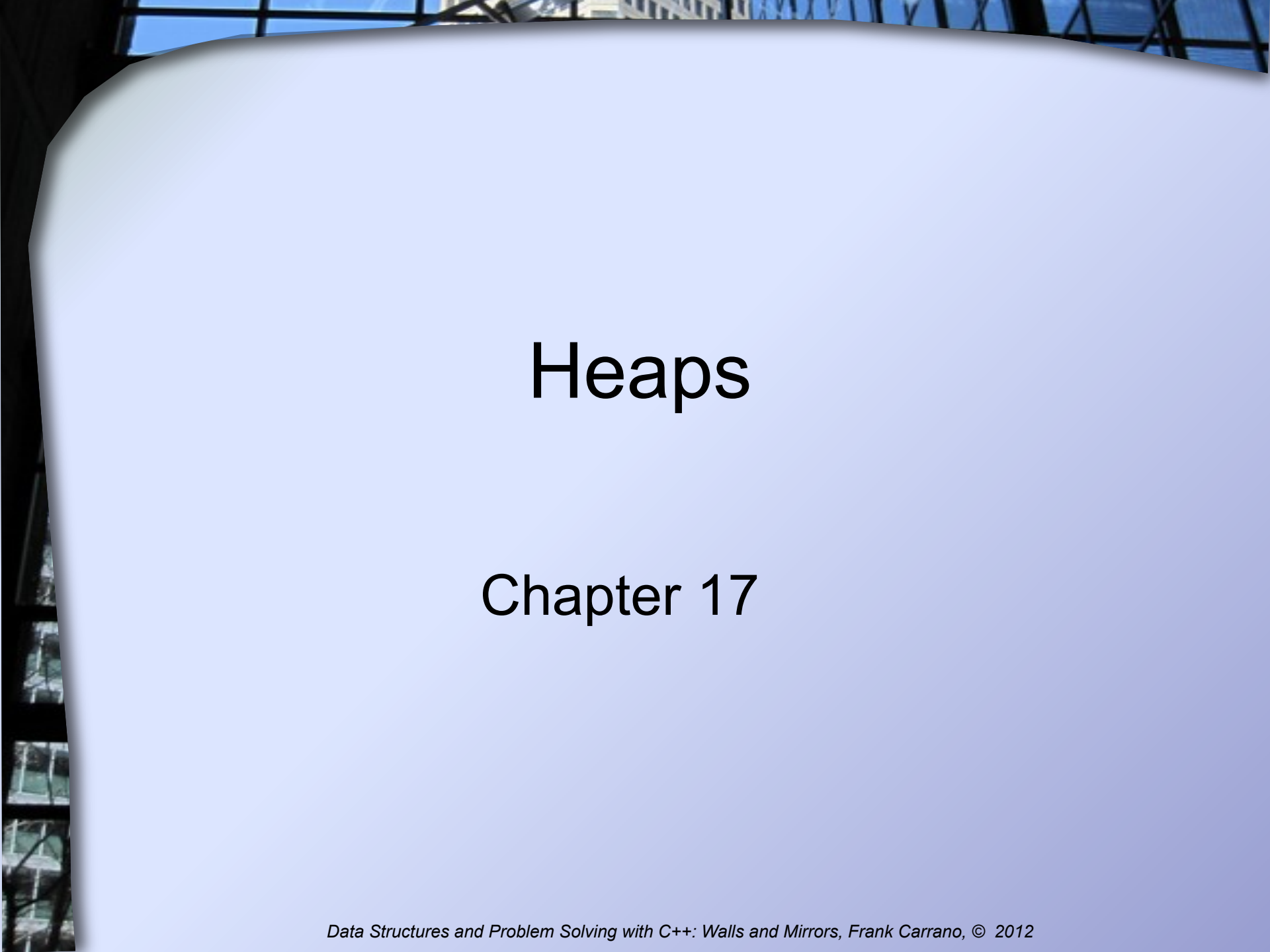


Heaps

Chapter 17

Contents

- The ADT Heap
- An Array-Based Implementation of a Heap
- A Heap Implementation of the ADT Priority Queue
- Heap Sort

The ADT Heap

- Definition
  - A heap is a complete binary tree that either is empty … or …
  - It's root
    - Contains a value greater than or equal to the value in each of its children, and
    - Has heaps as its subtrees

The ADT Heap

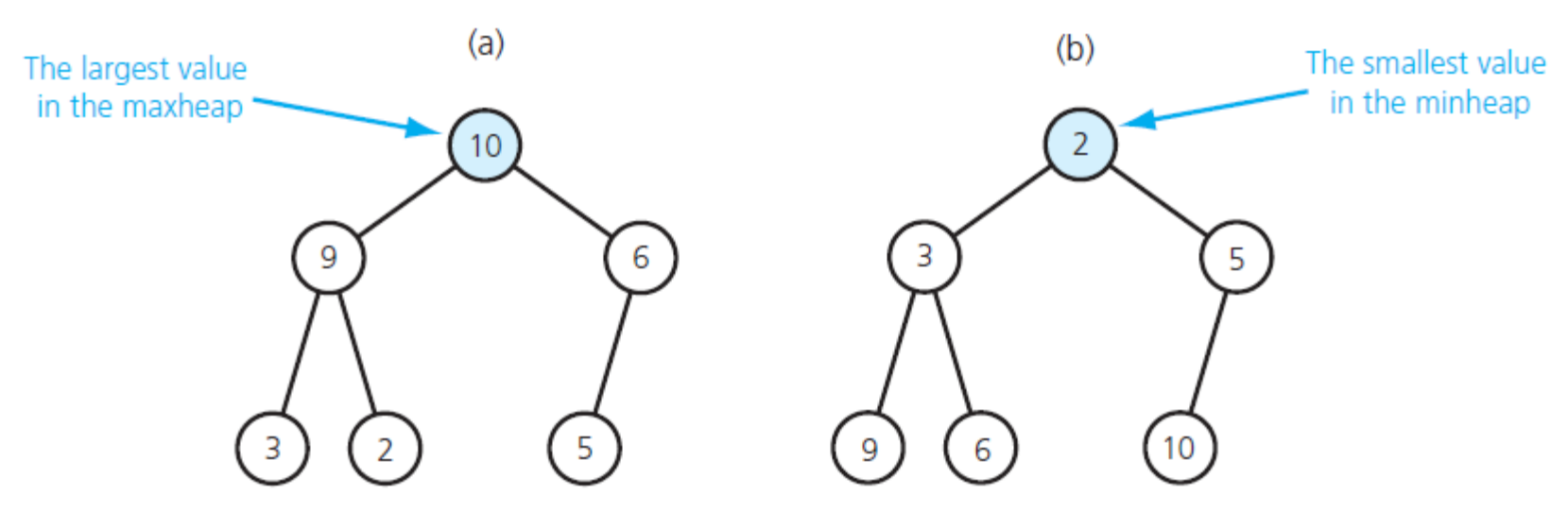
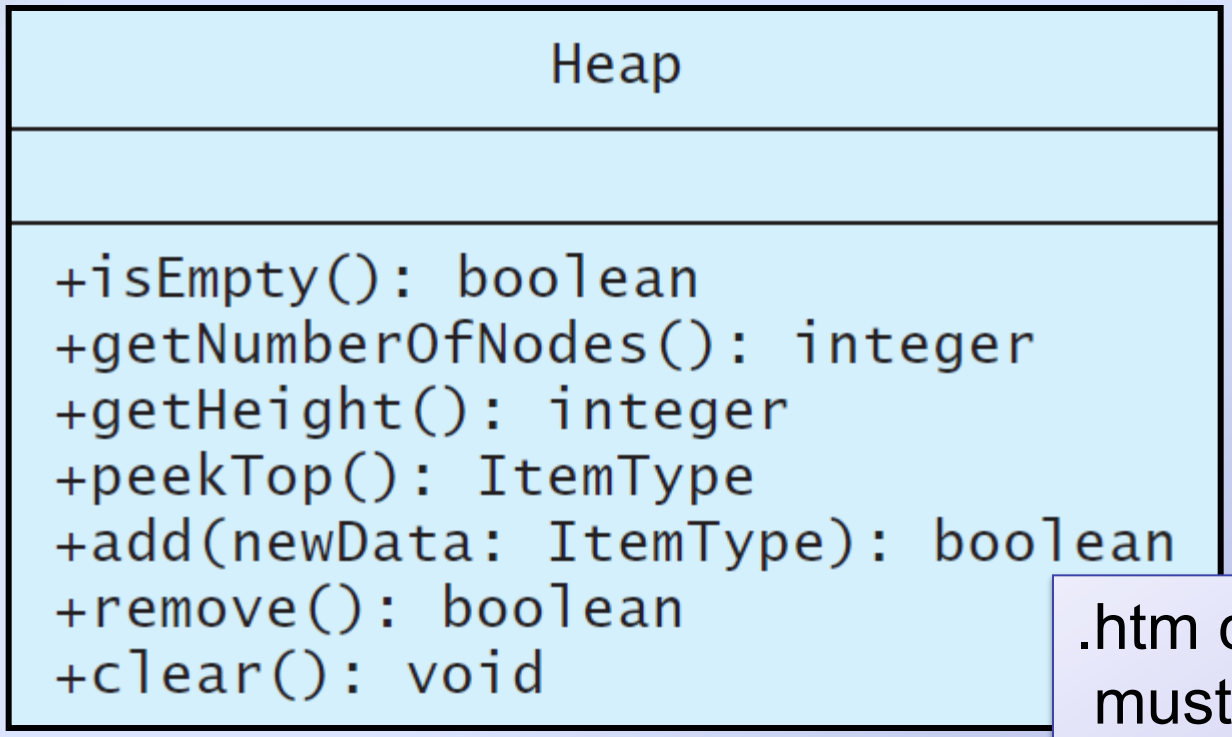


FIGURE 17-1 (a) A maxheap and (b) a minheap

The ADT Heap



View interface, Listing 17-1

.htm code listing files must be in the same folder as the .ppt files for these links to work

FIGURE 17-2 UML diagram fo

Array-Based Implementation of a Heap

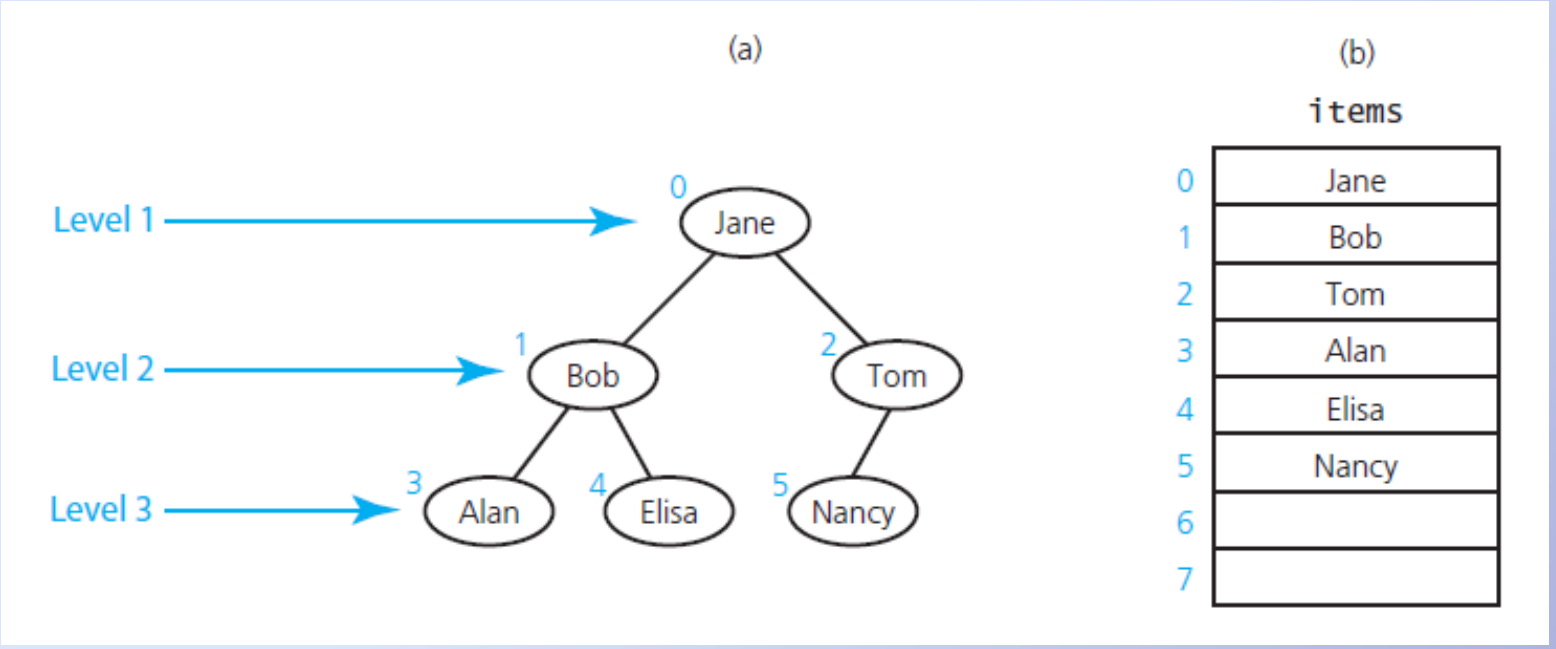


FIGURE 17-3 (a) Level-by-level numbering of a complete binary tree; (b) its array-based implementation

Algorithms for Array-Based Heap Operations

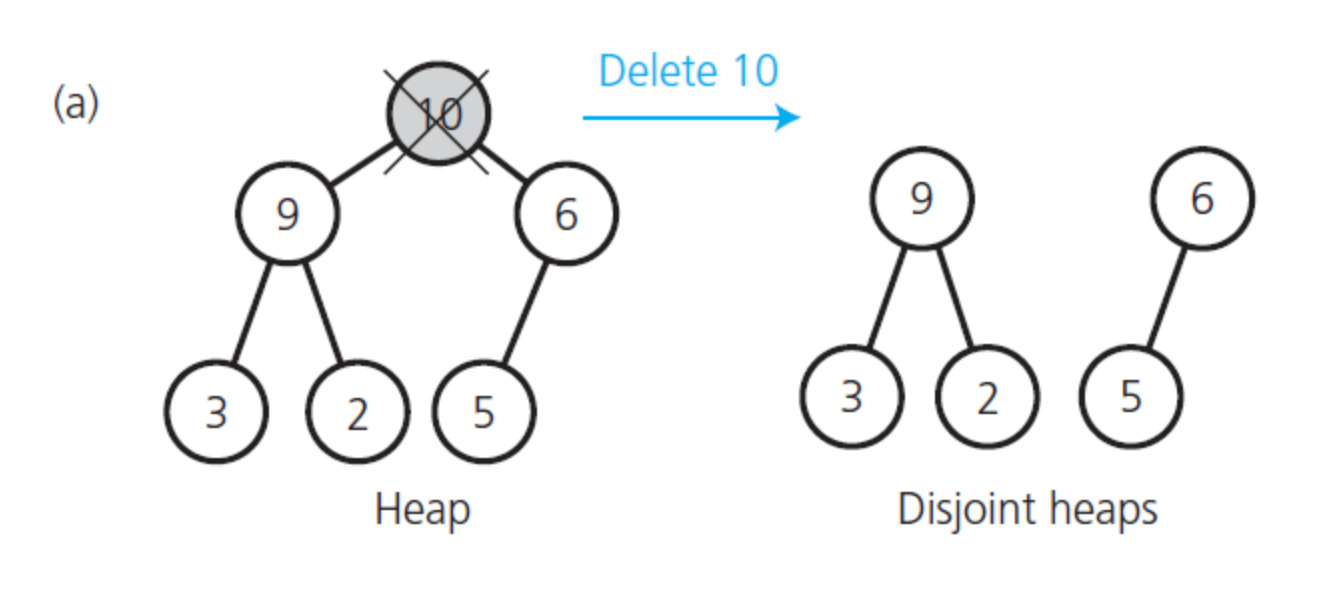


FIGURE 17-4 (a) Disjoint heaps after removing the heap's root;

Algorithms for Array-Based Heap Operations

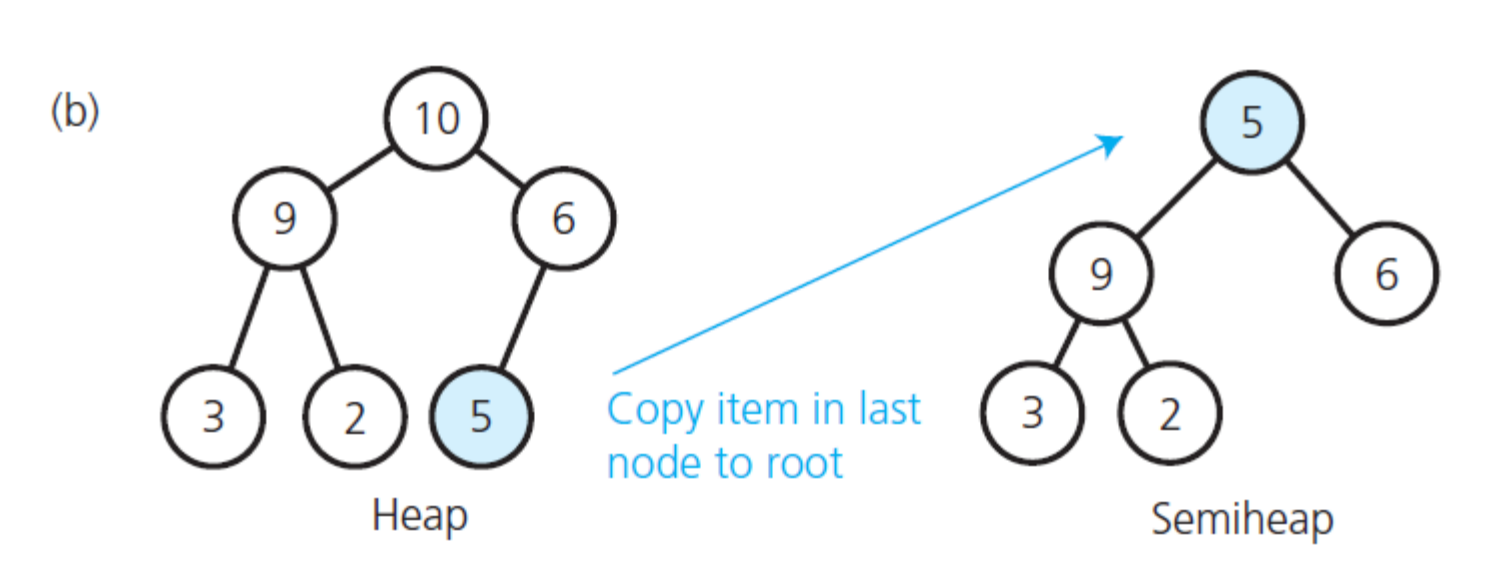


FIGURE 17-4 (b) a semiheap

Algorithms for Array-Based Heap Operations

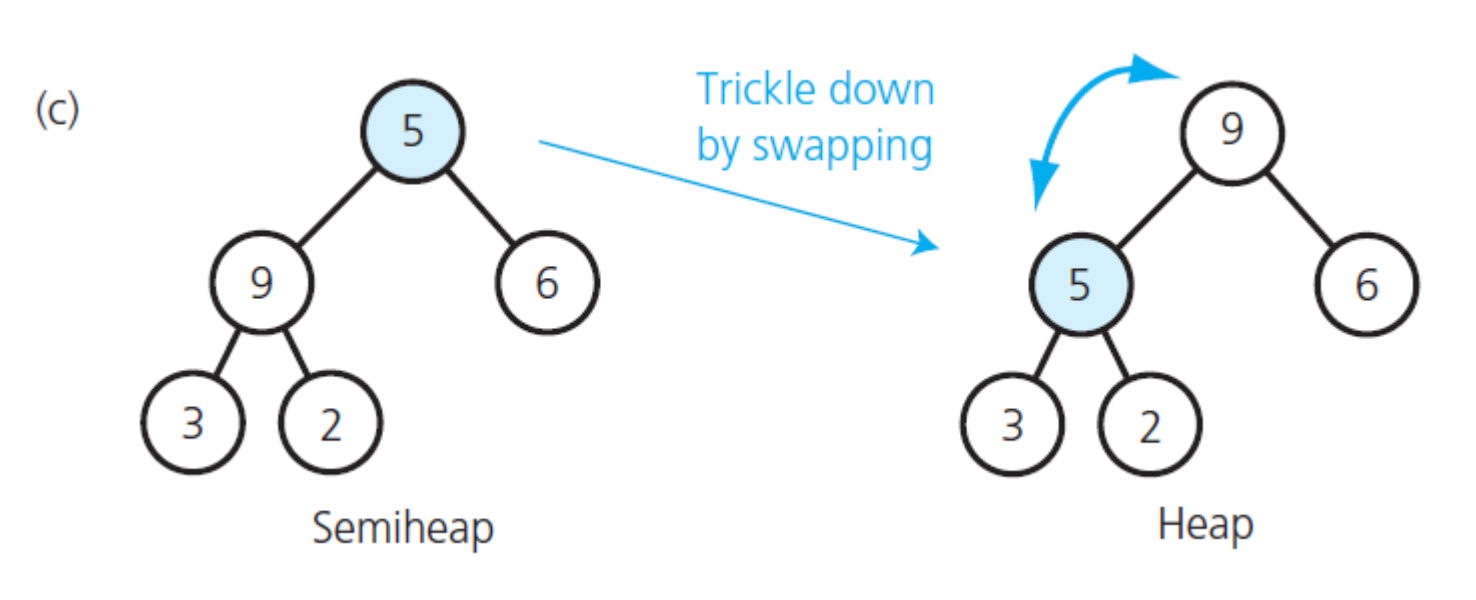


FIGURE 17-4 (c) the restored heap

View algorithm to make this conversion, Listing 17-A

Algorithms for Array-Based Heap Operations

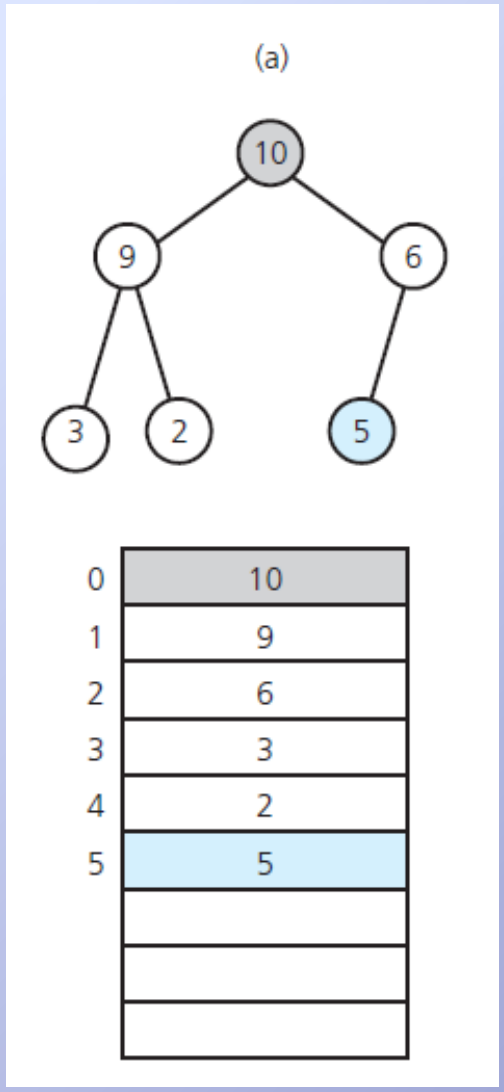


FIGURE 17-5 The array representation of
(a) the heap in Figure 17-4 a;

Algorithms for Array-Based Heap Operations

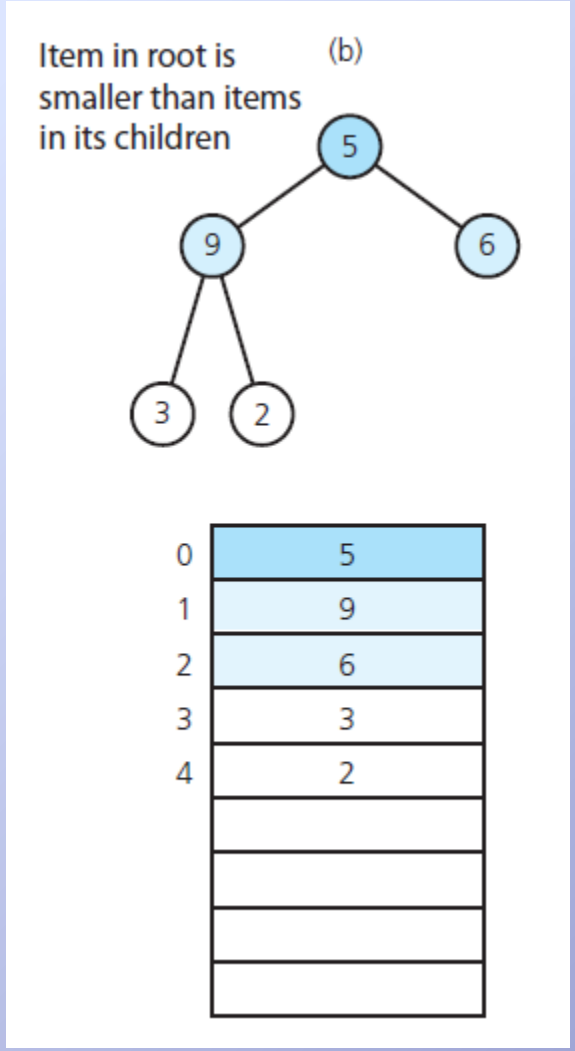


FIGURE 17-5 The array representation of (b) the semiheap in Figure 17-4 b;

Algorithms for Array-Based Heap Operations

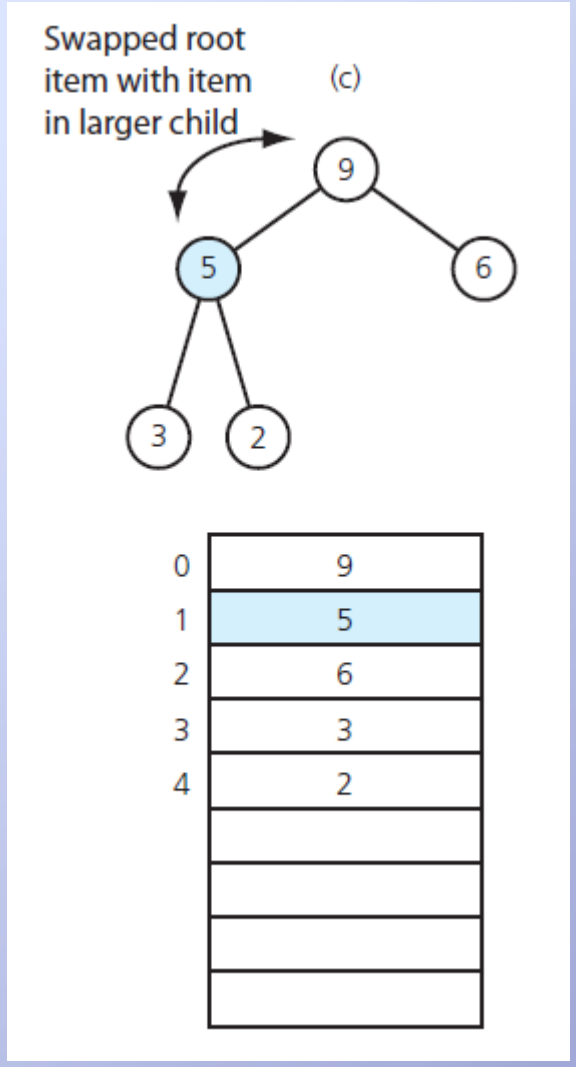


FIGURE 17-5 The array representation of (c) the restored heap in Figure 17-4 c

Algorithms for Array-Based Heap Operations

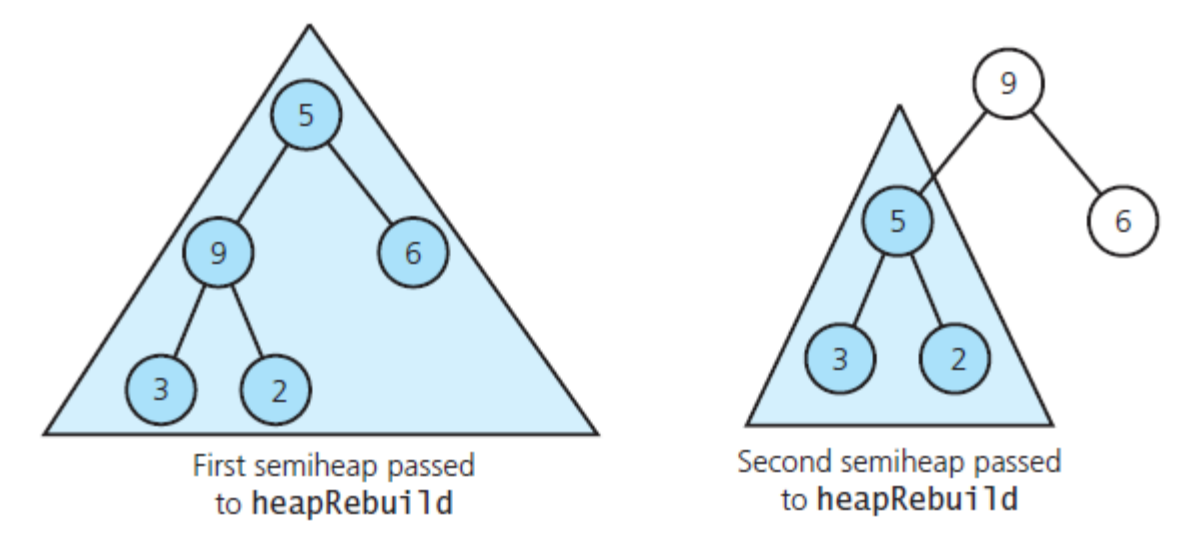


FIGURE 17-6 Recursive calls to **`heapRebuild`**

Algorithms for Array-Based Heap Operations

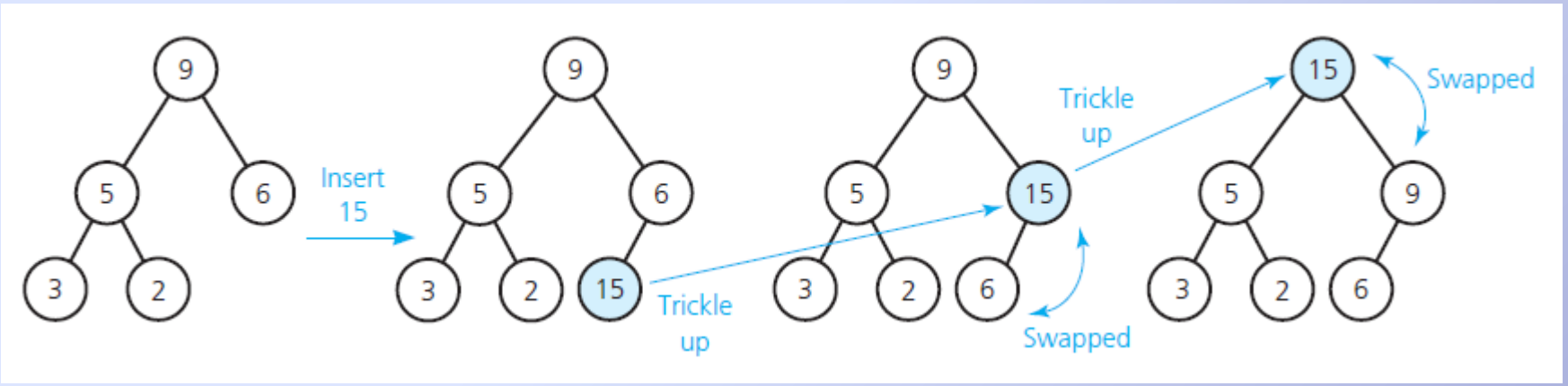


FIGURE 17-7 Insertion into a heap

Algorithms for Array-Based Heap Operations

- Pseudocode for **`add`**

```
// Insert newData into the bottom of the tree
items[itemCount] = newData

// Trickle new item up to the appropriate spot in the tree
newDataIndex = itemCount
inPlace = false
while ( (newDataIndex >= 0) and !inPlace)
{
   parentIndex = (newDataIndex - 1) / 2
   if (items[newDataIndex] < items[parentIndex])
       inPlace = true
   else
   {
       Swap items[newDataIndex ] and items[parentIndex]
       newDataIndex = parentIndex
   }
}
itemCount++
```

The Implementation

- The header file for class ArrayMaxHeap
  - An array-based implementation of the ADT heap
- View Listing 17-2.
- This heap is a maxheap.

The Implementation

- Definition of constructor

```
template<class ItemType>
ArrayMaxHeap<ItemType>::
ArrayMaxHeap(const ItemType someArray[], const int arraySize):
             itemCount(arraySize), maxItems(2 * arraySize)
{
   // Allocate the array
   items = new ItemType[2 * arraySize];

   // Copy given values into the array
   for (int i = 0; i < itemCount; i++)
      items[i] = someArray[i];

   // Reorganize the array into a heap
   heapCreate();
}  // end constructor
```

The Implementation

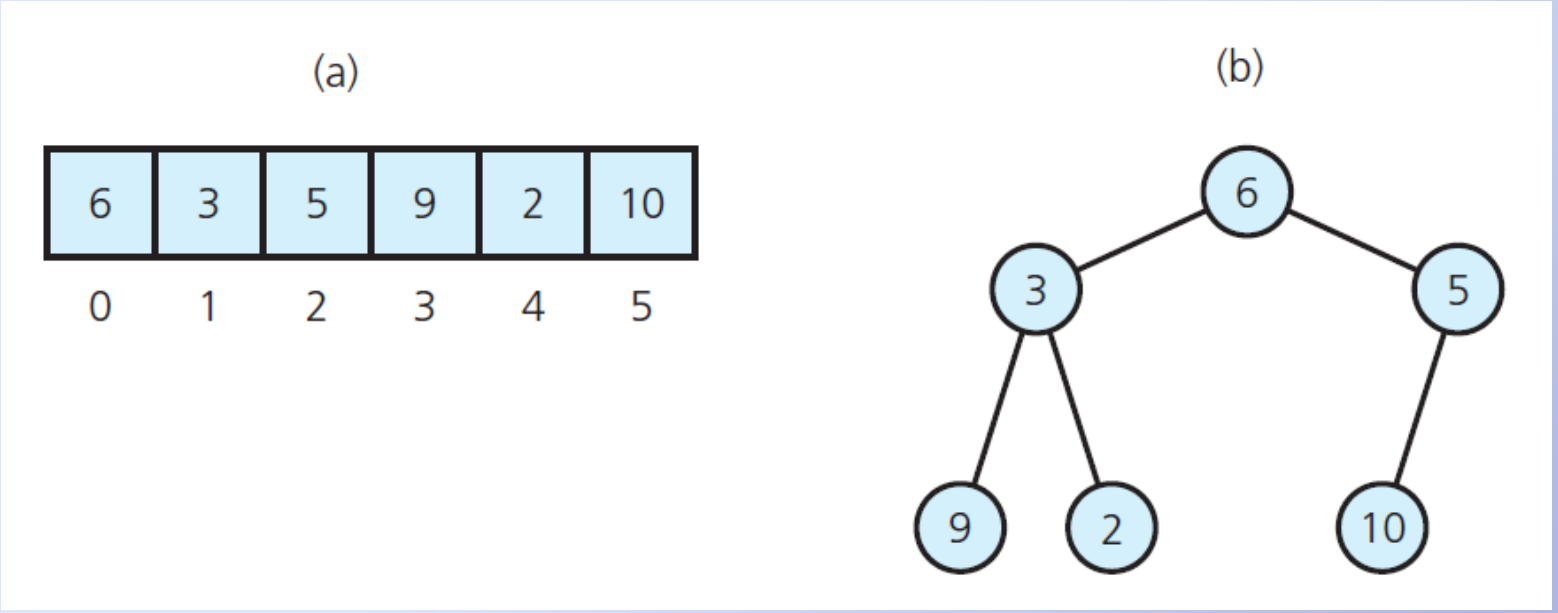


FIGURE 17-8 (a) The initial contents of an array; (b) the array's corresponding complete binary tree

The Implementation

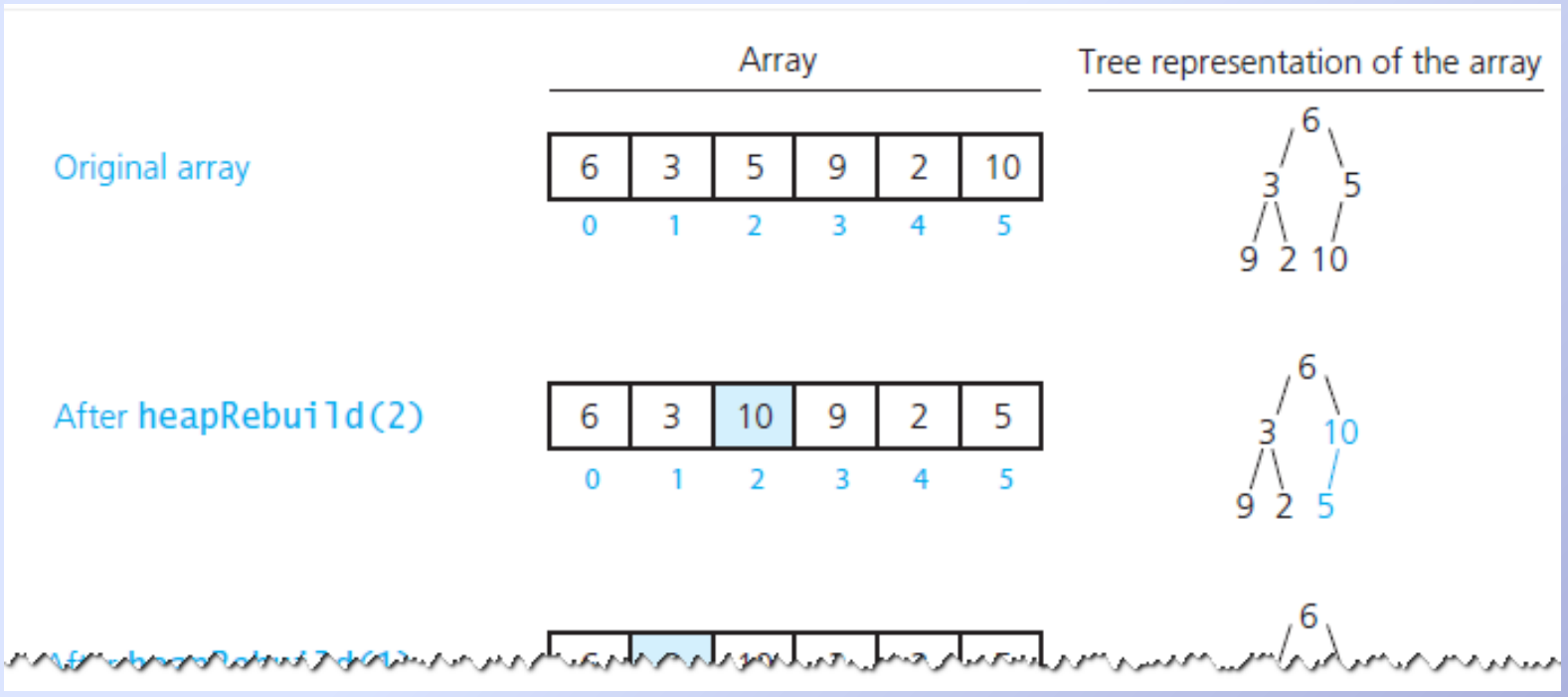


FIGURE 17-9 Transforming an array into a heap

The Implementation

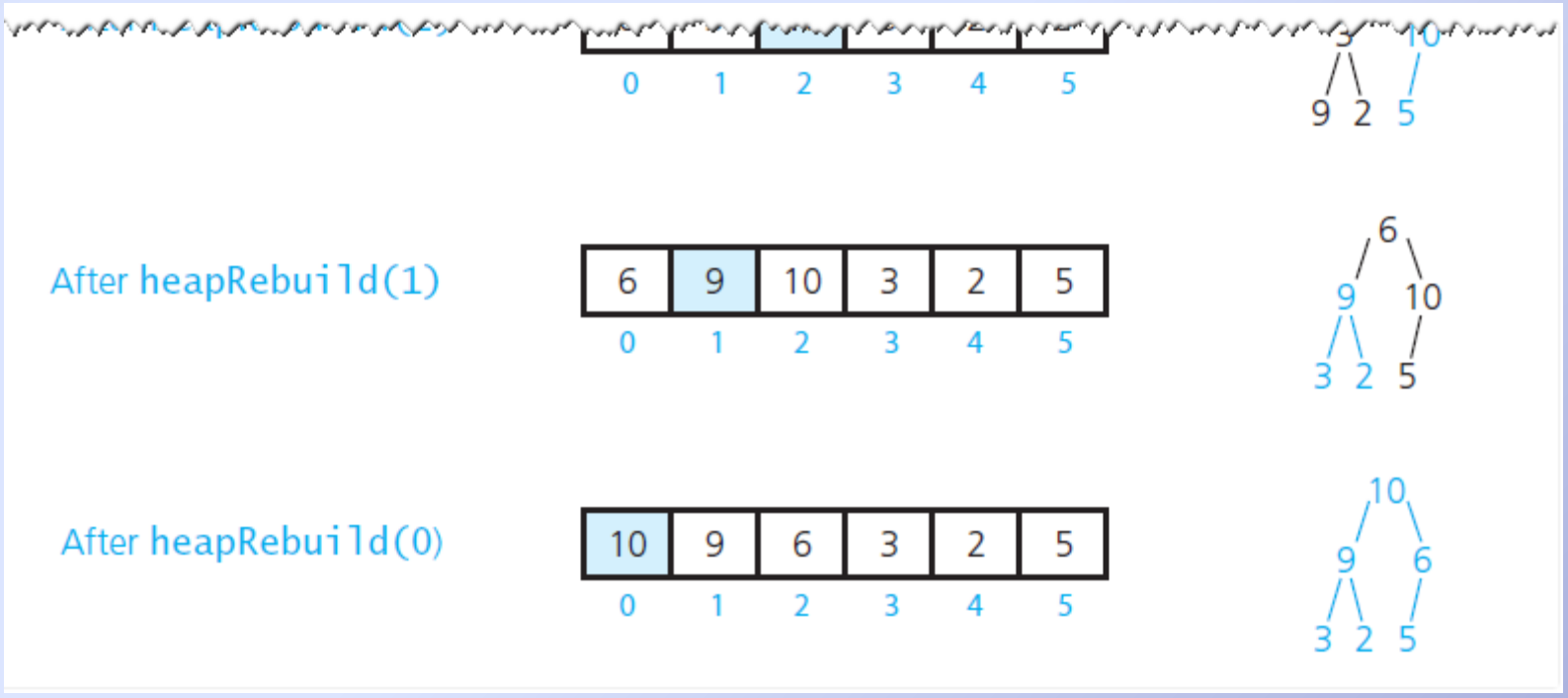


FIGURE 17-9 Transforming an array into a heap

The Implementation

- Method **heapCreate**

```
template<class ItemType>
void ArrayMaxHeap<ItemType>::heapCreate()
{
   for (int index = itemCount / 2; index >= 0; index--)
      heapRebuild(index);
}  // end heapCreate
```

The Implementation

- Method **`peekTop`**

```
template<class ItemType>
ItemType ArrayMaxHeap<ItemType>::peekTop() const throw(PrecondViolatedExcep)
{
   if (isEmpty())
      throw PrecondViolatedExcep("Attempted peek into an empty heap.");

   return items[0];
}  // end peekTop
```

Heap Implementation of the ADT Priority Queue

- Using a heap to define a priority queue results in a more time-efficient implementation
- Listing 17-3 contains a header file for a class of priority queues.
- View implementation, Listing 17-4

Heap Sort

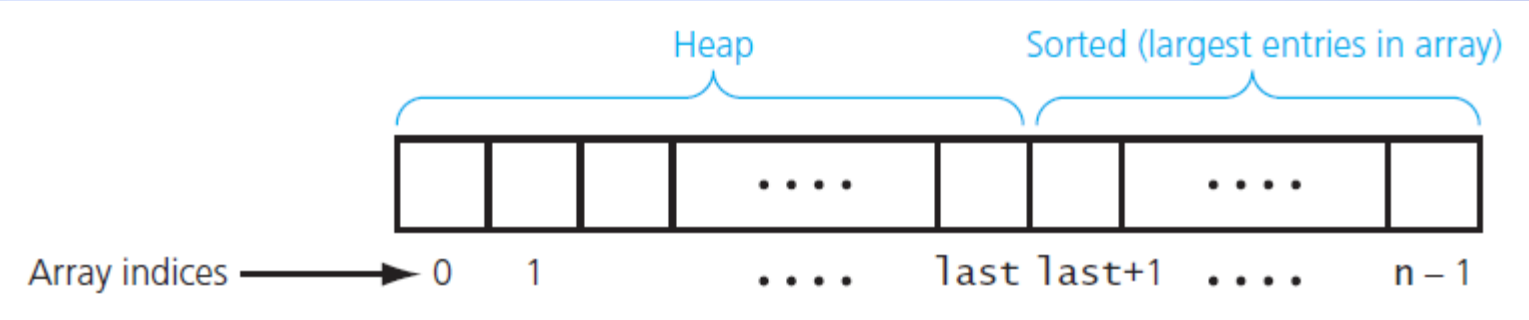


FIGURE 17-10 Heap sort partitions an array into two regions

View heap sort algorithm, Listing 17-B

Heap Sort

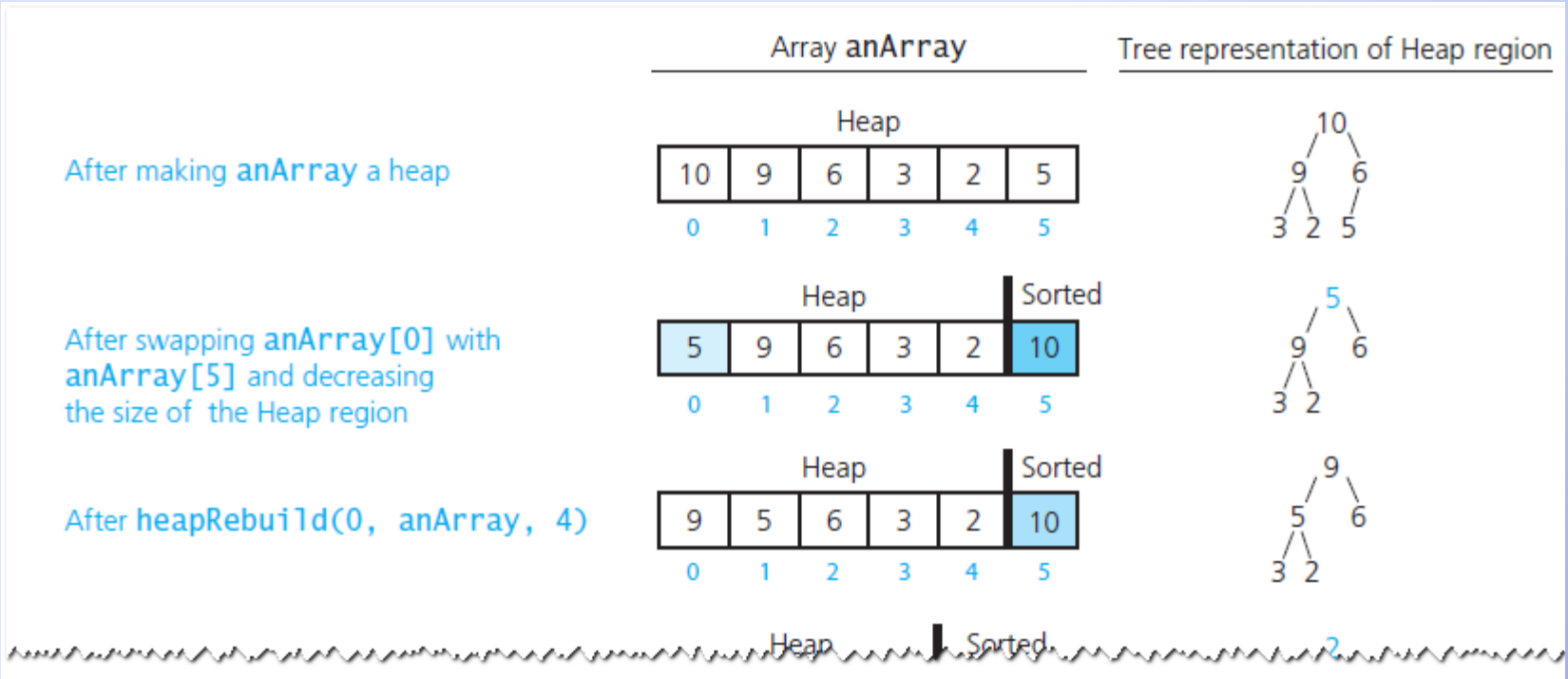


FIGURE 17-11 A trace of heap sort, beginning with the heap in Figure 17-9

Heap Sort

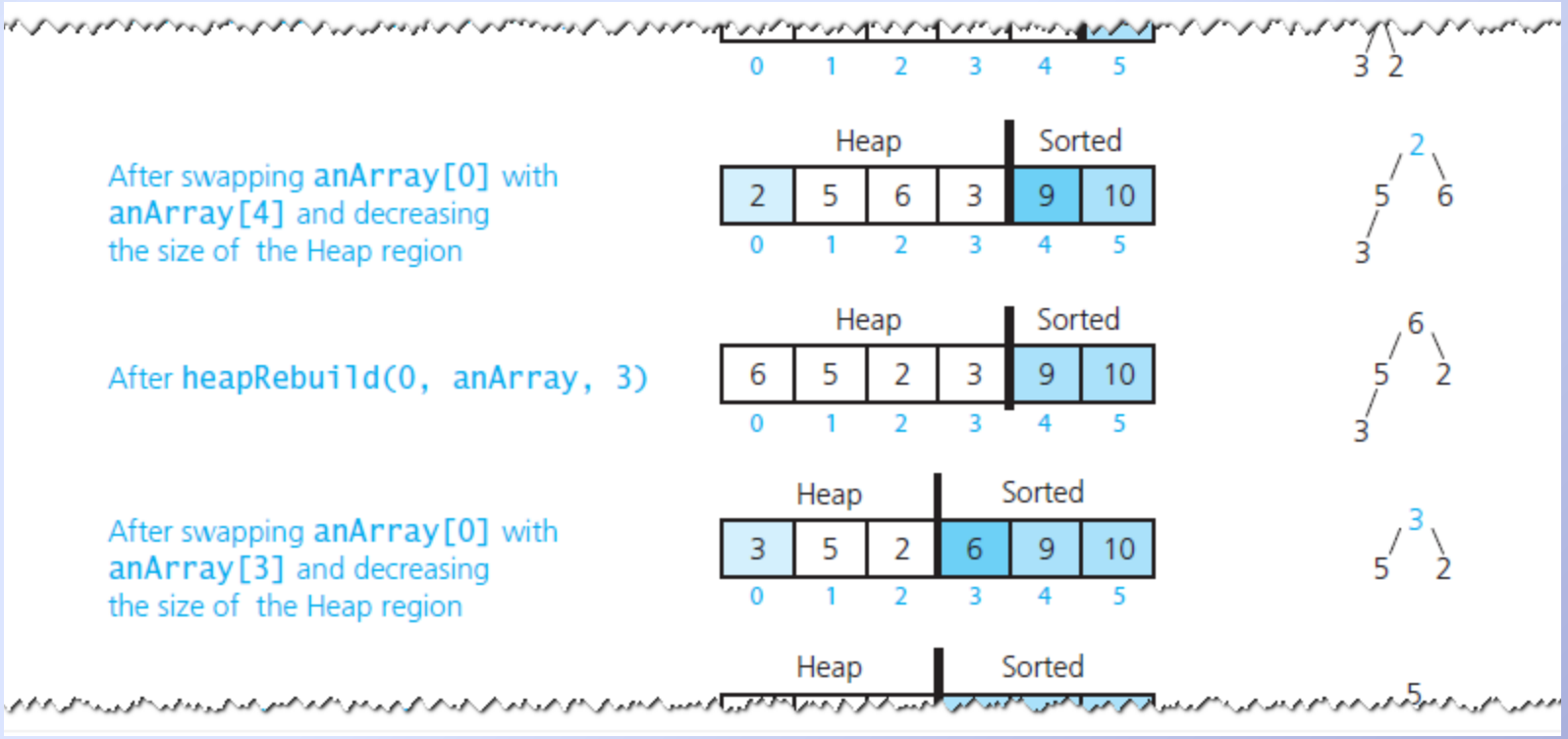


FIGURE 17-11 A trace of heap sort, beginning with the heap in Figure 17-9

Heap Sort

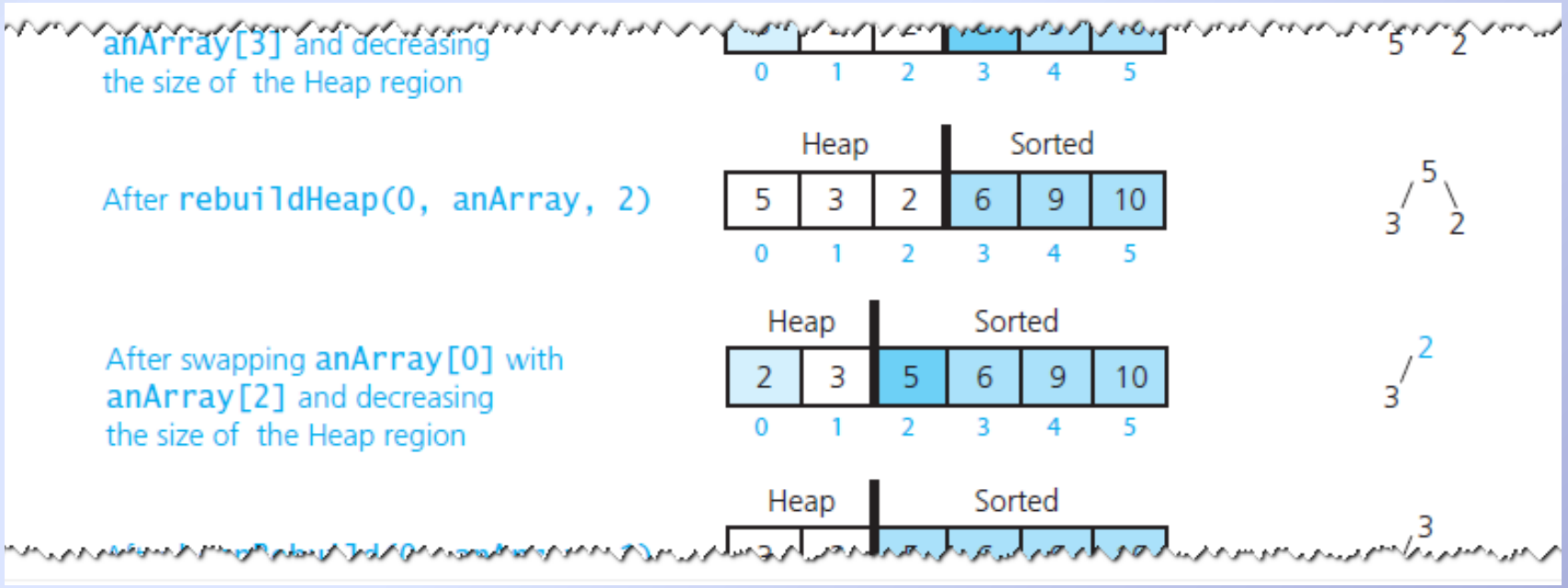


FIGURE 17-11 A trace of heap sort, beginning with the heap in Figure 17-9

Heap Sort

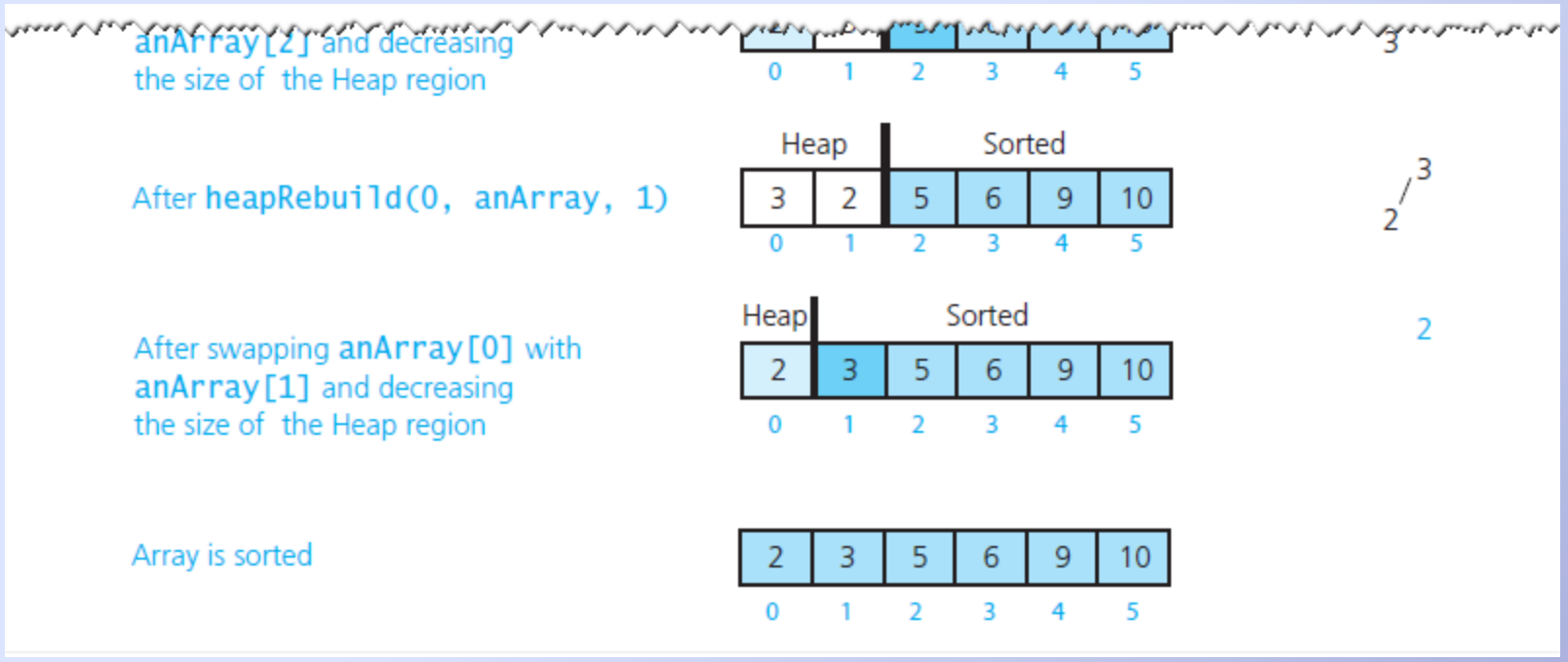
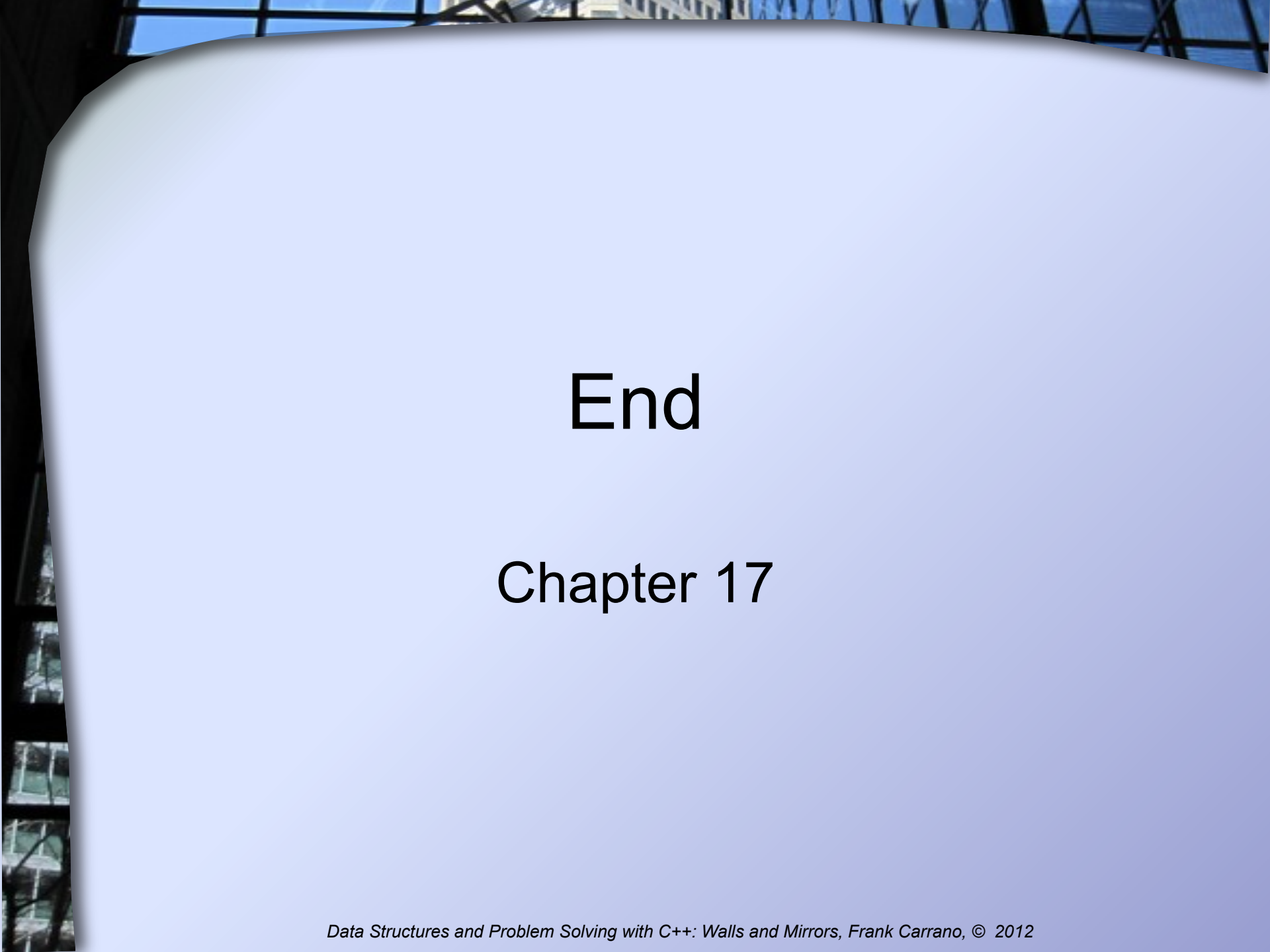


FIGURE 17-11 A trace of heap sort, beginning with the heap in Figure 17-9



End

Chapter 17