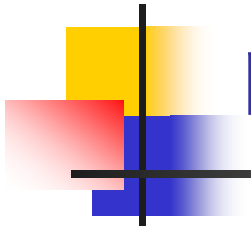
Dynamic Memory Allocation with malloc() and free()

Pei-yih Ting

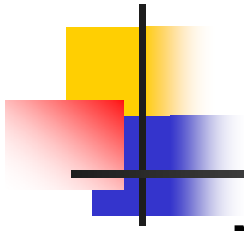


malloc() and free()

- Library routines for managing the heap

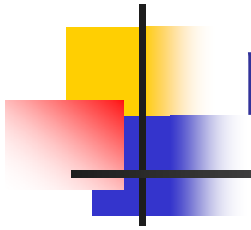
```
int *ary;
ptr = (int *) malloc(sizeof(int) * 100);
ary[5] = 3;
free(ary);
```

- Allocate and free arbitrary-sized chunks of memory in any order



malloc() and free()

- More flexible than automatic variables (stacked)
- More costly in time and space
  - malloc() and free() use complicated non-constant-time algorithms
  - Each block generally consumes two additional words
    - Pointer to next empty block
    - Size of this block
- Common source of errors
  - Using uninitialized memory
  - Using freed memory
  - Not allocating enough
  - Neglecting to free disused blocks (memory leaks)

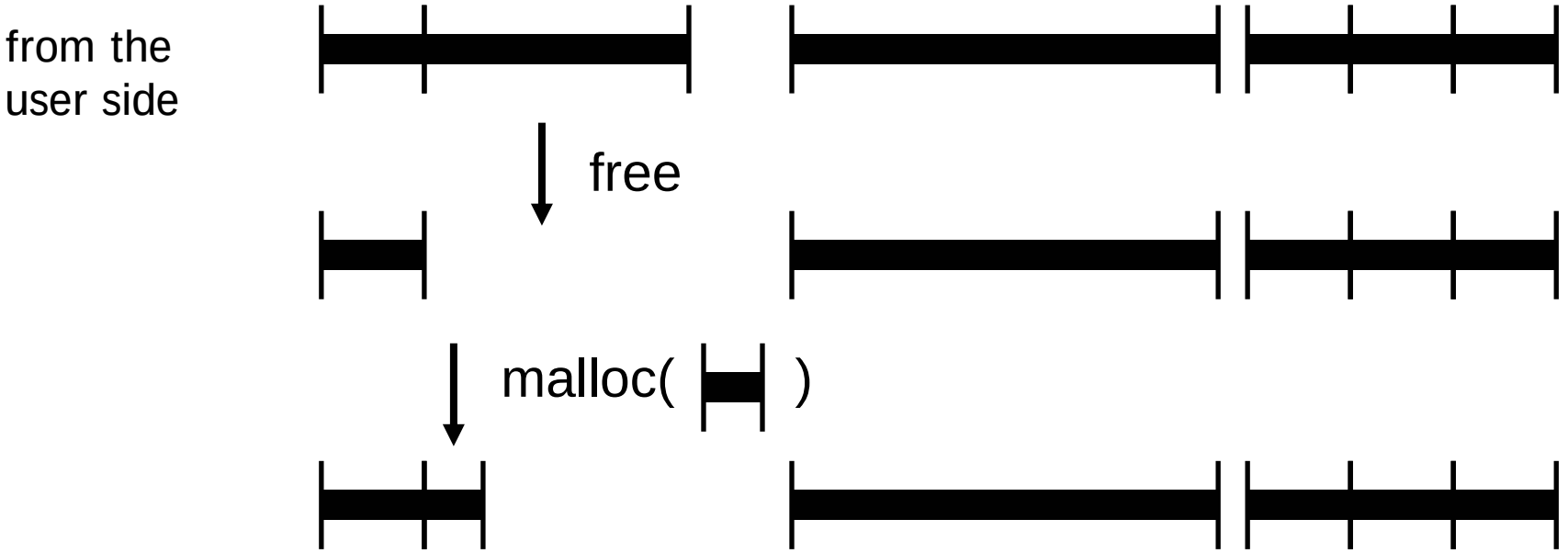


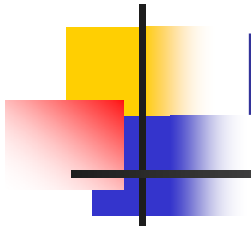
malloc() and free()

- Memory usage errors so pervasive, entire successful company (Pure Software) founded to sell tool to track them down
- Purify tool inserts code that verifies each memory access
- Reports accesses of uninitialized memory, unallocated memory, etc.
- Publicly-available Electric Fence tool does something similar

Dynamic Storage Allocation

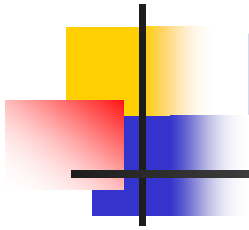
- What are malloc() and free() actually doing?
- Pool of memory segments:





Dynamic Storage Allocation

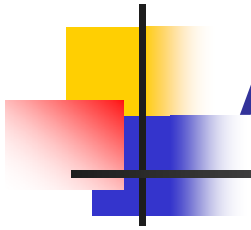
- Rules:
  - Each segment contiguous in memory (no holes)
  - Segments do not move once allocated
- malloc()
  - Find memory area large enough for segment
  - Mark that memory as allocated
- free()
  - Mark the segment as unallocated



Dynamic Storage Allocation

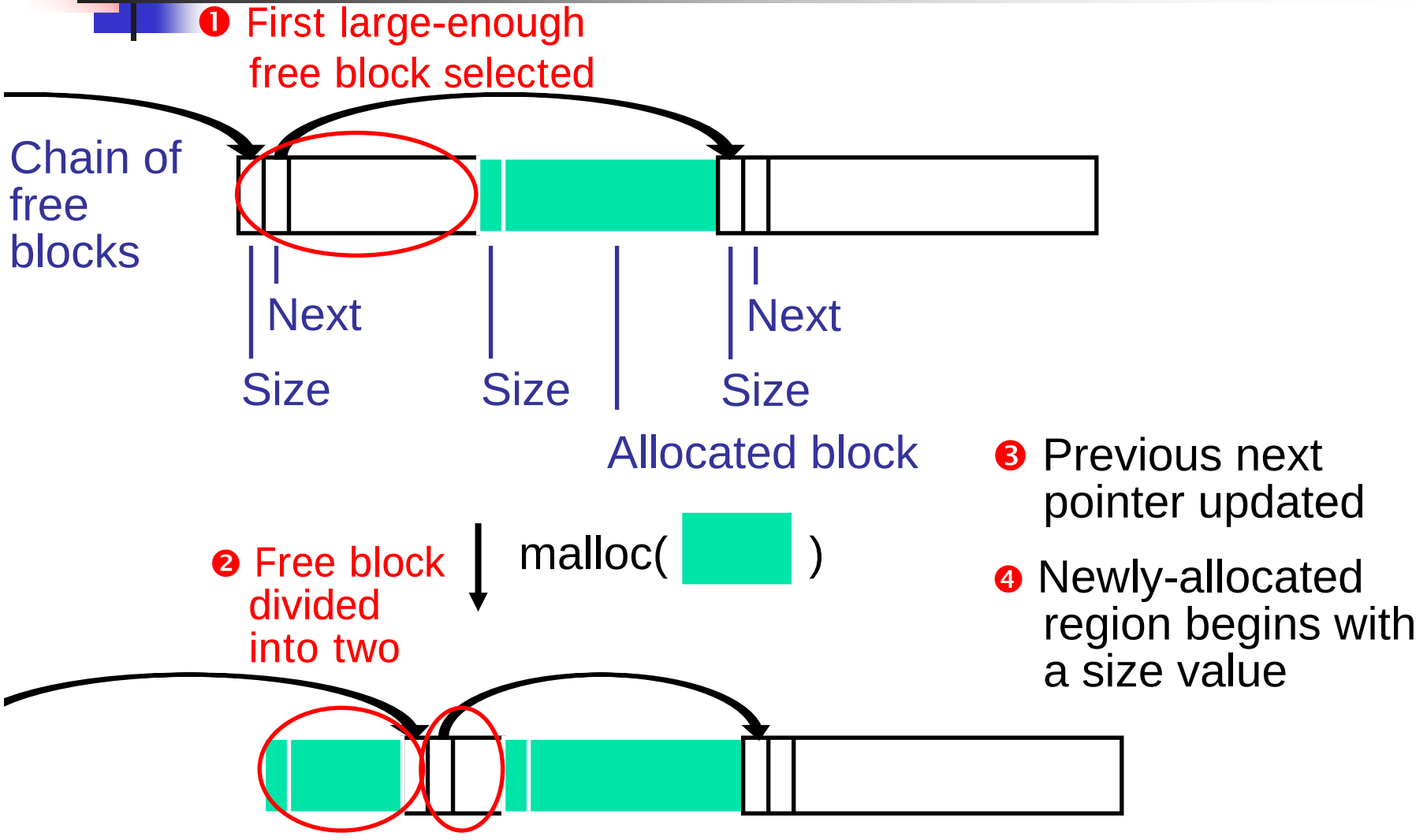
- Three issues:
  - How to maintain information about free memory
  - The algorithm for locating a suitable block
  - The algorithm for freeing an allocated block

Simple Dynamic Storage Allocation

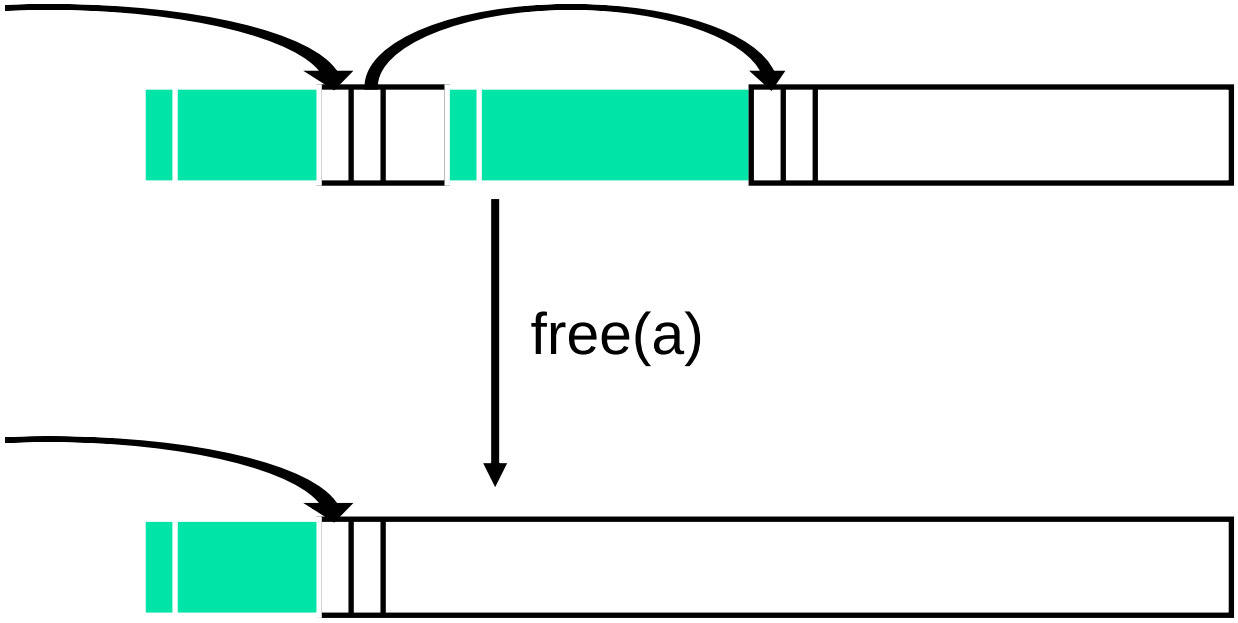


- Three issues:
  - How to maintain information about free memory
    - Linked list
  - The algorithm for locating a suitable block
    - First-fit
  - The algorithm for freeing an allocated block
    - Coalesce adjacent free blocks

Simple Dynamic Storage Allocation

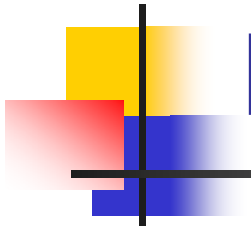


Simple Dynamic Storage Allocation



Appropriate position in free list identified

Newly-freed region added to adjacent free regions



Dynamic Storage Allocation

- Many, many variants
- Other “fit” algorithms
- Segregation of objects by sizes
  - 8-byte objects in one region, 16 in another, etc.
- More intelligent list structures