20CYS304 Artificial Intelligence and Neural Networks

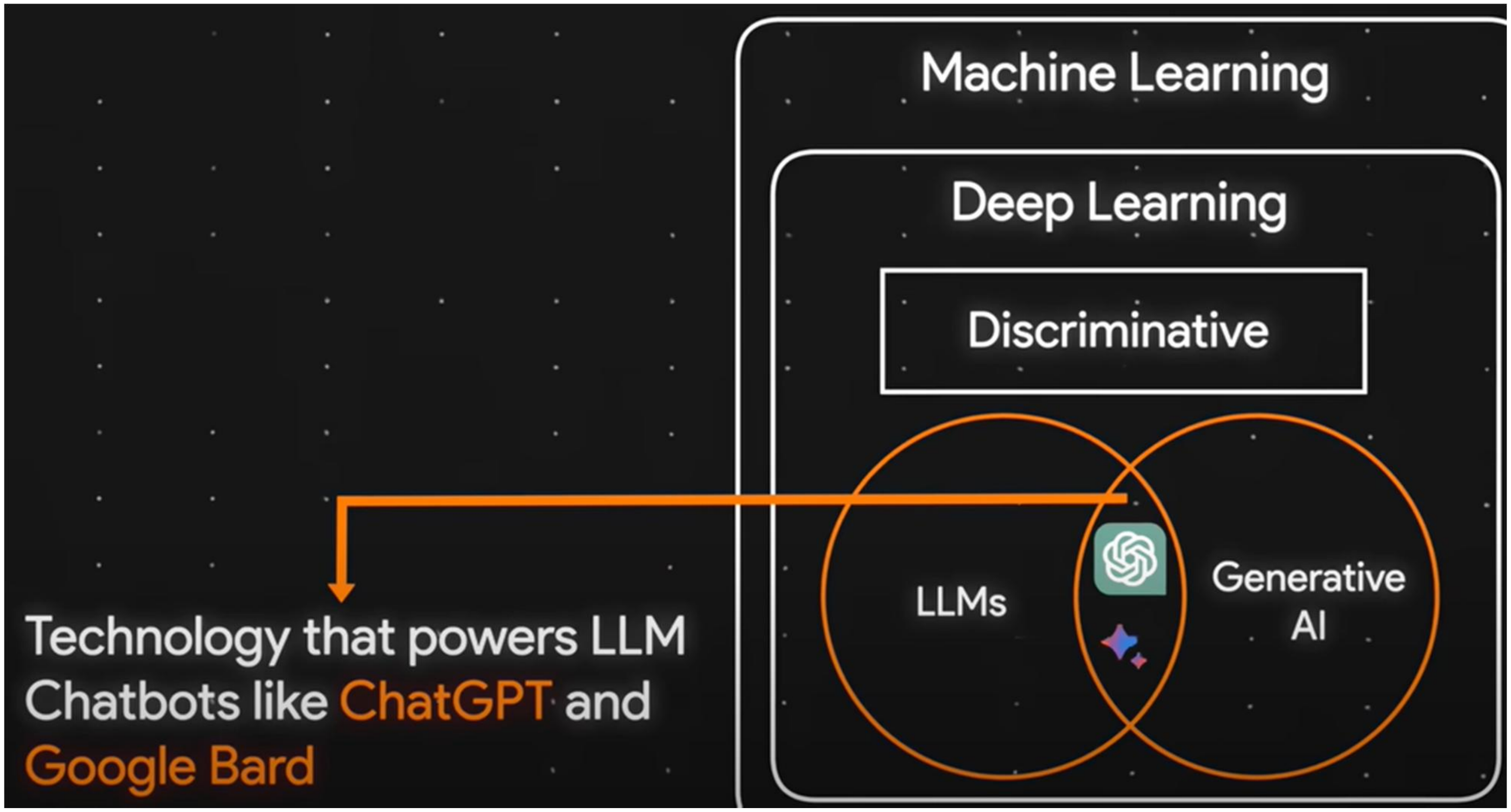
AY 2025-2026 (Sem V)

-Dr. S. Durga, M.E., PhD

Unit 1 – Part 1

- Introduction to AI and systems - Problem formulation, problem definition, Control Strategies, Search Strategies - Depth first, Breadth first, problem characteristics, system characteristics, problem solving methods - problem graphs, matching, indexing, heuristic functions, A* search algorithm, Hill climbing, Constraint satisfaction - related algorithms, handling uncertainty in terms of probability, measure of performance.

What is Artificial Intelligence?


Machine Learning
Deep Learning
Discriminative
LLMs
Generative AI
Technology that powers LLM Chatbots like ChatGPT and Google Bard

What is Artificial Intelligence?

- It is a branch of Computer Science that pursues creating the computers or machines as intelligent as human beings.
- Definition:

According to the father of Artificial Intelligence, John McCarthy, it is “The science and engineering of making intelligent machines, especially intelligent computer programs”.

Goals and Application area

- Science based goals of AI pertain to developing concepts, mechanisms and understanding biological intelligent behaviour. The emphasis is on understanding intelligent behaviour.
- Engineering based goals of AI relate to developing concepts, theory and practice of building intelligent machines. The emphasis is on system building.
- AI Techniques depict how we represent, manipulate and reason with knowledge in order to solve problems. Knowledge is a collection of 'facts'.
- To manipulate these facts by a program, a suitable representation is required. A good representation facilitates problem solving.
- Applications of AI refers to problem solving, search and control strategies, speech recognition, natural language understanding, computer vision, expert systems, etc.

The following questions are to be considered before we can step forward:

1. What are the underlying assumptions about intelligence?
2. What kinds of techniques will be useful for solving AI problems?
3. At what level human intelligence can be modelled?
4. When will it be realized when an intelligent program has been built?

Branches of AI:

Search — Artificial Intelligence programs often examine large numbers of possibilities – for example, moves in a chess game and inferences by a theorem proving program. Discoveries are frequently made about how to do this more efficiently in various domains.

- Pattern Recognition — When a program makes observations of some kind, it is often planned to compare what it sees with a pattern. For example, a vision program may try to match a pattern of eyes and a nose in a scene in order to find a face. More complex patterns are like a natural language text, a chess position or in the history of some event. These more complex patterns require quite different methods than do the simple patterns that have been studied the most.

AI Research and Application Areas

- Game Playing
- Automated Reasoning and Theorem Proving
- Expert Systems
- Natural Language Understanding and Semantic Modelling
- Modelling Human Performance
- Planning and Robotics
- Languages and Environments for AI
- Machine Learning
- Alternative Representations: Neural Nets and Genetic Algorithms
- AI and Philosophy

Important Features of Artificial Intelligence

1. The use of computers to do reasoning, pattern recognition, learning, or some other form of inference.
2. A focus on problems that do not respond to algorithmic solutions. This underlies the reliance on heuristic search as an AI problem-solving technique.
3. A concern with problem-solving using inexact, missing, or poorly defined information and the use of representational formalisms that enable the programmer to compensate for these problems.
4. Reasoning about the significant qualitative features of a situation.

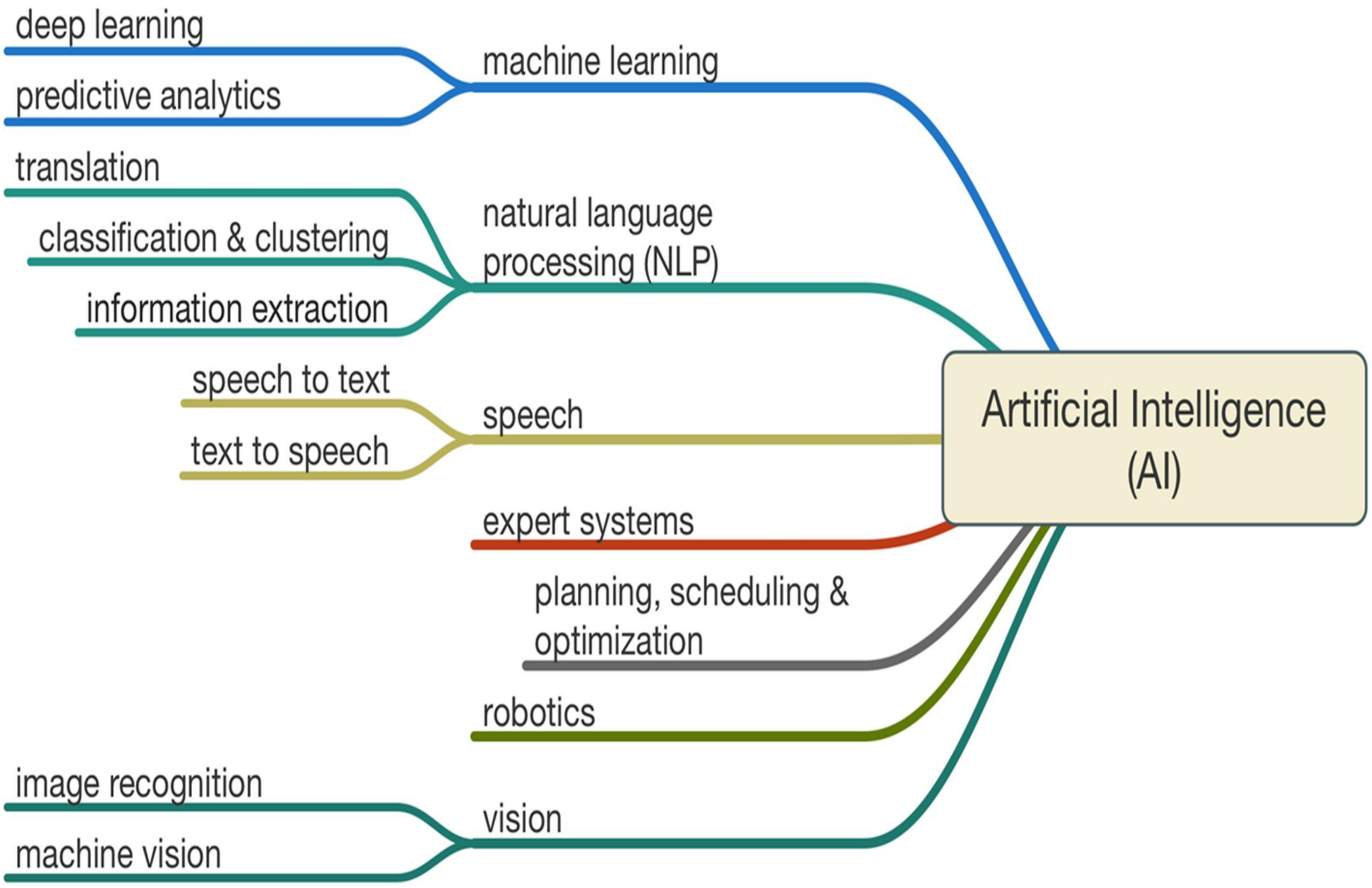
Important Features of Artificial Intelligence (Cont.)

5. An attempt to deal with issues of semantic meaning as well as syntactic form.
6. Answers that are neither exact nor optimal, but are in some sense “sufficient”. This is a result of the essential reliance on heuristic problem-solving methods in situations where optimal or exact results are either too expensive or not possible.
7. The use of large amounts of domain-specific knowledge in solving problems. This is the basis of expert systems.
8. The use of meta-level knowledge to effect more sophisticated control of problem-solving strategies. Although this is a very difficult problem, addressed in relatively few current systems, it is emerging as an essential are of research.

Problems , Types

- AI is being applied to solve a wide range of problems:
  - Automation (Autonomous vehicle, Robotics)
  - Data analysis, Prediction and forecasting
  - Communication, Speech Recognition and Information extraction (Natural Language Processing for Sentimental analysis, chatbots etc)

Problems , Types



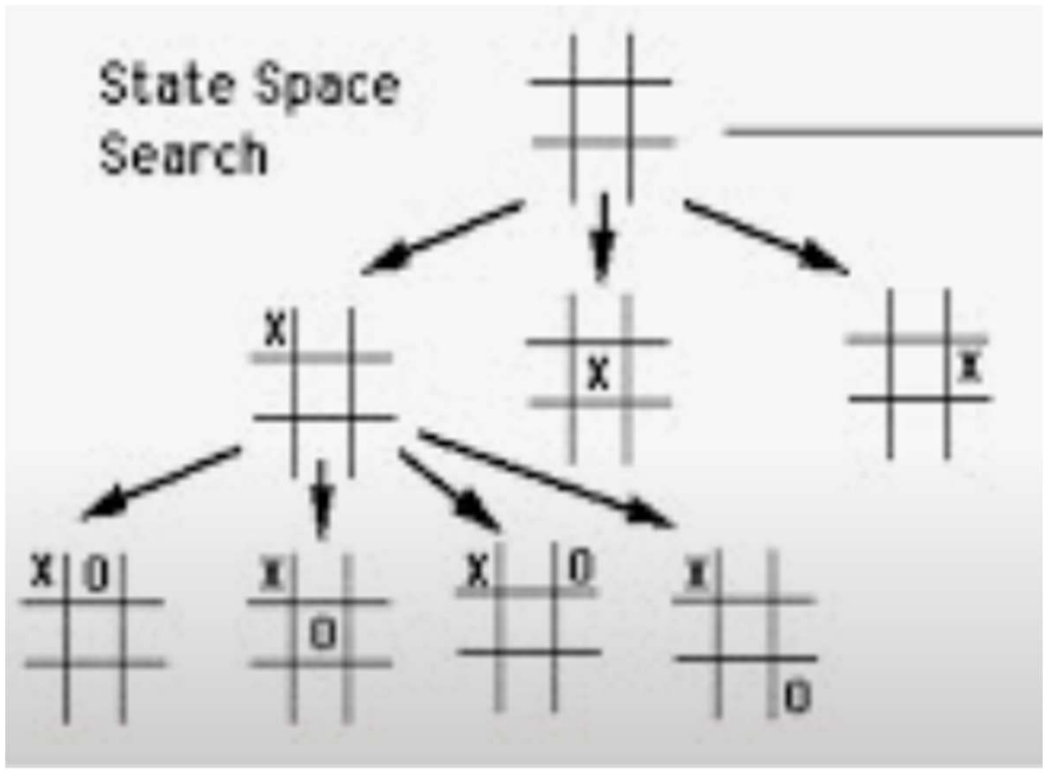
Problem Solving

- Problem solving agents

- agents are entities that perceive their environment and take actions to achieve specific goals.
- Navigate through the problem space and finds solution
- a problem space defines the entire range of possible states, actions, and transitions relevant to solving a specific problem

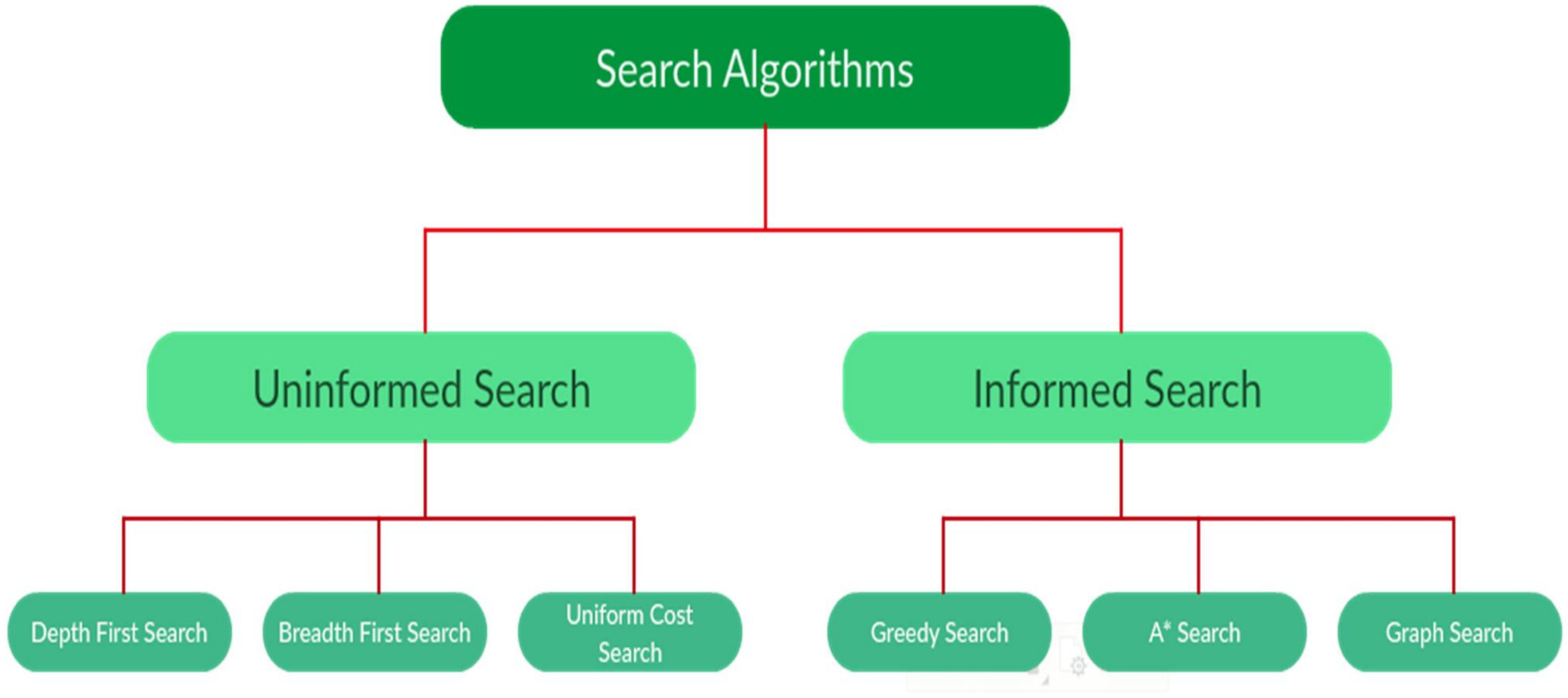
- States: A state represents a specific configuration or situation within the problem. For example, in a game of chess, a state would be the arrangement of pieces on the board at a particular point in time.
- Initial state, current state , operator, Goal state

- Example: Consider the classic 8-puzzle problem (a sliding tile game). The problem space would include:
- **States:** All possible arrangements of the 8 tiles and the blank space.
- **Operators:** Moving the blank space up, down, left, or right.
- **State Space:** All the different board configurations reachable from the initial configuration.
- **Goal:** Reaching a specific arrangement of the tiles.



Initial state

Search Strategies in AI



Search Strategies in AI

Un Informed Search (blind)

- Uninformed strategies explore systematically without additional knowledge
- **Breadth-First Search (BFS):** Explores all neighboring nodes at the current depth before moving to the next level. Useful for finding the shortest path.
- **Depth-First Search (DFS):**Explores as deeply as possible along each branch before backtracking. Good for finding solutions when the goal is deep within the search space.
- **Uniform Cost Search (UCS):**Similar to BFS but considers the cost of paths, finding the least-cost path.

Informed Search (Heuristic)

- informed strategies use heuristics to guide the search more efficiently.
- **Greedy Best-First Search:** Uses a heuristic function to estimate the cost to reach the goal, selecting the most promising path.
- **A Search:*** Combines the cost of reaching a node with the heuristic estimate, often used for pathfinding.
- **Genetic Algorithms:** Inspired by natural selection, these algorithms evolve solutions over generations to find optimal solutions to complex problems.

Other

- **Bidirectional Search:**Starts searching from both the start and goal states simultaneously, meeting in the middle for faster results.
- **Minmax:** A search algorithm used in game playing, particularly for two-player games, to find optimal moves.

uninformed search algorithms

- Depth First Search
- Breadth First Search
- Uniform Cost Search

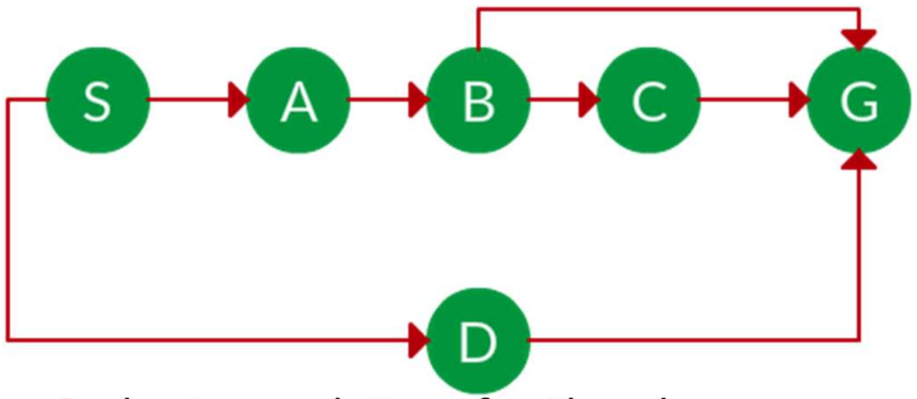
Each of these algorithms will have:

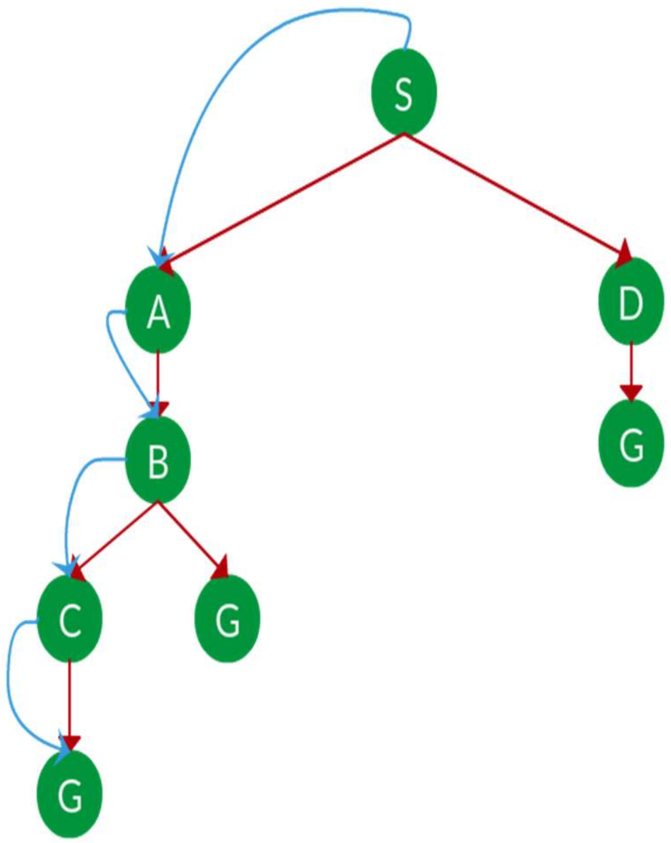
- A problem **graph,** containing the start node S and the goal node G.
- A **strategy,** describing the manner in which the graph will be traversed to get to G.
- a data structure used to store all the possible states (nodes) that you can go from the current states.
- A **tree,** that results while traversing to the goal node.
- A solution **plan,** which the sequence of nodes from S to G.

Depth First Search:

- Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures.
- The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.
- It uses last in- first-out strategy and hence it is implemented using a stack.

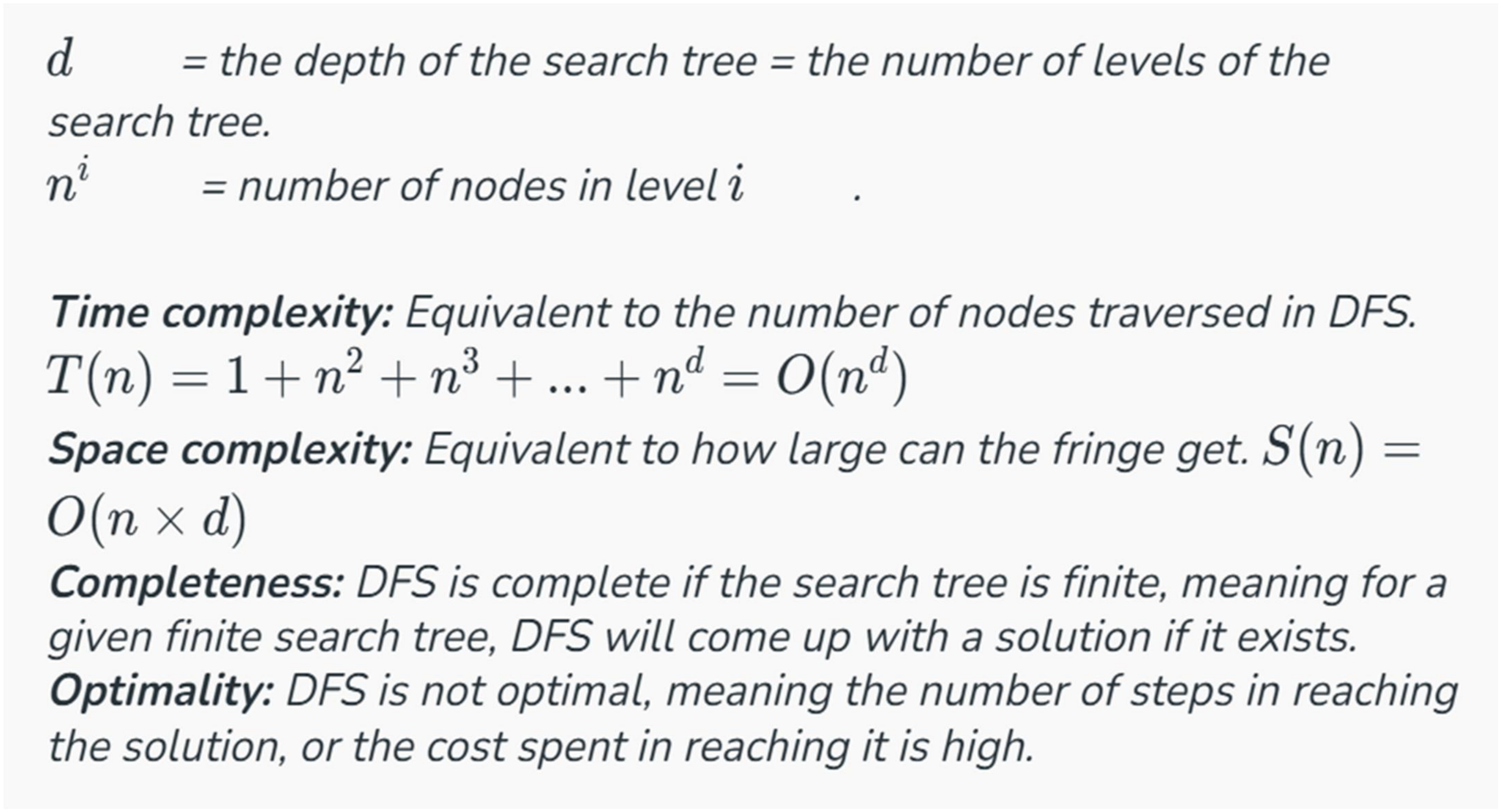
Eg. *Which solution would DFS find to move from node S to node G if run on the graph below?*





- The equivalent search tree for the above graph is as follows. As DFS traverses the tree "deepest node first", it would always pick the deeper branch until it reaches the solution (or it runs out of nodes, and goes to the next branch). The traversal is shown in blue arrows.
- Path is S -> A -> B -> C -> G

DFS: Analysis



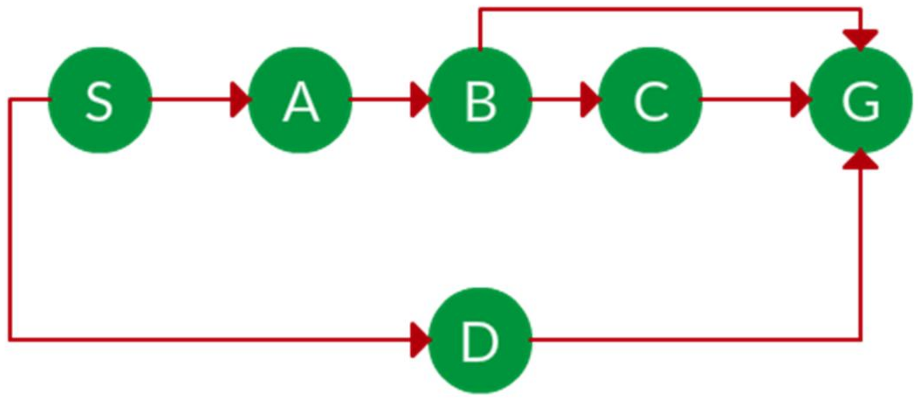
d *= the depth of the search tree = the number of levels of the search tree.*
n^i *= number of nodes in level* i .

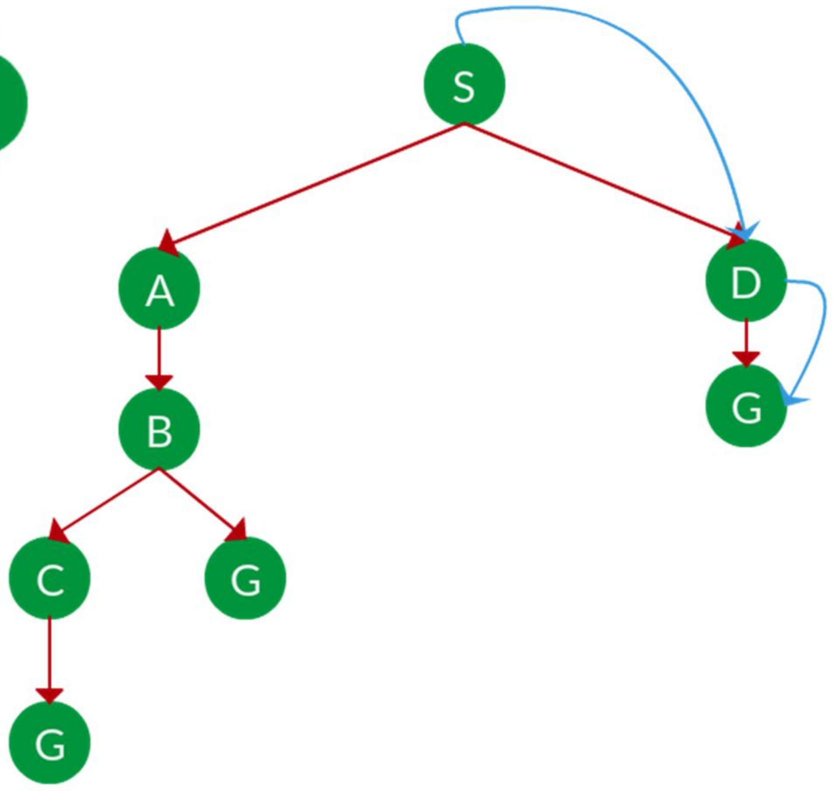
Time complexity: *Equivalent to the number of nodes traversed in DFS.*
$T(n) = 1 + n^2 + n^3 + ... + n^d = O(n^d)$
Space complexity: *Equivalent to how large can the fringe get.* $S(n) = O(n \times d)$
Completeness: *DFS is complete if the search tree is finite, meaning for a given finite search tree, DFS will come up with a solution if it exists.*
Optimality: *DFS is not optimal, meaning the number of steps in reaching the solution, or the cost spent in reaching it is high.*

Breadth First Search

- Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures.
- It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level. It is implemented using a queue.

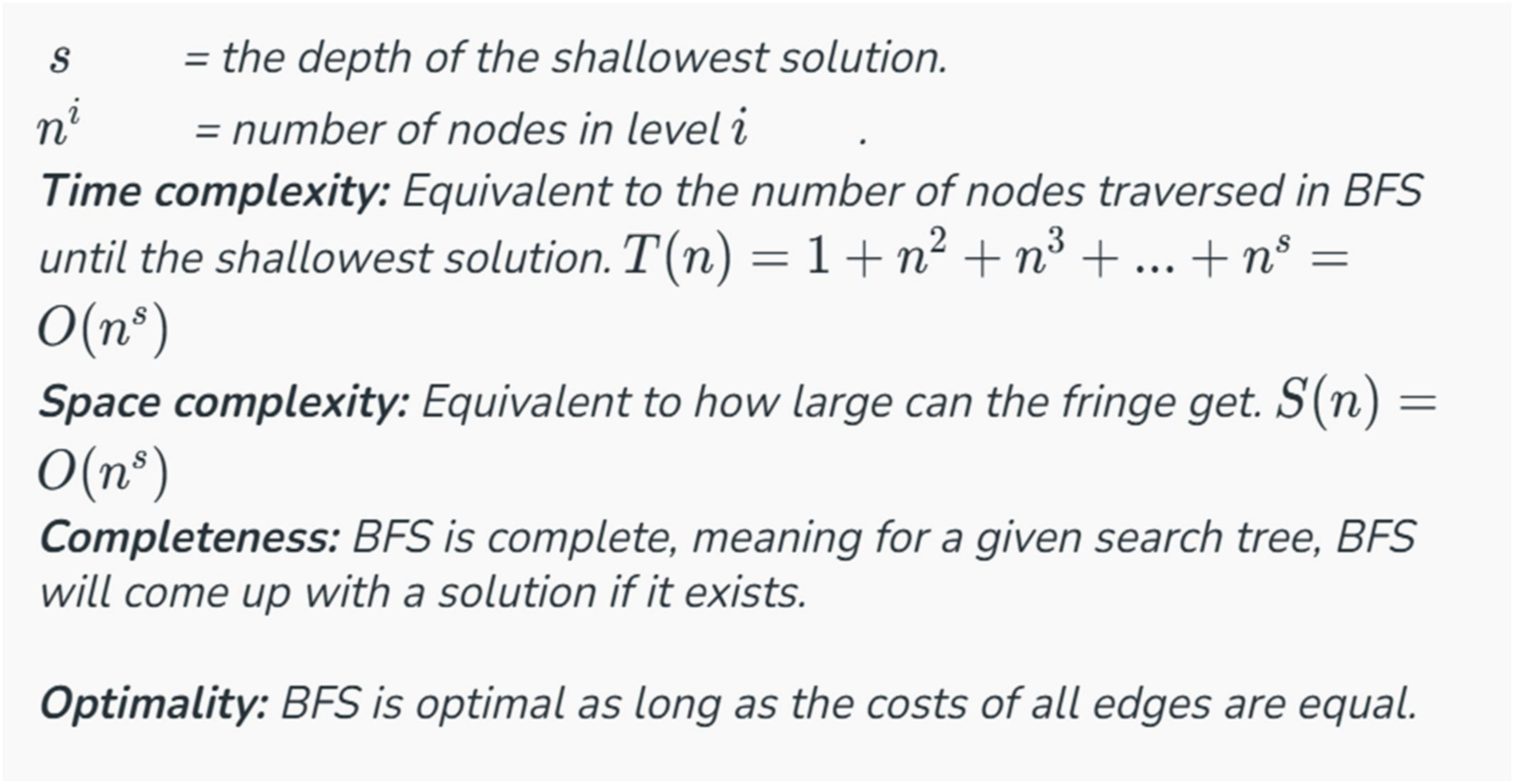
Which solution would BFS find to move from node S to node G if run on the graph below?





Path: S -> D -> G

BFS : Analysis



s = the depth of the shallowest solution.

n^i = number of nodes in level i .

Time complexity: *Equivalent to the number of nodes traversed in BFS until the shallowest solution.* $T(n) = 1 + n^2 + n^3 + ... + n^s = O(n^s)$

Space complexity: *Equivalent to how large can the fringe get.* $S(n) = O(n^s)$

Completeness: *BFS is complete, meaning for a given search tree, BFS will come up with a solution if it exists.*

Optimality: *BFS is optimal as long as the costs of all edges are equal.*