CS261 Data Structures

Dynamic Arrays

Arrays: Pros and Cons

- Pro: only core data structure designed to hold a collection of elements
- Pro: random access: can quickly get to any element → O(1)
- Con: fixed size:
  - Maximum number of elements must be specified when created

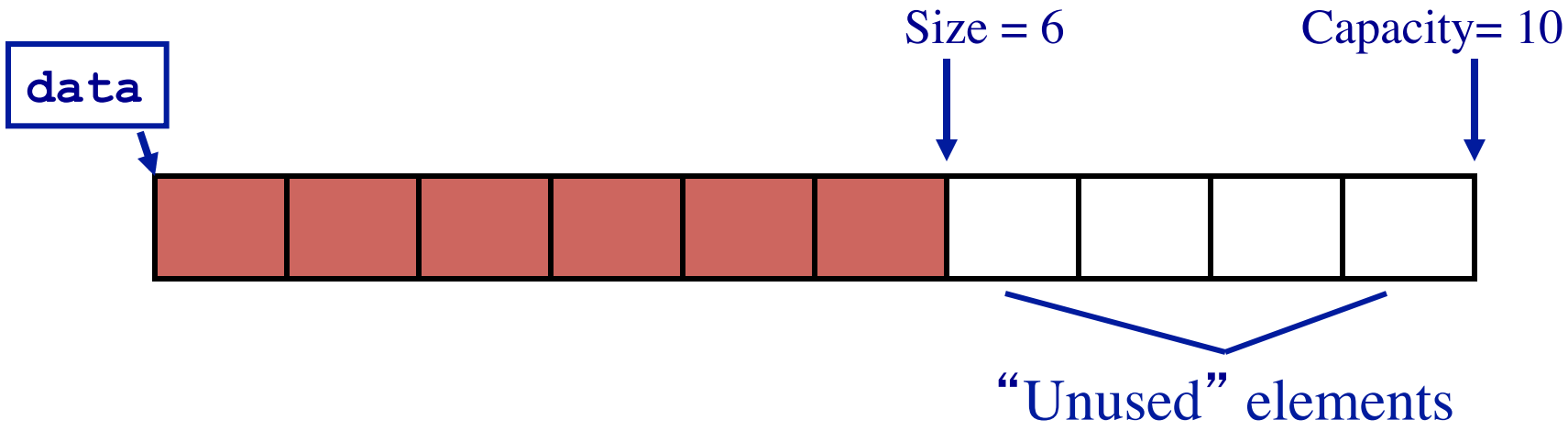
Dynamic Array (Vector or ArrayList)

- The dynamic array (called Vector or ArrayList in Java, same thing, different API) gets around this by encapsulating a partially filled array
- Hide memory management details behind a simple API
- Is still randomly accessible, but now it grows as necessary

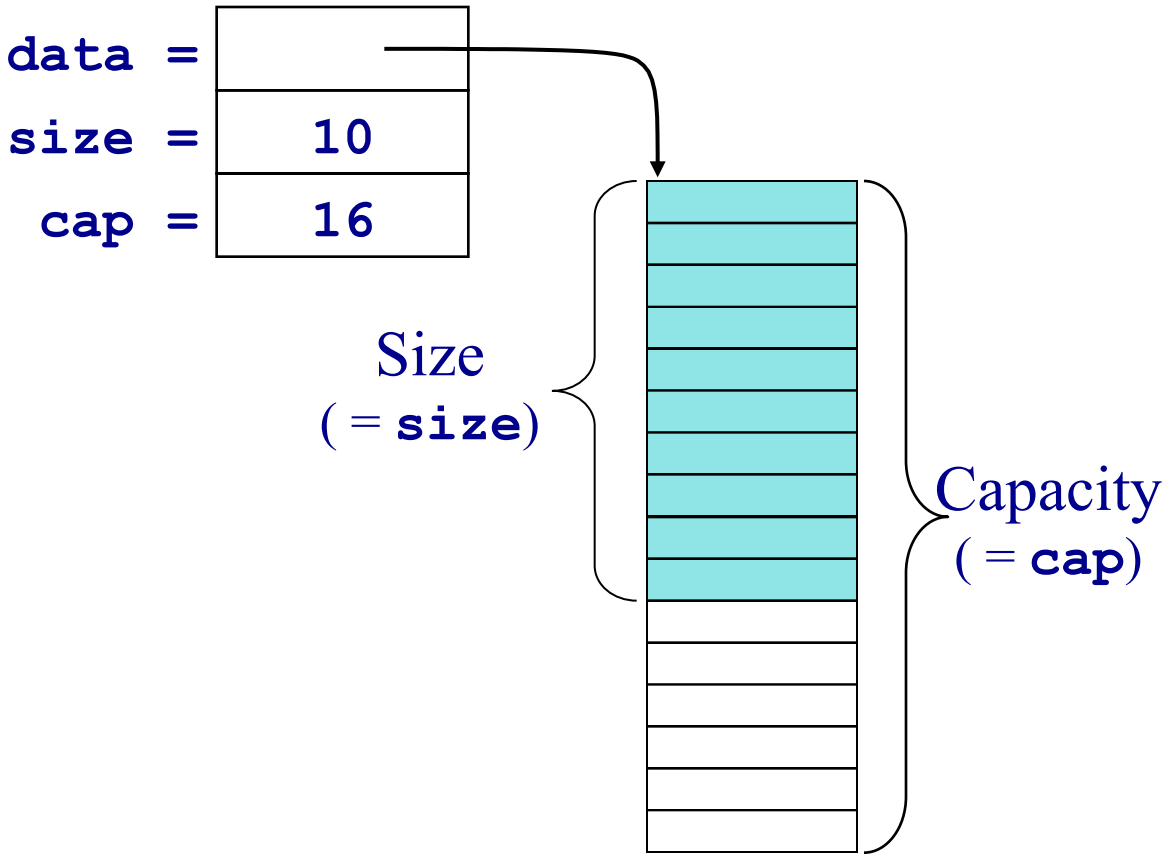
Size and Capacity

- Unlike arrays, a dynamic array can change its capacity
- *Size* is logical collection size:
  - Current number of elements in the dynamic array
  - What the programmer thinks of as the size of the collection
  - Managed by an internal data value
- *Capacity* is physical array size: # of elements it can hold before it must resize

Size and Capacity



Partially Filled Dynamic Array

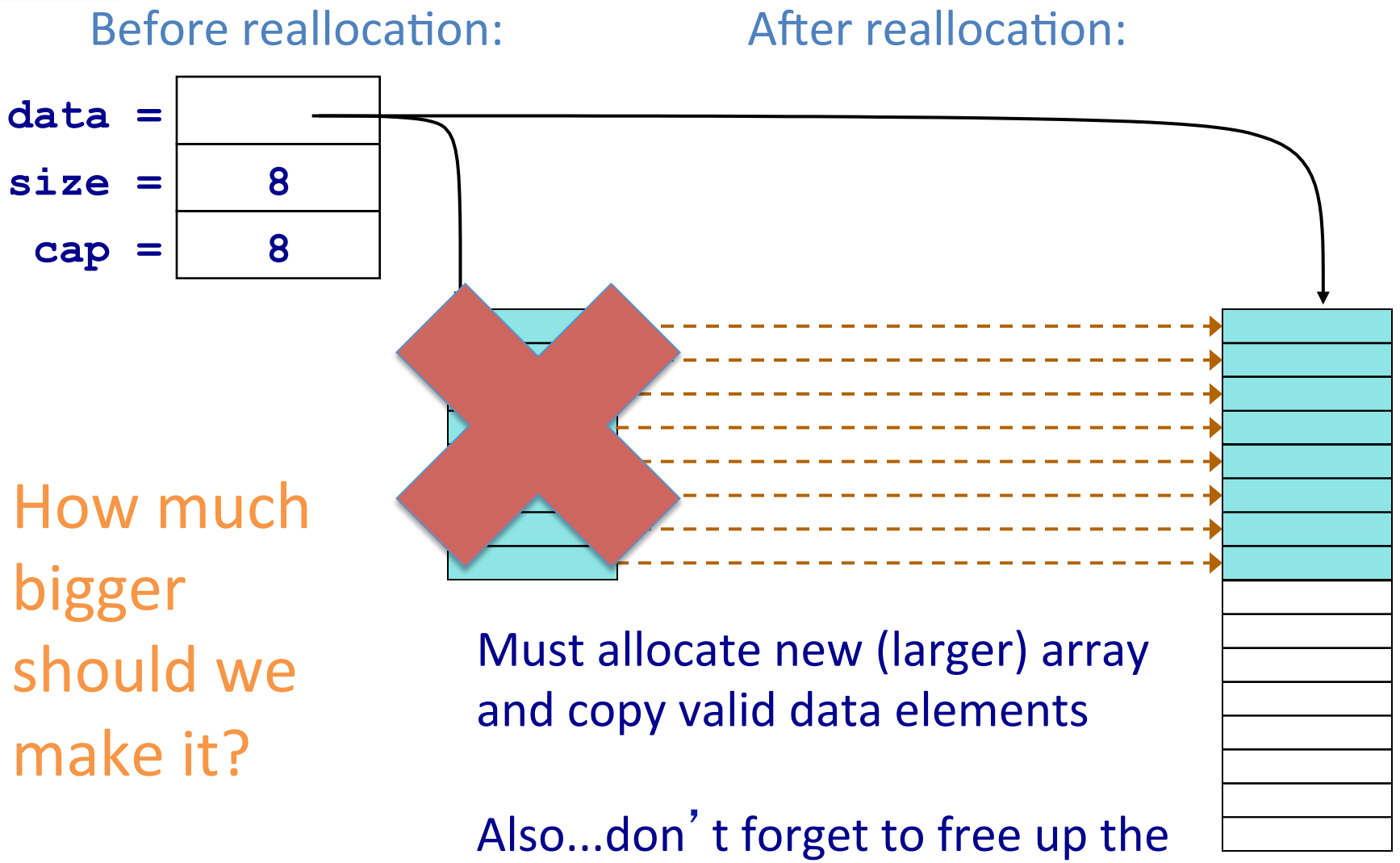


Adding an element

- Adding an element to end is usually easy — just increase the (logical) size, and put new value at end

- What happens when size reaches capacity?
  - Must reallocate new data array - but this detail is hidden from user

Set Capacity: Reallocate and Copy (animation)



How much bigger should we make it?

Must allocate new (larger) array and copy valid data elements

Also...don't forget to free up the old array

Adding to Middle

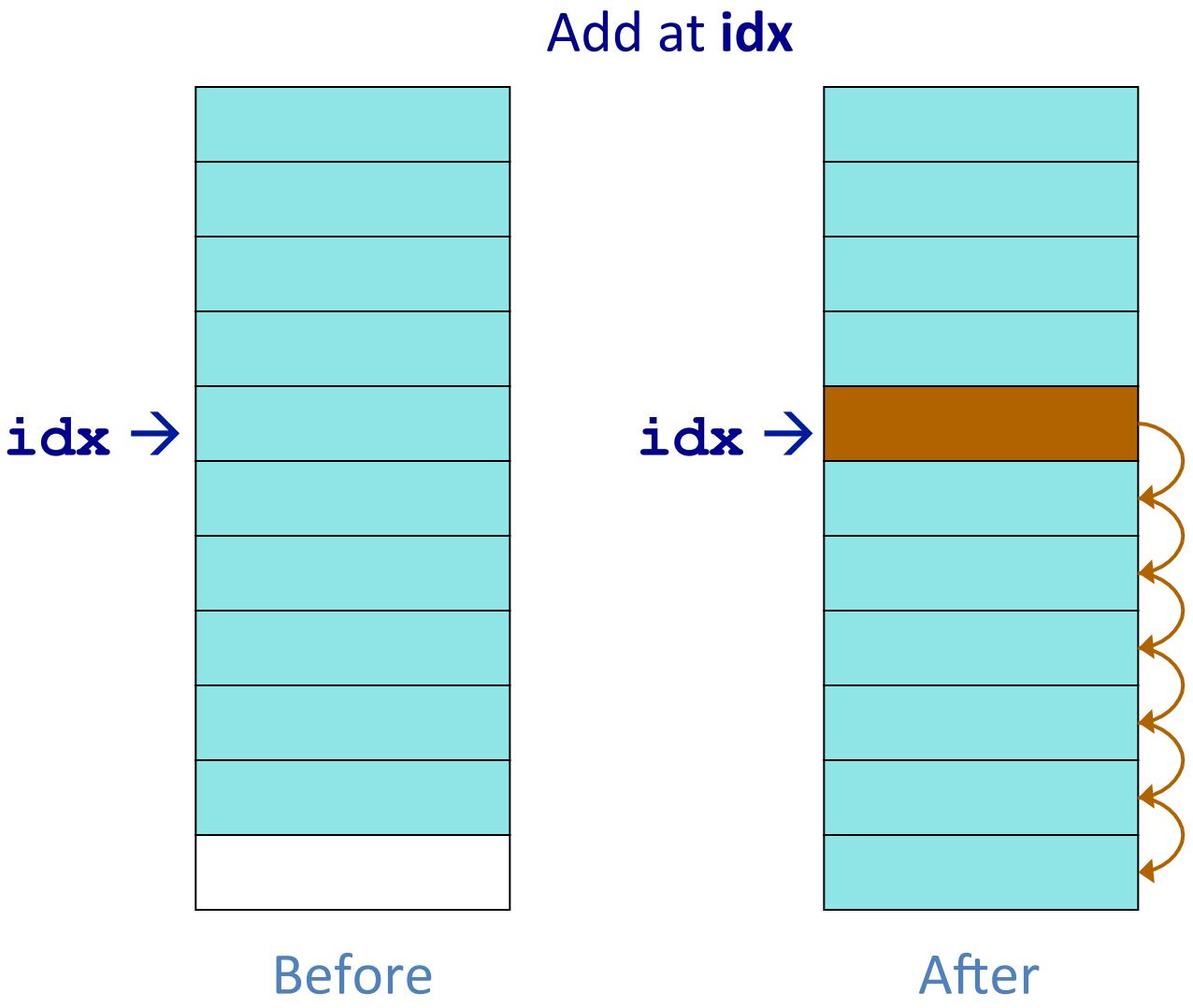
- Adding an element to middle can also force reallocation (if the current size is equal to capacity)
- But will ALWAYS require that elements be moved to make space
- Is therefore O(*n*) worst case

Adding to Middle (cont.)

Must make space for new value

Be Careful!

Loop from bottom up while copying data



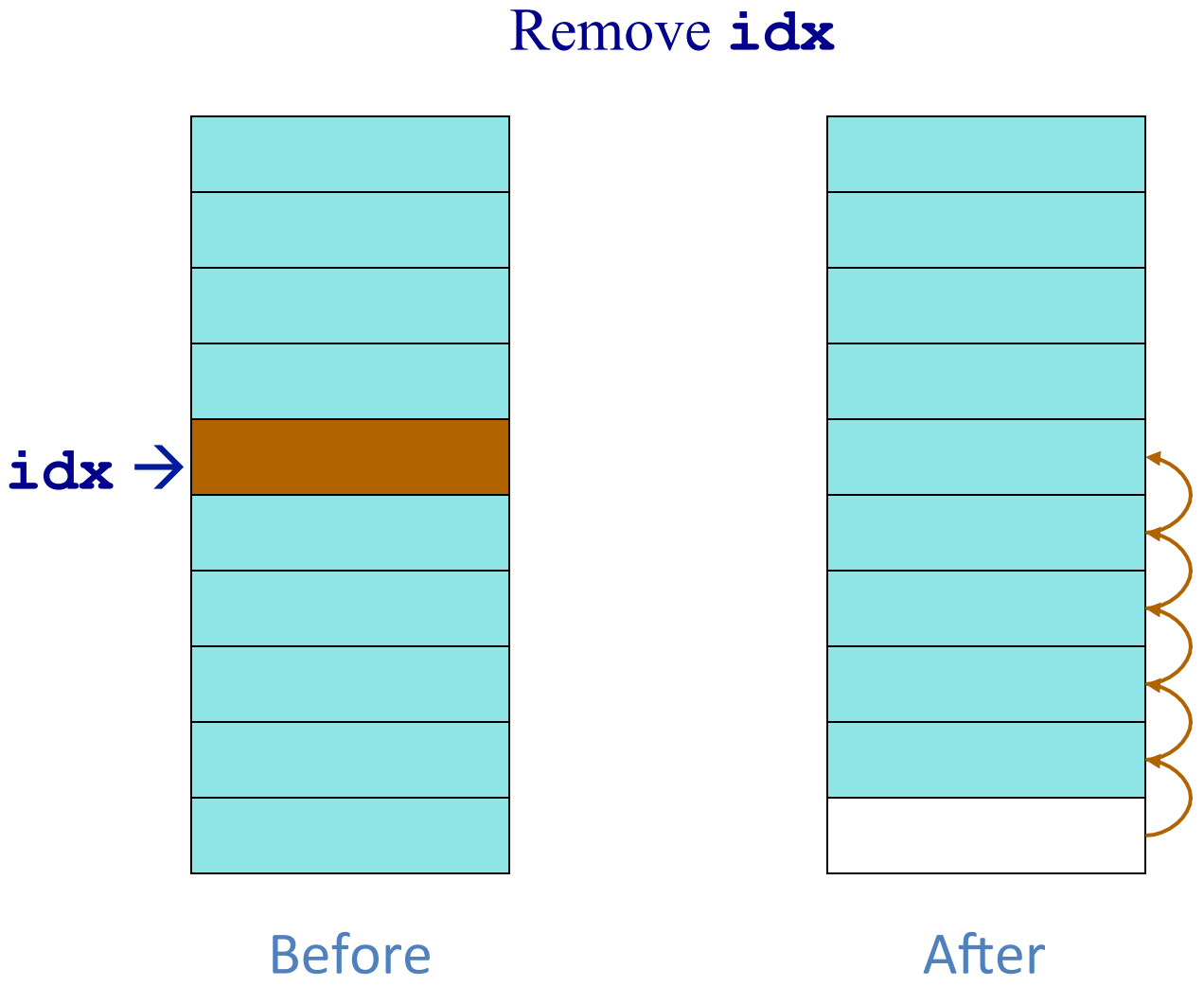
Removing an Element

- Removing an element will also require "sliding over" to delete the value
  - We want to maintain a contiguous chunk of data so we always know where the next element goes and can put it there quickly!
- Therefore is O(*n*) worst case

Remove Element

Remove also requires a loop

This time, should it be from top (e.g. at idx) or bottom?



Side Note

- realloc() can be used in place of malloc() to do resizing and will avoid ‘copying’ elements if possible
- For this class, use malloc only (so you’ll have to copy elements on a resize)

Something to think about…

- In the long term, are there any potential problems with the dynamic array?
  - hint: imagine adding MANY elements in the long term and potentially removing many of them.