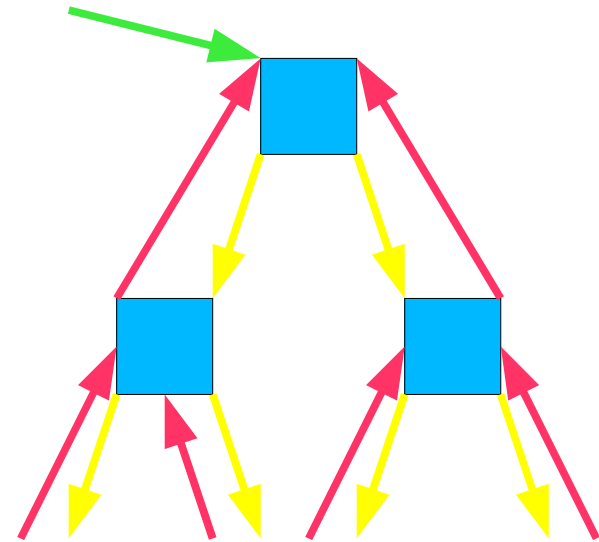
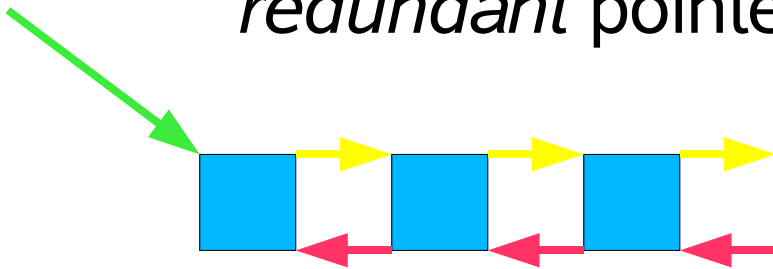


Imagine a world...

- ✗ With a static analysis that identifies *reachability-redundant* pointer fields:



- ✗ Redundant because *reconstructible*:

$$\forall n: n.next.prev == n$$

- ✗ How to infer such invariants? Some ideas, but I don't know for sure!

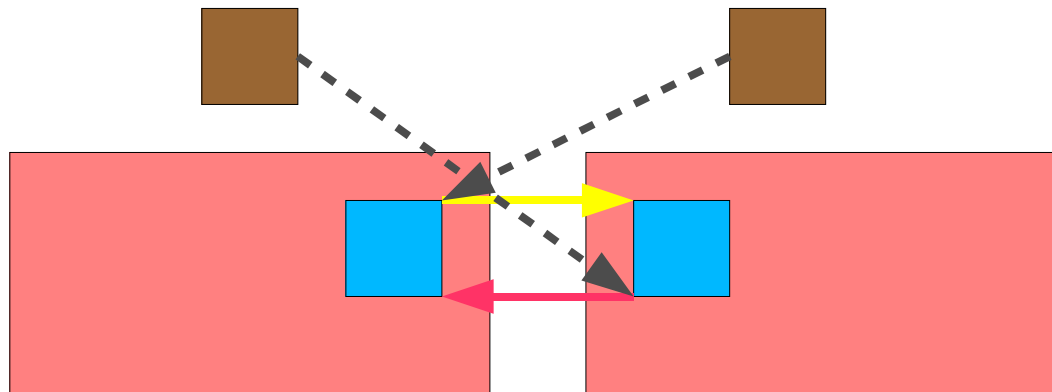
What could we do if we could infer such invariants?

- ✗ Decrease marking work: specialized object field iterators that skip the redundant fields.
- ✗ Reference-counting:
 - ✗ Decrease necessary work: don't count refs from redundant fields.
 - ✗ More importantly: eliminate garbage cycles!



What could we do if we could infer such invariants?

- ✗ In *space-incremental* collectors (Train, Garbage-First, MC²):
 - ✗ Decrease remembered set overhead – don't track redundant pointers. (But we must know how to reconstruct them!)



- ✗ As with ref-counting, eliminating cycles may be more important than just reducing cost.