CST 311
Algorithm Analysis & Design

Al Lake
Oregon Institute of Technology

Chapter 2

Getting Started

Sorting Problem

- **An algorithm contains input, output and a procedure.**
- **The sorting problem is defined as:**
  - **Input: A sequence of *n* numbers $(a_1, a_2, \ldots, a_n)$.**
  - **Output: A permutation or reordering $(a'_1, a'_2, \ldots, a'_n)$ of the input sequence such that $a'_1 \leq a'_2 \leq \ldots \leq a'_n$.**
- **The numbers that are sorted are know as keys.**

Pseudocode

- **The pseudocode for insertion sort is presented as a procedure called Insertion-Sort, which takes as a parameter an array A[1..n] containing a sequence of length *n* that is to be sorted.**
- **The input numbers are sorted in place, rearranged within the array.**
- **A constant number of the array elements are stored outside the array at any time.**
- **The inpute array A contains the sorted output sequence when the sort is finished.**

Insertion-Sort

```
INSERTION-SORT(A)
for j ← 2 to length[A]
     do key ← A[j]
        ▷ Insert A[j] into the sorted sequence A[1 .. j − 1].
        i ← j − 1
        while i > 0 and A[i] > key
            do A[i + 1] ← A[i]
               i ← i − 1
        A[i + 1] ← key
```

Loop invariants and the correctness of insertion sort

Analyzing an Algorithm

- **Analyzing an algorithm means to predict the resources that the algorithm requires.**
- **Principally, the computational time is measured.**

Analysis of Insertion Sort

- **Input size: depends on the problem being solved.**
  - **Number of items in the input**
  - **Total number of bits needed to represent the input**
- **Running time of an algorithm on a particular input is the number of primitive operations or "steps" executed.**

Insertion-Sort

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ **to** $length[A]$	c_1	n
do $key \leftarrow A[j]$	c_2	$n-1$
▷ Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$
$i \leftarrow j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
do $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$i \leftarrow i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
$A[i+1] \leftarrow key$	c_8	$n-1$

Worst-case Running Time

- **Worst-case running time is generally the longest running time for any input of size n.**
- **Worst-case running time is an upper bound on the running time for any input.**

Average Case

- **Average-case or the expected running time of an algorithm is a probabilistic analysis technique.**
- **Average-case will require defining what constitutes the *'average'* input for the problem.**
- **Often, we will assume that all inputs of a given size are equally likely. [how realistic is this?]**
- **Sometimes we will use a randomized algorithm [what are the drawbacks for this?]**

Order of Growth

- **In order to easily accomplish the analysis of the insertion sort we will use simplifying abstractions.**
- **The worst-case running time is:**
  - $an^2 + bn + c$
  - **For some constants *a, b, and c, that depend on the statement costs* c_i.**
  - **So the rate of growth will be** an^2
  - **Since the lower-order terms are relatively insignificant for large n.**
  - **Worst-case time of** $\Theta(n^2)$

Divide-and-conquer

- **Algorithms are generally recursive in structure.**
- **Algorithm analysis is typically a divide-and-conquer approach.**
  - **Divide the problem into a numer of subproblems**
  - **Conquer the subproblems by solving them recursively**
  - **Combine the solutions to the subproblems**

Analyzing Divide-and-conquer Algorithms

- **When an algorithm contains a recursive call to itself, its running time can be describedby a recurrence equation or recurrence.**
- **The recurrence describes the overall running time on a problem of size *n* in terms of the running time on smaller inputs.**