Analysis of Algorithms

Unit 5 - Intractable Problems

Intractable Problems

- Tractable Problems vs. Intractable Problems
- Polynomial Problems
- NP Problems
- NP Complete and NP Hard Problems

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

In this unit we will:

•Learn about a special collection of problems called Intractable Problems

•Discuss the relationship between the P class problems and NP class problems

•Look at a few techniques to solve intractable problems

•Introduce the NP Complete class and NP Hard class of problems

Tractable Problems vs. Intractable Problems

- An algorithm for a given problem is said to be a *polynomial time* algorithm if it's worst case complexity belongs to **$O(n^k)$** for a fixed integer **k** and an input size of **n**.
- The set of all problems that can be solved in *polynomial time* are called **Tractable Problems**.
- The set of all problems that cannot be solved in polynomial time are called **Intractable Problems**.

Education and Research

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

In this unit our focus is on the problem itself and not on the algorithm. The algorithms that we had analyzed till now had a polynomial time complexity. Polynomial time algorithms are said to be feasible. Algorithms whose worst case complexity belongs to **$O(2^n)$** are called exponential algorithms.

But why are we interested in polynomial time algorithms?

Tractable Problems vs. Intractable Problems (Contd…)

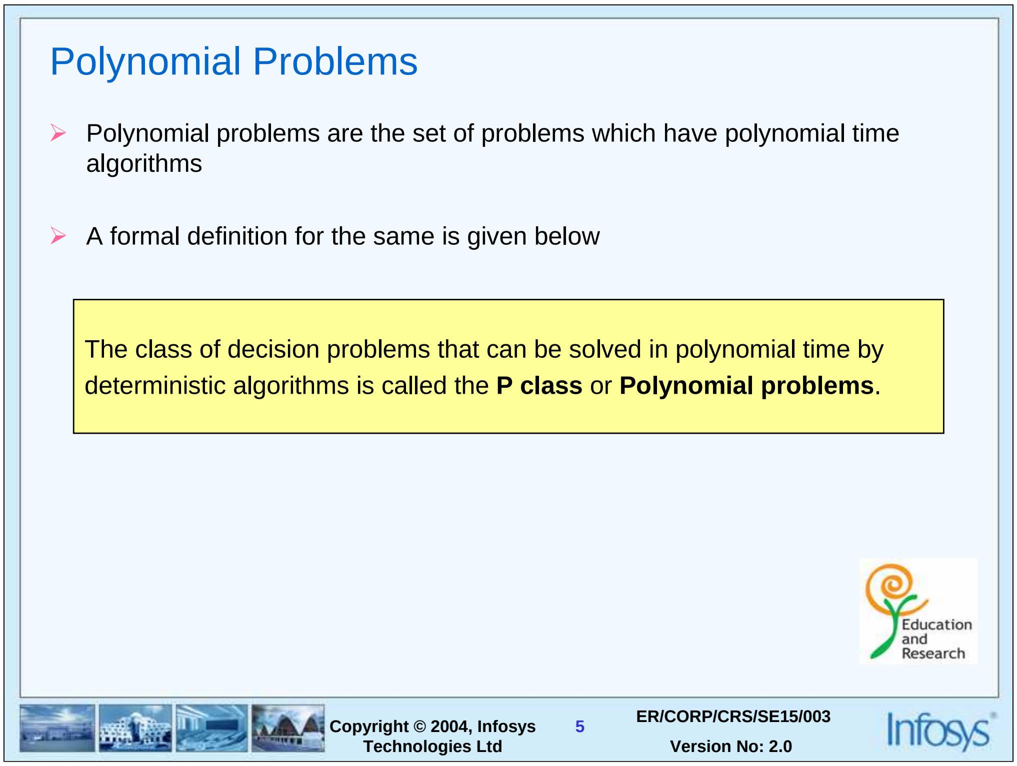
1	1	1	1	1
logn	3.3	4.3	4.9	5.3
n	10	20	30	40
nlogn	33	86	147	212
n^2	100	400	900	1600
n^3	1000	8000	27000	64000
2^n	1024	1 million	1.1 billion	1.1 trillion

Education and Research

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

The above table gives the growth rates of a few functions with respect to the input size **n**. The growth rate of these functions are given row wise. A constant function denoted by **1** has a constant growth (denoted by 1) as the input size varies. The growth rate of an exponential algorithm (**O(2^n)**) is very high, i.e. the number of operations performed by an exponential algorithm is very high.

The reason why we are interested in polynomial time algorithms is evident from the above table.



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

What are deterministic algorithms?

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

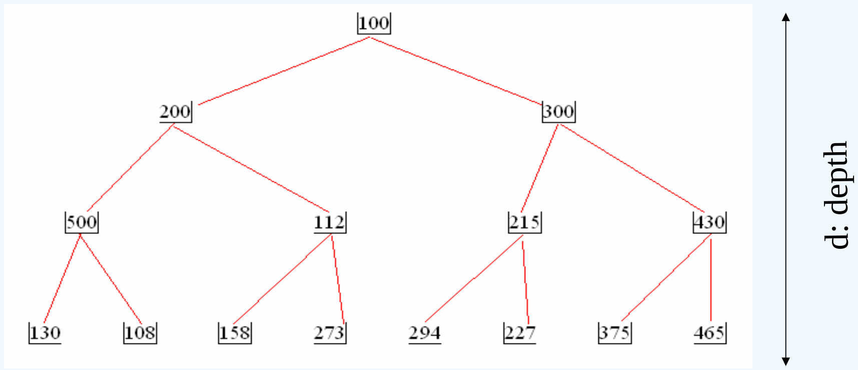
•Conventional Digital Machines are Deterministic

•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. The following problem will help us to understand the concept of determinism in algorithms.

Polynomial Problems (Contd…)



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

Education and Research

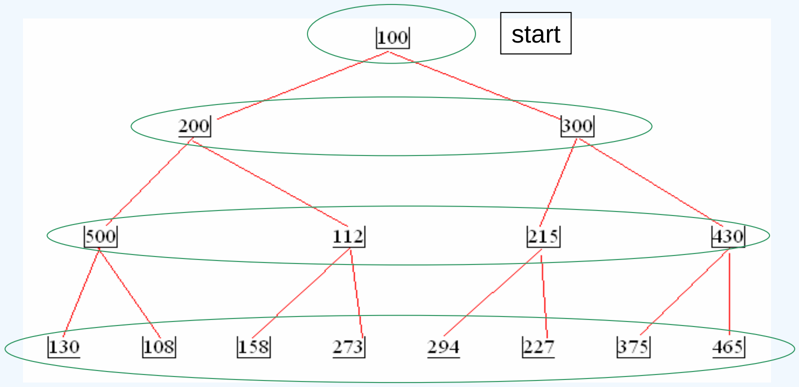
6
ER/CORP/CRS/SE15/003
Version No: 2.0
Infosys

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.

Polynomial Problems (Contd…)



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 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)**. The key point to note here is that the search is not conducted in a serial fashion. Machines of such capability (hypothetical machines) are **Non Deterministic Machines**.

The above example shows that not all problems belong to the **P Class**. The **Halting Problem** is one of the famous problem which shows that not all decision problems are solvable in polynomial time. In fact the **Halting Problem** cannot be solved by any algorithm. Such problems are called **Undecidable Problems**.

Polynomial Problems (Contd…)

Halting Problem (Alan Turing): Given a program *P* and an input *I* to this program determine whether this program will halt on that input or continue working indefinitely

- The halting problem is an Undecidable problem. The halting problem cannot be solved by any algorithm
- The next obvious question would be to see if there exists decidable problems which are Intractable. The answer to this question is yes, but these problems are not of much importance
- But there are a large number of important (real world) problems for which no polynomial time algorithm have been found till date nor it has been proved that it is impossible to get a polynomial time algorithm for such problems

Education and Research

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

The halting problem was given by **Alan Turing**. He showed that there does not exist any algorithm that solves the halting problem. The halting problem is one of the classical example for Undecidable problems.

We will discuss some of the important problems for which no polynomial time algorithm has been found nor the impossibility been proved.

NP Problems

Traveling Salesman Problem (TSP): Given n cities and a salesman who is required to visit each city once, find a route that minimizes the total distance traveled

Printed Circuit Board Problem (PCB): Consider a PCB (Printed Circuit Board) manufacturing unit that churns out several thousands of etched and drilled PC boards of the same type every day. The final stage of the process is drilling. To minimize the drilling time required for each board, we need to decide on the sequence in which holes will be drilled (The drilling time can be cut down by as much as 50% by sequencing the holes using sophisticated algorithms rather than relying on an intuitive or visually judged sequence)

Education and Research

Infosys

The **TSP** problem is one of the famous problem for which there is known polynomial time algorithm. These problems are solved using *Heuristics* or *Approximate Algorithms*.

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**.

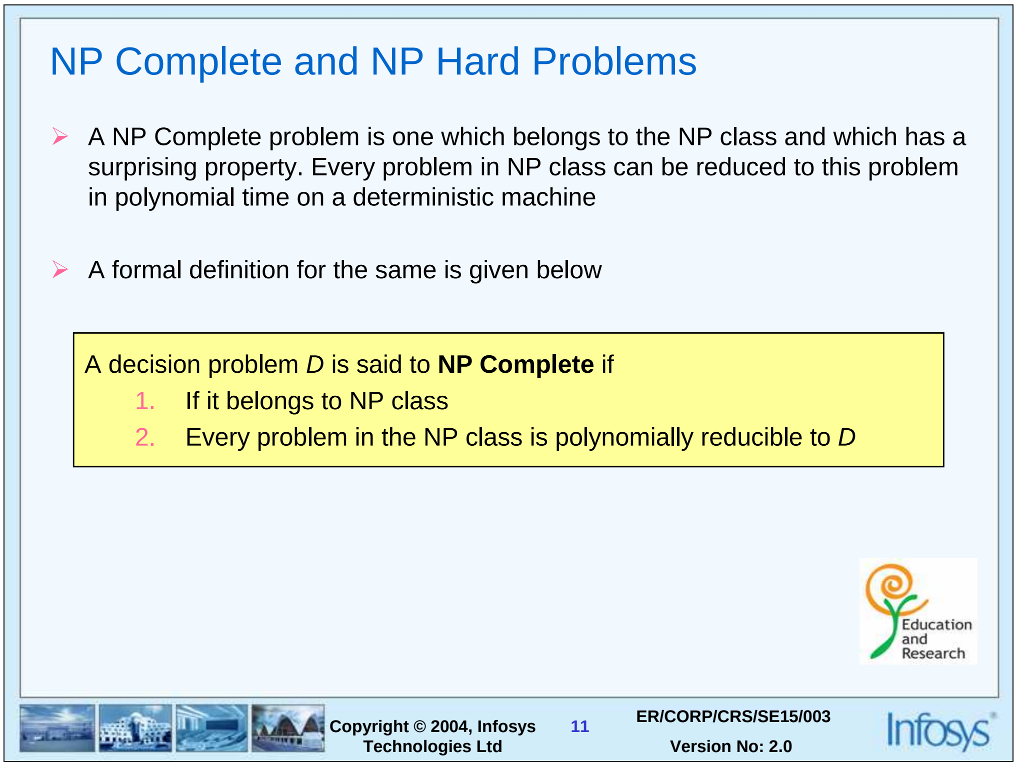
- $P \subseteq NP$
- $NP \subseteq P$ Not known

Education and Research

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

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

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!



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.

NP Complete and NP Hard Problems (Contd…)

Bin Packing Problem: Given a set of items, each with a certain weight and a set of bins, each with a constant weight capacity. You are required to pack the items using the minimum number of bins

Knapsack Problem: Given a set of items, each with a certain weight and value and a knapsack with a certain weight capacity. You are required to pack the knapsack with items so as to maximize the value of the packed items

Node Cover Problem: You are given a network that consists of a set nodes, and a set of edges. An edge is a connection between two nodes. You are required to find a minimal set of nodes S such that every other edge in this network is connected to at least one node in S

The above mentioned problems are all NP Complete problems!

NP Complete and NP Hard Problems (Contd…)

- A problem *P* is said to be a NP Hard problem if any problem (not necessarily in NP) is polynomially reducible to *P*
- NP Hard problems are basically the optimization versions of the problems in NP Complete class
- The NP Hard problems are not mere yes/no problems. They are problems where in we need to find the optimal solution

Summary

- Tractable Problems vs. Intractable Problems
  - Need for polynomial time solutions
- Polynomial Problems
  - Deterministic vs. Non Deterministic Algorithms
- NP Problems
  - P vs. NP
- NP Complete and NP Hard Problems

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

In this chapter we had:

•Studied about tractable and intractable problems

•Understood the need for polynomial time solutions

•Studied the difference between Deterministic and Nondeterministic algorithms

•Studied the NP class of problems and understood the NP Complete class of problems

•Understood the difference between NP Complete and NP Hard Problems