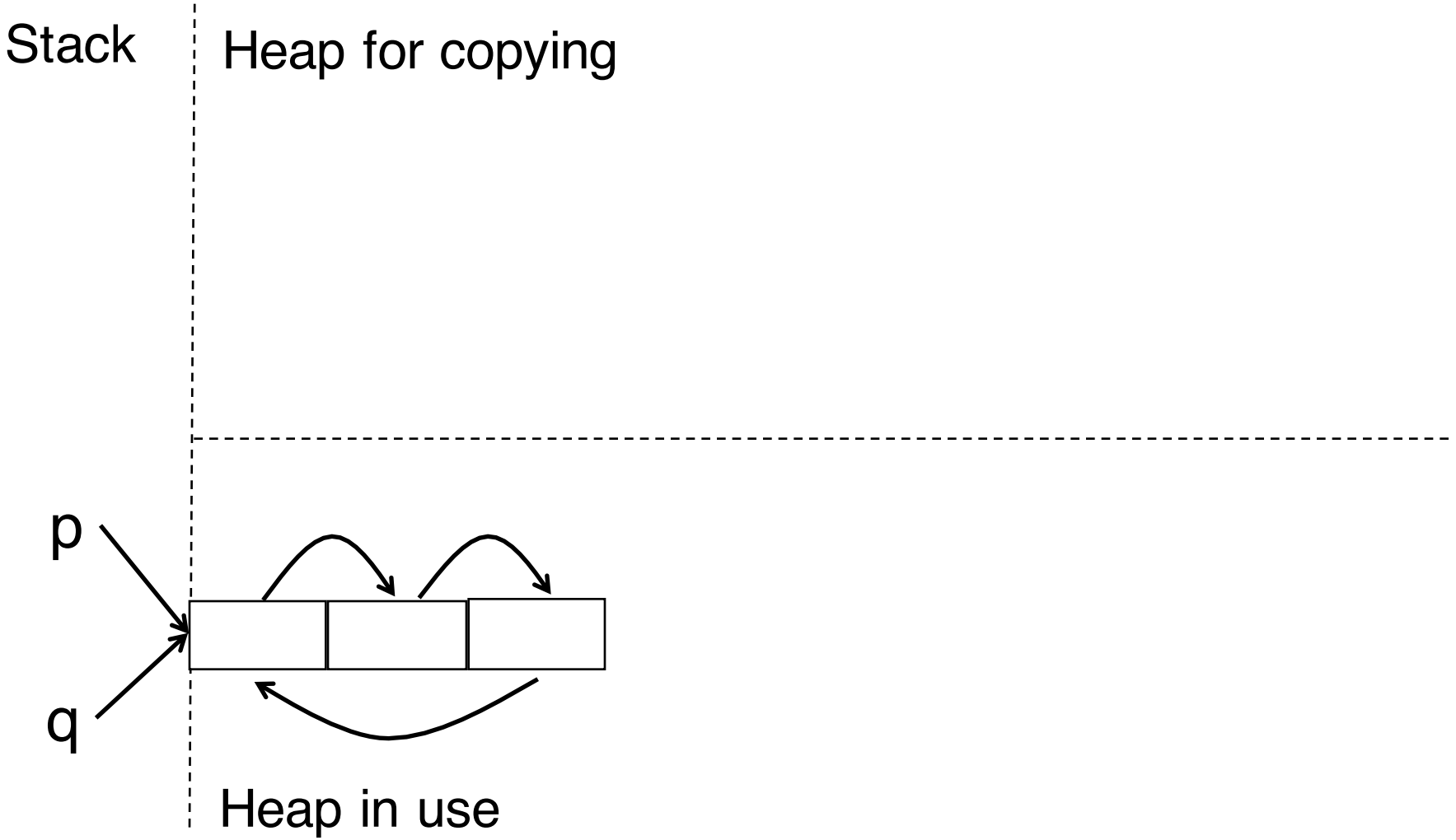
Example

Retarget root references



Example

Swap Heaps (no need to clean the old one)



Stop-and-Copy

Pros:

Eliminate fragmentation

Collection time proportional to live objects

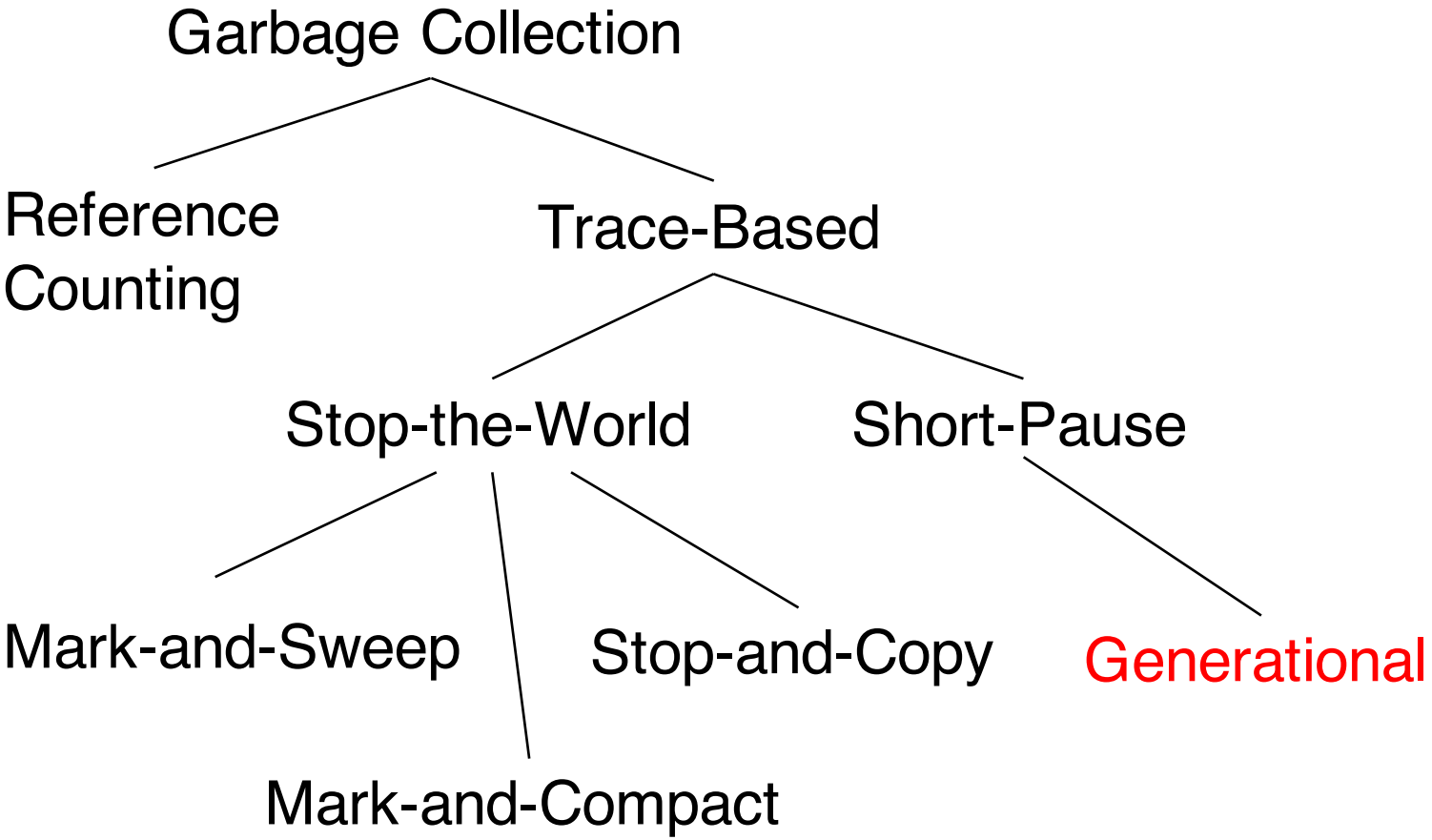
Great for short-lived data

Cons:

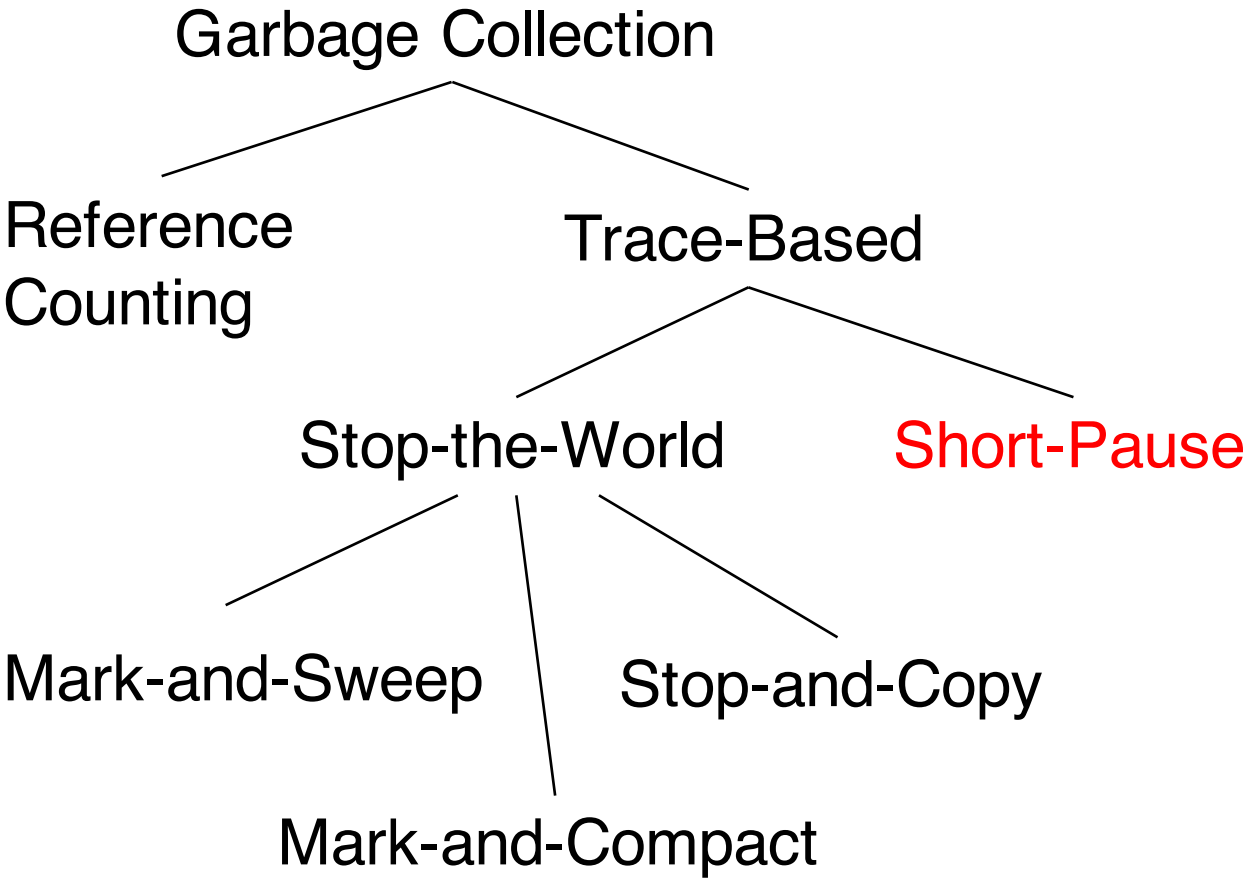
Only half of heap is in use, requires more collecting (virtual memory alleviates this limitation)

Bad for long living data (copying data back and forth)

Taxonomy


Garbage Collection
Reference Counting
Trace-Based
Stop-the-World
Short-Pause
Mark-and-Sweep
Mark-and-Compact
Stop-and-Copy
Generational

Taxonomy


Garbage Collection
Reference Counting
Trace-Based
Stop-the-World
Short-Pause
Mark-and-Sweep
Stop-and-Copy
Mark-and-Compact

Short-Pause Collection

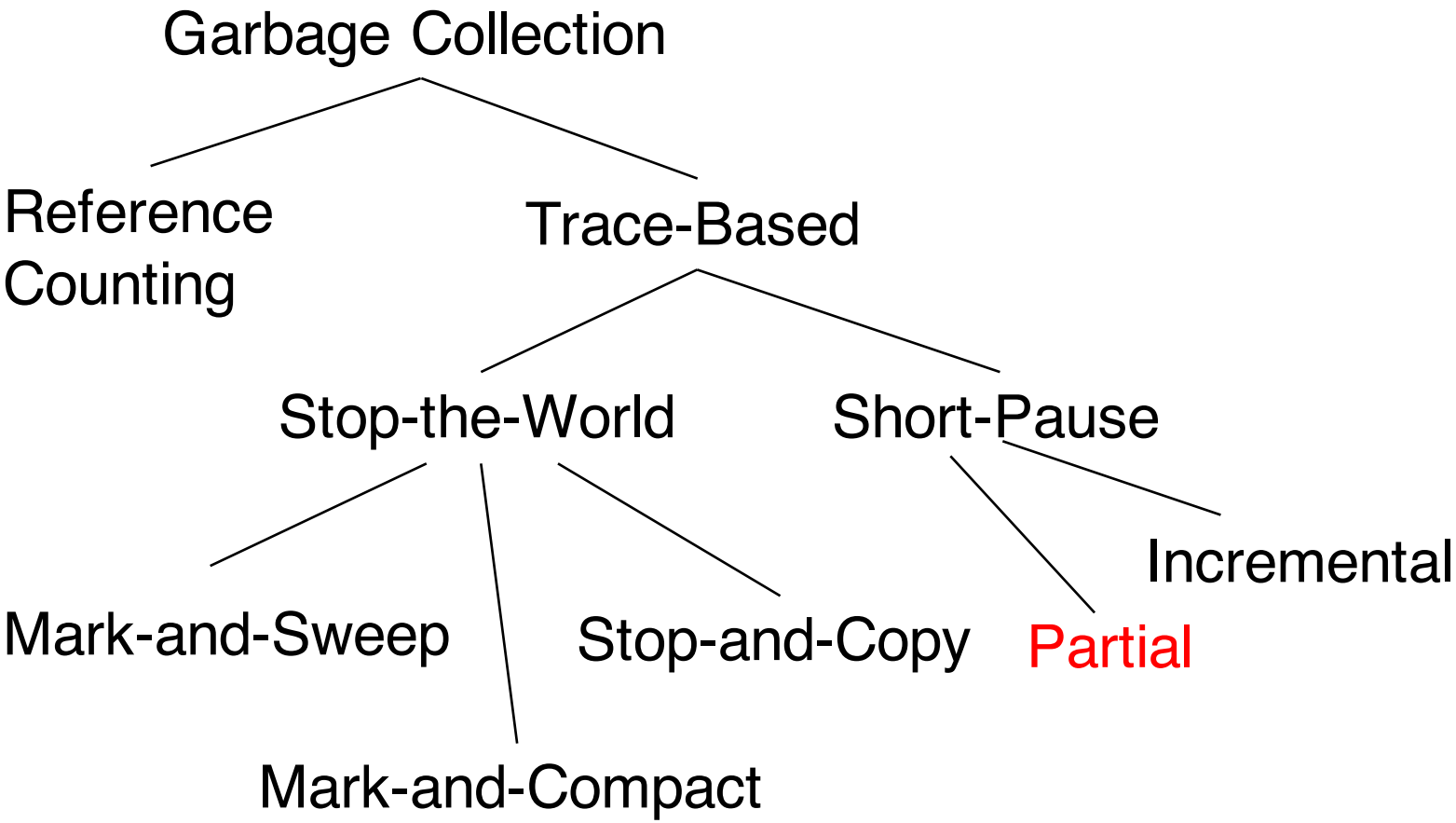
Partial

- stop the program, but only briefly, to garbage collect **a part of** the heap

Incremental

- Run garbage collection **in parallel** with operation of the program

Taxonomy


Garbage Collection
Reference Counting
Trace-Based
Stop-the-World
Short-Pause
Mark-and-Sweep
Mark-and-Compact
Stop-and-Copy
Partial
Incremental

Partial Garbage Collection

We collect one partition of the heap

- The target set

We maintain for a set of objects outside the partition that refer to objects in the target set

- the stable set

Collecting a Partition

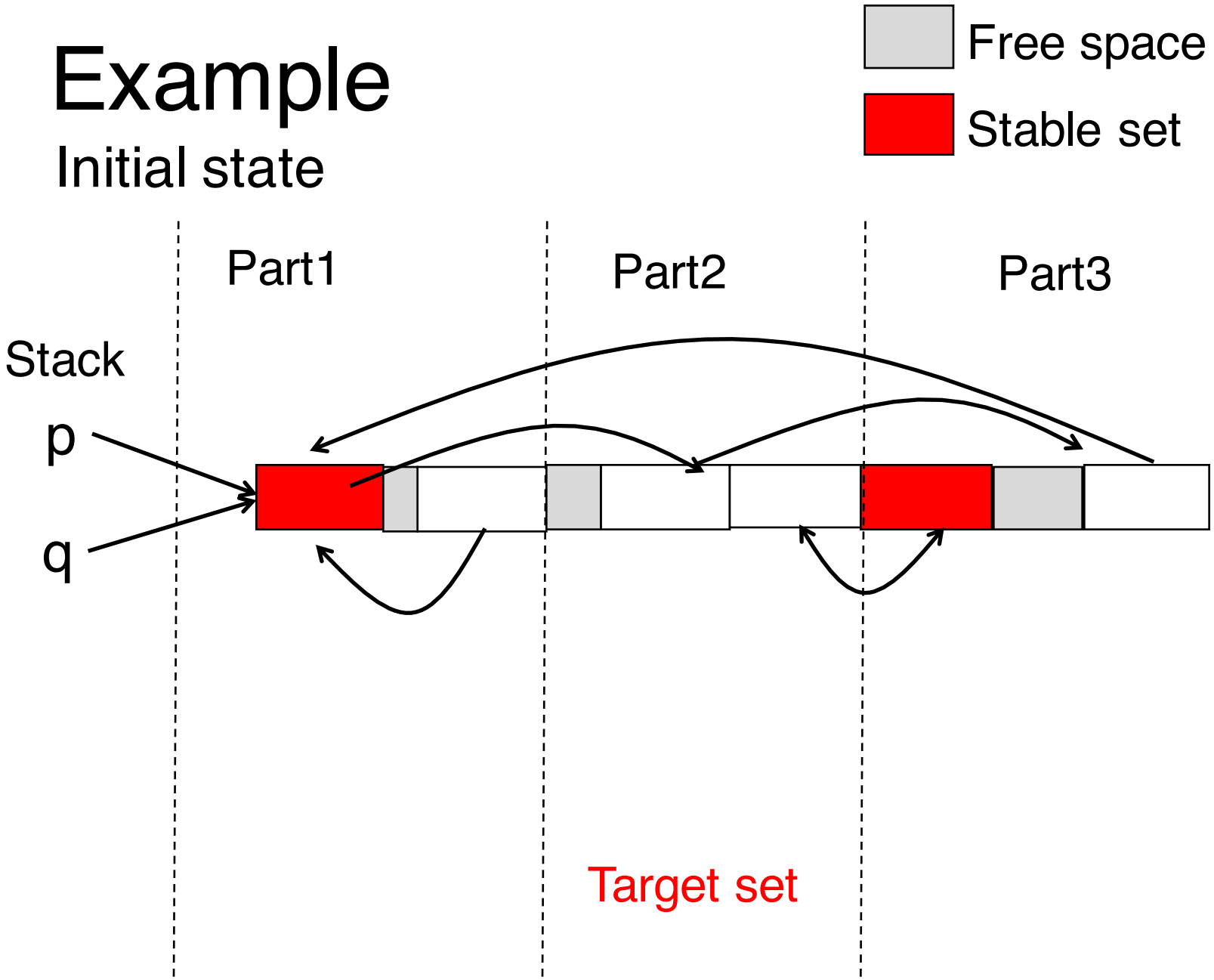
Add stable set to **root set**

Do GC as before (use algorithms discussed earlier)

Note: garbage might survive the collection

Example

Initial state



Example

After collection