16.070

Introduction to Computers & Programming

Introduction to: Data Structures and Algorithm Analysis

Prof. Kristina Lundqvist
Dept. of Aero/Astro, MIT

Slides based on:

A Practical Introduction to
Data Structures and Algorithm Analysis
Second Edition

Clifford A. Shaffer
Department of Computer Science
Virginia Tech

Abstract Data Types

- **Abstract Data Type** (ADT): a definition for a data type solely in terms of a set of values and a set of operations on that data type.
- Each ADT operation is defined by its inputs and outputs.
- **Encapsulation**: Hide implementation details.

Data Structure

- A **data structure** is the physical implementation of an ADT.
  - Each operation associated with the ADT is implemented by one or more subroutines in the implementation.
- **Data structure** usually refers to an organization for data in main memory.
- **File structure** is an organization for data on peripheral storage, such as a disk drive.

Data Structure Philosophy

- Each data structure has costs and benefits.
- Rarely is one data structure better than another in all situations.
- A data structure requires:
  - **space** for each data item it stores,
  - **time** to perform each basic operation,
  - **programming** effort.

Data Structure Philosophy (cont)

- Each problem has constraints on available space and time.
- Only after a **careful analysis** of problem characteristics can we know the best data structure for the task.

Efficiency

- A solution is said to be **efficient** if it solves the problem within its **resource constraints**.
  - Space
  - Time
- The **cost** of a solution is the amount of resources that the solution consumes.

Estimation Techniques

- Known as “back of the envelope” or “back of the napkin” calculation

1. Determine the major parameters that effect the problem.

2. Derive an equation that relates the parameters to the problem.

3. Select values for the parameters, and apply the equation to yield and estimated solution.

Estimation Example

- How many library bookcases does it take to store books totaling one million pages?
  - Estimate:
    - Pages/inch
    - Feet/shelf
    - Shelves/bookcase

Algorithm Efficiency

There are often many approaches (algorithms) to solve a problem. How do we choose between them?

At the heart of computer program design are two goals.

1. To design an algorithm that is easy to understand, code, debug.
2. To design an algorithm that makes efficient use of the computer's resources.

How to Measure Efficiency?

1. Empirical comparison (run programs)
2. Asymptotic Algorithm Analysis

Critical resources: ?

Factors affecting running time: ?

For most algorithms, running time depends on "size" of the input.

Running time is expressed as **T**(*n*) for some function **T** on input size *n*.

Ada Side Bar

- Array Construct

```
<constant for array size> : constant Integer := <array size>;

type <type name> is array (1 .. <constant for array size>)
                                        of <element type>;
<variable name> : <type name>;
```

- List of Homogenous Elements
- Operations
  - Store `A(i):= C;`
  - Retrieve `C := A(i);`
  - Assignment `A := B;`
  - Equality test `A = B;` `A /= B;`
  - Search
  - Sort

Examples of Growth Rate

Example 1.

Write ada code to find the largest number in the array

```
// Find largest value
int largest(int array[], int n) {
int currlarge = 0; // Largest value seen
  for (int i=1; i<n; i++) // For each val
    if (array[currlarge] < array[i])
      currlarge = i;      // Remember pos
  return currlarge;       // Return largest
}
```

Examples of Growth Rate

Example 1.

```
function Linear_Largest (Input_Array :My_Integer_Array )
     return Integer is

      Largest : Integer := 1;
   begin
      for I in 2.. My_Array_Max loop
         if (Input_Array(I) > Input_Array(Largest))then
            Largest:= I;
         end if;
      end loop;

      return (Largest);
   end Linear_Largest;
```

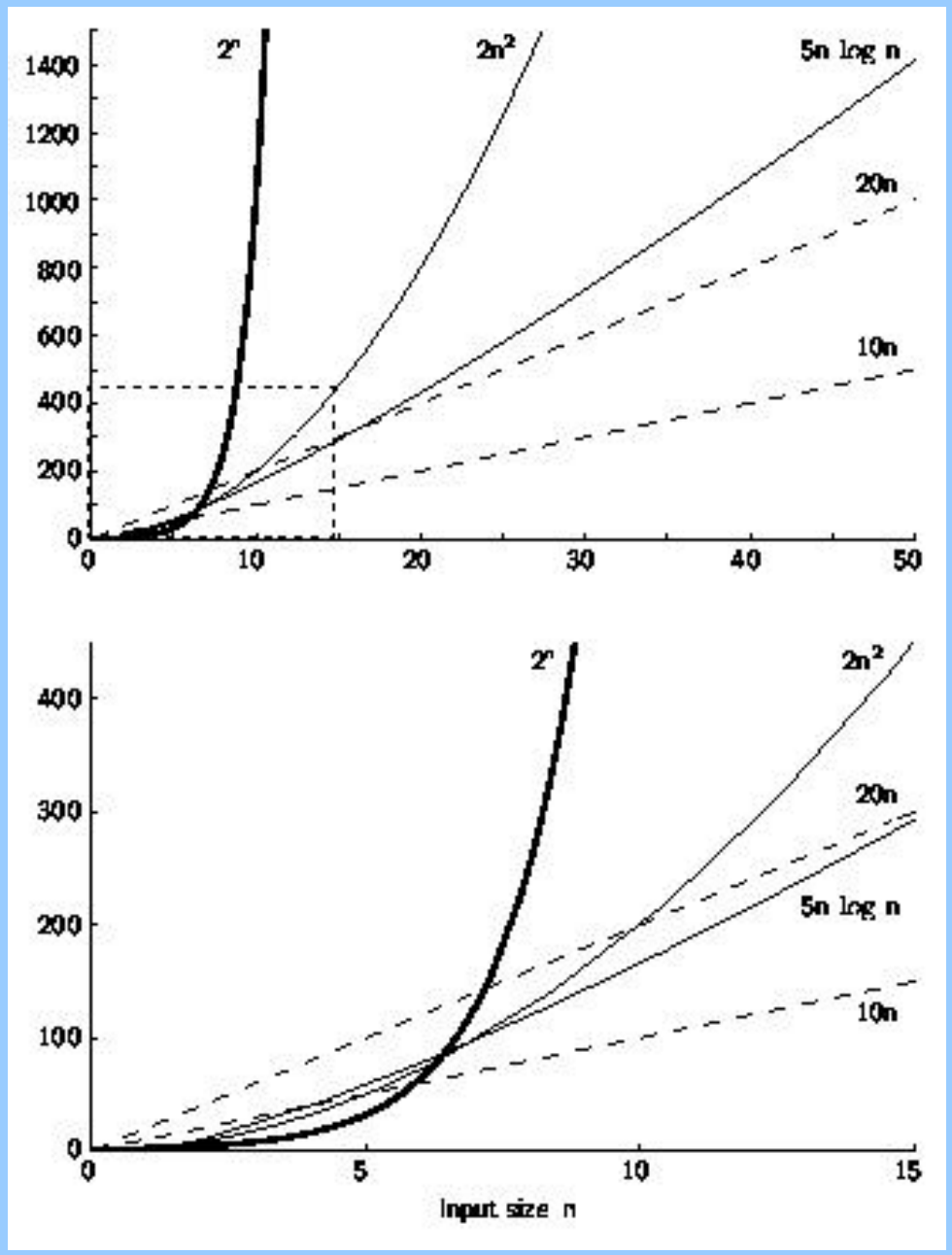
Examples (cont)

Example 2: Assignment statement.

Example 3:

```
sum := 0;
for i in 1 .. N loop
  for j in 1 .. N-1 loop
    sum := sum + 1;
  end loop;
end loop;
```

Growth Rate Graph



Best, Worst, Average Cases

- Not all inputs of a given size take the same time to run.
- Sequential search for *K* in an array of *n* integers:
  - Begin at first element in array and look at each element in turn until *K* is found

Best case: ?

Worst case: ?

Average case: ?

Which Analysis to Use?

- While average time appears to be the fairest measure, it may be difficult to determine.

- When is the worst case time important?

Faster Computer or Algorithm?

- What happens when we buy a computer 10 times faster?

$\mathbf{T}(n)$	n	n'	Change	n'/n
$10n$	1,000	10,000	$n' = 10n$	10
$20n$	500	5,000	$n' = 10n$	10
$5n \log n$	250	1,842	$\sqrt{10}\, n < n' < 10n$	7.37
$2n^2$	70	223	$n' = \sqrt{10}n$	3.16
2^n	13	16	$n' = n + 3$	-----