TDDA69 Data and Program Structure
Garbage Collection

Cyrille Berger

LINKÖPING UNIVERSITY

Lectures

Lecture content

- Memory Management
  - Garbage Collection
    - Allocation Algorithms
    - Garbage Collection Algorithms

Memory Management

LINKÖPING UNIVERSITY

Drawbacks of imperative programming

- Difficulty in reasoning
- Side effects
- Concurrency
- Memory management
  - Compared to functional programming, states outlives function calls

Garbage Collection

Low-level

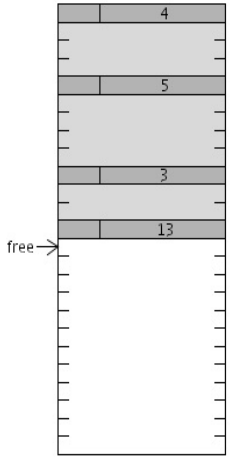
- Two operations
  - malloc: allocate a continuous array of bytes
  - free: disallocate the array
- Problem: people forget to free memory, which introduces leaks in the program
- Solution: use a garbage collector

Garbage Collection

- Automatically free unused memory
- Need an allocation algorithm
- Need a garbage collection algorithm

Allocation Algorithms

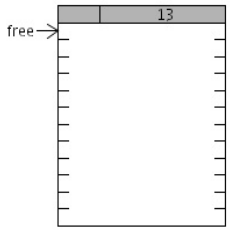
One Big Free Space (1/2)



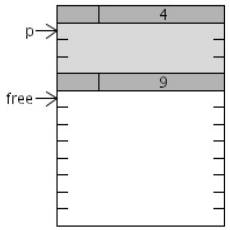
- Pros
  - Very simple and fast allocation algorithm
  - Free space not subject to fragmentation
- Cons
  - Requires a compacting garbage collection algorithm (slower)
  - That is, in-use blocks have to be shifted and pointers have to be adjusted during garbage collection

One Big Free Space (2/2)

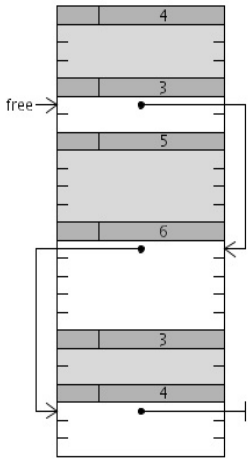
- Before:



- After:



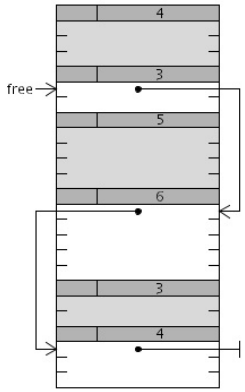
Free List



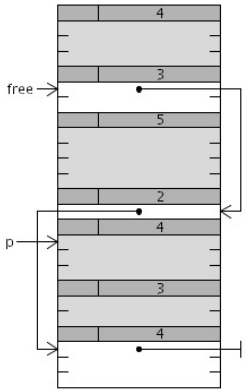
- Pros
  - Can use a non-compacting garbage collection algorithm (faster)
  - That is, in-use blocks do not have to be shifted and pointers do not have to be adjusted during garbage collection
- Cons
  - More complicated and slower allocation algorithm
  - Free space subject to fragmentation
  - In the figure, you can't allocate a 10-word block, even though there are 10 total free words

First fit Allocation

- Before:

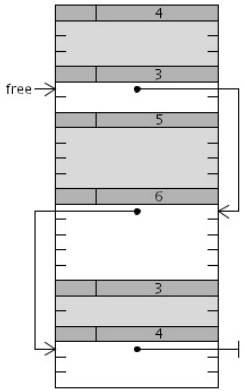


- After (allocate 3):

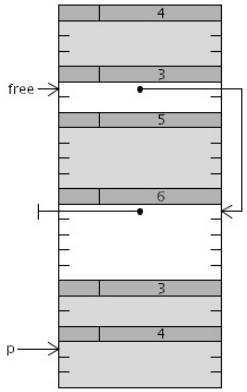


Best fit Allocation

- Before:



- After (allocate 3):

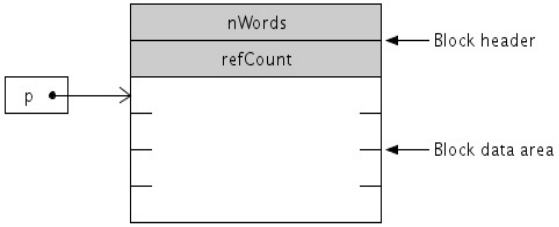


When allocation fails

- Try to allocate
  - If there's no free block big enough...
- Do a garbage collection, then try to allocate again
  - If there's still no free block big enough...
- Increase the size of the heap (if possible), then try to allocate again
  - If you can't increase the size of the heap...
- OutOfMemoryError!

Garbage Collection Algorithms

Reference Counting (1/2)

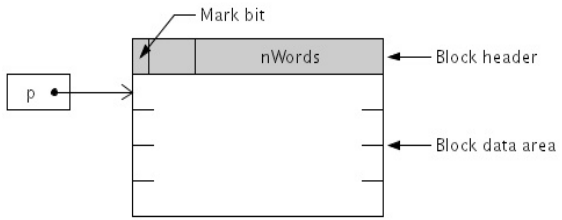


- Header
  - nWords: the size of the block area
  - refCount: The number of pointers pointing to this block

Reference Counting (2/2)

- Pros
  - Very simple, non-compacting garbage collection
  - Heap maintenance spread throughout program execution (instead of suspending the program when the garbage collector runs)
- Cons
  - Extra word in block header to hold reference count
  - Fragile; if you forget to adjust reference counts on any pointer assignment (including passing pointers as subroutine arguments), disaster can happen
  - Major problem: Cannot garbage collect circularly linked data structures

Mark-Sweep (1/2)

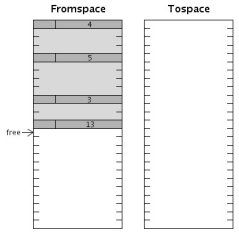


- Header
  - nWords: the size of the block area
  - mark: 1 if in use, 0 if garbage
- Algorithm:
  - Mark all blocks reachable from the root
  - At this point, all unmarked blocks are unreachable, hence garbage
  - Sweep all unmarked blocks into the free list (non-compacting)

Mark-Sweep (2/2)

- Pros
  - Minimal overhead in block header
  - Heap maintenance not required on every pointer assignment (unlike reference counting)
  - Can garbage collect circularly linked data structures
- Cons
  - Doesn't deal with heap fragmentation
  - Suspend the entire system during collection

Copying Garbage



- Variation of the mark-sweep
- Copy in-use blocks from the fromspace to the top of the tospace (compacting)
- Pros
  - One big free space organization means very simple allocation
  - Compacting garbage collection means heap fragmentation does not occur
- Cons
  - Heap requires twice the memory that would otherwise be needed — most of the time, half of this memory is "wasted"

Tri-color marking

- Three colors:
  - White: candidate for deletion
  - Black: objects with no connection to white objects and reachable from root
  - Gray: objects reachable from root but with possible references to white objects

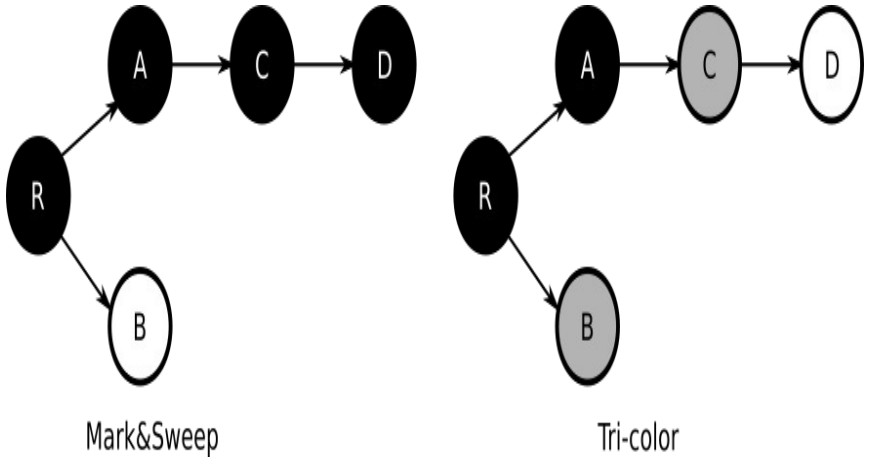
Tri-color marking - Algorithm

- Initialisation
  - White set: all objects not directly reachable from root
  - Black set: empty
  - Gray set: all objects directly reachable from root
- Pick an object in the gray set
  - Mark as gray all the white objects it references
  - Mark as black this object
- End when gray set is empty

Tri-color marking - Advantage

- Can be performed *on-the-fly*
  - No need for freezing the application

mark&swee vs tri-color marking



Generational

- In many programs, the newest objects are the most likely to become unreachable
- Hence the idea to only put recent objects in the white set

Summary

- Memory allocation
- Garbage collection