Mark-sweep GC

Models GC's two phase abstraction

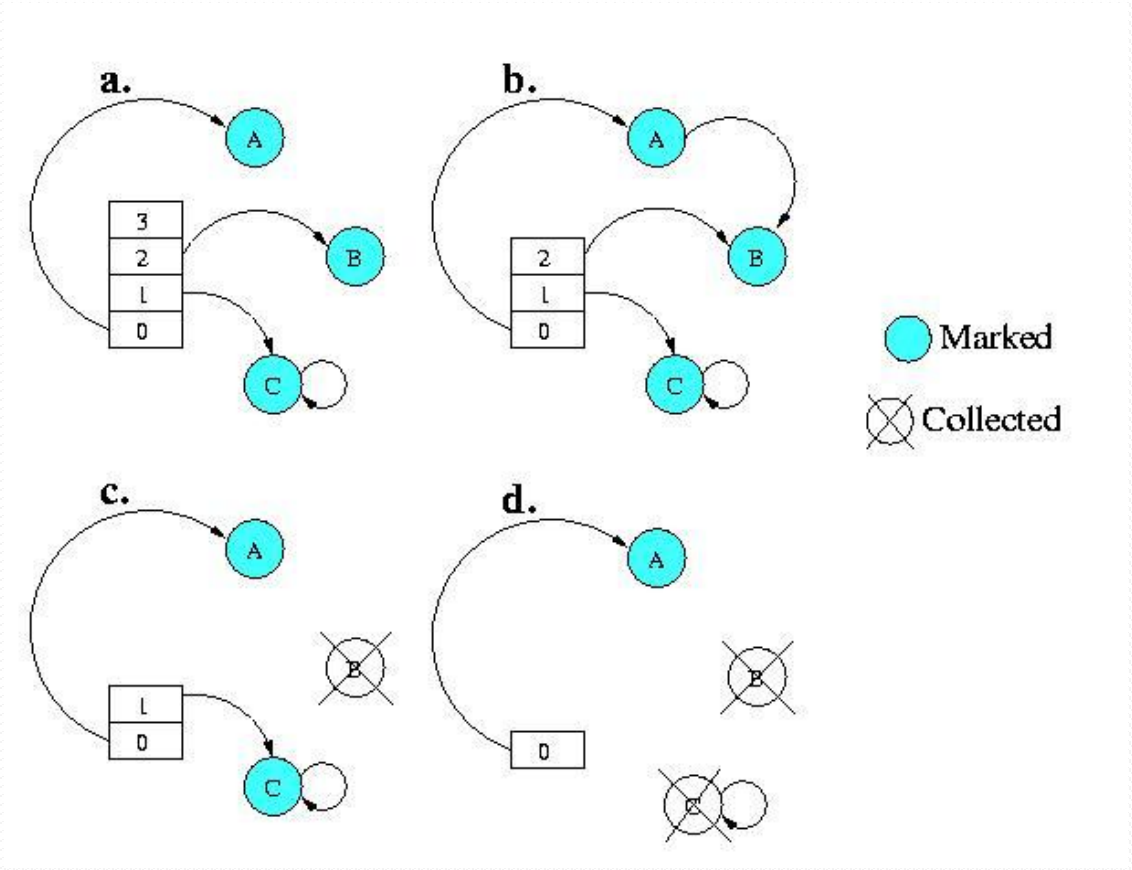
Mark-sweep GC Defined

- First algorithm for automated storage reclamation
- Is a stop-the-world collector
- Is an example of a tracing collector
- Has two phases
    - Mark phase
        - marks all objects reachable from the root set of the currently executing program
    - Sweep phase
        - reclaims all objects not marked in the in the previous phase

Implement 2-phase abstract GC alg.

- Distinguish live objects from garbage
  - Done by tracing, the mark step
  - Starts at the root set
  - Traverse graph of pointer relationships
    - Depth-first or breadth-first search
  - Mark reached object in some way
    - bitmap, bit in object, some other table
- Reclaim the garbage
  - Done in sweep phase
  - Memory exhaustively examined to find garbage
    - Linked to one or more free lists

Mark-sweep operations

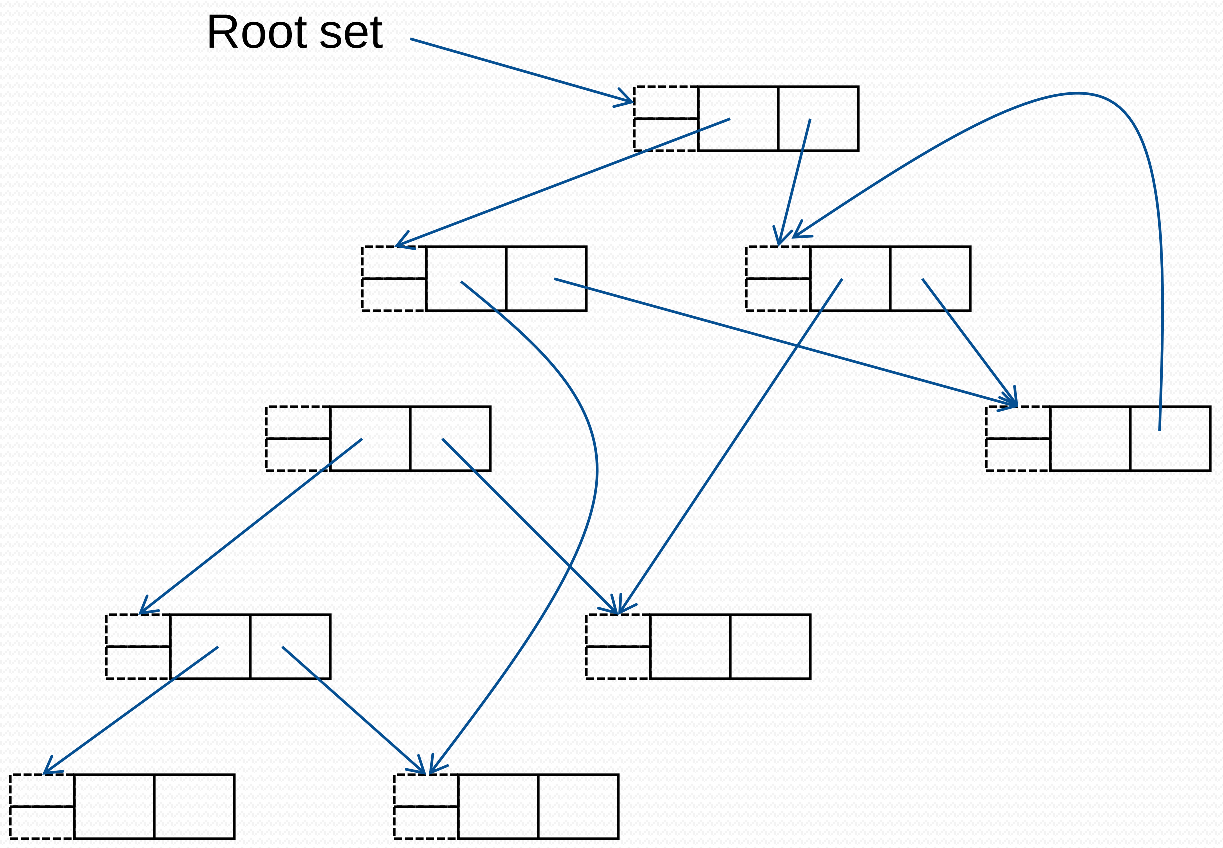


Mark-sweep algorithm

```
// The mark-sweep collector
mark_sweep() {
    for R in Roots
        mark(R)
    sweep()
    if free_pool is empty
        abort "Memory exhausted"
}
// Simple recursive marking
mark(N) {
    if mark_bit(N) == unmarked
        mark_bit(N) = marked
        for M in Children(N)
            mark(*M)
}
```

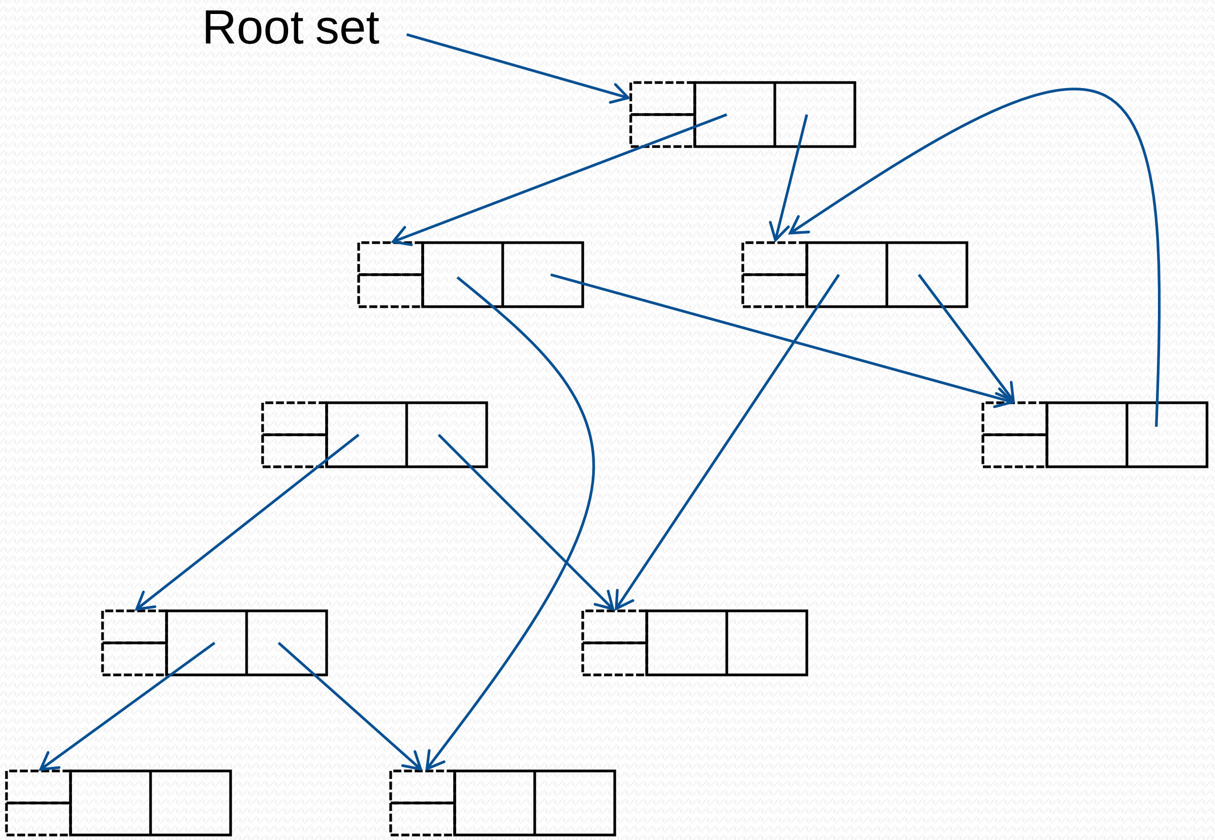
```
// The eager sweep of the heap
sweep() {
    N = Heap_bottom
    while N < Heap_top
        if mark_bit(N) == unmarked
            free(N)
        else mark_bit(N) = unmarked
        N = N + size(N)
}
```

Graph of object relationships



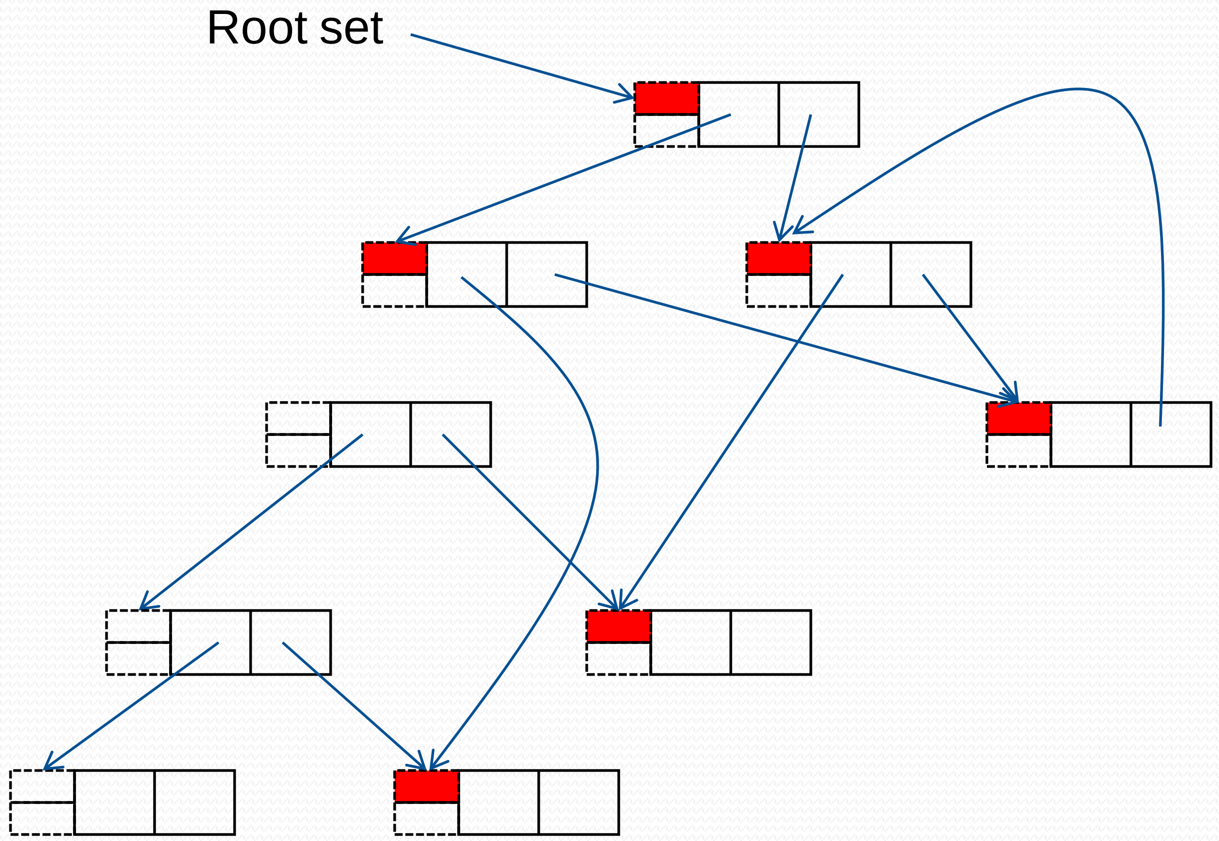
Before MS GC Runs

Graph of object relationships



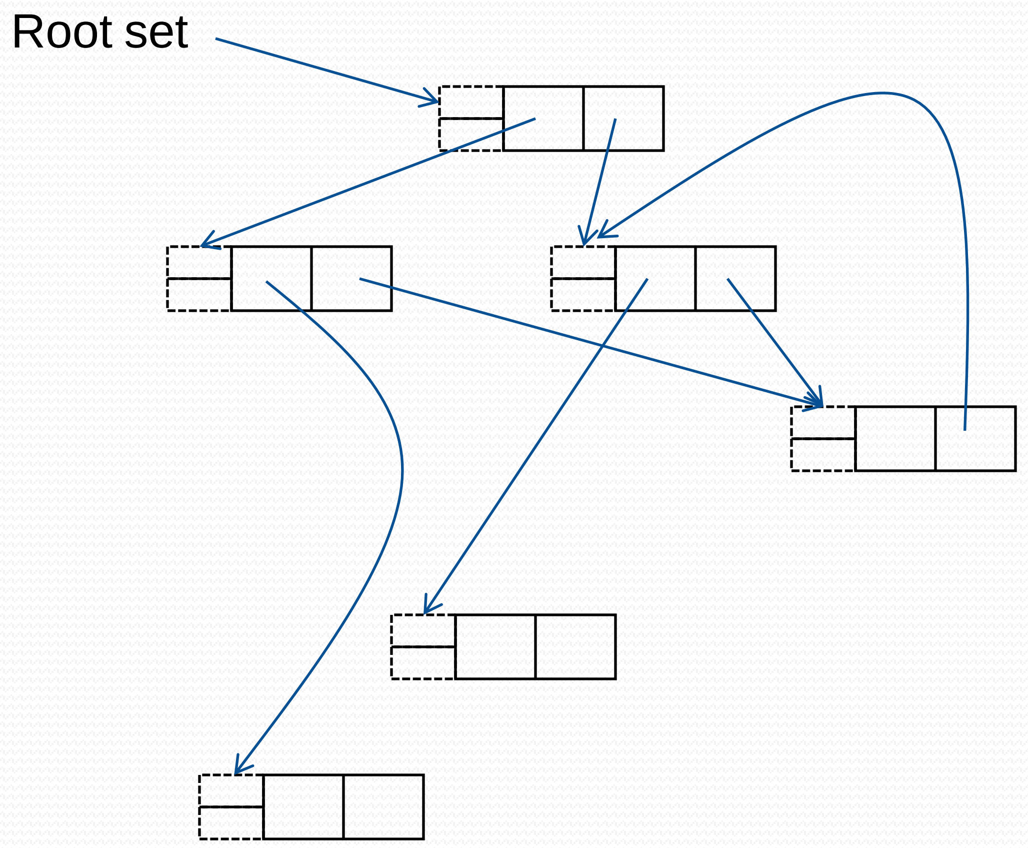
Graph after mark phase

Graph of object relationships



Graph after sweep phase

Graph of object relationships



After MS GC Runs

Advantages of mark-sweep GC

- Reclaims ‘*all*’ garbage
  - Including cyclic data structures
- No overhead on manipulating pointers
- Low space overhead
  - Only a mark bit per object

Disadvantages of mark-sweep GC

- Stop-the-world algorithm
  - Computation suspended while GC runs
  - Pause time may be high
    - Not practical for real-time, interactive applications, video games
- High cost:
  - proportional to size of heap (not just live objects)
  - Why?
    - Active objects visited by mark phase
    - All of memory visited by sweep phase

Disadvantages of mark-sweep GC

- Tending to fragment memory
  - Programs may ‘*thrash*’
- High heap occupancy
  - GC runs frequently

How do we address these cons?

Subject of next class