Lecture 7
Memory Management II

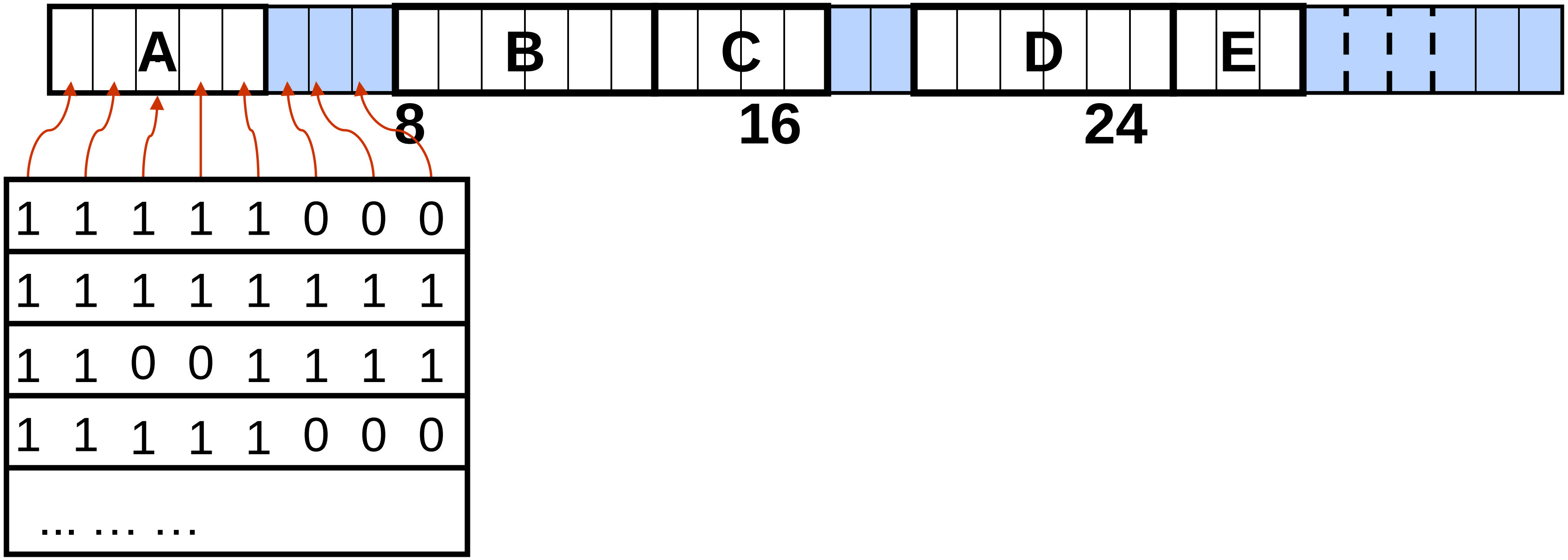
Operating Systems

How to keep track of Memory

- We need to keep track of
  - Allocated memory
  - Free memory
- Memory management with bitmaps
- Memory management with linked lists

Memory management with bitmaps

- Memory is divided up into allocation units
  - □ Few words
  - □ Or several kilobytes



1 1 1 1 1 0 0 0

1 1 1 1 1 1 1 1

1 1 0 0 1 1 1 1

1 1 1 1 1 0 0 0

... ...

Memory management with bitmaps

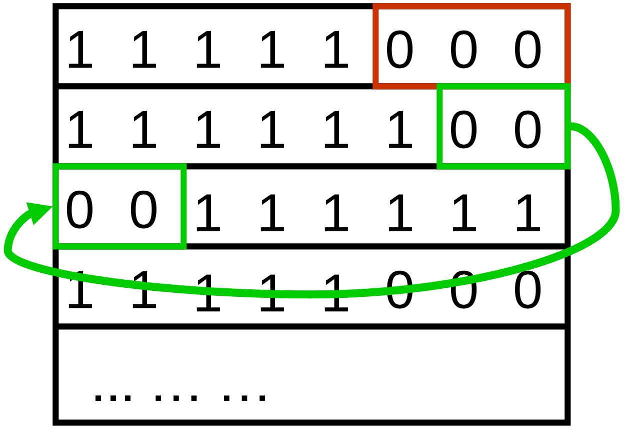
- Size of allocation unit is important
- Smaller allocation unit
  - Larger bitmap required
- Larger allocation unit
  - Smaller bitmap required
  - More memory will be wasted
    - If the process size is not an exact multiple of the allocation unit

Memory management with bitmaps

- To bring a k unit process in memory
- Search for a k run consecutive 0 bits in the map
- Search can be slow
- Since, k run may cross word boundaries

Find run of length = 3

Find run of length = 4



1 1 1 1 1 0 0 0

1 1 1 1 1 1 0 0

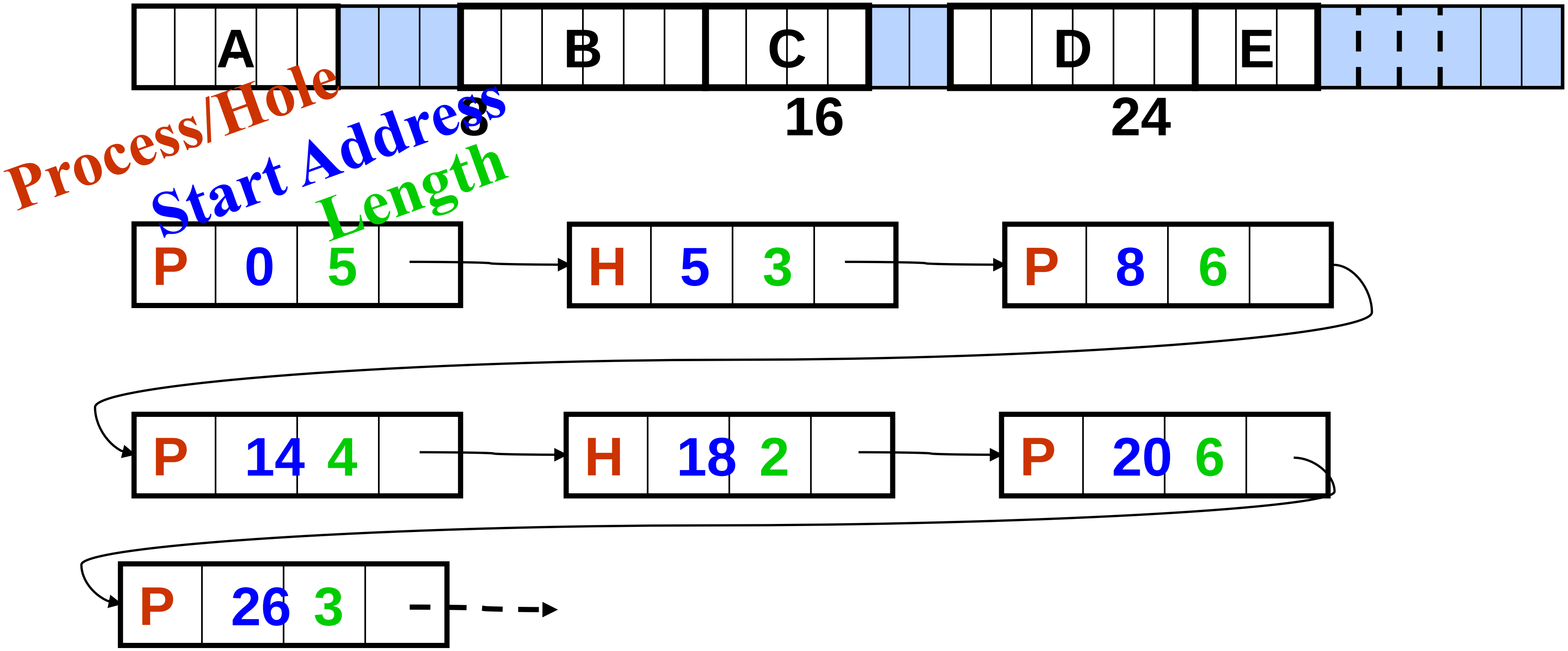
0 0 1 1 1 1 1 1

1 1 1 1 1 0 0 0

... ...

Memory management with Linked Lists

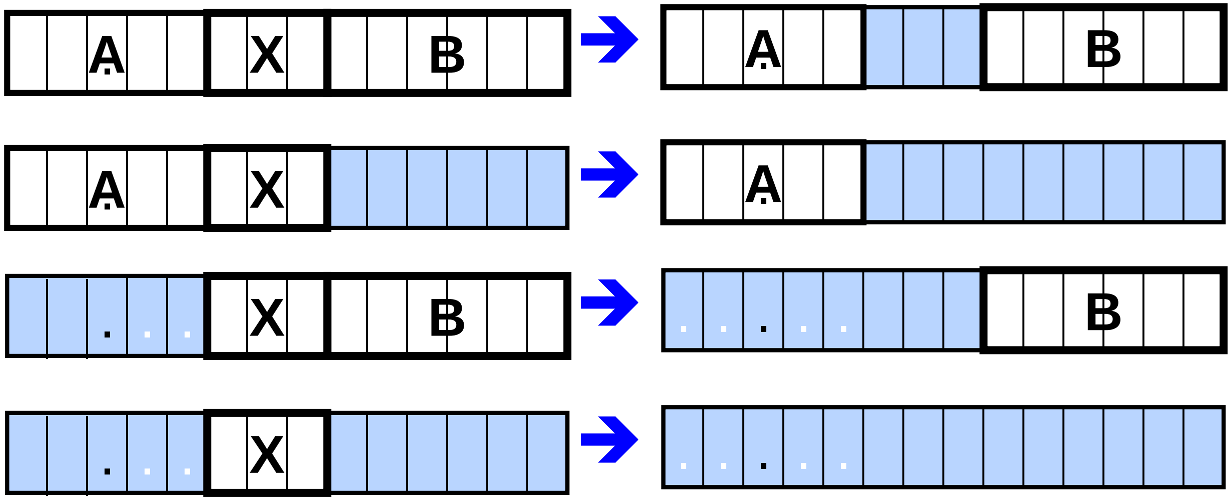
- Linked list of memory segments
  - □ Free segments ➔ Holes
  - □ Allocated segments ➔ Processes



Memory management with Linked Lists

- Segment list is sorted by addresses
- Sorting helps in updating the list, when a process is swapped out or exits
- A process usually has two neighbors

Updating the list requires



Replace P by H

Two entries are merged

Two entries are merged

Three entries are merged

Memory Allocation with Linked Lists

- Several algorithms for allocating memory with linked list
- FIRST FIT:
  - Scan the segment list, until
  - A hole large enough is found
  - Split the hole into two pieces (if not exact match)
  - One for the new process
  - One for the unused memory
- The algorithm is fast since it searches as little as possible

Memory Allocation with Linked Lists

- NEXT FIT
  - Works the same as First Fit, except,
  - Does not always start searching from the beginning
  - Rather starts searching the list, from where it left last time
- Simulations show, that gives no better performance than First Fit

Memory Allocation with Linked Lists

- BEST FIT
  - Search the entire list
  - Take the smallest hole that is adequate
- Best Fit tries to find the hole closest to the size
- Slower than First Fit
  - Every time has to search the entire list
- But still results in more memory wastage
  - Tends to fill up memory, with tiny useless holes
  - First Fit generates larger holes on the average

Memory Allocation with Linked Lists

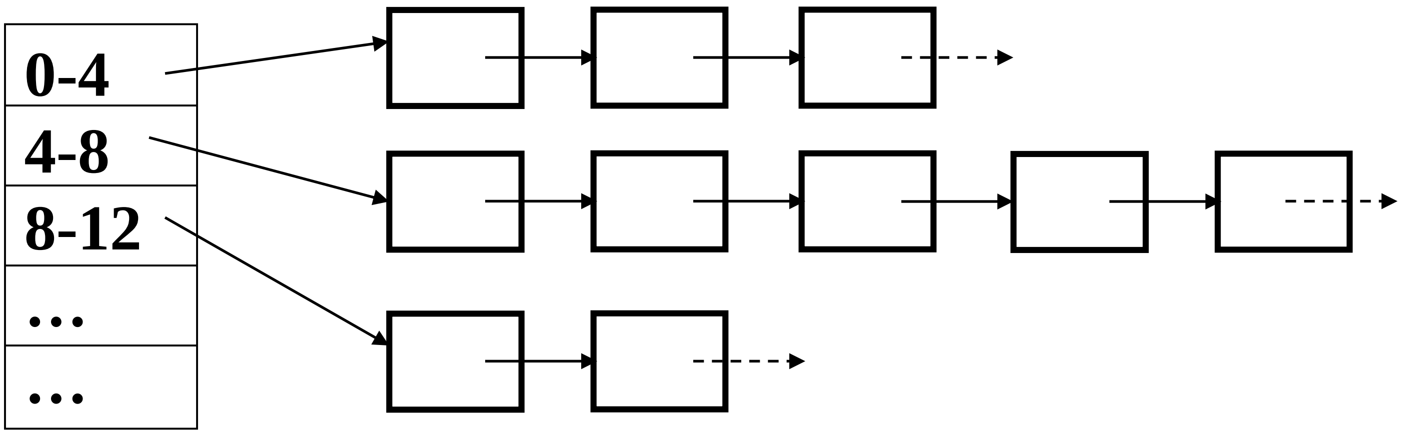
- WORST FIT
  - Take the largest hole available
  - So that, the hole broken off will be large enough to be useful
  - What if the larger holes left by worst fit are not useful
  - ➔ more memory wasted than Best Fit

Memory Allocation with Linked Lists

- Search time of all the four algorithms can be improved by
- Keeping separate lists for Process and Holes
- While allocating memory only have to search the lists of holes
- DRAWBACK:
  - Problem while deallocating memory
  - A node from the process list has to be inserted in the hole list

Memory Allocation with Linked Lists

- The Hole list can be kept sorted by size
- As soon as a hole that fits is found, no more searching is required
- Another improvement can be,
  - □ Maintain different lists for different sizes of holes
  - □ QUICK FIT



Memory Allocation with Linked Lists

- QUICK FIT
- Finding a hole of required size is extremely fast
- DRAWBACK:
  - When a process terminates/swapped out
  - Finding the neighbor to see whether the merge is possible
  - If merge is not done, then sooner memory will be fragmented into large holes