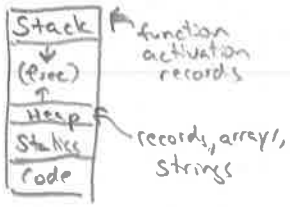
Garbage Collection

Lecture topics

I. Intro to GC
II. Reference counting
III. Mark-Sweep GC

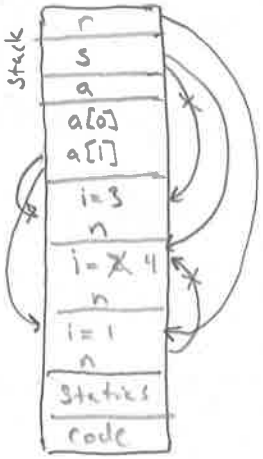
I. Intro to GC

Storage organization



Heap allocation example

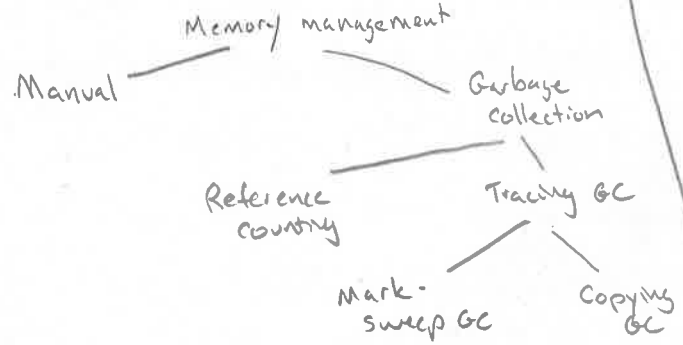
```
r = (i=1, n=());
r.n:= (i=2, n=());
s = (i=3, n=());
a = [r, s];
s:= null;
r.n := null;
s = (i=4, n=());
```



Garbage = heap object that will not be used in the future
(e.g., (i=2, n=1) is unreachable)

Garbage collection (GC) = reclaim memory of garbage, for reuse in later allocations

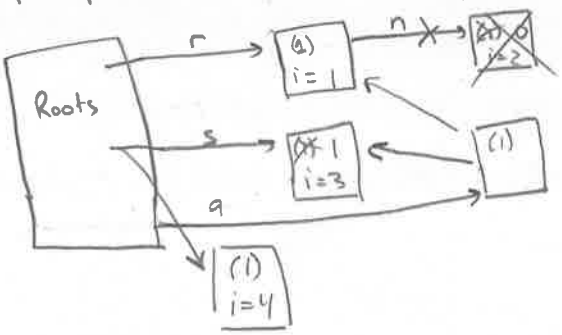
Various techniques:



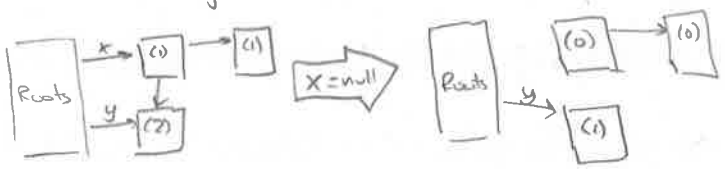
TACK leaves memory management unspecified.

II. Reference Counting

Keep implicit "count" field in every heap object.



- when creating pointer: increment count
- when overwriting pointer: decrement count
- when count drops to ∅:
  - Recursively decrement counts
  - Free object



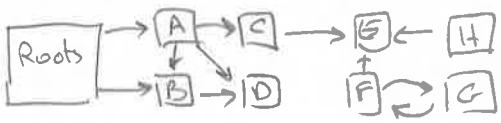
Problem: reference counting does not reclaim cycle

III Mark-Sweep GC

Traverse objects from roots to find reachable
Reclaim all unreached objects
Use mark stack (reached, must scan pointer)
mark bit in each heap object

States of object:

Example:



Mark	B	D			
Stack	A	A	A	C	E

→ time