Data Structures
Lesson 11

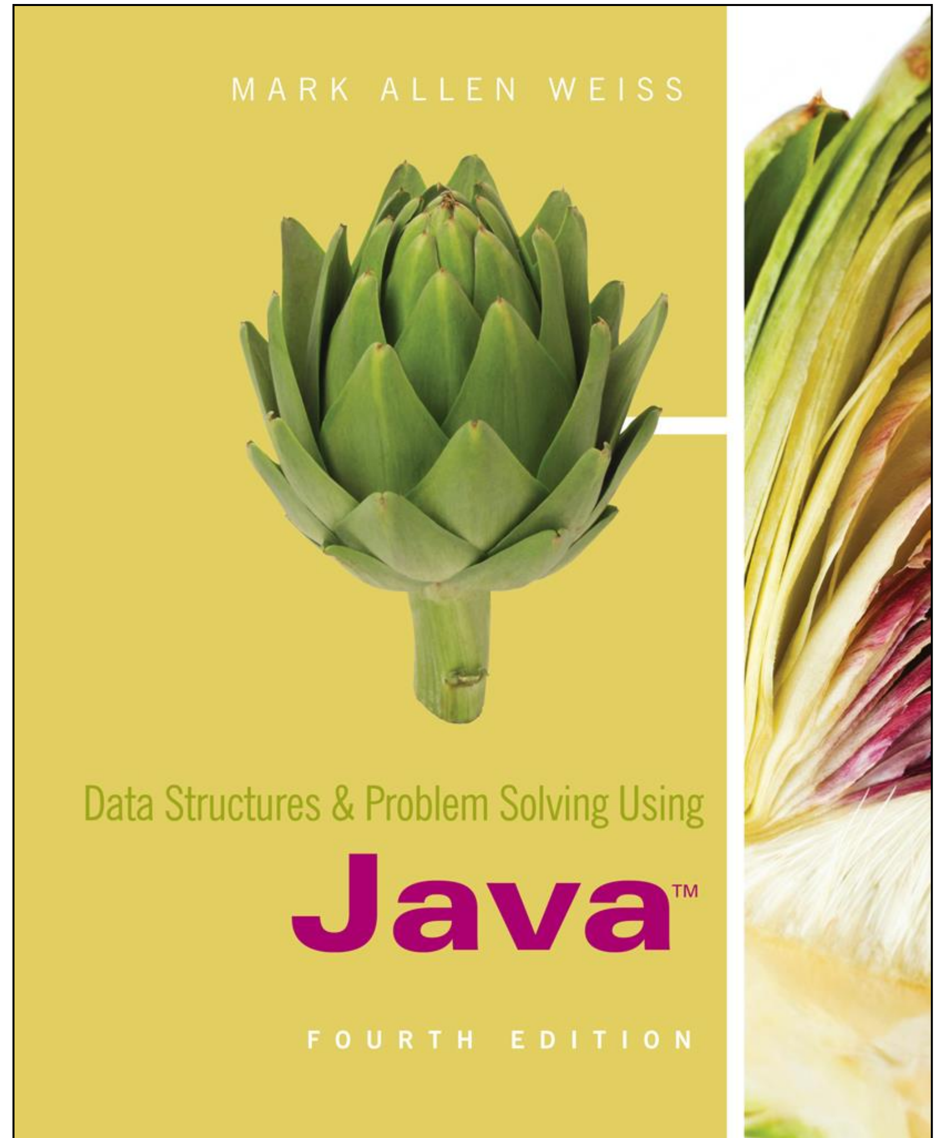
BSc in Computer Science
University of New York, Tirana

Assoc. Prof. Marenglen Biba

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Chapter 8

Sorting Algorithms



Addison-Wesley
is an imprint of
PEARSON

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Sorting

- Sorting is a fundamental application for computers.
- Much of the output eventually produced by a computation is sorted in some way, and many computations are made efficient by invoking a sorting procedure internally.
- Thus sorting is perhaps the most intensively studied and important operation in computer science.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Real-world applications 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).
- The card catalog in a library is sorted by both author and title.
- A listing of course offerings at a university is sorted, first by department and then by course number.
- 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.
- In the programs printed for graduation ceremonies, departments are listed in sorted order and then students in those departments are listed in sorted order.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Topics on sorting

Today's lesson:

- That simple sorts run in quadratic time
- How to code Shellsort, which is a simple and efficient algorithm that runs in subquadratic time
- How to write the slightly more complicated O(N log N): the mergesort algorithm.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Remove duplicates

```
1     // Return true if array a has duplicates; false otherwise
2     public static boolean duplicates( Object [ ] a )
3     {
4         for( int i = 0; i < a.length; i++ )
5             for( int j = i + 1; j < a.length; j++ )
6                 if( a[ i ].equals( a[ j ] ) )
7                     return true;   // Duplicate found
8
9         return false;              // No duplicates found
10    }
```

figure 8.1

A simple quadratic algorithm for detecting duplicates

This algorithm requires quadratic worst-case time.
Sorting provides an alternative algorithm.
If we sort a copy of the array, then any duplicates will be adjacent to each other and can be detected in a single linear-time scan of the array.
The cost of this algorithm is dominated by the time to sort, so if we can sort in subquadratic time, we have an improved algorithm.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Comparison-based sorting

- An algorithm that makes ordering decisions only on the basis of comparisons is called a comparison-based sorting algorithm.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Bubble sort

- A simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order.
- The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted.
- The algorithm gets its name from the way smaller elements "bubble" to the top of the list.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Bubble sort

- Demo
  - Bubble-sort.gif
- Bubble sort has worst-case and average complexity both $O(n^2)$, where n is the number of items being sorted.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Selection sort

- A sorting algorithm, specifically an in-place comparison sort.
- It has O(n^2) time complexity, making it inefficient on large lists, and generally performs worse than the similar insertion sort.
- Selection sort is noted for its simplicity, and also has performance advantages over more complicated algorithms in certain situations, particularly where auxiliary memory is limited.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Selection sort

The algorithm works as follows:

- Find the minimum value in the list
- Swap it with the value in the first position
- Repeat the steps above for the remainder of the list
  - Starting at the second position and advancing each time

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Selection sort

- Demo
  - Selection sort

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Insertion Sort

- Insertion sort is a simple sorting algorithm: a comparison sort in which the sorted array (or list) is built one entry at a time.
- It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort.
- More efficient in practice than most other simple quadratic (i.e., $O(n^2)$) algorithms such as selection sort or bubble sort;
- In the best case (nearly sorted input) is $O(n)$

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Insertion sort

- Every repetition of insertion sort removes an element from the input data, inserting it into the correct position in the already-sorted list, until no input elements remain.
- Sorting is typically done in-place.
- The resulting array after k iterations has the property where the first $k + 1$ entries are sorted.
- In each iteration the first remaining entry of the input is removed, inserted into the result at the correct position, thus extending the result.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Insertion sort

- Demo
  - Insertion-sort.gif

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

figure 8.2

Insertion sort implementation

```
1      /**
2       * Simple insertion sort
3       */
4      public static <AnyType extends Comparable<? super AnyType>>
5      void insertionSort( AnyType [ ] a )
6      {
7          for( int p = 1; p < a.length; p++ )
8          {
9              AnyType tmp = a[ p ];
10             int j = p;
11
12             for( ; j > 0 && tmp.compareTo( a[ j - 1 ] ) < 0; j-- )
13                 a[ j ] = a[ j - 1 ];
14             a[ j ] = tmp;
15         }
16     }
```

Insert tmp in the right position

Shift one element on the right

Counter goes back to find the appropriate position to insert tmp

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Shell sort

- The first algorithm to improve on the insertion sort substantially was Shellsort, which was discovered in 1959 by Donald Shell.
- Though it is not the fastest algorithm known, Shellsort is a subquadratic algorithm whose code is only slightly longer than the insertion sort, making it the simplest of the faster algorithms.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Shell sort

- Shell's idea was to avoid the large amount of data movement, first by comparing elements that were far apart and then by comparing elements that were less far apart, and so on, gradually shrinking toward the basic insertion sort.
- Shellsort uses a sequence $h_1, h_2, \ldots, h_i$, called the increment sequence.
- Any increment sequence will do as long as $h_1 = 1$, but some choices are better than others.
- After a phase, using some increment h_k, we have $a[i] < a[i + h_k]$ for every i where $i + h_k$ is a valid index; all elements spaced h_k apart are sorted.
- The array is then said to be h_k-sorted.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Shell sort

Shellsort is a multi-pass algorithm. Each pass is an insertion sort of the sequences consisting of every *h*-th element for a fixed gap *h* (also known as the increment). This is referred to as *h*-sorting.

figure 8.5

Shellsort after each pass if the increment sequence is {1, 3, 5}

Original	81	94	11	96	12	35	17	95	28	58	41	75	15
After 5-sort	35	17	11	28	12	41	75	15	96	58	81	94	95
After 3-sort	28	12	11	35	15	41	58	17	94	75	81	96	95
After 1-sort	11	12	15	17	28	35	41	58	75	81	94	95	96

After a 5-sort, elements spaced five apart are guaranteed to be in correct sorted order.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Shell sort: what are the gaps?

- Shell suggested starting gap at N/2 and halving it until it reaches 1, after which the program can terminate.
- Using these increments, Shellsort represents a substantial improvement over the insertion sort, despite the fact that it nests three for loops instead of two, which is usually inefficient.
- By altering the sequence of gaps, we can further improve the algorithm's performance.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Running time of Shell sort

		Shellsort		
N	Insertion Sort	Shell's Increments	Odd Gaps Only	Dividing by 2.2
10,000	575	10	11	9
20,000	2,489	23	23	20
40,000	10,635	51	49	41
80,000	42,818	114	105	86
160,000	174,333	270	233	194
320,000	NA	665	530	451
640,000	NA	1,593	1,161	939

figure 8.6

Running time of the insertion sort and Shellsort for various increment sequences

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Performance of Shell sort

- The running time of Shellsort depends heavily on the choice of increment sequences, and in general the proofs can be rather involved.
- The average-case analysis of Shellsort is a long-standing open problem except for the most trivial increment sequences.
- When Shell's increments are used, the worst case is $O(N^2)$.
- Best case: $N\log^2 N$

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

```
1       /**
2        * Shellsort, using a sequence suggested by Gonnet.
3        */
4       public static <AnyType extends Comparable<? super AnyType>>
5       void shellsort( AnyType [ ] a )
6       {
7           for( int gap = a.length / 2; gap > 0;
8                       gap = gap == 2 ? 1 : (int) ( gap / 2.2 ) )
9               for( int i = gap; i < a.length; i++ )
10              {
11                  AnyType tmp = a[ i ];
12                  int j = i;
13
14                  for( ; j >= gap && tmp.compareTo( a[j-gap] ) < 0; j -= gap )
15                      a[ j ] = a[ j - gap ];
16                  a[ j ] = tmp;
17              }
18      }
```

Gonnet Sequence 2.2

compute the gap

Back by gap elements

figure 8.7

Shellsort implementation

Swap elements

Update j with -gap

A gap insertion sort.

Loop 1: gap = 6, i = 6, j = 6; tmp = 17

Loop 2: gap = 6, i = 7, j = 7; tmp = 95

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Merge sort

- Recursion can be used to develop subquadratic algorithms.
- Specifically, a divide-and-conquer algorithm in which two half- size problems are solved recursively with an O(N) overhead results in the algorithm O(N log N).
- Mergesort is such an algorithm.
- It offers a better bound, at least theoretically, than the bounds claimed for Shellsort.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Merge Sort

- An O(N log N) comparison-based sorting algorithm.
- A divide and conquer algorithm that was invented by John von Neumann in 1945.

Conceptually, a merge sort works as follows

- Divide the unsorted list into *n* sublists, each containing 1 element (a list of 1 element is considered sorted).
- Repeatedly Merge sublists to produce new sublists until there is only 1 sublist remaining.
  - This will be the sorted list.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Merge Sort

- Demo
  - Merge-sort.gif

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Merge Sort

- O(N log N) algorithm, because:
  - merging of two sorted groups can be performed in linear time. (more in depth in Algorithms course).
  - How many couples of sorted groups: logN

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Quicksort

- An alternative algorithm is quicksort.
- Quicksort is the algorithm used in C++ to sort all types, and it is used in Java.uti1. Arrays. sort to sort arrays of primitive types.

- Quicksort: next year in Algorithms course.

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

Sorting code

- http://users.cis.fiu.edu/~weiss/dsj4/code/code.html

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved

End of class

- Readings
  - Chapter 8

Copyright © 2010 Pearson Education, publishing as Addison-Wesley. All rights reserved