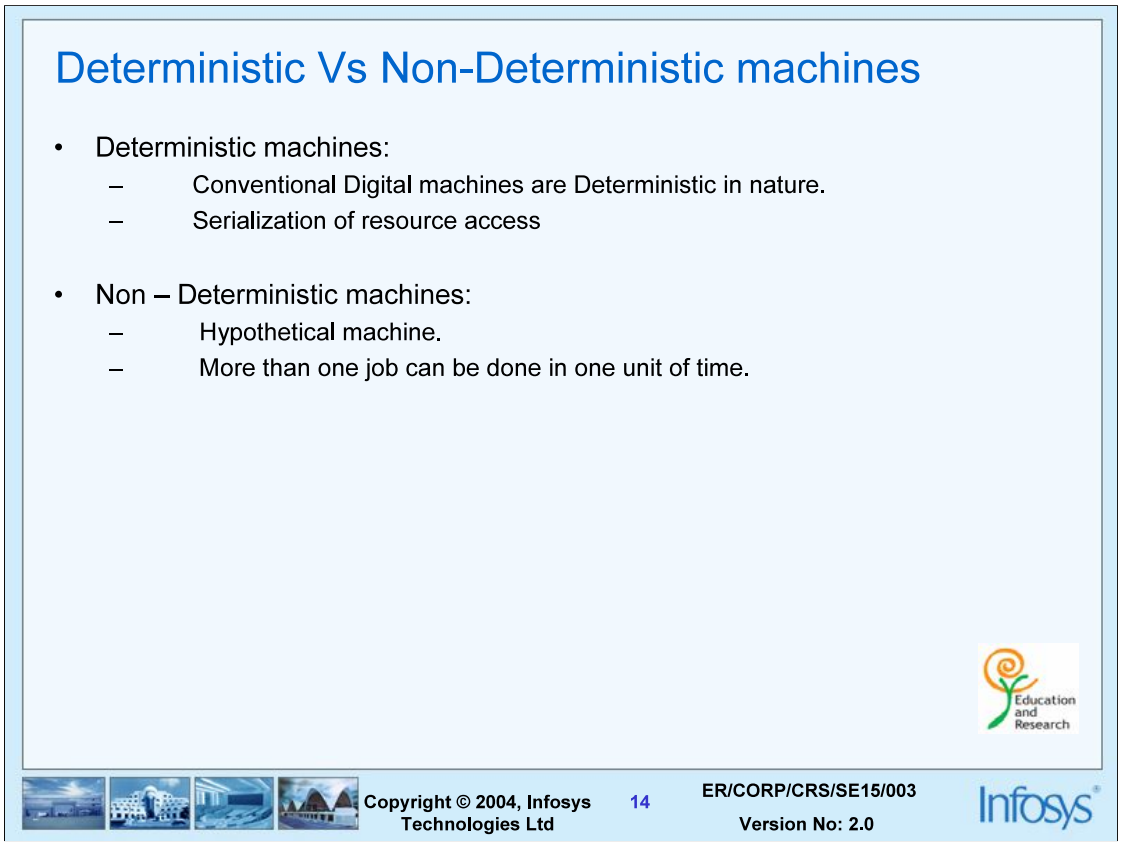


What are deterministic algorithms?

Before we could address this question, let us try to recall how the our conventional digital machines work.

•Conventional Digital Machines are Deterministic

•Conventional Digital Machines do a Sequential Execution. This execution is based on

- •Von Neumann Architecture
- •Serialization of resource access

Such machines are called Deterministic Machines.

In a non-deterministic machine problems can be solved in lesser amount of time than in a deterministic machine because in a non-deterministic machine we can do the jobs in a parallel fashion.

Note: Not to confuse the term non-determinism with Parallel computers, where there will be more than one processor. In Non-deterministic machines there will be only one hypothetical processor which can do more than one job at any instance of time.

Example:

Consider Linear search. Let us consider that the scanning of an element takes 1 unit of time.

In deterministic machines, searching is done by scanning every element. If there are n elements, then the average time taken will be of O(n).

In non-deterministic machines, searching is done in parallel fashion. So the time taken will be of O(1) = 1.

P-Class problems

- Polynomial problems are the set of problems which have polynomial time algorithms
- A formal definition for the same is given below

> The class of decision problems that can be solved in polynomial time **by deterministic algorithms** is called the **P class** or **Polynomial problems**.

Polynomial problems

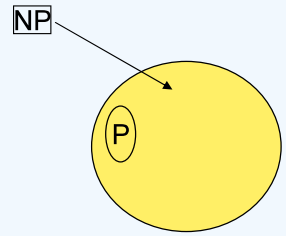
O(1)	--	**Constant**
O(log *n*)	--	**Sub-linear**
O(*n*)	--	**Linear**
O(*n* log *n*)	--	**Nearly linear**
O(n^2)	--	**Quadratic**

NP Problems (Contd…)

- NP problems are the set of problems which have nondeterministic polynomial time algorithms
- A formal definition for the same is given below

> The class of decision problems that can be solved in polynomial time by nondeterministic algorithms is called the **NP class** or **Nondeterministic Polynomial problems**.

- $P \subseteq NP$
- $NP \subseteq P$ Not known

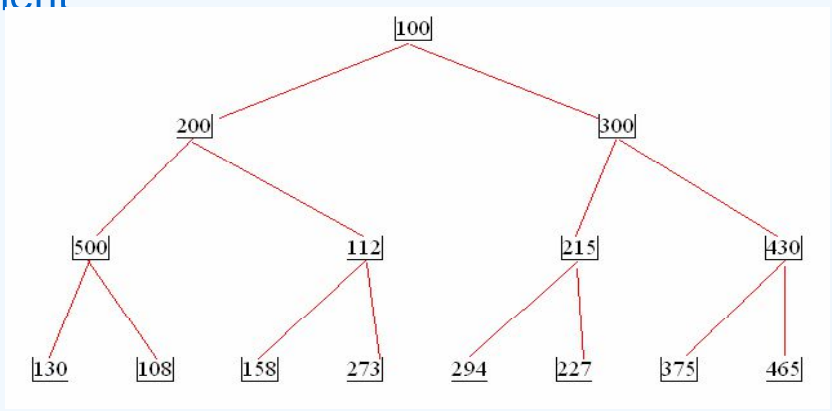


Algorithms which run in Polynomial time on a nondeterministic machine are called nondeterministic polynomial time algorithms.

Decision Problems are problems with **yes/no** answers.

It is easy to see that the P Class is contained in the NP Class. The vice versa is not known. In fact it is one of the most prized open problem which carries a prize money of 1 million US Dollars!

P Vs NP – Deterministic algorithm for searching an element



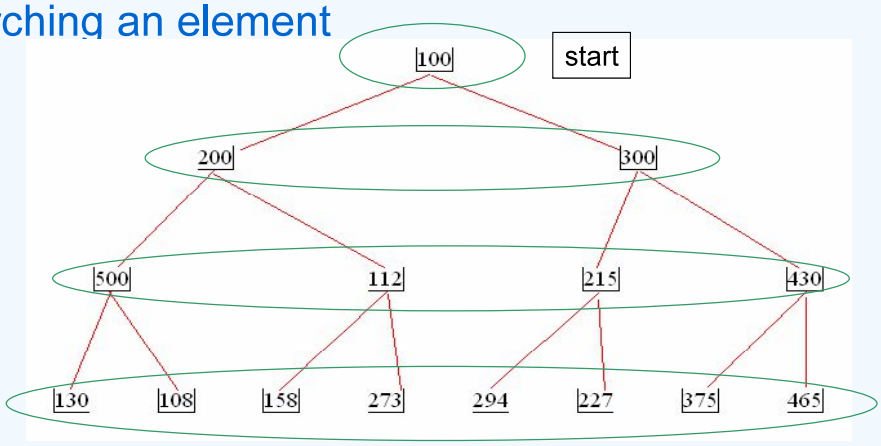
- A binary tree
- No of elements $2^{d+1}-1$
- Searching an element.

The above mentioned problem is to search for a particular element in a binary tree. The depth of the binary tree is **d** and the number of elements in the tree are $\mathbf{2^{(d+1)} - 1}$.

A deterministic algorithm for this problem would be as follows:

Start from the root node (100) and keep traversing *inorder* comparing each element with the search element. In the worst case this would take $\mathbf{O(2^{(d+1)})}$ comparisons for the simple reason that there is a serialization of search. The determinism here refers to the serialization of the search.

P Vs NP (Contd…)- Non deterministic algorithm for searching an element



Machine having special capability- will check all numbers at one level and it will take one unit of time. Hence this will take only 4 unit of time

Suppose that we have a machine (Non-Deterministic machine) which can compare the search element with all the elements at one level in one single operation (one unit of time) then in this case the search algorithm with have a worst case complexity of **O(d),** where d is the depth of the tree. The key point to note here is that the search is not conducted in a serial fashion.

The above example shows that not all problems belong to the **P Class**.

The problems that cannot be solved by any algorithms are called **Undecidable Problems**.

NP Complete problems

- A NP Complete problem is one which belongs to the NP class and which has a surprising property. Every problem in NP class can be reduced to this problem in polynomial time on a deterministic machine
- A formal definition for the same is given below

A decision problem *D* is said to **NP Complete** if

1. If it belongs to NP class
2. Every problem in the NP class is polynomially reducible to *D*

NP Complete problems is a set of problems with the following features:

If one instance of such a problem can be solved using a polynomial algorithm, the complete class of problems can be solved using a polynomial algorithm

NP Hard problems are basically the optimization versions of the problems in NP Complete class. They are problems where in we need to find the optimal solution

One could immediately conclude that if a deterministic polynomial time algorithm is found for an NP Complete problem then every problem in NP class can be solved in deterministic polynomial time.

There are a number of interesting real world problems which belong to the NP Complete class. The Traveling Salesman Problem (TSP), Printed Circuit Board (PCB) problem are examples of NP Complete problems. Some more interesting NP Complete Problems are listed in the next slide.