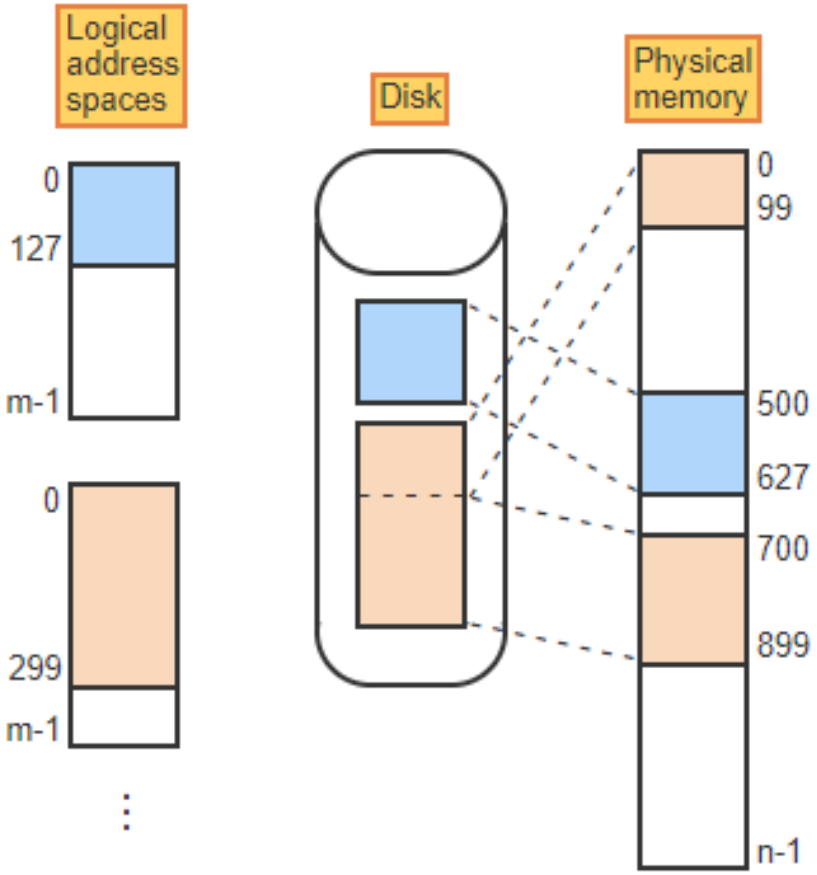
Memory Management

CS3600 Spring 2022

Memory

- **Logical vs physical memory**
  - A computer's ***physical memory*** (***RAM***) is a hardware structure consisting of a linear sequence of words that hold a program during execution.
  - A ***logical address space*** is an abstraction of physical memory, consisting a sequence of imaginary memory locations in a range [0 : m-1], where m is the size of the logical address space.

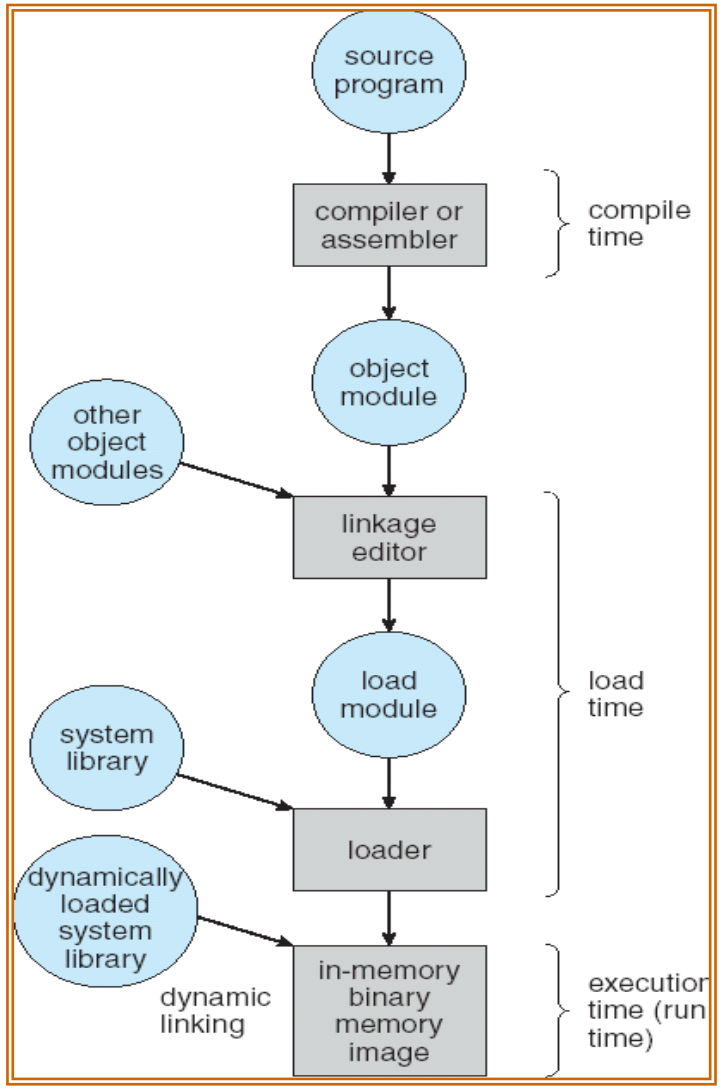


Program transformations

- A ***source module*** is a program, or a program component written in a language, like C, or an assembly language, that must be translated by a compiler or assembler into executable machine code.

- An ***object module*** is the machine-language output of a compiler or assembler generated from a source module. An object module may be self-contained and executable or multiple object modules may be linked together into a load module by a linker or linkage editor.

- A ***load module*** is a program or combination of programs in a form ready to be loaded into main memory and executed.



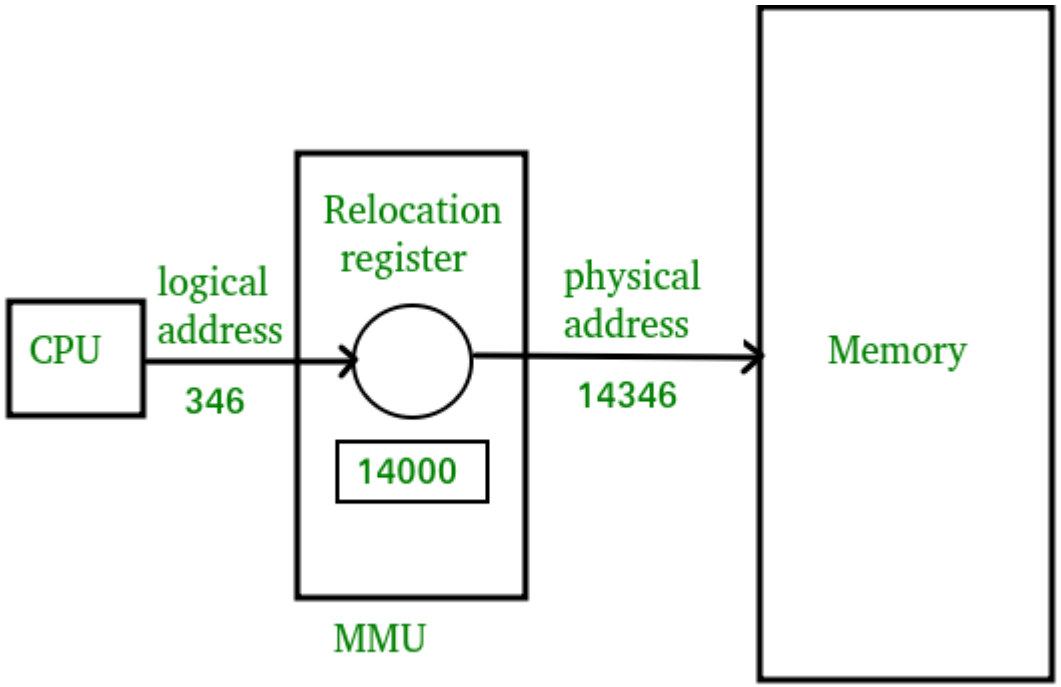
Relocation and address binding

Program ***relocation*** is the process of moving a program component from one address space to another. The relocation may be between two logical address spaces or from a logical address space to a physical address space.

Static relocation binds all logical addresses to physical addresses prior to execution.

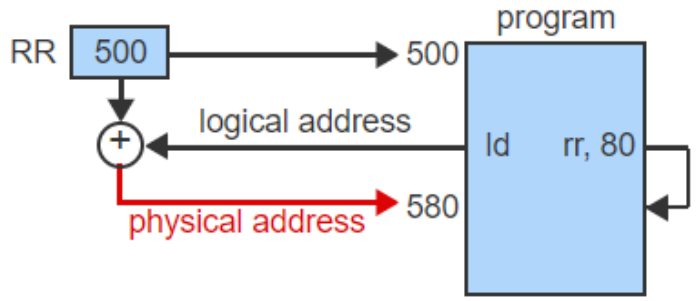
Dynamic relocation postpones the binding of a logical address to a physical address until the addressed item is accessed during execution.

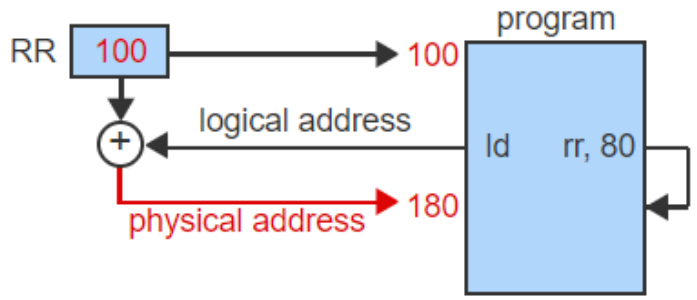
Static relocation



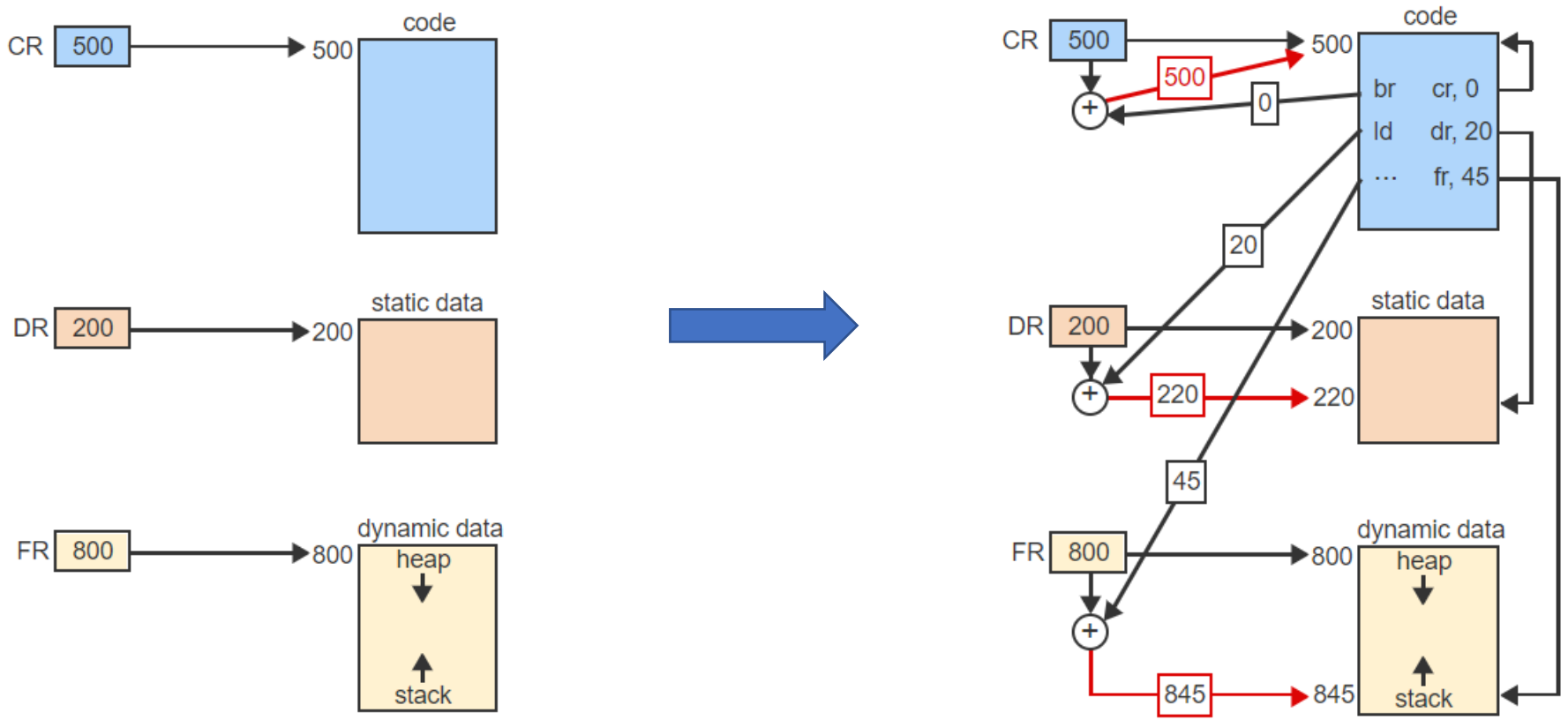
Dynamic relocation

- A ***relocation register*** contains the physical starting address of a program or program component in memory.
- Programs
  - code, static data, and dynamic data.
- Dynamic relocation can be implemented using a single relocation register, which is loaded with the program's starting address.
- A more flexible management scheme treats the 3 components as separate modules, each of which may reside in a different area of memory. Three relocation registers, each loaded with the starting address of one of the modules, then accomplish dynamic relocation by being added to all logical addresses at runtime





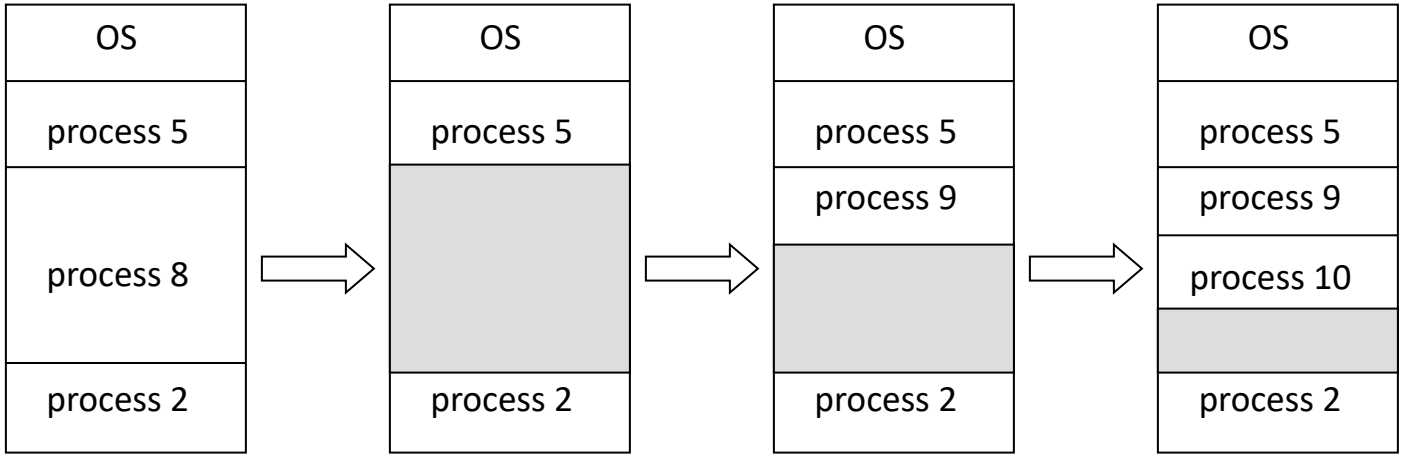
Dynamic Relocation



Memory Allocation

- *Hole* – block of available memory; holes of various size are scattered throughout memory
- When a process arrives, it is allocated memory from a hole large enough to accommodate it
- Operating system maintains information about:
  a) allocated partitions

  b) free partitions (hole)



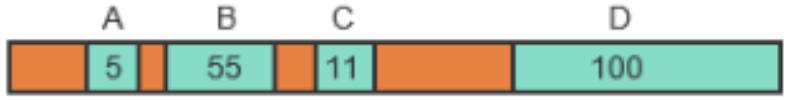
Free space management

- The OS must also coalesce any neighbouring holes resulting from the removal of programs from memory to prevent the memory from becoming a fragmented collection of increasingly smaller holes.

- Different search strategies have been explored:
  - ***First-fit*** always starts the search from the beginning of the list and allocates the first hole large enough to accommodate the request.
  - ***Next-fit*** starts each search at the point of the last allocation.
  - ***Best-fit*** searches the entire list and chooses the smallest hole large enough to accommodate the request.
  - ***Worst-fit*** takes the opposite approach from best-fit by always choosing the largest available hole for any request.

Example

- The list contains 4 holes, A through D, with the given sizes.



1. Using First Fit request for 10 bytes will be placed in _____ hole.

2. Best-fit will place a request for 10 bytes into hole _____.

3. Worst-fit will place a request for 10 bytes into hole _____.

4. If the last request was placed into hole D then next-fit will place a subsequent request for 20 bytes into hole _____.

5. Which strategy tends to preserve large holes?

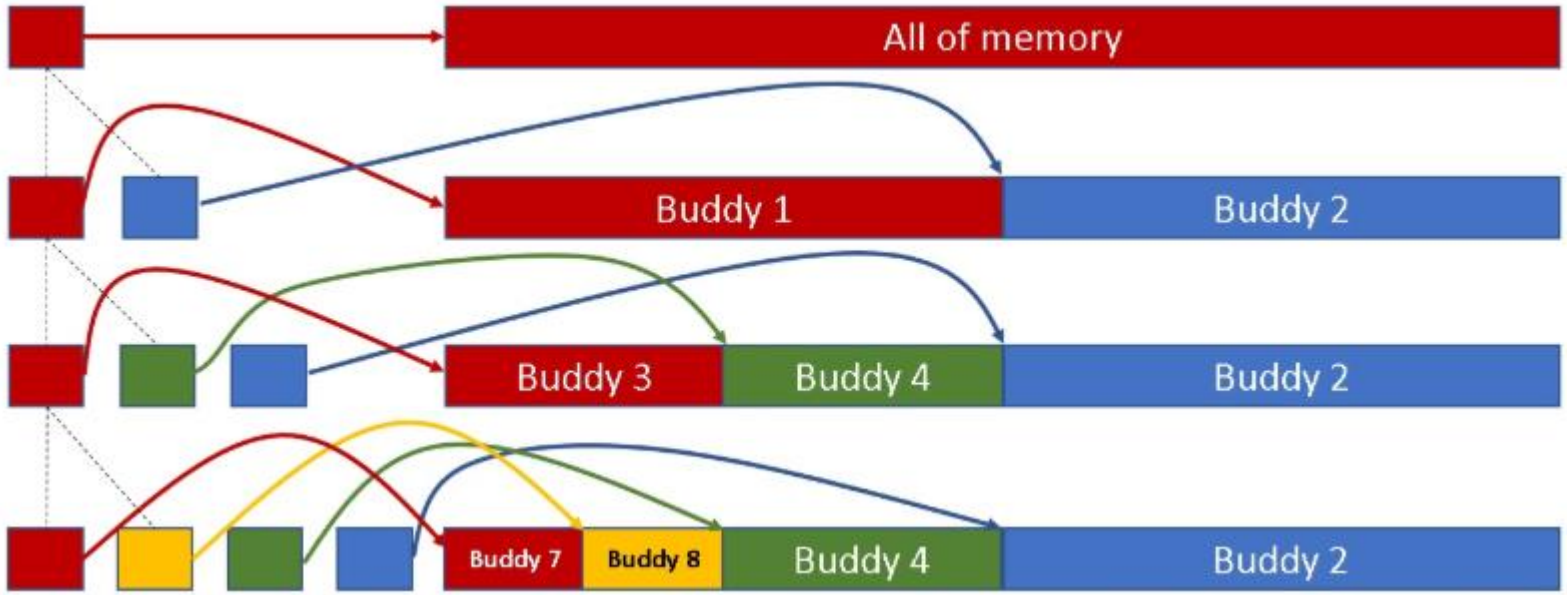
6. Which strategy always needs to search all holes?

Other Approaches (Not it Textbook)

- Slab Allocators – Kernel Memory Allocation
  - Common objects- Semaphores, Process descriptors etc….
    - Jeff Bonwick and Jonathan Adams. Magazines and vmem: Extending the slab allocator to many CPUs and arbitrary resources. In *Proceedings of the 2001 USENIX Annual Technical Conference (USENIX-01)*, pages 15–34, Berkeley, CA, June 25–30 2001

Other Approaches Cont.

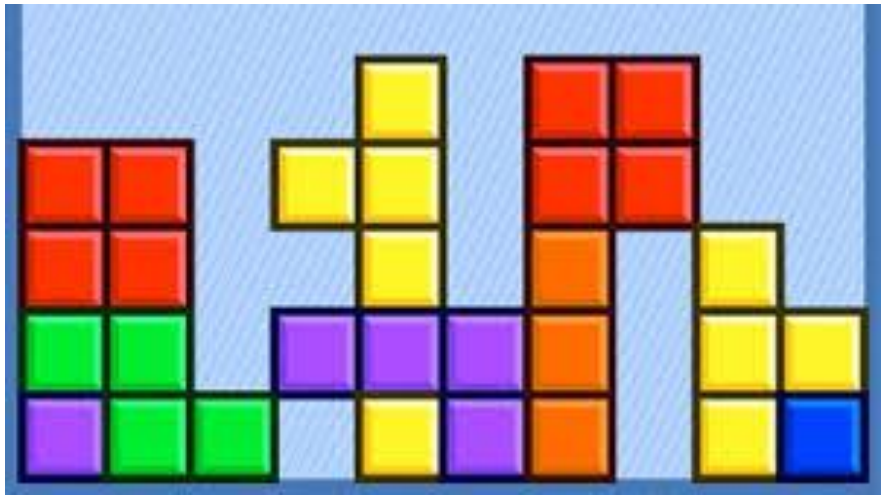
- Buddy Allocation
  - Power of 2 allocation



- Complete Worksheet Q 1 to 4

Managing Insufficient memory

- ***External memory fragmentation*** is the loss of usable memory space due to holes between allocated blocks of variable sizes.



Managing Insufficient memory Cont.

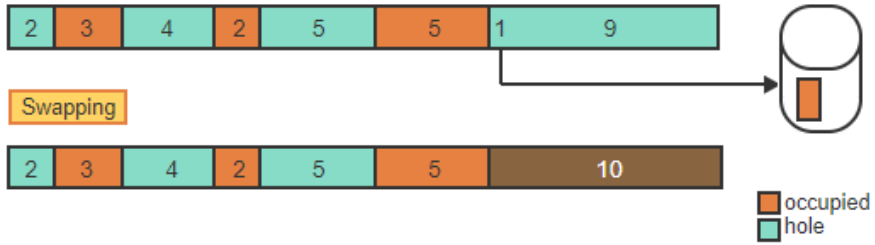
- The ***50% rule*** states that, if the probability of finding an exact match for a request approaches 0, 1/3rd of all memory partitions are holes and 2/3rd

$$n = 0.5\ m$$

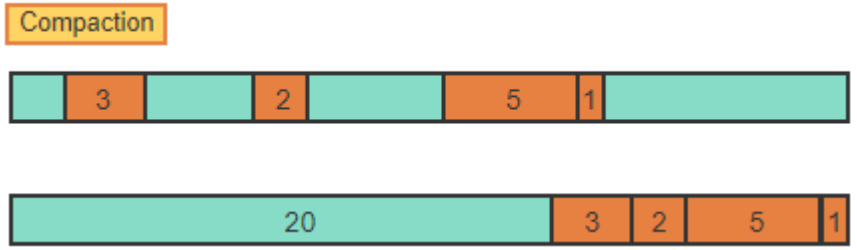
  - where n is the number of holes and m is the number of occupied blocks.

Swapping and Memory compaction

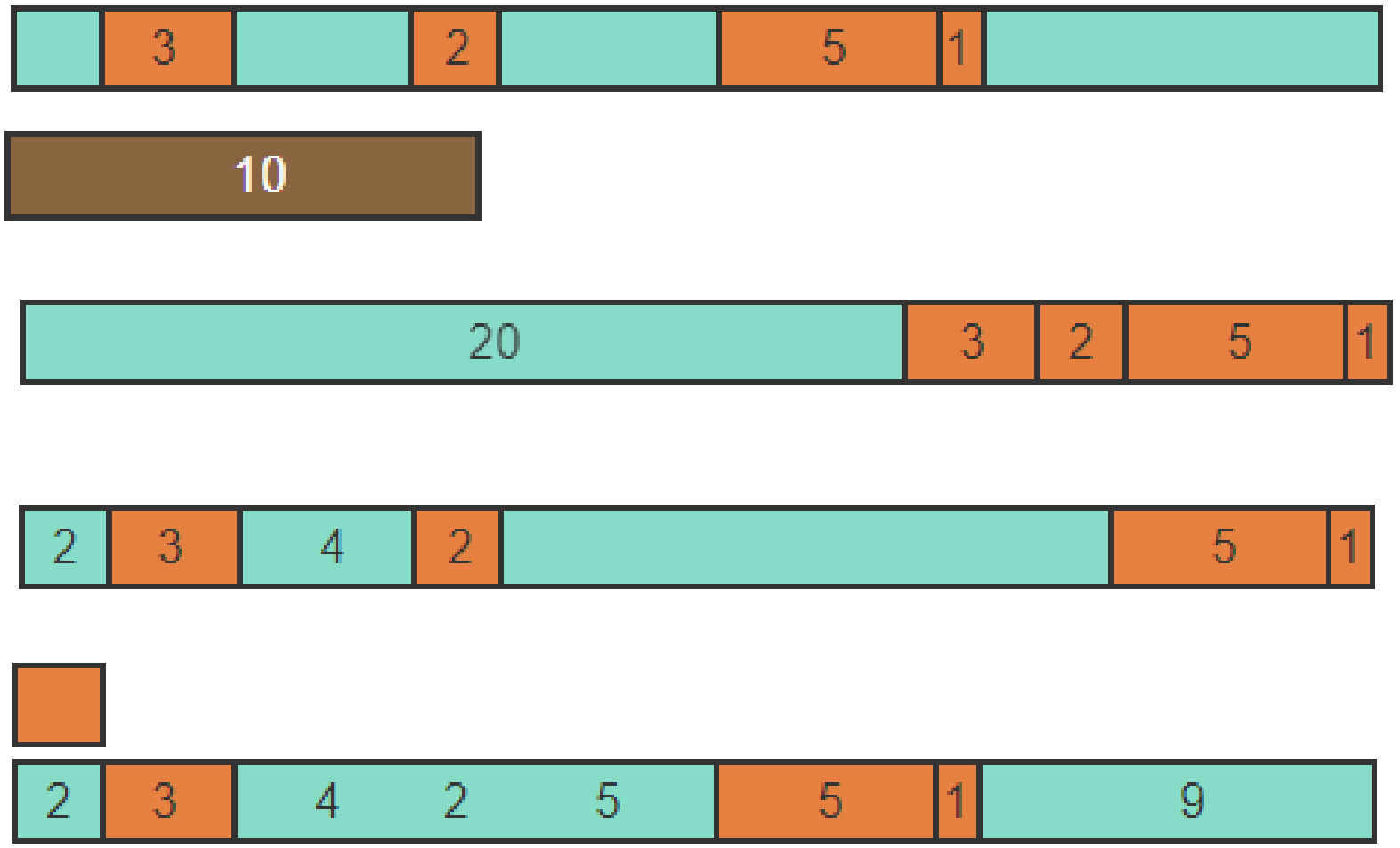
- ***Swapping*** is the temporary removal of a module from memory. The module is saved on a disk and later moved back to memory. Dynamic relocation is necessary so that the module can be placed into a different location without modification.



- ***Memory compaction*** is the systematic shifting of modules in memory, generally in one direction, to consolidate multiple disjoint holes into one larger hole.



Memory compaction



Example

- Memory consists of 3 occupied blocks of size 20 KB each and 4 holes of size 30 KB each:

Moving data between memory and a disk is 3 times slower than moving data within memory so a smart compaction algorithm could create a hole of size 80 by moving at most _____ block(s).

- Complete Worksheet Q5 and Q6

To Do as of 03/29/2022

- Complete Worksheet 07_Memory1
- Next Class: Paging