Memory Management: Simple Systems

Bibliography

- Understanding Operating Systems Seventh Edition by Ann McIver, McHoes Ida, M. Flynn; Chapter 2

Learning Objectives

After completing this chapter, you should be able to describe:

- The basic functionality of the four memory allocation schemes presented in this chapter: single user, fixed partitions, dynamic partitions, and relocatable dynamic partitions
- Best-fit memory allocation as well as first-fit memory allocation
- How a memory list keeps track of available memory

Learning Objectives (cont'd.)

- The importance of memory deallocation
- The importance of the bounds register in memory allocation schemes
- The role of compaction and how it can improve memory allocation efficiency

Introduction

- Main memory management is critical
- Entire system performance is dependent on two items:
  - Amount of memory available
  - Optimization of memory during job processing
    - **Memory optimization** is a range of techniques related to improving computer memory, such as identifying memory leaks and corruption, to optimize memory usage and increase performance and application usability
    - Memory leak is a type of resource leak that occurs when a computer program incorrectly manages memory allocations in a way that memory which is no longer needed is not released. A memory leak may also happen when an object is stored in memory but cannot be accessed by the running code

Introduction (cont'd.)

- This chapter introduces:
  - Role of main memory (RAM)
  - Four types: memory allocation schemes
    - Single-user systems
    - Fixed partitions
    - Dynamic partitions
    - Relocatable dynamic partitions

Single-User Contiguous Scheme

- Entire program: loaded into memory
- Contiguous memory space: allocated as needed
- Jobs: processed sequentially
- Memory Manager: performs minimal work
  1. Evaluates incoming process size: loads if small enough to fit; otherwise, rejects and evaluates next incoming process
  2. Monitors occupied memory space; when process ends, makes entire memory space available and returns to Step 1

Single-User Contiguous Scheme (cont'd.)

- Disadvantages
  - Multiprogramming or networking not supported
  - Not cost effective: unsuitable for business when introduced in late 1940s and early 1950s

Memory Allocation Schemes

- Single-user systems
- Fixed partitions
- Dynamic partitions
- Relocatable dynamic partitions

Fixed Partitions

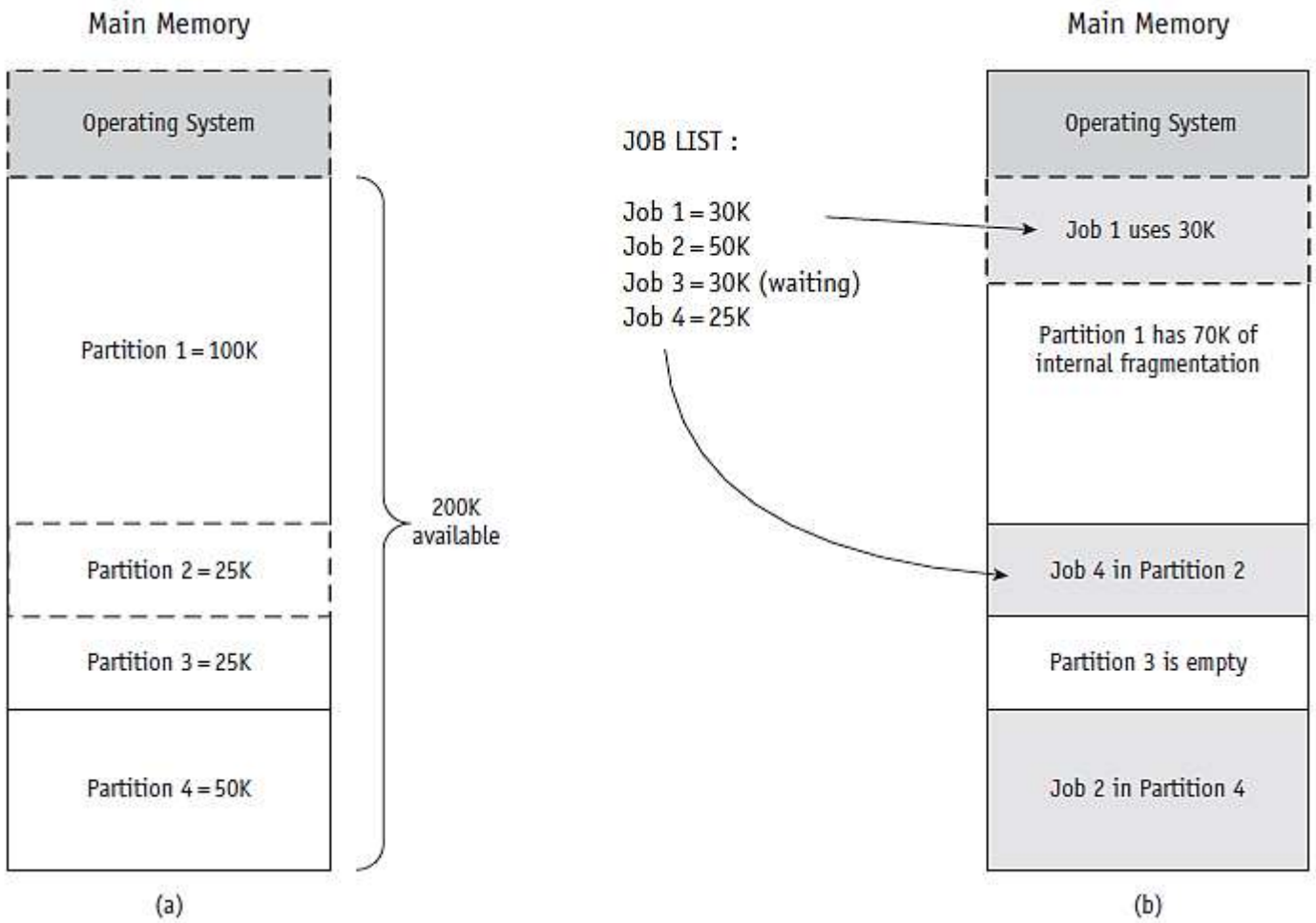
- Permits multiprogramming
- Main memory: partitioned
  - Each partition: one job
  - Static: reconfiguration requires system shut down
- Responsibilities
  - Protecting each job's memory space
  - Matching job size with partition size

Fixed Partitions (cont'd.)

- Memory Manager: allocates memory space to jobs
  - Information: stored in a table

Partition Size	Memory Address	Access	Partition Status
100K	200K	Job 1	Busy
25K	300K	Job 4	Busy
25K	325K		Free
50K	350K	Job 2	Busy

(table 2.1)
A simplified fixed-partition memory table with the free partition shaded.



(figure 2.2)
As the jobs listed in Table 2.1 are loaded into the four fixed partitions, Job 3 must wait even though Partition 1 has 70K of available memory. Jobs are allocated space on the basis of "first available partition of required size."

Fixed Partitions (cont'd.)

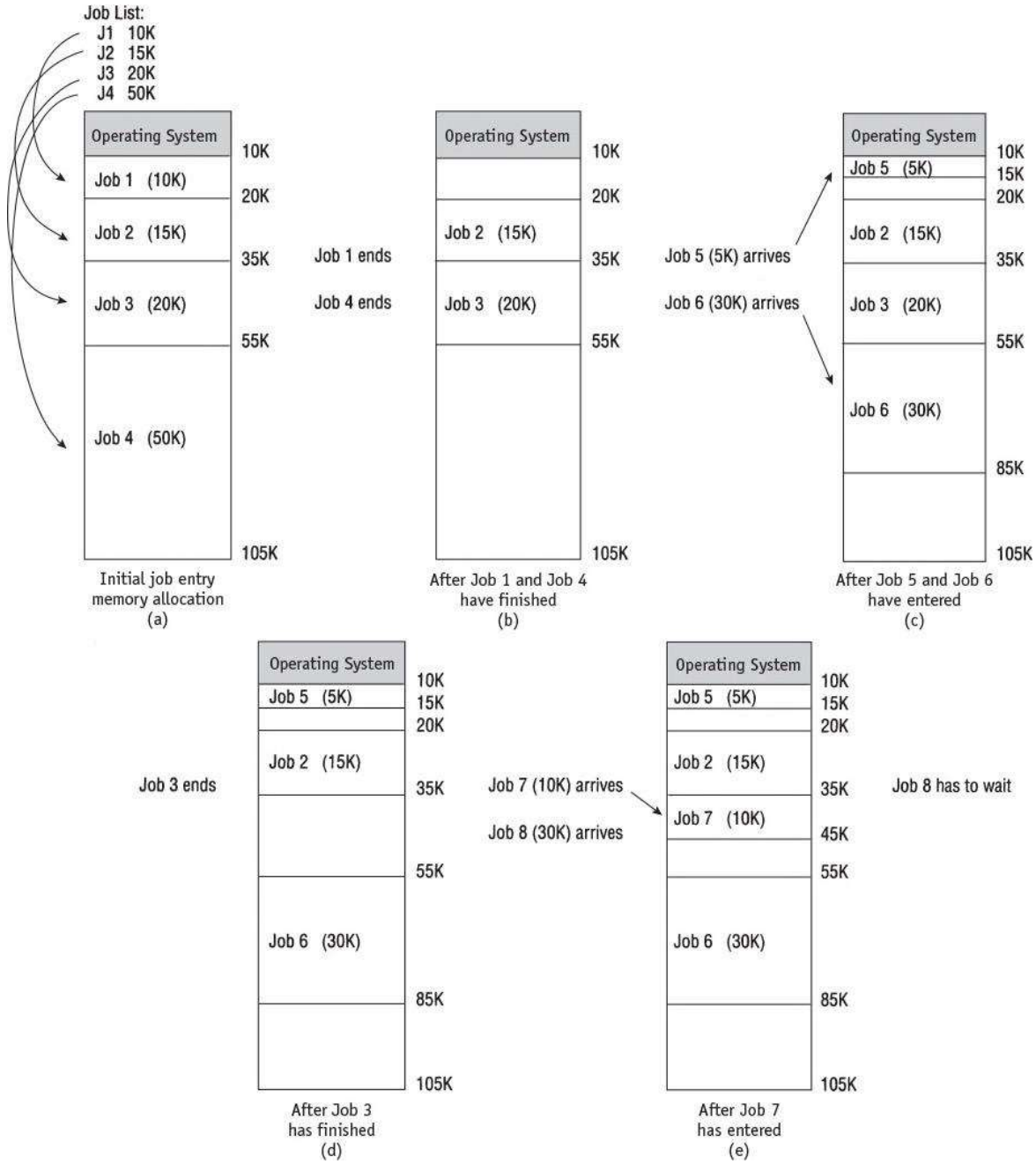
- Characteristics
  - Requires contiguous loading of entire program
  - Job allocation method
    - First available partition with required size
  - To work well:
    - All jobs have similar size and memory size known ahead of time
  - Arbitrary partition size leads to undesired results
    - Partition too small
      - Large jobs have longer turnaround time
    - Partition too large
      - Memory waste: internal fragmentation

Memory Allocation Schemes

- Single-user systems
- Fixed partitions
- Dynamic partitions
- Relocatable dynamic partitions

Dynamic Partitions

- Main memory: partitioned
  - Jobs: given requested memory when *loaded*
  - One contiguous partition per job
- Job allocation method
  - First come, first serve
  - Memory waste: comparatively small within partitions
- Disadvantages
  - Full memory utilization: only when first jobs loaded
  - Subsequent allocation: memory waste
    - External fragmentation: fragments between blocks



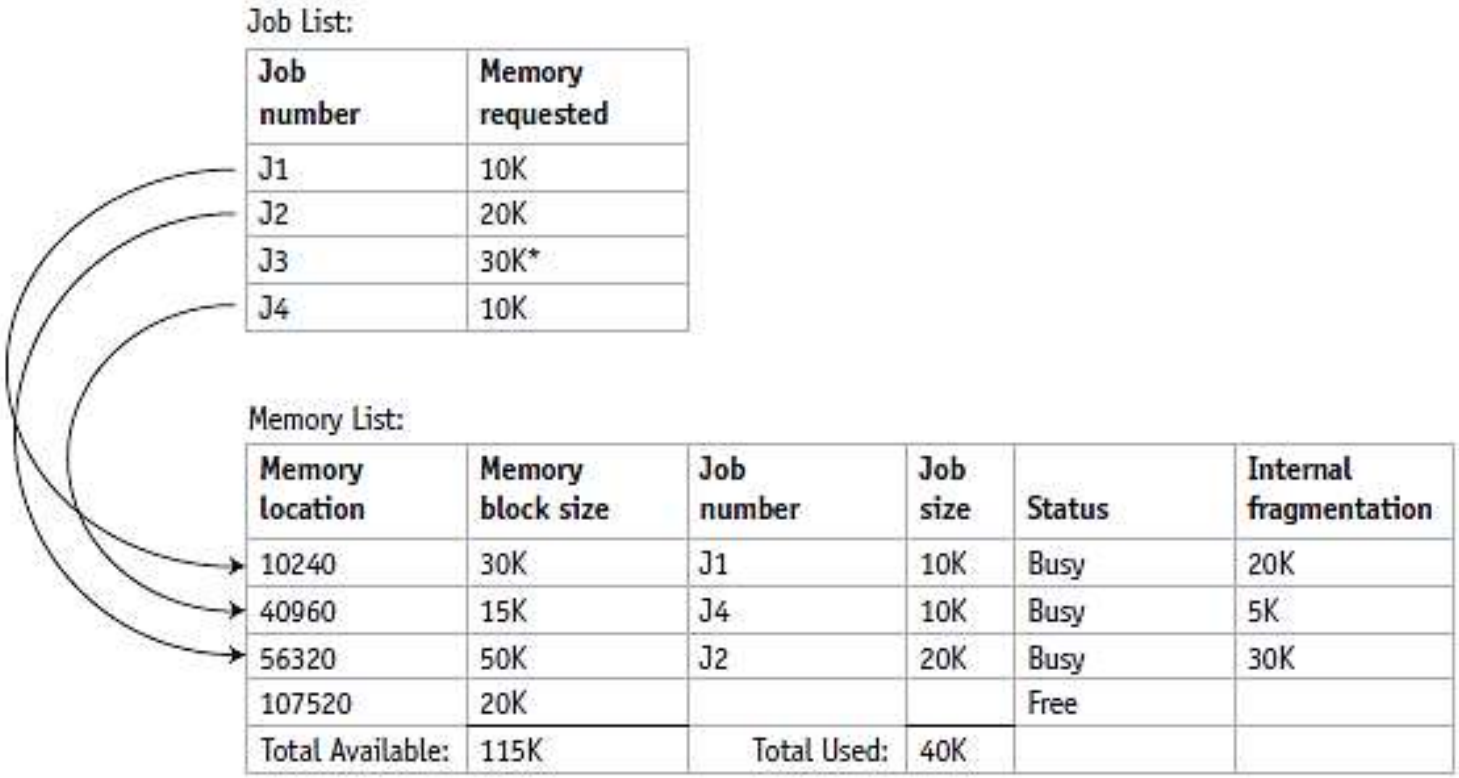
(figure 2.3) Main memory use during dynamic partition allocation. Five snapshots (a-e) of main memory as eight jobs are submitted for processing and allocated space on the basis of "first come, first served." Job 8 has to wait (e) even though there's enough free memory between partitions to accommodate it.

Best-Fit and First-Fit Allocation

- Two methods for free space allocation
  - First-fit memory allocation
    - Memory Manager: free/busy lists organized by memory locations (low- to high-order memory)
    - Job: assigned first partition large enough
    - Fast allocation
  - Best-fit memory allocation
    - Memory Manager: free/busy lists ordered by size (smallest to largest)
    - Job: assigned smallest partition large enough
    - Least wasted space; internal fragmentation reduced

Best-Fit and First-Fit Allocation (cont'd.)

- Fixed and dynamic memory allocation schemes: use both methods
- First-fit memory allocation
  - Advantage: faster allocation
  - Disadvantage: memory waste
- Best-fit memory allocation
  - Advantage: best use of memory space
  - Disadvantage: slower allocation



Job List:

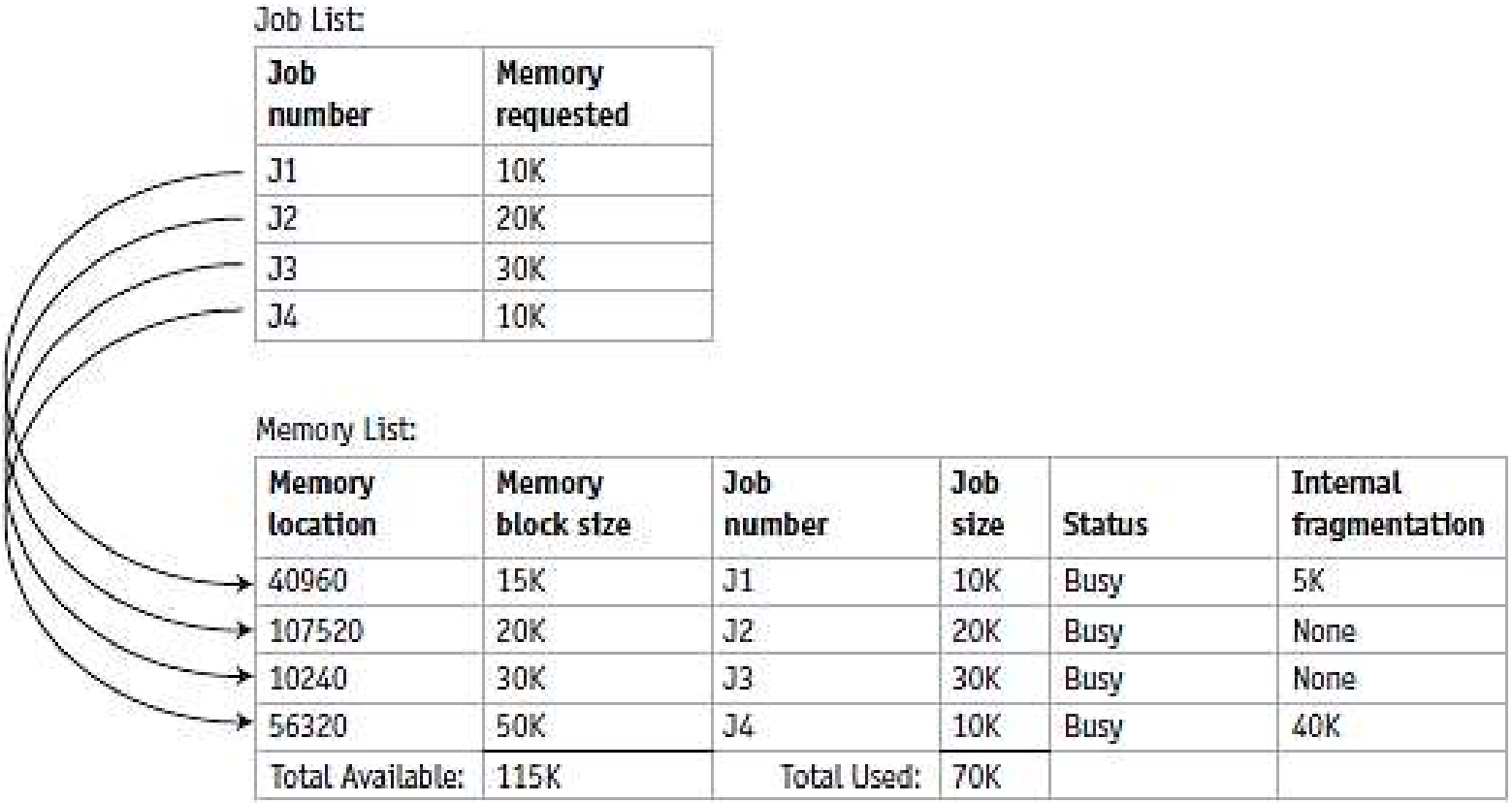
Job number	Memory requested
J1	10K
J2	20K
J3	30K*
J4	10K

Memory List:

Memory location	Memory block size	Job number	Job size	Status	Internal fragmentation
10240	30K	J1	10K	Busy	20K
40960	15K	J4	10K	Busy	5K
56320	50K	J2	20K	Busy	30K
107520	20K			Free	
Total Available:	115K	Total Used:	40K		

(figure 2.5)

Using a first-fit scheme, Job 1 claims the first available space. Job 2 then claims the first partition large enough to accommodate it, but by doing so it takes the last block large enough to accommodate Job 3. Therefore, Job 3 (indicated by the asterisk) must wait until a large block becomes available, even though there's 75K of unused memory space (internal fragmentation). Notice that the memory list is ordered according to memory location.



Job List:

Job number	Memory requested
J1	10K
J2	20K
J3	30K
J4	10K

Memory List:

Memory location	Memory block size	Job number	Job size	Status	Internal fragmentation
40960	15K	J1	10K	Busy	5K
107520	20K	J2	20K	Busy	None
10240	30K	J3	30K	Busy	None
56320	50K	J4	10K	Busy	40K
Total Available:	115K	Total Used:	70K		

(figure 2.6)
Best-fit free scheme. Job 1 is allocated to the closest-fitting free partition, as are Job 2 and Job 3. Job 4 is allocated to the only available partition although it isn't the best-fitting one. In this scheme, all four jobs are served without waiting. Notice that the memory list is ordered according to memory size. This scheme uses memory more efficiently but it's slower to implement.

Best-Fit and First-Fit Allocation (cont'd.)

- First-fit algorithm
  - Memory Manager: keeps two lists
    - One for free memory
    - One for busy memory blocks
  - Loop: compares job size to each memory block size
    - Until large enough block found: fits the job
  - Job: stored into that memory block
  - Memory Manager: moves out of the loop
    - Fetches next job: entry queue

Best-Fit and First-Fit Allocation (cont'd.)

- First-fit algorithm (cont'd.)
  - If entire list searched: no memory block large enough
    - Job: placed into waiting queue
    - Memory Manager: fetches next job
  - Process repeats

Status Before Request		Status After Request	
Beginning Address	Free Memory Block Size	Beginning Address	Free Memory Block Size
4075	105	4075	105
5225	5	5225	5
6785	600	*6985	400
7560	20	7560	20
7600	205	7600	205
10250	4050	10250	4050
15125	230	15125	230
24500	1000	24500	1000

(table 2.2)
These two snapshots of memory show the status of each memory block before and after 200 spaces are allocated at address 6785, using the first-fit algorithm. (Note: All values are in decimal notation unless otherwise indicated.)

Best-Fit Versus First-Fit Allocation (cont'd.)

- Best-fit algorithm
  - Goal: find smallest memory block where job fits
  - Entire table searched before allocation

Status Before Request		Status After Request	
Beginning Address	Free Memory Block Size	Beginning Address	Free Memory Block Size
4075	105	4075	105
5225	5	5225	5
6785	600	6785	600
7560	20	7560	20
7600	205	*7800	5
10250	4050	10250	4050
15125	230	15125	230
24500	1000	24500	1000

(table 2.3)
These two snapshots of memory show the status of each memory block before and after 200 spaces are allocated at address 7600, using the best-fit algorithm.

Best-Fit Versus First-Fit Allocation (cont'd.)

- Hypothetical allocation schemes
  - Next-fit: starts searching from last allocated block for next available block
  - Worst-fit: allocates largest free available block
    - Opposite of best-fit
    - Good way to explore theory of memory allocation
    - Not best choice for an actual system