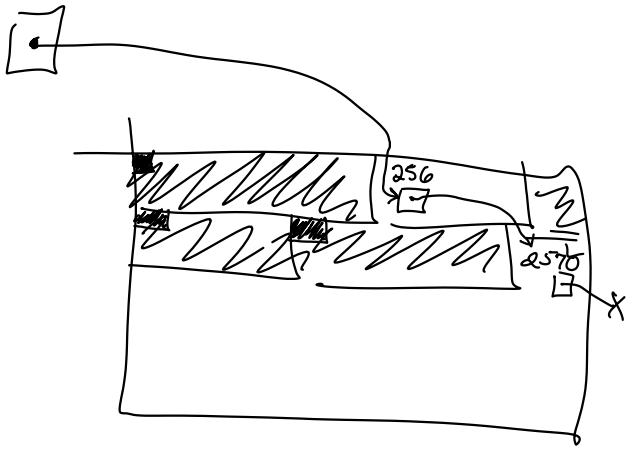
Malloc

malloc(400);



Mark & sweep

Mark reachable allocations
Free unmarked objs.

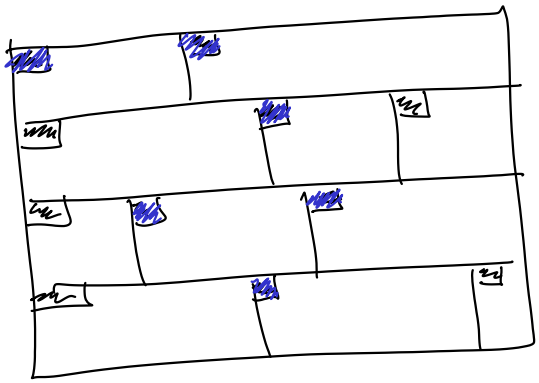
* Need to ID ptr
* Need to iterate the heap
+ Mem doesn't move (no ptr updates)
- Fragmentation

Mark & Don't Sweep

- When we fail to alloc mem:
  1. Flip which bit means "free"
  2. DFS mark all reachable

vs. M&S:

+ No sweep work
- Metadata for free & alloc must be ~same



Either 0 or 1 means free
& other means alloc

Reference Counting

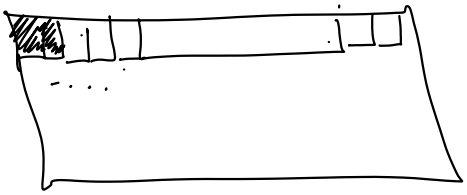
Each heap object records how many ptrs to it
When heap object has 0 referers, free

+ Very simple
- Bad w/ cycles
- Slow
- Bad w/ mem copies

Using RC

Java (a while ago)
C++ (via std libs)
Python
(uses RC mostly; occasionally, sweeps heap)

Scenario:

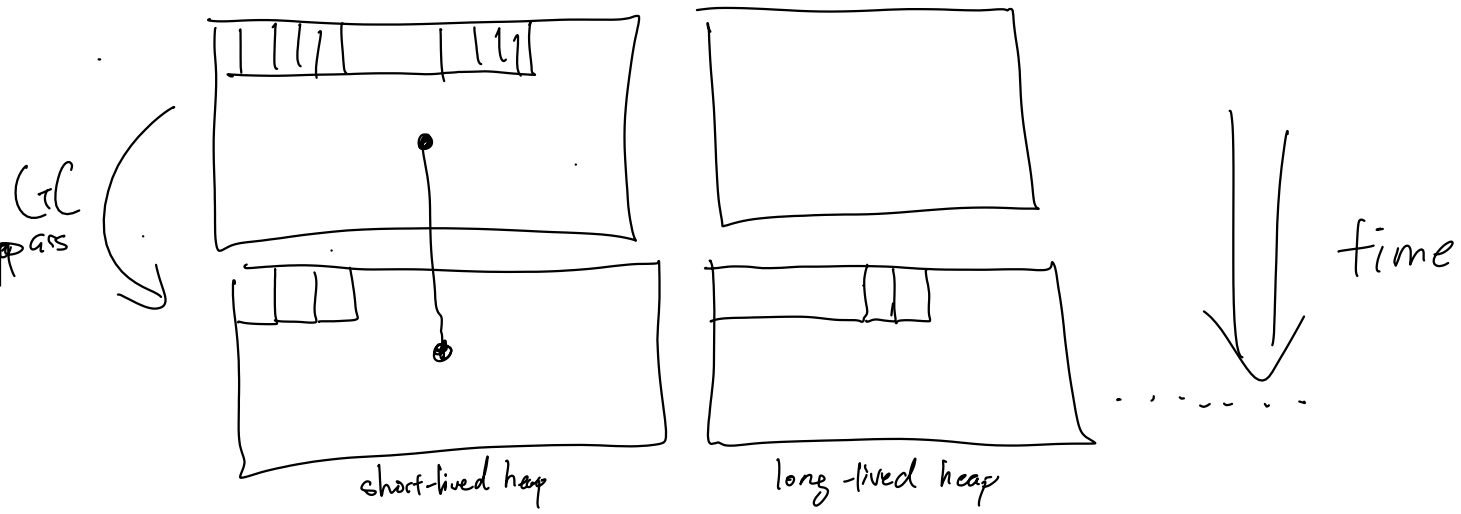


Small allocs
Big, long-lived allocs
Small allocs
GC time: 1st small alloc is freed

Problem: small allocs are short-lived, but cause long-lived, big alloc to move

Generational Hypothesis: most allocs are small & don't live long;
alloc that lives a little while usually lives a long while

Generational GC



Long-lived heap fills up slowly
Short-lived heap tends to clear out

0: 8s
1: 8.001s
2: 8.003s
.
.
.
16: 8.027s
17: 9.582s
18: 9.584s

"Stop-the-world" gc

or

Incremental gc

Ex.: (incremental gc on mark/compact)

after 2/3 full on each allocation, mark some of heap

after 3/4 full on each alloc, forward some of heap