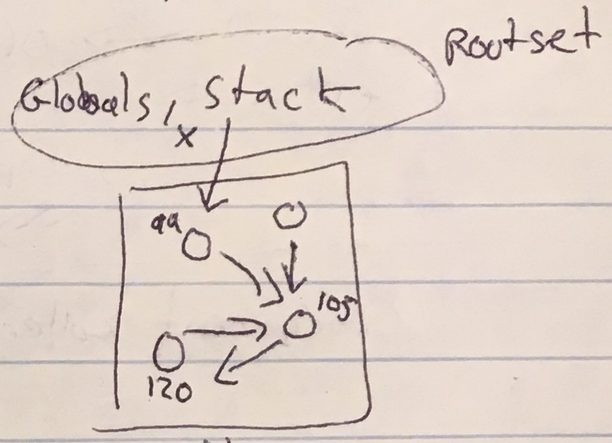
3

garbage collection ⟹ automatic memory management

- collector
- mutator
- root set
  - Intrinsically accessible
    - x ∈ RS
    - X, rest, rest, rest = 122



- live set := (heap − garbage)
- garbage := things with no references

Promises

① GC can tell what structure a ptr points to

108 ⟹ [58 | 57 | 92] ... how big

ptr

which are ptrs

easy, but not a given (OOs)

② Pointers cannot be ~~manufactured~~

scanf ("%d", &p)

*p = 17;

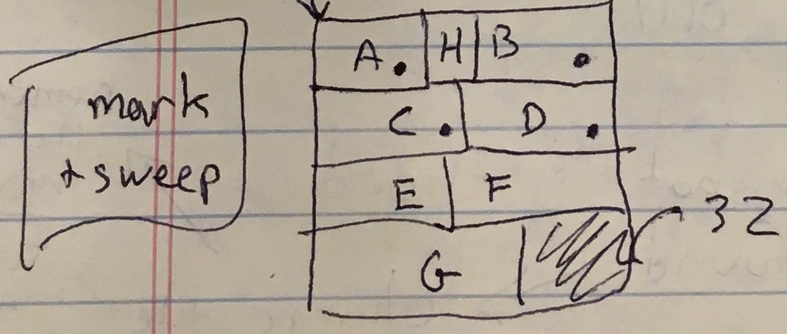
BiBoP

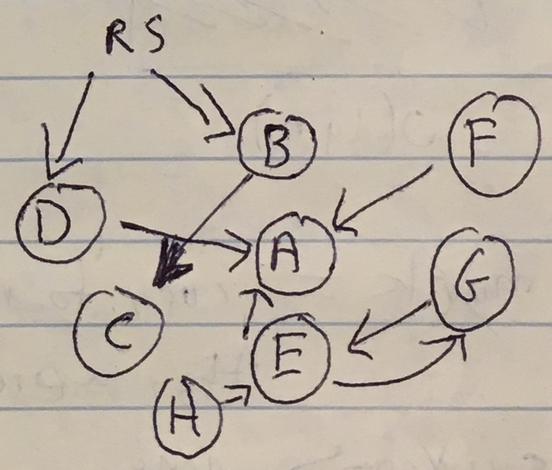
Big Bag of Pages

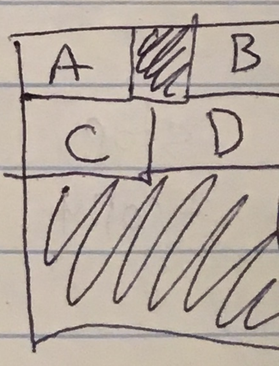
x64 ptrs store in 64 add space is 48

---

malloc (64)







addr=0 ⟹ 40 ⟹ 44 ⟹ 60 ⟹ 80

A H B C D

Free spot ⟸ 130 ⟸ 114 ⟸ 100

G F E

If there's a •

then remove •

else put on free list (& merge)

(4)

MRS costs

malloc() ⇒ same as manual

free() ⇒ O(manual)

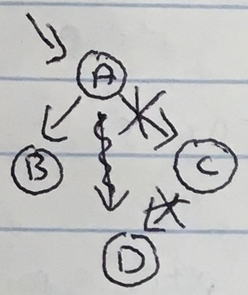
memory overhead – O(1) because 1 bit

extra ⇒ mark = O(live)

sweep = O(mem) ⇒ includes garbage

scheduling
- ↳ out of space
- ↳ every X calls to malloc
- ↳ every X ms
- ↳ every X bytes

concurrent collection

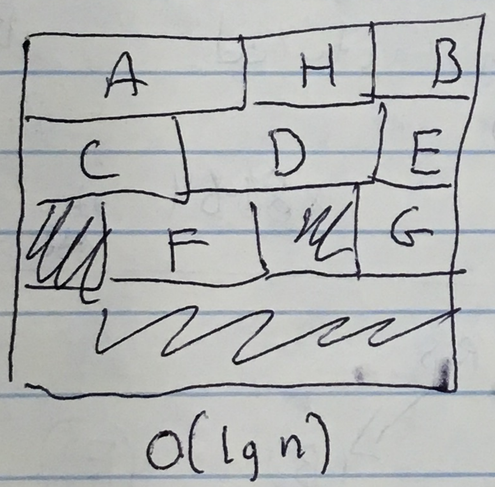


Marked A
add B, C
change
mark B
mark C

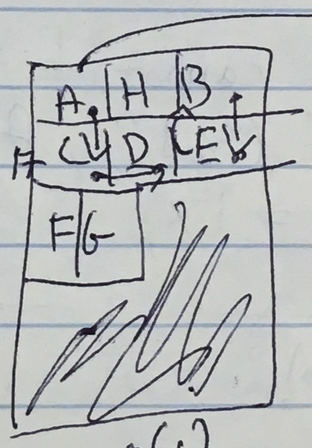
move allocation out of hot spots

solution: if mutator changes marked thing, tell GC

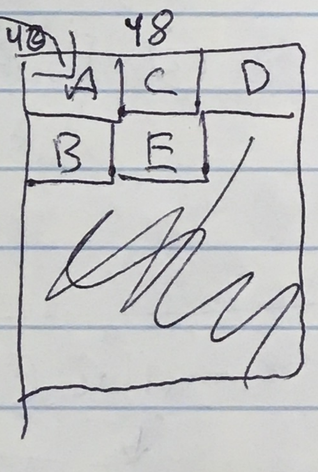
possible, but not great



O(lg n)



O(1)



stop + copy

mark ⇒ copy to new spot in the order they were found ⇒ change the ref ptr

remember that is was moved / forward addresses

sweep ⇒ gone

malloc() = O(1)

free() = O

extra() = O(live)

memory overhead = x2

before: a.c = 17

after: a'.c = 48 (42)

(17, 42)