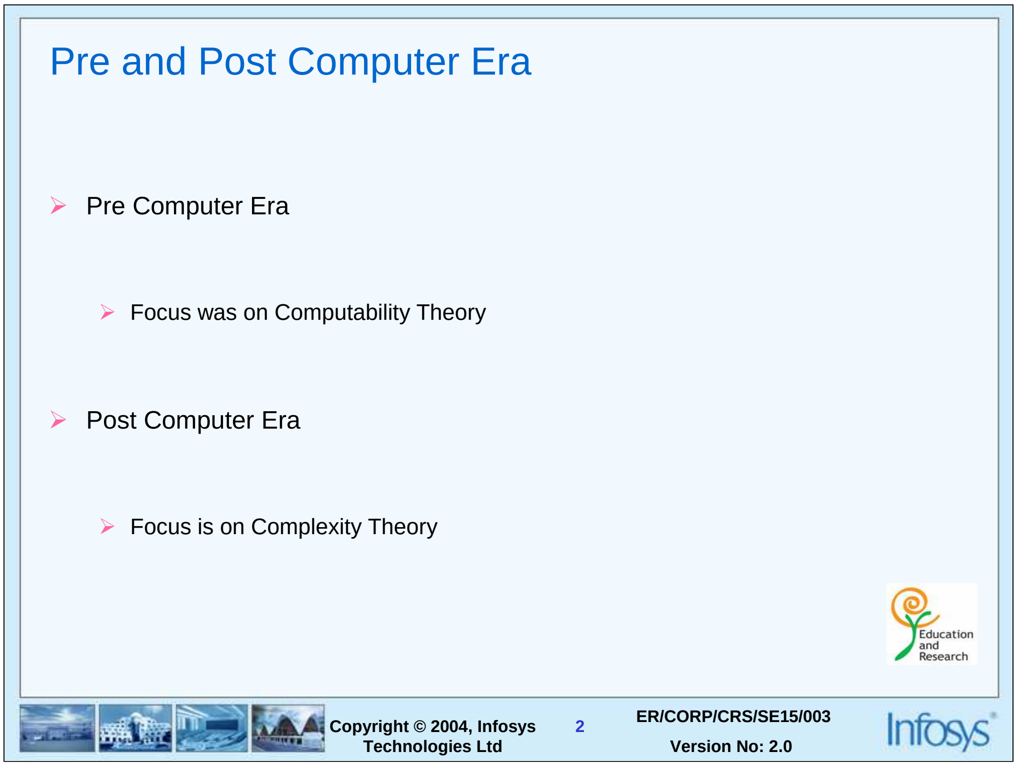
Infosys®
POWERED BY INTELLECT
DRIVEN BY VALUES

Analysis of Algorithms

Unit 1 - Introduction

Education and Research



<u>Pre Computer Era:</u>

Problem Solving using various models of computation existed much before the computer was born. The developments in the field of computer science can be broadly categorized into the Pre and the Post Computer Era.

In the pre computer era various models of computation were proposed. The models worked on a predefined set of operations. The primary focus in this era was on **<u>“What can be computed and what cannot be computed?”</u>** This field of study was called **<u>Computability Theory</u>**.

<u>Post Computer Era:</u>

With ultimate computability in terms of Turing Machines, the primary focus in the post computer era was **<u>Complexity Theory</u>**. People were more interested to find out **<u>“How well can it be computed?”</u>**

Efficiency Measures

- Performance of a solution
  - Most of the software problems do not have a single best solution. Then how do we judge these solutions
  - Performance Measures
    - Time
    - Quality
    - Simplicity…

Education and Research

3
ER/CORP/CRS/SE15/003
Version No: 2.0
Infosys

<u>Why Performance?</u>

Since most of the software problems do not have a unique solution, we are always interested in finding the better solution. A better solution is judged based on its performance. Some of the performance measures include the time taken by the solution, the quality of the solution, the simplicity of the solution, etc.

For any solution to a problem we would always ask the following questions:

<u>“Is it feasible to use this solution?”</u> → In other words is it efficient enough to be used in practice? The efficiency measure which we normally look for is time and space. *How much time does this solution take?*. *How much space (memory) does this solution occupy?*

Improving the performance of a solution can be done by improving the algorithm design, database design, transaction design and by paying attention to the end-user psychology. Also continuous improvements in hardware and communication infrastructure aid in improving the performance of a solution.

Efficiency Measures (Contd …)

- Space Time Tradeoff

Example 1: Consider a personnel management product that an organization can purchase and use to maintain information about its employees. If employee details were to be stored in an array, the array would have to be declared large enough to be able to hold the maximum number of records the system was rated to handle. This would always take up a large amount of memory. With a linked list implementation on the other hand, there would be better utilization of memory.

Which implementation would provide faster access to an employee with a given employee number?

Which implementation would be easier to code?

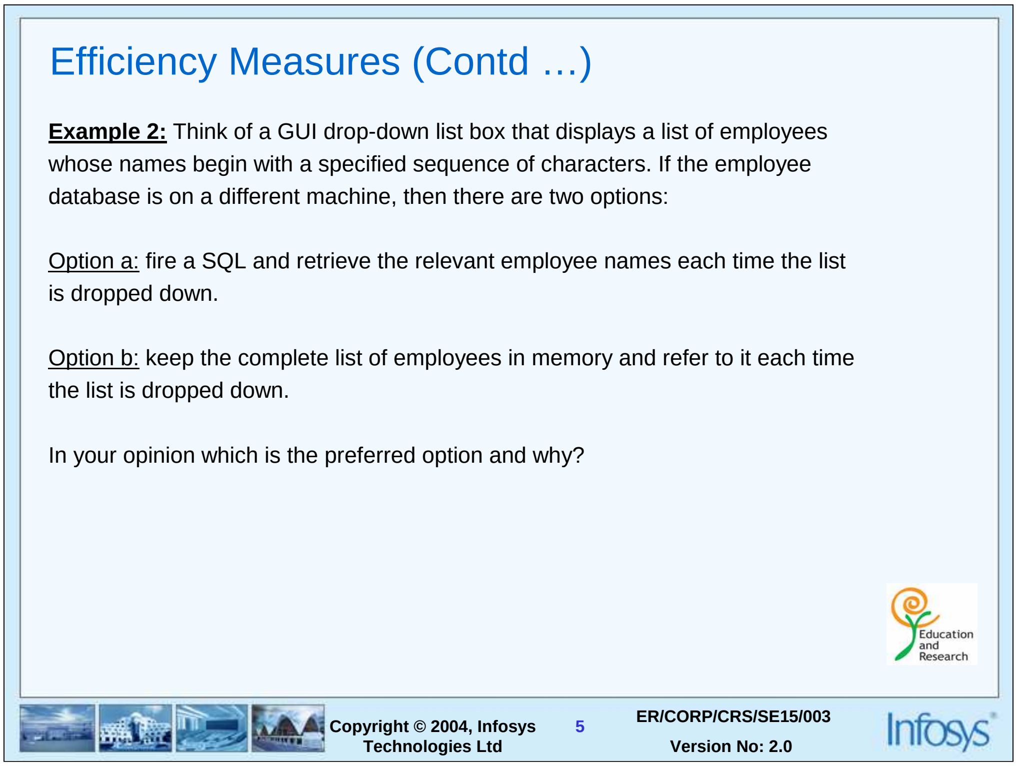
Which implementation would be easier to test?

 4 ER/CORP/CRS/SE15/003 Version No: 2.0

The above mentioned example tries to highlight the need for performance. Each of the three questions asked are aimed at some performance measure.

The array data structure is a better data structure for each of these questions. However if a different company also plans to buy this product, then the size of the array must be very high (which could as well lead to wastage of space). In this case a linked list data structure might be a better option.

This example also highlights an universal problem called the **space time tradeoff**, which we will be discussing shortly.



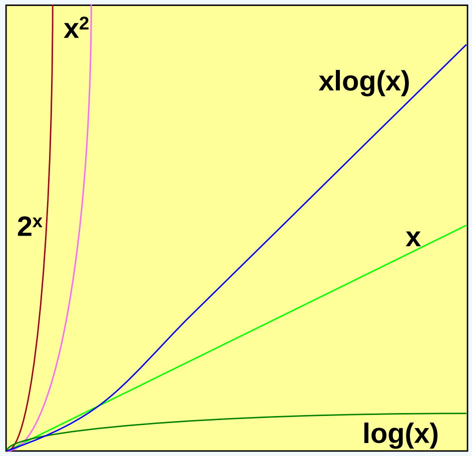
This example again does not have a unique solution. It depends on various parameters which include:

- The number of employees
- The transmission time from the database server to the client machine
- The volume of data transmission each time
- The frequency of such requests.
- The network bandwidth

Neither of the solutions is the better one. The main point here is the ***tradeoff*****.** When ever we need a better performance in terms of time taken, then we could opt for the **option b** which would however lead to more memory requirements. The vice versa is also true. When we want our solution to occupy less memory (space) then we need to strike a compromise for the efficiency in terms of time taken. This tradeoff is called the **space time tradeoff** which is an universal principle. We could never get a solution which is optimal in terms of both the time and space.

Basic Mathematical Principles

- Growth of a few mathematical functions



Education and Research

6
ER/CORP/CRS/SE15/003
Version No: 2.0
Infosys

•As part of Basic Mathematical Principles we will introduce applicable mathematics as required for this course.

•**Growth of functions:** The above figure shows the growth of a few mathematical functions. The *x-axis* varies from 0 to 50 and the *y-axis* varies from 0 to 100. The point to observed here is that the growth rate of the function ***log(x)*** is smaller when compared to the other functions namely ***x, xlog(x),*** x^2 and 2^x. An exponential function like 2^x will ultimately over take any polynomial function. The need to understand the growth of these basic functions will be well appreciated in the later chapters wherein we analyze algorithms.

Basic Mathematical Principles (Contd…)

- A few mathematical formulae.
  - $1 + 2 + \ldots + n = n * (n + 1) / 2$
  - $1 + 2^2 + \ldots + n^2 = n * (n + 1) * (2n + 1) / 6$
  - $1 + a + a^2 + \ldots + a^n = (a^{(n+1)} – 1) / (a – 1)$
  - $\log_a(x) = \log_a(b) * \log_b(x)$
  - The floor of a function f(x), which is written as floor(x) or $\lfloor x \rfloor$, is the largest integer not greater than f(x)

•$\log_a(x)$ is a constant multiple of $\log_b(x)$ for fixed a, b

Basic Mathematical Principles (Contd…)

- How many times should we half the number n (discarding reminders if any) to reach 1?
  - Since n is being divided by 2 consecutively we need to consider two cases.
  - **Case – 1:** n is a power of 2:
    Say for example $n = 8$ in which case 8 must be halved 3 times to reach 1
    $8 \rightarrow 4 \rightarrow 2 \rightarrow 1$. Similarly 16 must be halved 4 times to reach 1.
    $16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$
  - **Case – 2:** n is not a power of 2:
    Say for example $n = 9$ in which case 9 must be halved 3 times to reach 1
    $9 \rightarrow 4 \rightarrow 2 \rightarrow 1$. Similarly 15 must be halved 3 times to reach 1
    $15 \rightarrow 7 \rightarrow 3 \rightarrow 1$. So if $2^m < n < 2^{(m+1)}$ then n must be halved m times to reach 1
  - In general, n must be halved m times if
    $2^m \le n < 2^{(m+1)} \rightarrow m \le \log_2(n) < m+1 \rightarrow m = \mathrm{floor}(\log_2 n)$

Education and Research

•The above mentioned result is necessary for analyzing most of the algorithms

•As a corollary to the above mentioned result we can easily see that a number n must be halved $\underline{\mathrm{floor}(\log_2 n) + 1}$ times to reach 0.

Summary

- Performance of a solution
- Performance Measures
- Space Time Tradeoff
- Basic Mathematical Principles

In this chapter we had studied the need for performance of a solution and looked at a few performance measures. The principle of Space Time Tradeoff was introduced using a couple of examples and finally we had revisited some of the basic mathematical principles which are required for the better understanding of the further chapters.

Thank You!