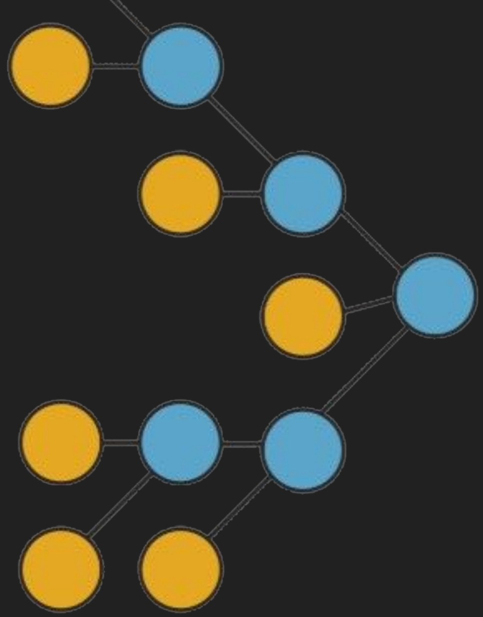
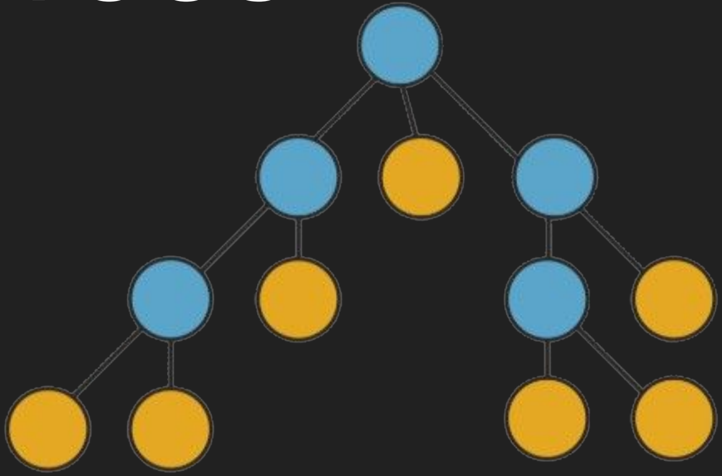


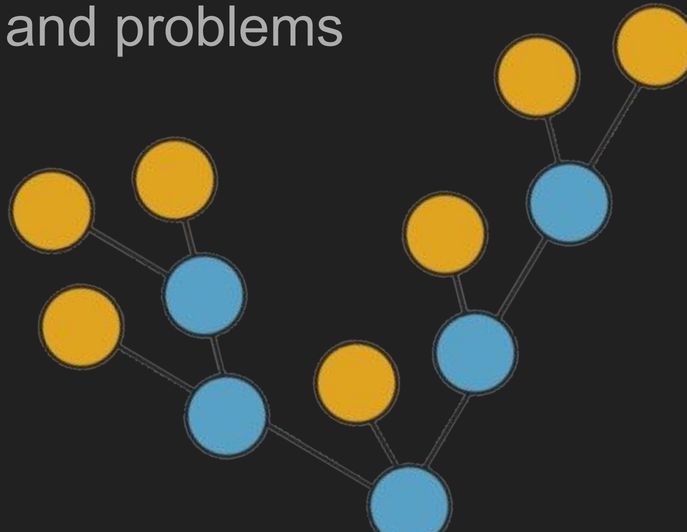
Decision Trees

Sivaani J



Topics To Discuss

- Limitations of Algorithmic Power
- Decision Trees
- Deterministic and Non Deterministic Algorithms
- P, NP, NP-Complete and NP-Hard classes and problems



Limitations of Algorithmic Power

- ➢ Some problems cannot be solved by any algorithm.
- ➢ Some problems can be solved algorithmically, but not in a polynomial time.
- ➢ Few problems have lower bound for their efficiency.
- ➢ Most problems cannot be solved exactly.

Travelling Salesman Problem, Hamiltonian Problem, Sudoku, The Halting Problem

Polynomial Time:

- ➢ The amount of time it takes for an algorithm to solve a polynomial function, which is a mathematical expression that does not contain fractions or negative numbers. The time is proportional to the input and very efficient. (Tractable)

Decision Trees

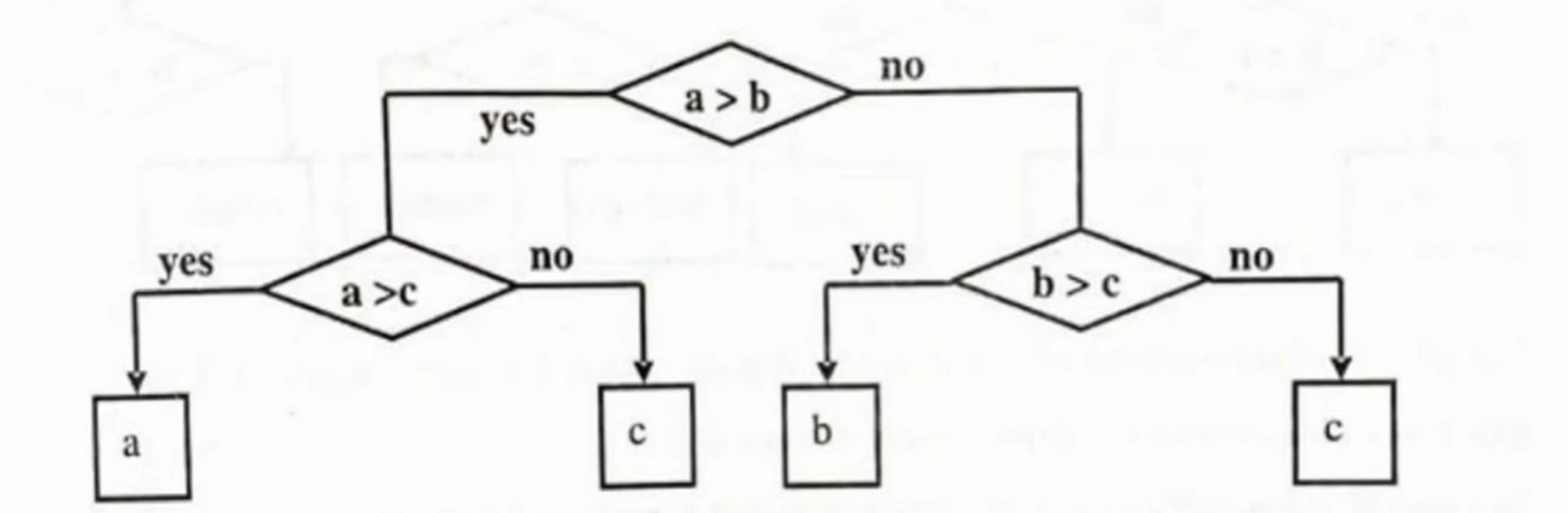
A Decision Tree is also called comparison tree is a binary tree that represents only the comparisons of given elements in the array while sorting or while searching.

A model of computation where decisions are made based on a sequence of comparisons or tests.

By studying properties of decision trees for such algorithms, we can derive important lower bounds on their efficiencies.

Decision trees are a tool for deterministic processes — where decisions are made one step at a time, in a fixed, logical way.

Obtain the decision tree to find minimum of three numbers.



Deterministic and Non Deterministic Algorithms

Deterministic Algorithms:

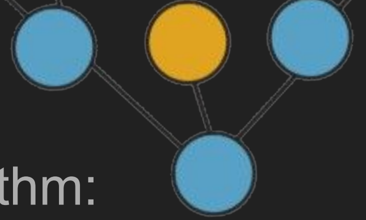
A **deterministic algorithm** is an algorithm that, for a given input, **always produces the same output** and follows **the same sequence of steps** every time you run it.

- Predictable - No randomness - One possible path of execution

Binary Search: For a sorted array, it always follows the same steps to find a target.

Bubble Sort: The steps and comparisons are fixed.

Analogy:
It's like following a **recipe** exactly the same way every time — the outcome is always the same.



Non-Deterministic Algorithm:

A **non-deterministic algorithm** may **produce different outcomes** for the same input on different runs, or can **choose** from multiple paths to reach a solution — often as if by "guessing" the right path.

A nondeterministic algorithm is a two-stage procedure, a guessing stage (non deterministic stage) and a verification stage (deterministic stage)..

-Multiple possible paths -May involve "guessing" -Used in theoretical models like **NP problems.**

Solving Sudoku: A machine guesses numbers in empty cells and instantly "magically" checks all valid combinations.

0/1 Knapsack: Many subsets of items; checking a solution is easy, but selecting the best one is hard. TSP O(n^2 2^n) Knapsack O(2^n/2)

P, NP, NP-Complete and NP-Hard classes and problems

P Problem (Polynomial Time Problem)

Definition:
Problems that can be solved in polynomial time by a deterministic algorithm.

Meaning:
The time it takes to solve the problem grows at a reasonable (polynomial) rate as the input size increases.

Examples:

- Sorting algorithms (like Merge Sort): `O(n log n)`
- Binary Search: `O(log n)`
- Dijkstra’s Algorithm (shortest path in graphs)

NP Problem (Nondeterministic Polynomial Time Problem)

Definition:
Class NP is the class of decision problems that can be solved by nondeterministic algorithms. Non-deterministic Polynomial problems are problems where positive solutions should be obtained in polynomial time.

Meaning:
Hard to solve, but **easy to check** if a proposed solution is correct.

Examples:

- **Sudoku**: Verifying a completed grid is easy.
- **Subset Sum Problem**: Does a subset of numbers add to a given value?
- **Hamiltonian Path**: Does a path visit all nodes exactly once?

NP-Complete Problem

Definition:
In NP (verifiable in polynomial time), and **As hard as any problem in NP** — if you solve one NP-complete problem efficiently, you can solve **all NP problems** efficiently. These are the **hardest** problems in NP.

Examples:

- **Traveling Salesman Problem** (decision version): Is there a route visiting all cities with total distance $\leq k$?

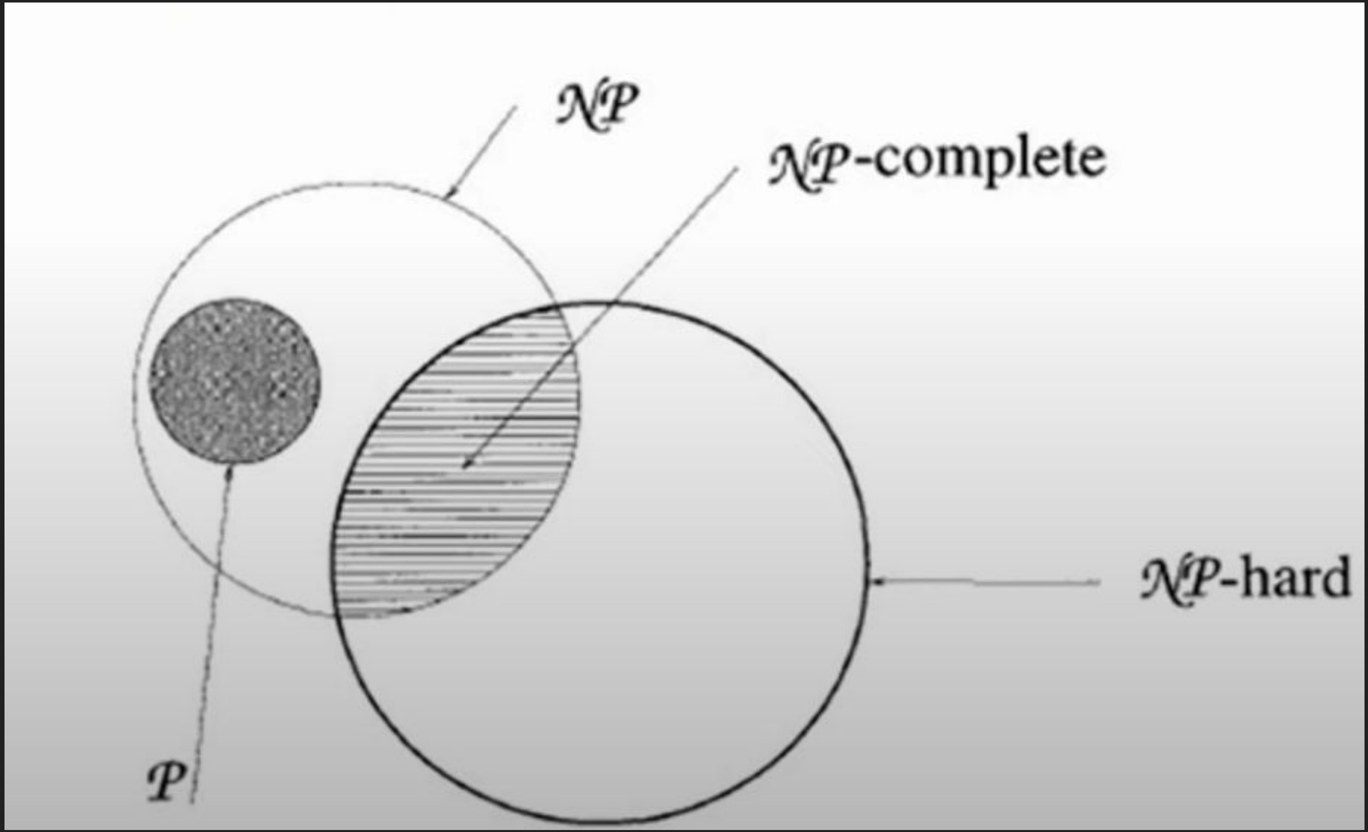
NP-Hard Problem

Definition:
Problems that are **at least as hard as NP-Complete problems**, but they **don't have to be in NP**. This means they **might not even be verifiable** in polynomial time. **All NP-Complete problems are NP-Hard**, but **not all NP-Hard problems are NP-Complete**.

Examples:

- **Halting Problem**: Will a program eventually stop? (Undecidable)
- **Chess Game Solver**: Will White win from a given position?

The big question: **Is P equal or subset to NP?** If yes, many problems in security, AI, and optimization could become easy to solve.
If no, there are problems we'll **never solve efficiently**.It's one of the **biggest unsolved problems** in computer science.



Thank You