CSC 548: Lecture 9

Overview

Garbage collection

Mark-sweek gc

Reference counting gc

Copying gc

Garbage collection

What is garbage collection (gc)? Why is gc useful? When is gc called?

What is a directed graph? A rooted directed graph? What are the reachable nodes in a rooted directed graph? What has this got to do with gc?

How can we measure the cost of a gc algorithm? Hint: if H is the size of the heap, and R is the size of the reachable heap, and C is the cost of running gc, then what is the *amortized cost* of allocating memory?

Garbage collection example

```
class List {
  field hd : Integer; field tl : List; field sz : Integer;
  method cons (hd : Integer) : List { return new List { hd=hd; tl=this; sz=this.sz+1; } }
  method snoc (last : Integer) : List {
    if (this.sz == 0) { return this.cons (last); }
    else { return this.tl.snoc (last).cons (this.hd); }
  }
  method reverse () : List {
    if (this.sz <= 1) { return this; }
    else { return this.tl.reverse ().snoc (this.hd); }
  }
  ...
}
object empty : List { hd=0; tl=empty; sz=0; }
thread Main {
  let x : List = empty.cons (10).cons (20).cons (30);
  let y : List = x.reverse ();
  // what can be gc'd here?
  stdout.print ("x = " + $x + ", y = " + $y);
}
```

Mark-sweek gc

What is the mark-sweep algorithm?

a) What is the cost of the mark phase (if it costs c_1 to mark one node)?

b) What is the cost of the sweep phase (if it costs c_2 to sweep one node)?

c) What is the amortized cost of memory allocation? (After allocation, we can make sure that $R = H/2$ by allocating or freeing memory from the OS).

Mark-sweek gc

The mark phase of mark-sweep gc is recursive. If we used recursion directly, how many activation records would we need?

Using recursion in gc is bad: what can we do instead?

How can we implement the mark phase in a fixed amount of stack space? How much extra memory do we need to use on the heap?

The sweep phase of mark-sweep gc adds nodes to a free list. How can we implement the free list? What is fragmentation? What can we do about it?

Reference counting gc

What is the reference counting algorithm?

Where is the cost of reference counting gc?

One improvement to reference counting gc: do the recursive decrementing when the object is removed from the free list, rather than when it is added. Why?

Again, we have a free list, so we have to worry about fragmentation.

Reference counting has a problem with cyclic heap: why? What can we do about it?

Copying gc

What is the copying algorithm?

What is from-space? To-space? Pointer forwarding?

Cheney's algorithm keeps three pointers into to-space: next, limit and scan. What are these?

a) What is the cost of the copying phase (if it costs c_3 to mark one node)?

b) What is the amortized cost of memory allocation? (After allocation, we can make sure that $R = H/4$ by allocating or freeing memory from the OS).

Copying gc

What extra information does copying gc require?

Copying gc needs a forwarding pointer for each object in from-space which has already been forwarded to to-space. Where can we put this forwarding pointer?

Copying gc has a problem with caching: *locality of reference*. What is this? What can we do about it?

Homework

Implement copying gc.

Summary

Garbage collection provides automatic memory management: programmers don’t need to explicitly free memory!

Three algorithms for gc are mark-sweep, reference counting and copying.

Next week: Generational and incremental gc.