CC4

Chapter I: Introduction to Algorithms and Data Structures

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

1

Theory of Computation

Early Leanings of Computer Science:

- Use mathematical models to explore **what** computers can do
  - Not restricted to mathematical computations
  - Includes networks, databases, simulations, artificial intelligence, games, security, etc.

Present Leanings:

- Explore **how** to solve computational problems and **how efficient** it can be solved.

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

2

Foundation of Algorithms

- Can a particular task be accomplished by a computing device?
- What is the minimum number of operations for any algorithm to perform a certain function?

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

3

Algorithms

- If a computer can solve a particular computational problem, then there must be an **algorithm** for that problem.
- An algorithm is any **well-defined**, **unambiguous** way/method to solve **all possible instances** of a problem.
- It is a **finite** set of instructions that is **usable by a computer** to accomplish a **particular task**

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

4

Algorithms

- Formally an algorithm

"Consists of a set of explicit and unambiguous finite steps, which when carried out for a given set of initial conditions, produce the corresponding output, and terminate in a fixed amount of time."

- Donald E. Knuth

Guiding Principles of Algorithm Design

Input

- 0 or more quantities which are externally supplied

Output

- At least 1 quantity that is produced

Finiteness

- Must terminate after a finite number of steps; traceability

Definiteness

- Each instruction must be clear and unambiguous

Effectiveness

- Simple but Feasible to implement

Creating Algorithms

- **Requirements:**
  - Information given (**input**) and the results to be produced (**output**) must be **explicitly specified**.
  - **Understand** the problem.
- **Design:**
  - Write an algorithm that will **solve the problem** according to the **requirement**.
- **Analysis:**
  - Trying to come up with **another algorithm** and **compare** it with the previous one.

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

7

Creating Algorithms (cont'd)

- **Refinement and Coding:**
  - A **representation** is chosen and a **complete version** of the program is made.
- **Verification:**
  - Consist of 3 aspects:
    - Program
    - Proving
      - Proof by Induction
      - Proof by Contradiction
    - Testing and Debugging

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

8

Algorithm Example

Algorithm: Bubble Sort, Ascending

Problem: Given an array of integers, arrange elements from lowest value to highest value.

Input: Array of unsorted integers

Output: Array of sorted integers

Algorithms

Algorithm: Bubble Sort, Ascending

Design:

Let j=0, temp=0, array=array of integers

a. Loop through each element pair of the array length
b. Check the value of the integer at array[j]
c. Check the value of the integer at array[j+1]
d. If array[j] > array[j+1], store the integer in temp
e. Place the integer in array[j+1] in array[j]
f. Place the integer in temp in array[j+1]
g. Increment j
h. Repeat steps b to g until length of array, print output

Algorithms

Algorithm: Bubble Sort, Ascending

Refinement and Coding:

```
//Method that accepts an array of integers
static void sort(int arr[]) {
    int n = arr.length;
    //Loop through each element pair given array length n
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            //Check and compare values in indeces j and j+1
            if (arr[j] > arr[j+1]) { //If the integer in index j is greater
                int temp = arr[j];   //Store it in a temporary variable
                arr[j] = arr[j+1];   //index j is now empty, move the integer in index j+1 here
                arr[j+1] = temp;     //index j+1 is now empy, move the integer in temp here
            }
        }
    }
}
```

Verification: Testing and debugging

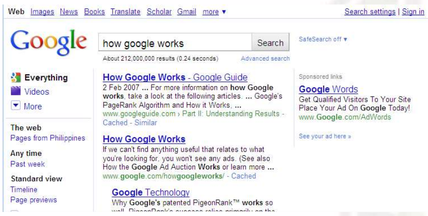
College of Information Technology and Computer Science

CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

11

Importance of Algorithms



The Internet enables people all around the world to quickly access and retrieve (even large amount) information through search algorithms

- Finding good routes from the source host to the destination host uses a clever algorithm
- Search engines that quickly find pages about a particular information, and rank the retrieved results, make use of well-studied algorithms

College of Information Technology and Computer Science

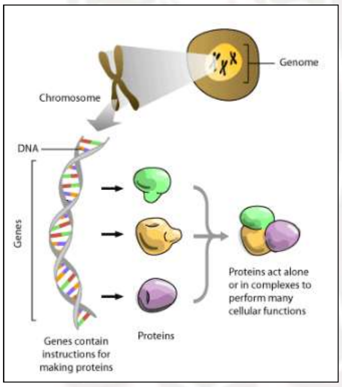
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

12

Importance of Algorithms

- The Human Genome Project involves sophisticated algorithms, enabling it to
  - identify all the (approx.100,000) genes in human DNA
  - determine the sequences of (approx. 3 billion) chemical base pairs that make up the human DNA
  - store all of the derived information in databases
  - develop tools for data analysis



College of Information Technology and Computer Science

by: RBMB,FAP

13

Importance of Algorithms

Far-reaching impact:

- Business (Stock Market Forecasting, Customer Segmentation, Logistics Set-up,…)
- Hard Science Research (Simulations and visualizations for Protein Folding, Chemical Reactions, Particle Collision,…)
- Security (Encryption/Decryption, Voting Machines, Fraud Detection,…)
- Others (Recommender Systems, File Compression, Cancer Detection, Games,…)

College of Information Technology and Computer Science

by: RBMB,FAP

14

Executing Algorithms

- **Machines** for executing algorithms involve the study of various **fabrications** and **organizations** for algorithms to be carried out.
- Languages for describing algorithms has 2 phases:
  - Language design
  - Translation

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

15

Algorithm Analysis

- Know the behavior of the algorithm. This includes the pattern and performance profile of an algorithm; measured in terms of **execution time** and the **amount of space** consumed.
- Computer-related technology is normally associated with **hardware**
  - **Processor** speed
  - **Memory** size

Which is better? Improving hardware or improving algorithms?

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

16

Algorithm Analysis

Making an algorithm more efficient is important:

Illustration 1: Suppose a complicated algorithm for n objects requires n^2 operations

- If each op requires 1 sec, how long will the algorithm run when n = 1000?
  - $n = 1000 \Rightarrow n^2 = 1\,000\,000$
  - Run-time = 1 000 000 ops * (1 sec / op) = 1 000 000 sec
  - Approx. 11 ½ days!
- If a (twice-as-) fast processor is developed so that each operation requires 0.5 second only, how long will the algorithm run?
- A "faster" algorithm requiring only $3n$ operations will finish in how much time?
  - $n = 1000 \Rightarrow 3n = 3000$
  - Run-time = 3000 ops * (1 sec / op) = 3000 sec or 50 min only!

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

17

Analyzing Programs

1. **Priori Estimates**

- Obtain a **function** which **bounds** the algorithm's **time complexity**. The amount of time a single execution will take or the number of times a statement is executed.
- However, it is **impossible** to **know** the **exact** amount of **time** to execute any command **unless**:

  a. the machine we are executing is known;

  b. machine instruction set

  c. time required by each machine instruction;

  d. translation of the compiler from source to machine lang.

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

18

Analyzing Programs

2. **Posteriori Estimates**
   - How memory space is consumed by the algorithm when it runs
   - Example: use of Arrays vs. Linked-lists

Asymptotic Growth and Notation

- Analyzing the **properties** of a growth **function** ***f(n)*** as *n* **increases** in size.
- Asymptotic growth is used to analyze/measure algorithm efficiency by:
  a) Benchmarking
  b) Big-Oh Notation
  c) Frequency Count

Benchmarking

- Actually run algorithms and get elapsed (running) time.
- Pitfalls:
  - Requires implementation of algorithm
    - Presence of bugs
    - Inefficient implementation
  - Requires exactly the same machine set-up
  - Interruption from software features
  - Results are valid only for the test cases used

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

21

Big-Oh Notation

Big-Oh Notation

Common time complexities:

O(1) **<** O(log n) **<** O(n) **<** O(n log n) **<** O(n^2) **<** O(nk) **<** O(2n) **<** O(n!)

Where n is the input size.

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

22

Big-Oh Notation

O(1)

- constant; most instructions are executed once or at most only a few times.

O(log n)

- program slightly slower as N grows; normally in programs that solve a big problem by transforming it into a small problem, cutting the size by some constant factor

O(n)

- linear; proportional to the size of N

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

23

Big-Oh Notation

O(n log n)

- occurs in algorithms that solve a problem by breaking it up into smaller sub-problems, solve them independently and then combining the solution.

O(n^2)

- quadratic; can be seen in algorithms that process all pairs of data items.

O(n^k)

- polynomial; algorithms that process polynomial of data items.

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

24

Big-Oh Notation

O(2^n)

- exponential; brute-force solution

O(n!)

- factorial.

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

25

Example

Given 2 algorithms performing the same task on N inputs:

	P1	**P2**
Actual time complexity:	10n	n^2 / 2
Big-Oh:	O(n)	O(n^2)

Which is faster and more efficient?

College of Information Technology and Computer Science
CENTER OF EXCELLENCE in Information Technology

by: RBMB,FAP

26

Example

Analysis:

N	**P1**	**P2**
1		
5		
10		
15		
20		
30		

Example

Analysis:

N	**P1**	**P2**
1	10	0.5
5	50	12.5
10	100	50
15	150	112.5
20	200	200
30	300	450

P2 is faster and more efficient for **N <= 20**;
BUT P1 proves to be better once **N > 20**

Efficiency Measure Example

N = 10,000

Efficiency	Big-O	Iterations	Estimated Time
Logarithmic	O(logn)	14	Microsecond
Linear	O(n)	10,000	Seconds
Linear Logarithmic	O(nlogn)	140,000	Seconds
Quadratic	$O(n^2)$	$10{,}000^2$	Minutes
Polynomial	$O(n^k)$	$10{,}000^k$	Hours
Exponential	$O(c^n)$	$2^{10{,}000}$	Intractable
Factorial	O(n!)	10,000!	Intractable

Frequency Count

Determine the frequency count and the corresponding Big-Oh Notation.

```
                                        Frequency Count
1.  K = 500;                            1
    for(j=1; j<=K; j++)                 1 + K + 1 + K
        x = x + 1;                      K
    n = 200;                            1

Substituting actual value for K.
    F.C. = 1504
    O(1)
```