Memory Management

Andrew C. Lee

EECS, Syracuse

Contents

Background

1. Responsibility of a typical operating systems
   - 1.1 Maintain free memory blocks
   - 1.2 Assign specific memory blocks to user programs when needed
   - 1.3 Release memory from unneeded blocks; return them back to the memory pool
2. Contiguous Memory Allocation in heaps (not the heap data structure)
3. External Fragmentation
4. Internal Fragmentation

Question Can data structure help ?

Sequential Fit Methods

FIGURE 12.1 Memory allocation using sequential-fit methods.

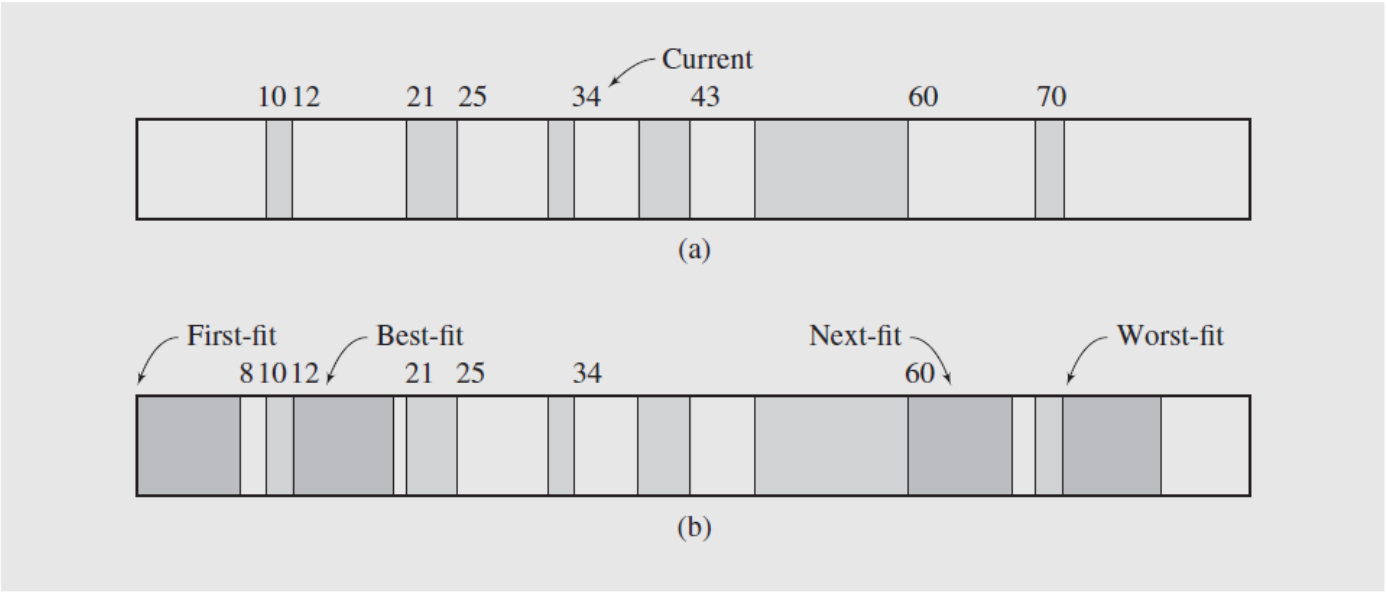

Figure : Sequential Fit Methods

Note: Reduce external fragmentation

Sequential Fit Methods

Maintain a list of available memory. Use one the following *online* algorithms to handle memory allocation requests:

1. First Fit: allocate the *first* block of memory that is large enough
2. Best Fit: allocates a block that is closest in size to the request
3. Next Fit: allocate the *second* block of memory that is large enough
4. Worst Fit: finds the largest block on the list available memory blocks
5. Many other variants

A Variant: Adaptive Exact-Fit

1. Maintain a *size-list* of block lists of a particular size b returned to the memory pool during the last T allocations is maintained
2. A block b is added to a particular block list when
   - 2.1 the block list holds blocks of memory of size b
   - 2.2 b has been returned by the program.
   - 2.3 For request comes for a block of size b: Allocate a block from the block list to meet the request
   - 2.4 For other requests, it triggers a time-consuming search for a block in memory via a sequential-fit method.

Adaptive Exact-Fit: Illiustrations

FIGURE 12.2 An example configuration of a size-list and heap created by the adaptive exact-fit method.

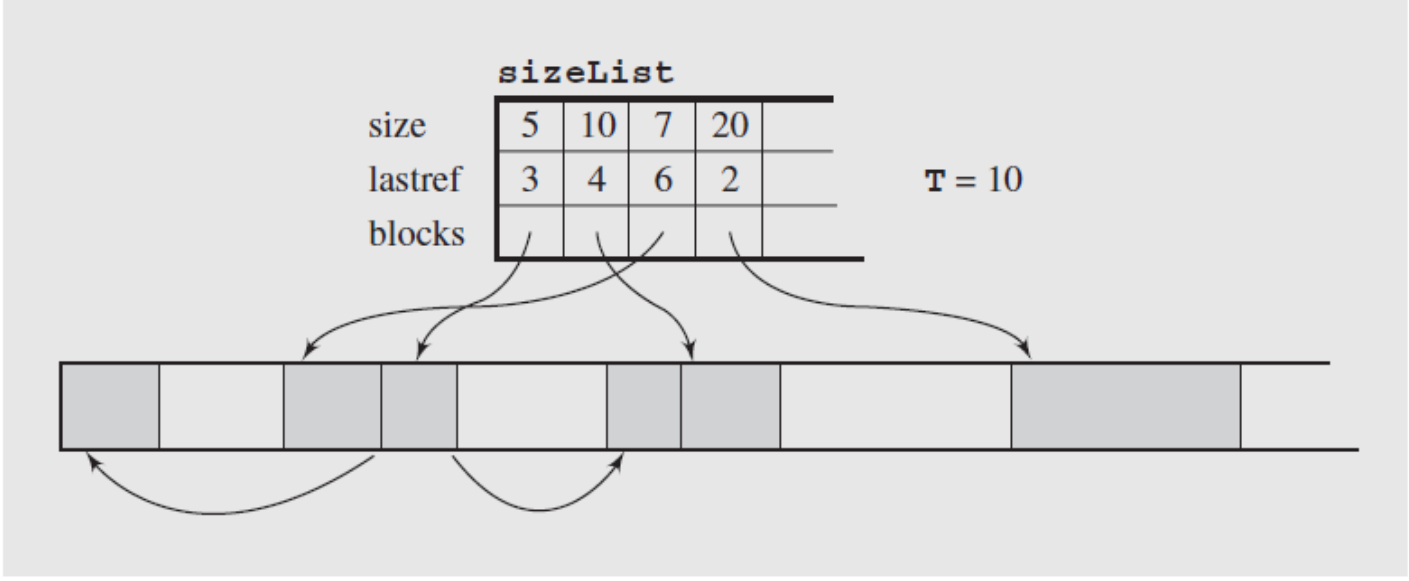

Figure : A block list

Non Sequential FIt Methods

Example: Binary buddy Memory systems
Ideas

1. Storage is of size 2^m locations for some integer m
2. location addresses are $0, \ldots, 2^m - 1$
3. Use an array, say `avail[]` such that, for each $i = 0, ..., m$, `avail[i]` is the head of a doubly linked list of blocks of the same size 2^i .

Block Structure

FIGURE 12.3 Block structure in the binary buddy system.

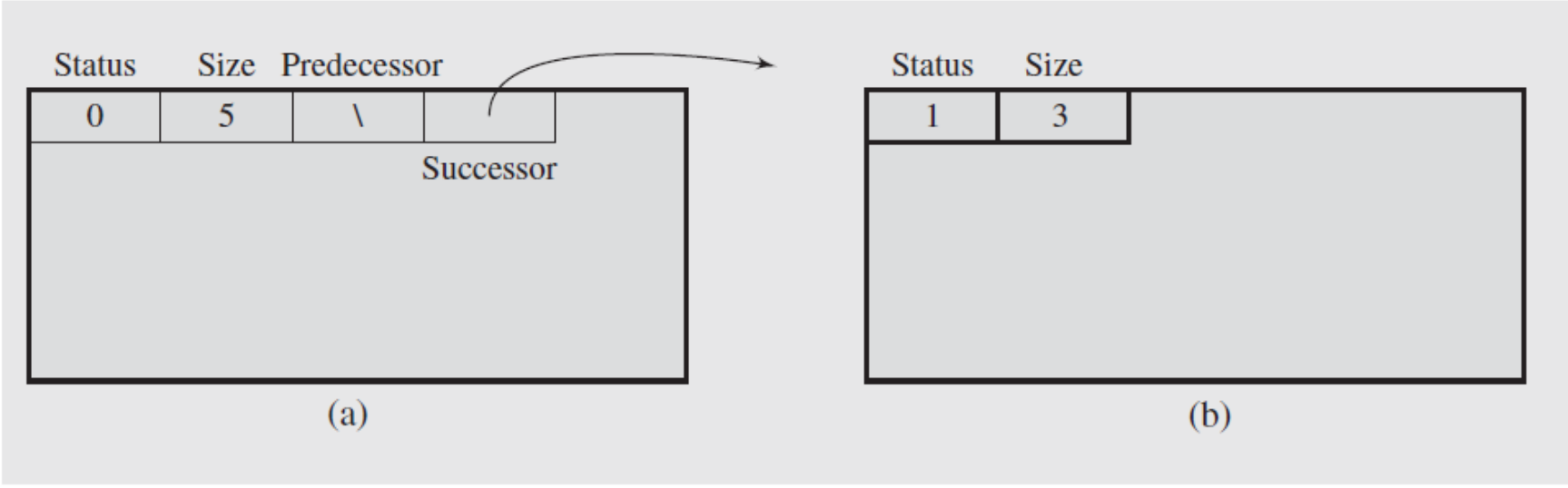

Figure : Block Structure used in Binary Buddy System

Example See the wikipedia example and CNPS (sections on memory management).

Garbage Collection I

Garbage Collection: Determine which part of storage is no longer being referenced and return it to the pool of storage

1. Java:
   An automatic process; JVM manage the runtime memory used by programs
2. C++:
   Not built in. See the use of smart pointers (see CNPS memory management sections)

Garbage Collection II

1. Mark and Sweep: Two phases
   1.1 The marking phase: identify all cells currently being used
   1.2 The Reclamation phase: return all unmarked cells to the memory pool
2. Copying methods

Copying

FIGURE 12.9 (a) A situation in heap before copying the contents of cells in use from semispace$_1$ to semispace$_2$ and (b) the situation right after copying. All used cells are packed contiguously.

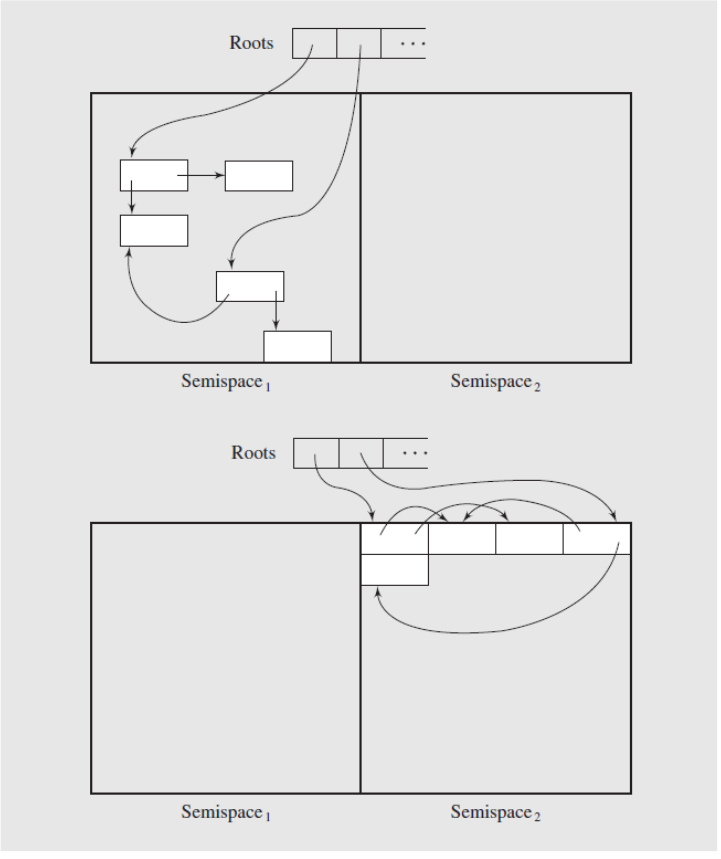

Figure : Use copying in garbage collection