computer science
illuminated

Algorithms: Sorting

Nell Dale & John Lewis
(adaptation by Michael Goldwasser)

Sorting

- Because sorting a large number of elements can be extremely time-consuming, a good sorting algorithm is very desirable
- We present several quite different sorting algorithms

Selection Sort

- List of names
  - Put them in alphabetical order
    - Find the name that comes first in the alphabet, and write it on a second sheet of paper
    - Cross out the name on the original list
    - Continue this cycle until all the names on the original list have been crossed out and written onto the second list, at which point the second list is sorted

Selection Sort (cont.)

- A slight adjustment to this manual approach does away with the need to duplicate space
  - As you cross a name off the original list, a free space opens up
  - Instead of writing the minimum value on a second list, exchange it with the value currently in the position where the crossed-off item should go

Selection Sort (pseudocode)

Selection Sort

set current to the index of first item in list

while (not sorted yet)

    find index of the smallest “unsorted” item

    swap current item, smallest unsorted item

    Increment current

Selection Sort (example)

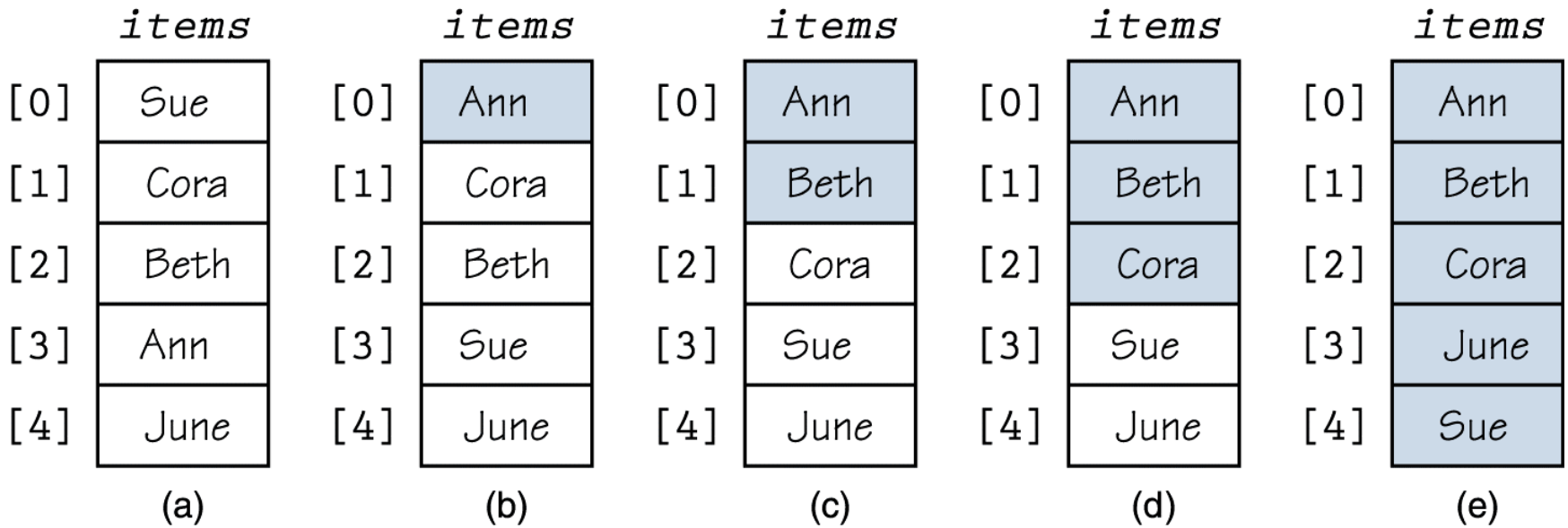


Figure 9.9 **Example of a selection sort (sorted elements are shaded)**

Bubble Sort

- A selection sort that uses a different scheme for finding the minimum value
  - Starting with the last list element, we compare successive pairs of elements, swapping whenever the bottom element of the pair is smaller than the one above it

Bubble Sort (pseudocode)

Bubble Sort

set current to index of first item in the list

do

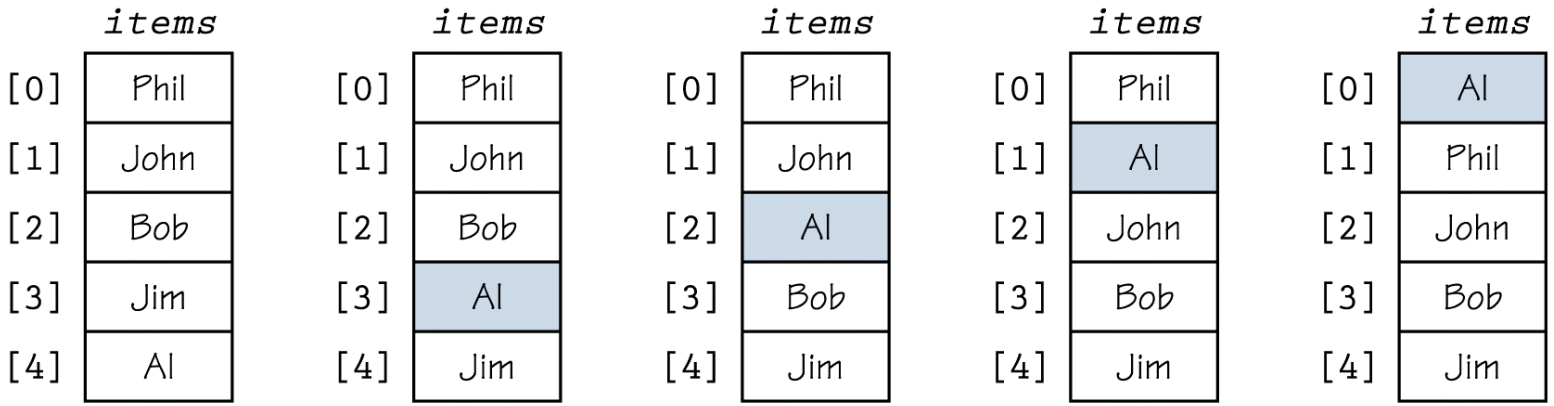
    Set swap to false

    Perform “bubble up” pass

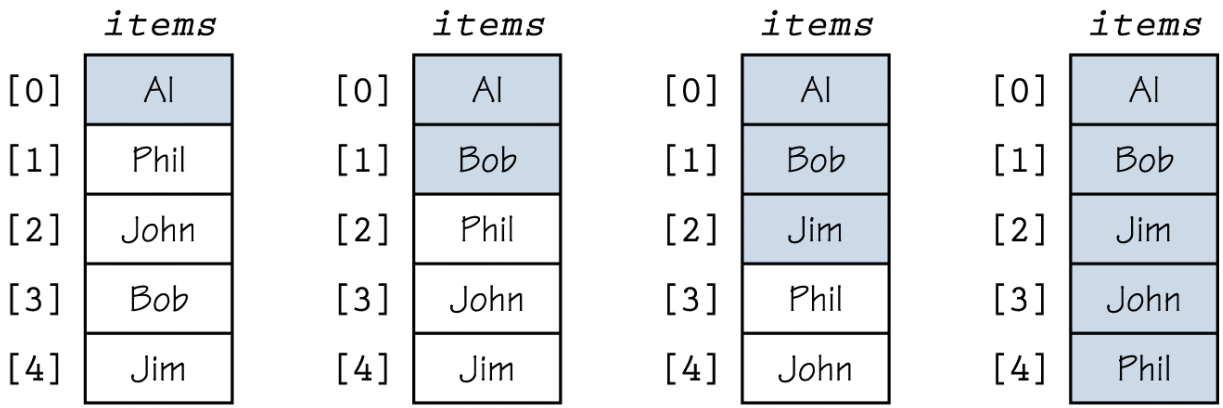
    Increment current

while (swap)

Bubble Sort (example)



a) First iteration (Sorted elements are shaded.)



b) Remaining iterations (Sorted elements are shaded.)

Figure 9.10
Example of a bubble sort

Running Times

- Selection sort

  $O(n^2)$ in all cases

- Bubble sort

  $O(n^2)$ in worst case

  may be as quick as O(n) in some cases.

Quicksort

- Based on the idea that it is faster and easier to sort two small lists than one larger one
  - Given a large stack of final exams to sort by name
  - Pick a splitting value, say L, and divide the stack of tests into two piles, A–L and M–Z
  - note that the two piles do not necessarily contain the same number of tests
  - Then take the first pile and subdivide it into two piles, A–F and G–L
  - This division process goes on until the piles are small enough to be easily sorted by hand

Quicksort

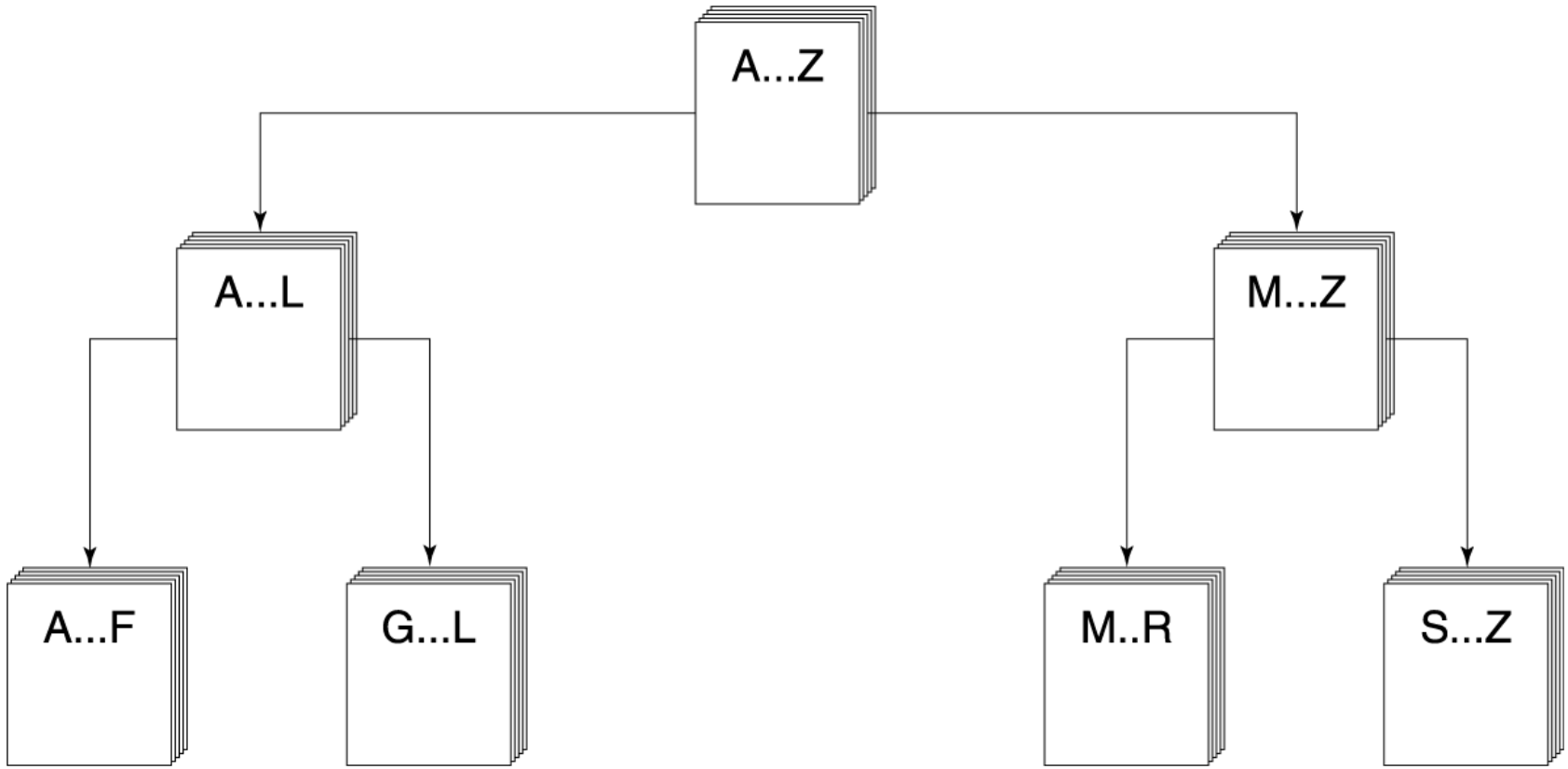


Figure 9.12 Ordering a list using the Quicksort algorithm

Quicksort

Quicksort(first, last)

IF (there is more than one item in list[first]..list[last])

Select splitVal

Split the list so that

    list[first]..list[splitPoint–1] <= splitVal

    list[splitPoint] = splitVal

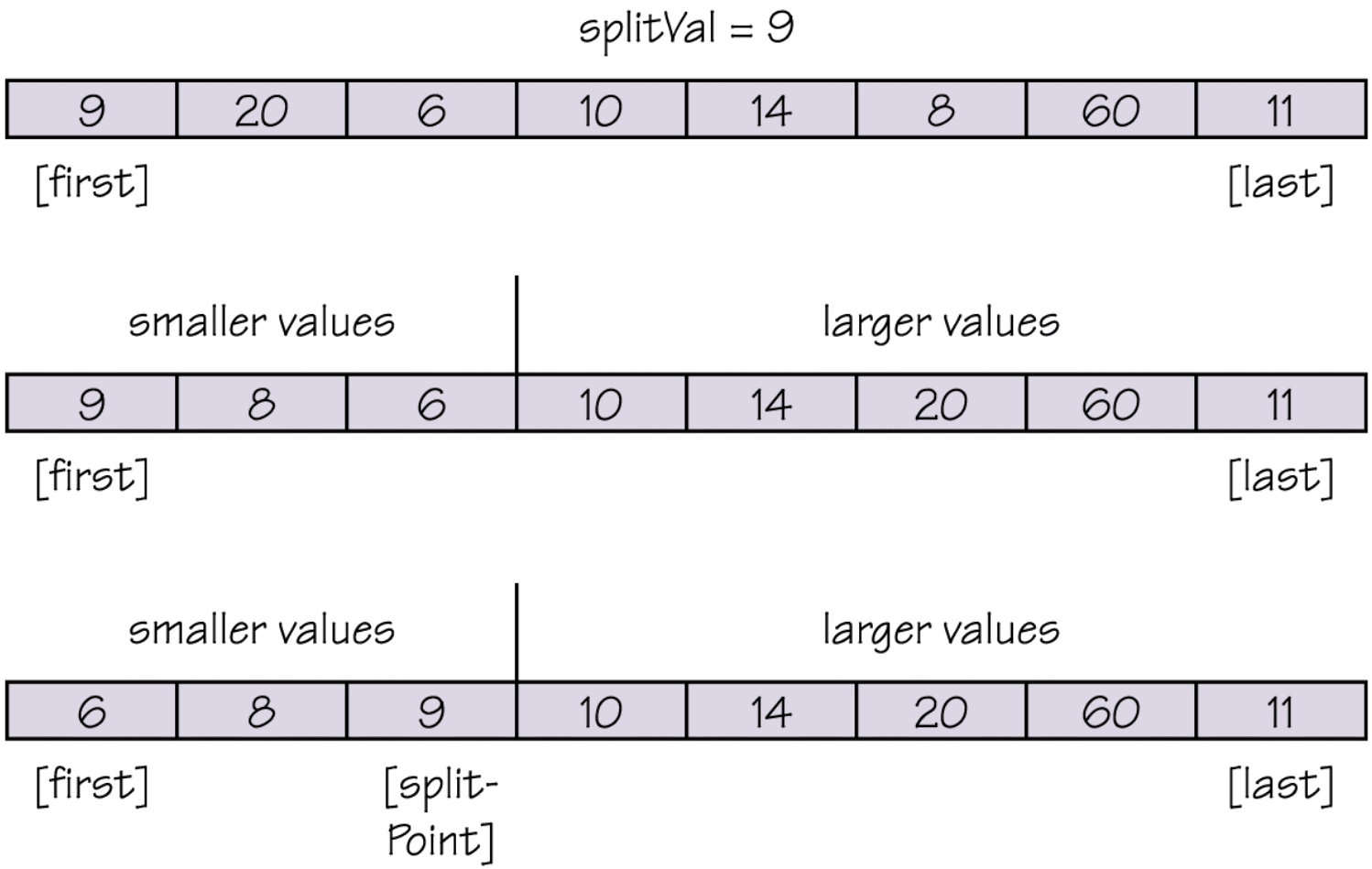
    list[splitPoint+1]..list[last] > splitVal

Quicksort the left half

Quicksort the right half

Page 292

Quicksort


splitVal = 9
9 20 6 10 14 8 60 11
[first]
[last]
smaller values
larger values
9 8 6 10 14 20 60 11
[first]
[last]
smaller values
larger values
6 8 9 10 14 20 60 11
[first]
[split-Point]
[last]

Quicksort Partitioning

```
Set left to first + 1
Set right to last
Do
        Increment left until list[left] > splitVal OR left > right
        Decrement right until list[right] < splitVal OR left > right
        Swap list[left] and list[right]
While (left <= right)
Set splitPoint to right
Swap list[first] and list[right]
```

Page 293

Running Times

- Quicksort

  can be implemented so that it runs in O(n log n) in average case.

  though worst case may be $O(n^2)$

- Other sorts exists that can guarantee

  O(n log n) time

Big-O Analysis

N	$\log_2 N$	$N \log_2 N$	N^2	N^3	2^N
1	0	1	1	1	2
2	1	2	4	8	4
4	2	8	16	64	16
8	3	24	64	512	256
16	4	64	256	4,096	65,536
32	5	160	1,024	32,768	4,294,967,296
64	6	384	4,096	262,144	About 5 years' worth of instructions on a supercomputer
128	7	896	16,384	2,097,152	About 600,000 times greater than the age of the universe in nano-seconds (for a 6-billion-year estimate)
256	8	2,048	65,536	16,777,216	Don't ask!

Table 17.2
Comparison of rates of growth