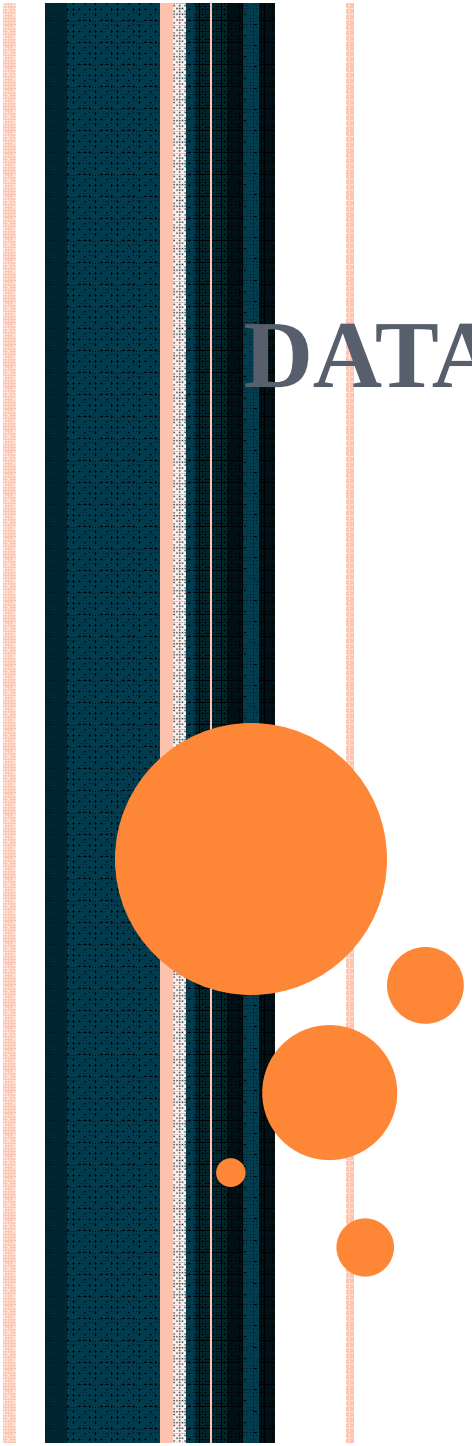
DATA STRUCTURES USING ‘C’



Lecture-10

Data Structures

Different types of Sorting Techniques used in Data Structures

Sorting: Definition

Sorting: an operation that segregates items into groups according to specified criterion.

A = { 3 1 6 2 1 3 4 5 9 0 }

A = { 0 1 1 2 3 3 4 5 6 9 }

Sorting

- Sorting = ordering.
- Sorted = ordered based on a particular way.
- Generally, collections of data are presented in a sorted manner.
- Examples of Sorting:
  - Words in a dictionary are sorted (and case distinctions are ignored).
  - Files in a directory are often listed in sorted order.
  - The index of a book is sorted (and case distinctions are ignored).

Sorting: Cont'd

- Many banks provide statements that list checks in increasing order (by check number).
- In a newspaper, the calendar of events in a schedule is generally sorted by date.
- Musical compact disks in a record store are generally sorted by recording artist.

- Why?
  - Imagine finding the phone number of your friend in your mobile phone, but the phone book is not sorted.

Review of Complexity

Most of the primary sorting algorithms run on different space and time complexity.

Time Complexity is defined to be the time the computer takes to run a program (or algorithm in our case).

Space complexity is defined to be the amount of memory the computer needs to run a program.

Complexity (cont.)

Complexity in general, measures the algorithms efficiency in internal factors such as the time needed to run an algorithm.

External Factors (not related to complexity):

- Size of the input of the algorithm
- Speed of the Computer
- Quality of the Compiler

O(n), Ω(n), & Θ(n)

- An algorithm or function T(n) is O(f(n)) whenever T(n)'s rate of growth is less than or equal to f(n)'s rate.
- An algorithm or function T(n) is Ω(f(n)) whenever T(n)'s rate of growth is greater than or equal to f(n)'s rate.
- An algorithm or function T(n) is Θ(f(n)) if and only if the rate of growth of T(n) is equal to f(n).

Types of Sorting Algorithms

There are many, many different types of sorting algorithms, but the primary ones are:

- Bubble Sort
- Selection Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Shell Sort
- Radix Sort
- Swap Sort
- Heap Sort

Bubble Sort: Idea

- Idea: bubble in water.
  - Bubble in water moves upward. Why?
- How?
  - When a bubble moves upward, the water from above will move downward to fill in the space left by the bubble.

Bubble Sort Example

9, 6, 2, 12, 11, 9, 3, 7

Bubblesort compares the numbers in pairs from left to right exchanging when necessary. Here the first number is compared to the second and as it is larger they are exchanged.

Now the next pair of numbers are compared. Again the 9 is the larger and so this pair is also exchanged.

In the third comparison, the 9 is not larger than the 12 so no exchange is made. We move on to compare the next pair without any change to the list.

The 12 is larger than the 11 so they are exchanged.

The twelve is greater than the 9 so they are exchanged

The end of the list has been reached so this is the end of the first pass. The twelve at the end of the list must be largest number in the list and so is now in the correct position. We now start a new pass from left to right.

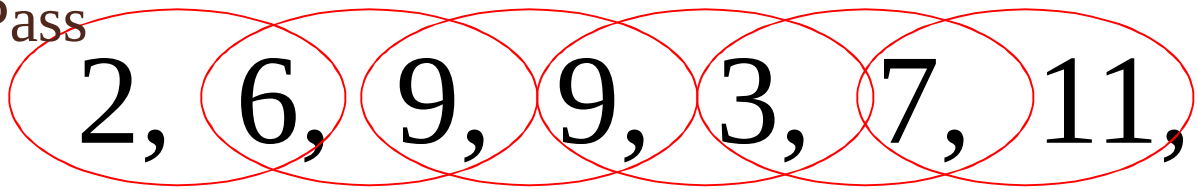
The 12 is greater than the 7 so they are exchanged.

Bubble Sort Example

First Pass

6, 2, 9, 11, 9, 3, 7, 12

Second Pass



Notice that this time we do not have to compare the last two numbers as we know the 12 is in position. This pass therefore only requires 6 comparisons.

Bubble Sort Example

First Pass

6, 2, 9, 11, 9, 3, 7, 12

Second Pass

2, 6, 9, 9, 3, 7, 11, 12

Third Pass

This time the 11 and 12 are in position. This pass therefore only requires 5 comparisons.

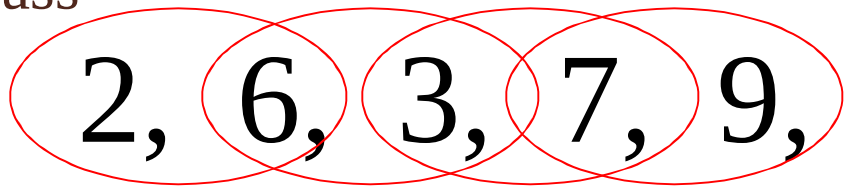
Bubble Sort Example

First Pass 6, 2, 9, 11, 9, 3, 7, 12

Second Pass 2, 6, 9, 9, 3, 7, 11, 12

Third Pass 2, 6, 9, 3, 7, 9, 11, 12

Fourth Pass 2, 6, 3, 7, 9, 9, 11, 12



Each pass requires fewer comparisons. This time only 4 are needed.

Bubble Sort Example

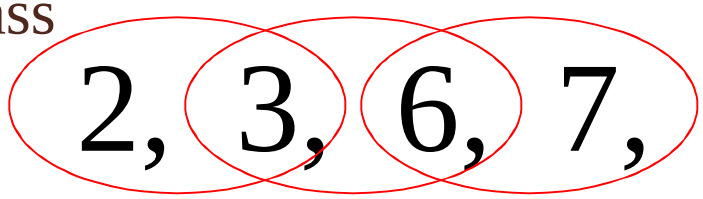
First Pass 6, 2, 9, 11, 9, 3, 7, 12

Second Pass 2, 6, 9, 9, 3, 7, 11, 12

Third Pass 2, 6, 9, 3, 7, 9, 11, 12

Fourth Pass 2, 6, 3, 7, 9, 9, 11, 12

Fifth Pass 2, 3, 6, 7, 9, 9, 11, 12



The list is now sorted but the algorithm does not know this until it completes a pass with no exchanges.

Bubble Sort Example

First Pass 6, 2, 9, 11, 9, 3, 7, 12

Second Pass 2, 6, 9, 9, 3, 7, 11, 12

Third Pass 2, 6, 9, 3, 7, 9, 11, 12

Fourth Pass 2, 6, 3, 7, 9, 9, 11, 12

Fifth Pa

This pass no exchanges are made so the algorithm knows the list is sorted. It can therefore save time by not doing the final pass. With other lists this check could save much more work.

Sixth Pass 2, 3, 6, 7, 9, 9, 11, 12

Bubble Sort Example

Quiz Time

1. Which number is definitely in its correct position at the end of the first pass?

Answer: The last number must be the largest.

2. How does the number of comparisons required change as the pass number increases?

Answer: Each pass requires one fewer comparison than the last.

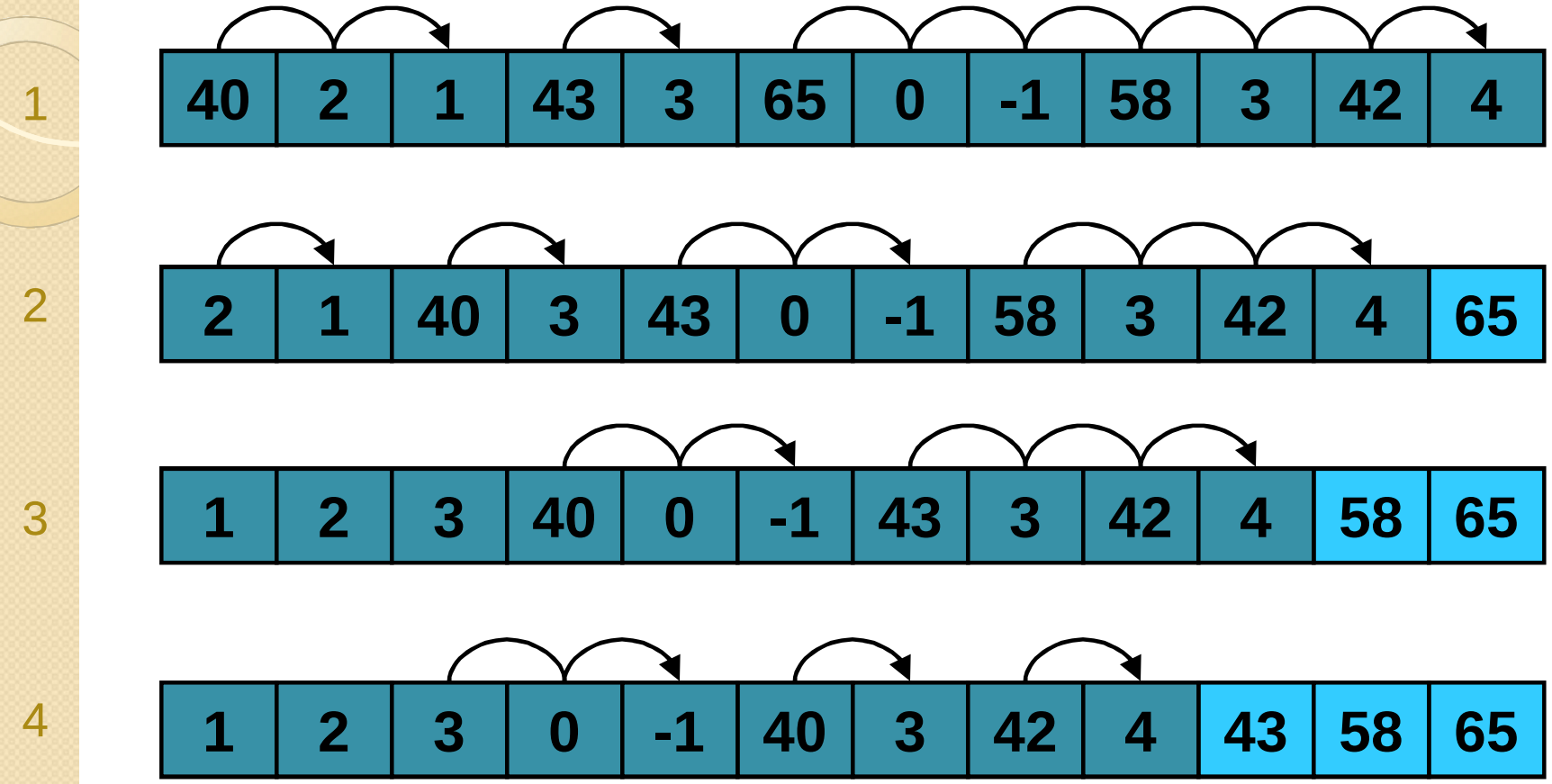
3. How does the algorithm know when the list is sorted?

Answer: When a pass with no exchanges occurs.

4. What is the maximum number of comparisons required for a list of 10 numbers?

Answer: 9 comparisons, then 8, 7, 6, 5, 4, 3, 2, 1 so total 45

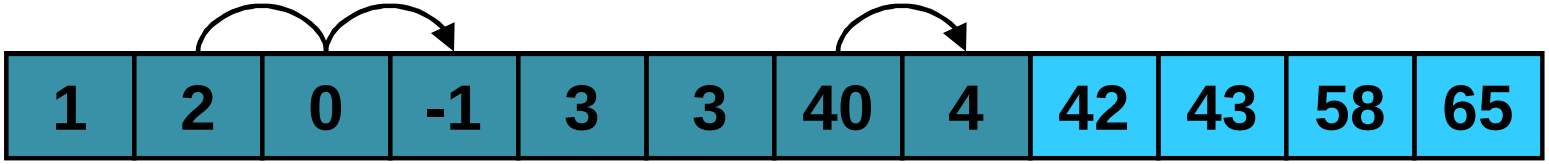
Bubble Sort: Example



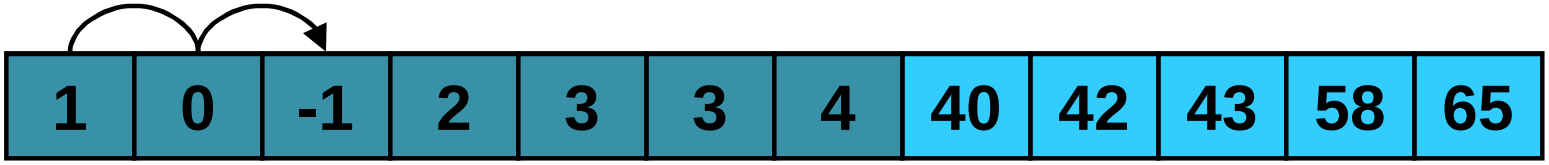
- Notice that at least one element will be in the correct position each iteration.

Bubble Sort: Example

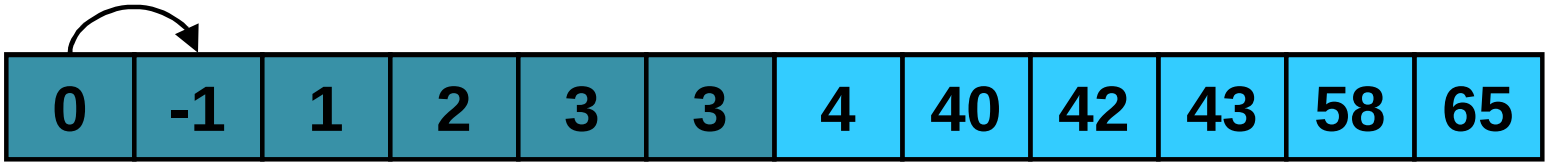
5



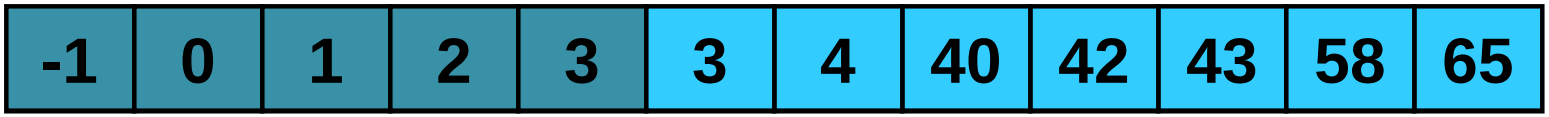
6



7



8



Bubble Sort: Analysis

- Running time:
  - Worst case: $O(N^2)$
  - Best case: O(N)
- Variant:
  - bi-directional bubble sort
    - original bubble sort: only works to one direction
    - bi-directional bubble sort: works back and forth.