A question about reference counting

• Q: What is the number of references that can refer to a specific given location?
  • A: Entire virtual memory.  
    (Implication: RC size = pointer size)

• Q: Suppose RC has only 3 bits. How can an overflow happen?
  • A: 9 pointers reference an object.

• Q: suppose we consider an RC that has reached “111” as “stuck” and never change it anymore. How does that influence the execution?
  – Will the program run correctly?
  – Will it consume more memory?

• A: It will run correctly and consume more memory.
A question about reference counting

- Q: Propose a manner to fix all stuck counts of live objects.
- A: Run tracing (like in mark-sweep).
  Upon checking an edge, increment the RC of the child.
  \(\text{(Of-course, RC may get stuck again, but those that should not will not...)}\)
A question about dangling pointers

• Q: Given updated reference counts, how can we tell that a “free” instruction creates a dangling pointer. (Assume no cycles, so RC is accurate.)
  • A: RC > 1.
• Q: Suppose one can tell if an object is allocated or freed (waiting for allocation). Use tracing to detect dangling pointers.
  • A: Trace and check.
• Q: Can we find all dangling pointers?
  • A1: No, as some of them may have been re-allocated.
  • A2: No, because we only go over live objects.
A question about dangling pointers

- Q: ignoring the tracing cost (i.e., can trace many times), detect all dangling pointers from live objects.
- A: trace after each delete!