Garbage Collection

CS520

Dept. of Computer Science
Univ. of New Hampshire

Garbage Collector

works with a memory allocator

identifies allocated blocks that are

garbage

- okay to be re-claimed and re-allocated

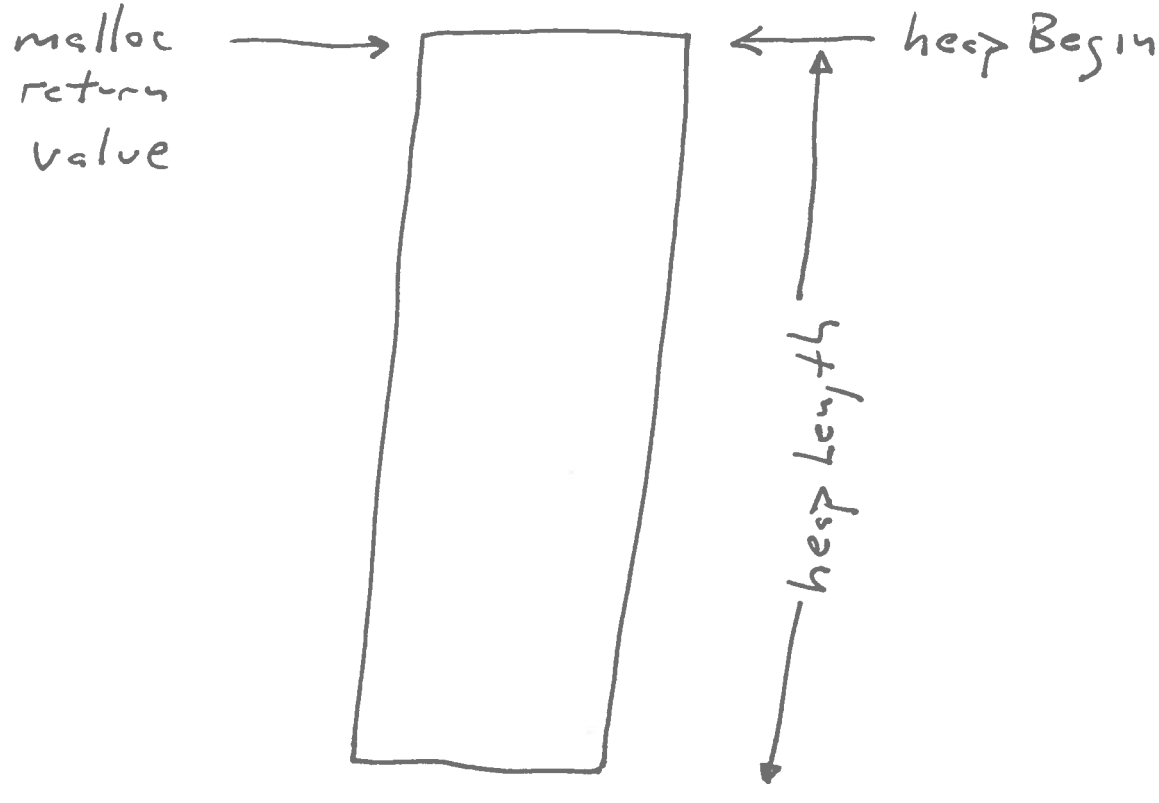
memory allocator

```
memInitialize(heapSize)

memAllocate(blockSize, finalizer)
```

~~memAlloc.te~~

use malloc to allocate the heap



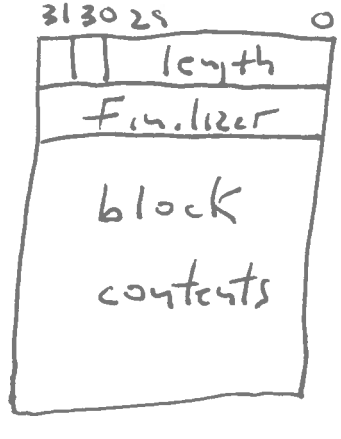
block headers

stores control information

1. is block allocated?
2. block length
3. if allocated, finalizer address
4. if allocated, is block reachable?

use single bits for 1 and 4 and pack with the block length

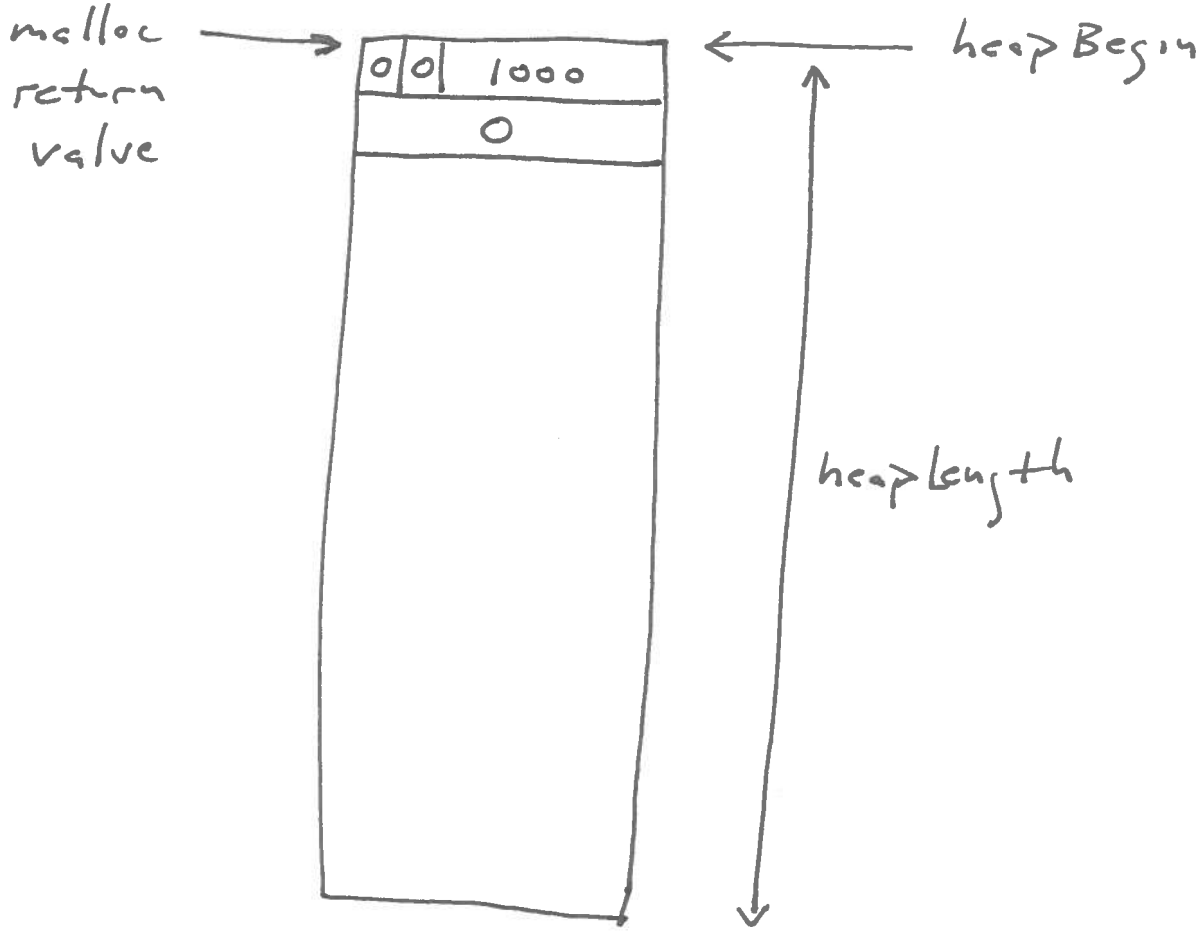
finalizer address is its own word



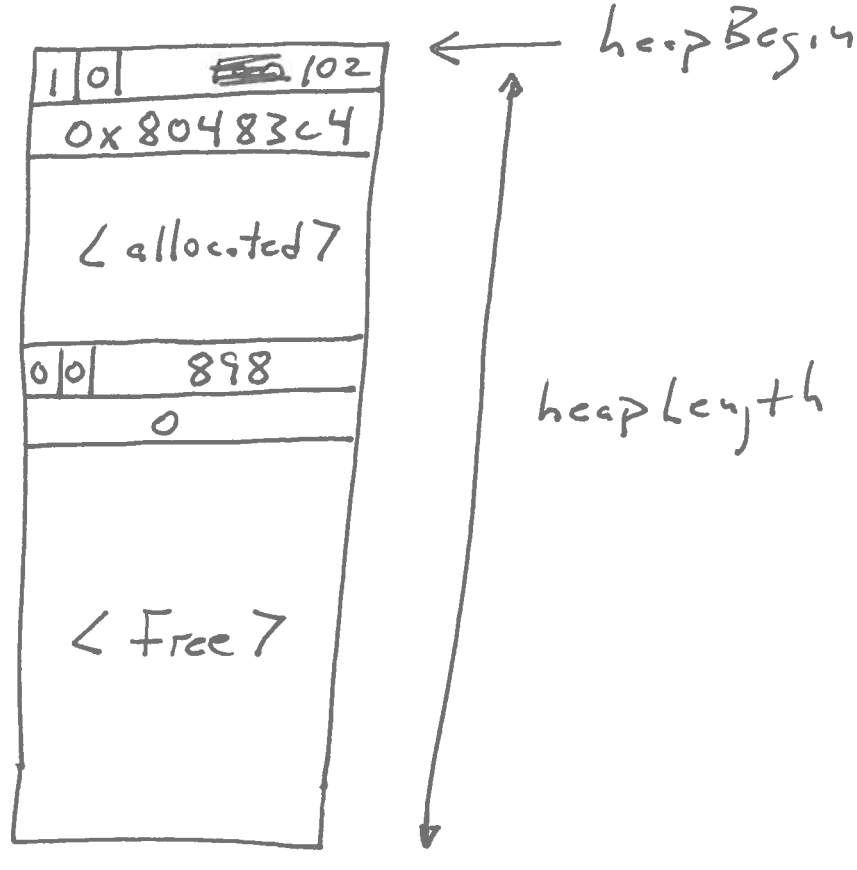
* does length include the header?

memInitialize

memInitialize(1000)



memAlloc.tc (100, 0x80483c4)



alignment

aligning values on appropriate memory boundaries is important for correctness on some machines and for performance on all machines

↑ memory system is designed to quickly read aligned values

typically

4-byte value (int/Float) should be on 4-byte boundary

↳ address evenly divisible by 4

8-byte value (double) should be on 8-byte boundary

So

allocator allocating in units of bytes
would round-up user's requested
length to be a multiple of 8

this, of course, ensures that header
words will be aligned

memFree?

could require user to call Free to explicitly deallocate a block

or when heap is full, run a garbage collector

Garbage collection

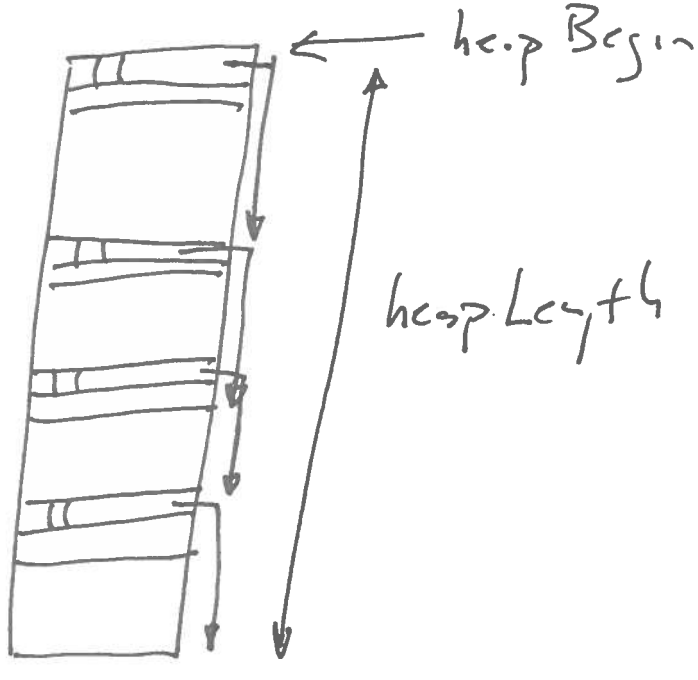
Identify allocated blocks that program can no longer reach
- no pointers exist to it

mark and sweep!

mark and sweep

1. mark all reachable, allocated blocks

2. sweep over the heap and re-claim the unmarked, allocated blocks

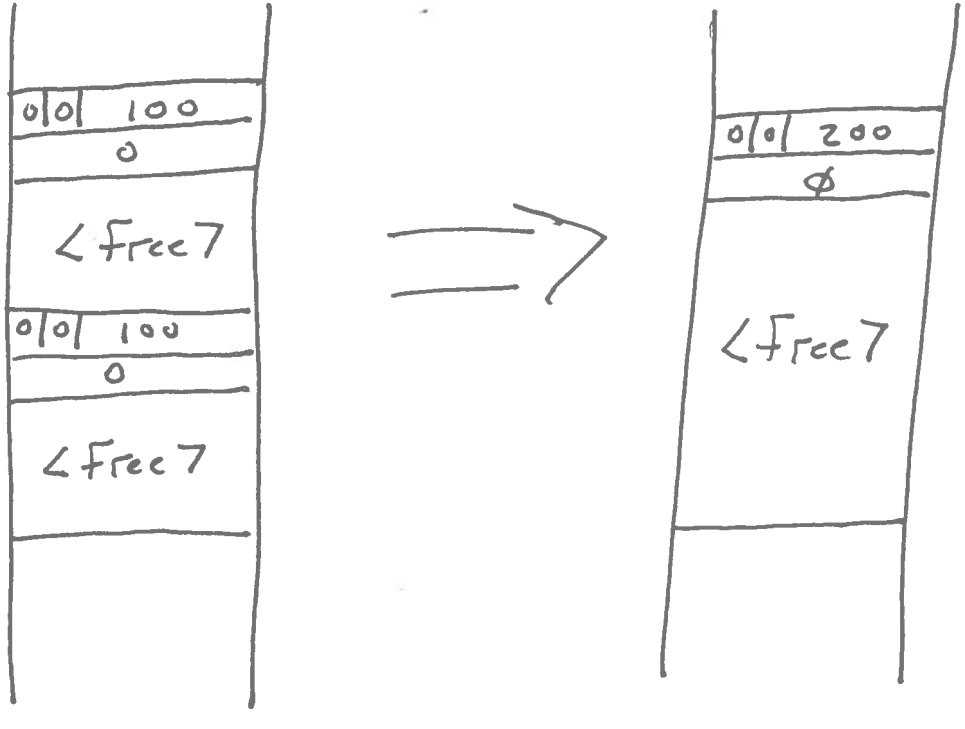


Re-claiming blocks

call the Finalizer function

mark the block as free

combine it with any adjacent free blocks



Which allocated blocks are reachable:

pointed to by global variable

or

pointed to by the stack

ie local variable
parameter
temporary
saved register

or

pointed to by a reachable block

ie marking procedure is recursive
base of recursion: block is already marked

or

pointed to by a register

only need to worry about callee-saved ones

roots of the garbage collection

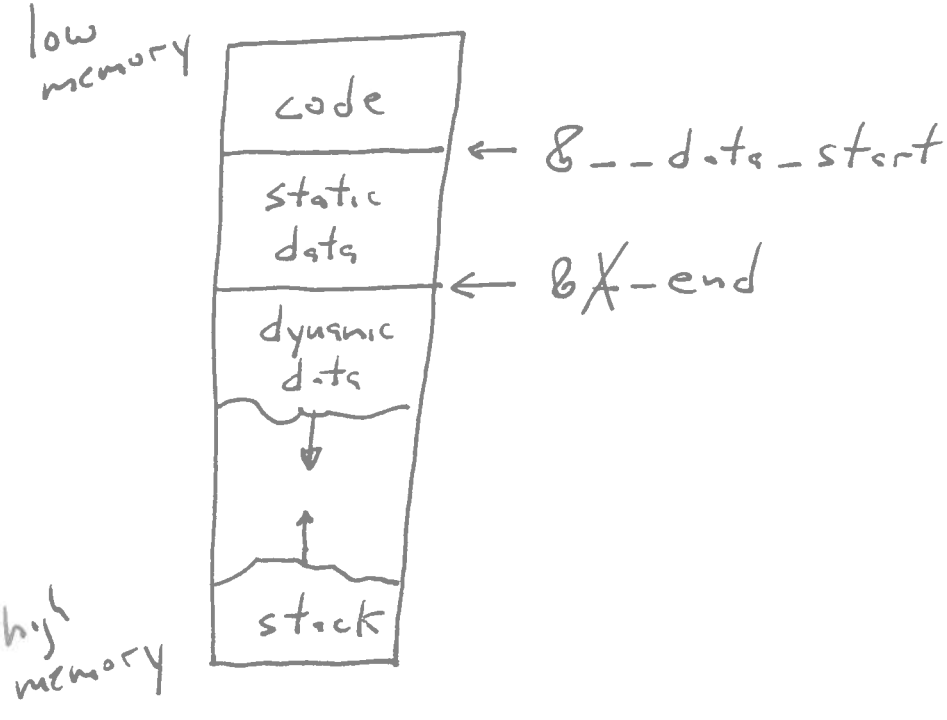
Globals

under Linux, linker inserts symbols that delimit the static data area

```
extern int __data_start;
extern int _end;
```

← just one underscore here!

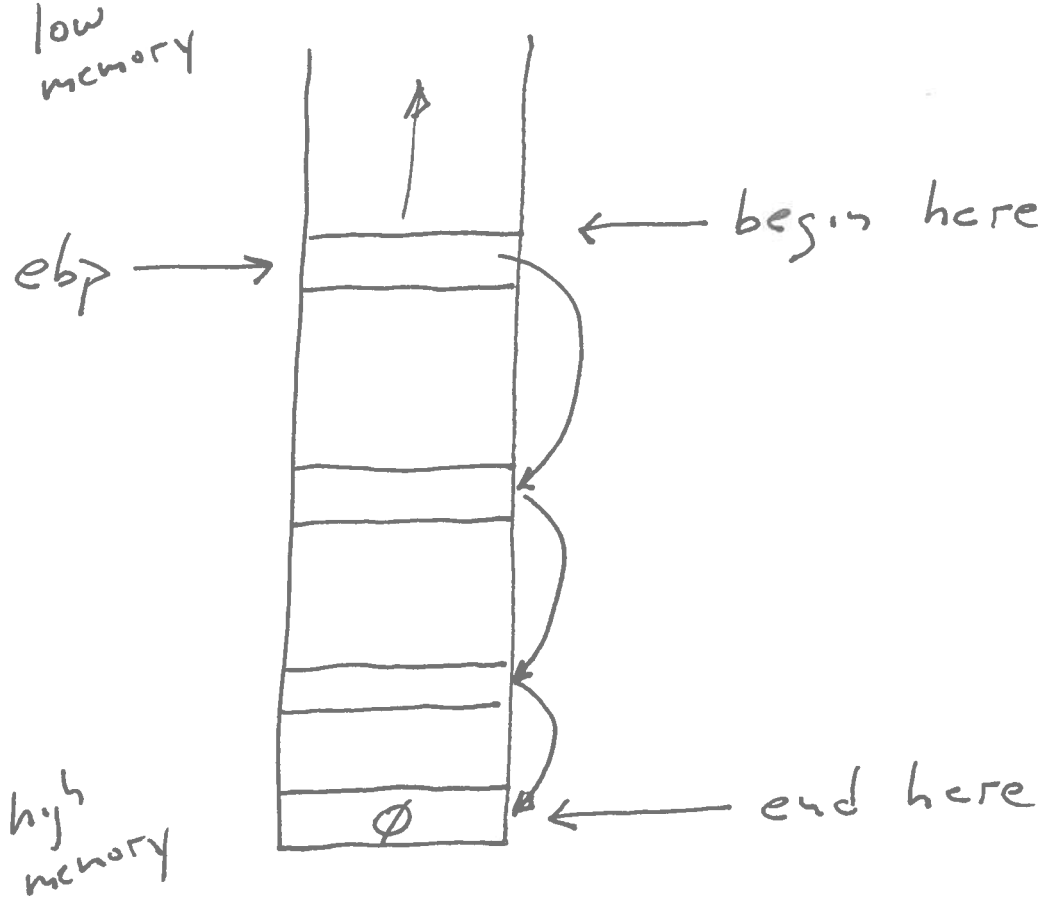
└ take the address of these to get the bounding addresses



note: be careful about the heap's control variables

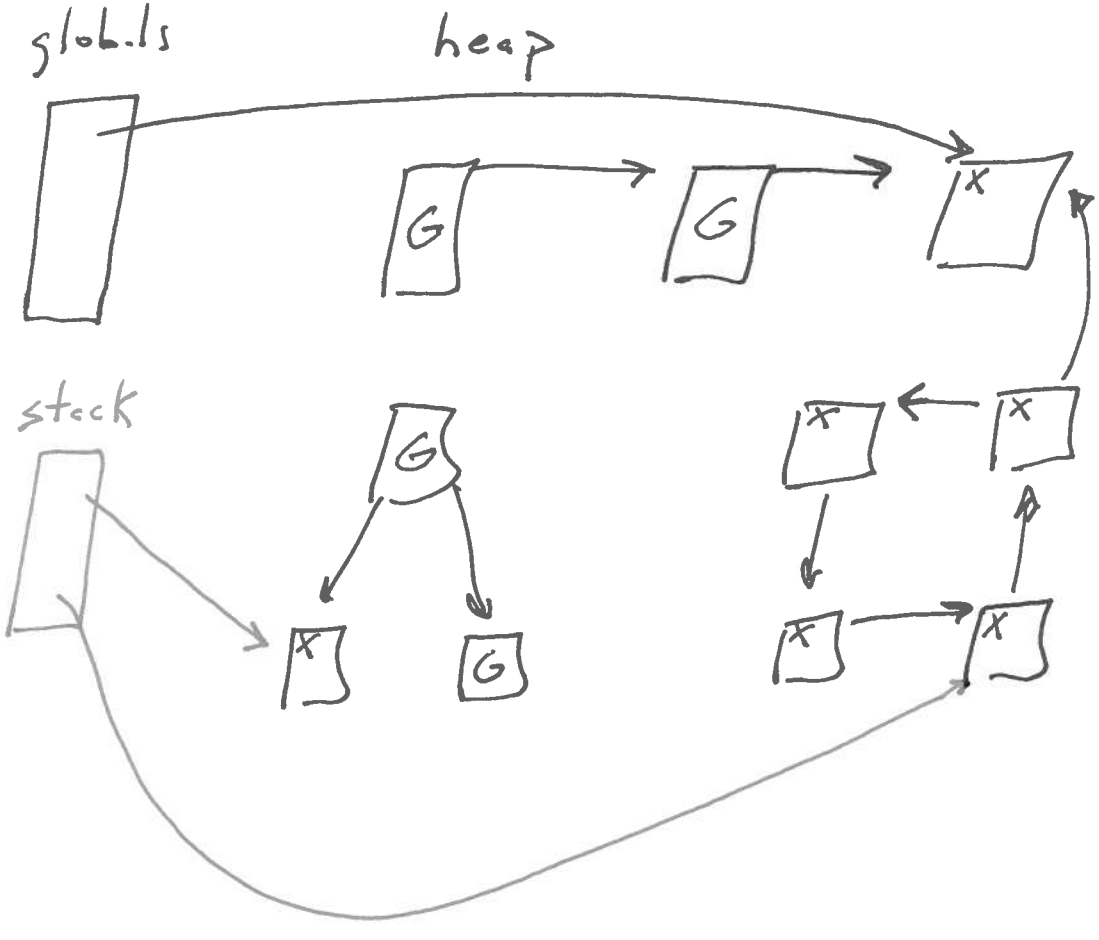
stack

on Intel follow the %ebp chains



pointers inside reachable blocks

consider these user data structures



note: need to remember to clear mark bits

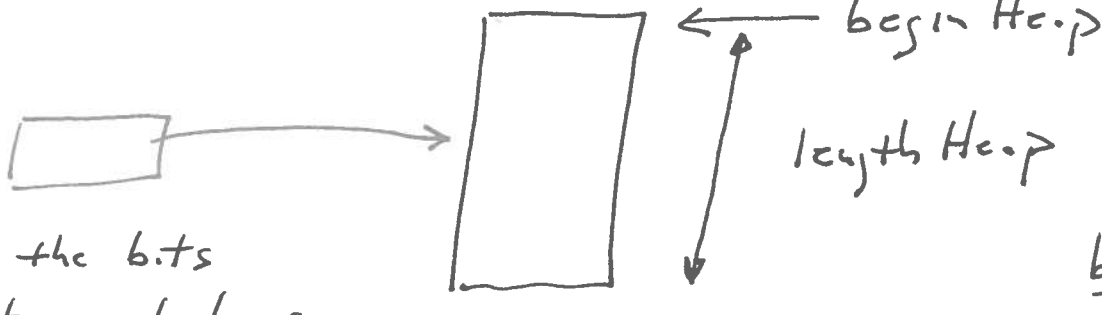
what is garbage?

Conservative Garbage Collection

Java has run-time type info
so it knows exactly where
the pointers are

What if you do not have run-time
type info?

must assume that bits that look like
a pointer into the heap are a
pointer into the heap



i.e. the bits
interpreted as
a pointer

but might actually be
an int or float

Conservative

This is conservative.

↳ might mark a block that does need to be marked

ok to overmark — heap utilization issue

not ok to undermark — correctness issue

advanced topics

concurrent garbage collection

incremental garbage collection

generational garbage collection

resource: the Garbage Collection Page
maintained by Richard Jones