HW Assignment
Copy Collection

1) Number using original algorithm:
For each root r: copy(r)

1 Copy A object to A’ memory location
16 Copy B object to B’ memory location
3 Copy C object to C’ memory location
5 Copy D object to D’ memory location
11 Copy E object to E’ memory location

2 Update As header with A’ address
17 Update Bs header with B’ address
4 Update Cs header with C’ address
6 Update Ds header with D’ address
12 Update Es header with E’ address

15 Return A’ address and update obj1 val
18 Return B’ address and update obj1 val
9 Return C’ address
7 Return D’ address
13 Return E’ address

10 Update A’ left field with C’s address
14 Update A’ right field with E’s address
8 Update C’ left field with D’s address
2) Re-write original algorithm to use breadth first traversal. Make sure to include the roots:

WL = create_worklist()
for each root r: ...
    WL.add(r, location_of_root)
for each x in WL, remove (x,location) from WL:
    if exists(forw_addr(x))
        location = forw_addr(x); return;
    x' = copy(x));
    forw_addr(x) = x'
    for (i = 0; i<child_objs; i++)
        if exists(forw_addr(x[i]))
            x'[i]= forw_addr(x[i])
        else WL.add(x[i], &x[i])

//contact Chandra (ckrintz@cs.ucsb.edu) if you see an error
//or a better way to do this
HW Assignment
Copy Collection

1) Number using original algorithm:
   For each root r: copy(r)
   1 Copy A object to A’ memory location
   16 Copy B object to B’ memory location
   3 Copy C object to C’ memory location
   5 Copy D object to D’ memory location
   11 Copy E object to E’ memory location

   2 Update As header with A’ address
   17 Update Bs header with B’ address
   4 Update Cs header with C’ address
   6 Update Ds header with D’ address
   12 Update Es header with E’ address

   15 Return A’ address and update obj1 val
   20 Return B’ address and update obj2 val
   9 Return C’ address
   7, 18 Return D’ address
   13 Return E’ address

   10 Update A’ left field with C’s address
   14 Update A’ right field with E’s address
   8 Update C’ left field with D’s address
   19 Update B’ field with D’s address

Here’s a case that is a bit different – B has a field that refers to D
Copying Collector pros/cons

• Strengths
  ■ No fragmentation
    ▸ Fast allocation (bump pointer)
  ■ Handles variable-sized objects naturally
  ■ No overhead on pointer updates

• Weaknesses
  ■ Required address space is doubled compared with non-copying collectors
    ▸ Performance degrades as residency increases (twice as quickly as mark-&-sweep b/c half the space)
  ■ Touches every page (VM) of the heap regardless of residency of the user program
    ▸ Unless both semispaces can be held in memory simultaneously
    ▸ Disrupts program locality (memory hierarchy and OS paging)