AI SEARCH ALGORITHMS

Agenda

- *Introduction*
- *Meaning of Search in AI*
- *State space*
- *Terminology*
- *Search Algortihms*
- *Questions*

INTRODUCTION

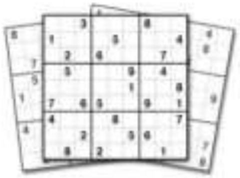
Introduction

- Many analytical problems can be solved by searching through a space of possible states.
- Starting from an initial state, try to reach a goal state.
- Solution - sequence of actions leading from initial to goal state.
- ***Example:*** n-queens
  - Initial state: an empty n x n chessboard
  - Actions (also called operators): place or remove a queen
  - Goal state: n queens placed, with no two queens on the same row, column, or diagonal
- ***Issues***
  - Large number of states and many choices to make in each state
  - Search must be performed in a systematic manner

Solving Problems by Searching

- A wide range of problems can be formulated as searches.

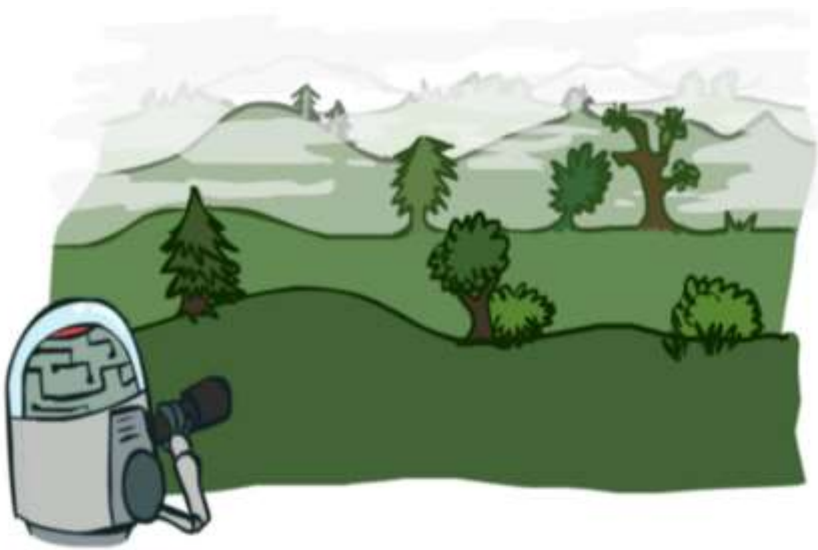
- The process of searching for a sequence of actions – takes from an initial state to a goal state.

Search

- The terms ‘search’, ‘search space’, ‘search problem’, ‘search algorithm’ are widely used in computer science and especially in AI.
- In this context the word ‘search’ has a somewhat technical sense - though it has a strong analogy with the meaning of the natural language word ‘search’.
- Technical meaning of ‘search’ as an AI problem solving method – to be understood.
- ***Examples of Search Problems***
  - Puzzles, Route finding, Motion control
  - Activity planning, Games AI
  - Scheduling
  - Mathematical theorem proving
  - Design of computer chips, drugs, buildings, etc.

Search in AI

- A lot of AI is about SEARCH.
- Search can be applied to many problems.
- **“State space"** - is the set of states of the problem that can be got to by applying operators to a state of the problem to get a new state.
- The state space can be HUGE! (Combinatorial explosion)



State Space Search - Terminologies

- **Given**
- an **initial state**
- a set of **operators** (actions the agent can take)
- a set of **goal states**
- a path **cost function**
- **Find**
- a sequence of operators that leads from the initial state to a goal state
- **Search space -** is the implicit tree (or graph) defined by the initial state and the operators
- **Search tree** (or graph) - is the explicit tree generated during the search by the control strategy
- **NOTE:** Each node of this graph is a complete description of the state of the problem.

Maze – Example problem for search

- To solve the maze, find a path from an initial location to a goal location.
- A path is determined by a sequence of choices.
- At each choice-point, choose between two or more branches.
- To find a path connecting physical locations - replace physical space with a state space.
- States in the space can correspond to any kind of configuration.
- For example, the possible settings of a device, positions in a game or (more abstract still) a set of assignments to variables.
- Paths in state space correspond to possible sequences of transitions between states.

Formulating Problems as Search Problems

- A huge variety of problems can be cast into the form of search problems.
- In order to apply a search approach, analyse the problem as follows:
- Conceptualise possible solutions and intermediate stages towards solutions as states (and identify an initial state).
- Identify transitions that allow a state to be transformed into other successor states.
- Devise an algorithm that will systematically search through possible paths from the initial state in order to find a path that ends in a goal.

Types of Search Problems

Type 1) Find a solution

- Search for a state or configuration satisfying known conditions.
- **Ex:** In scheduling, search for a timetable satisfying given constraints on when events can occur — such as Lecture A cannot be at the same time as Lecture B.

Type 2) Find a way to reach a solution

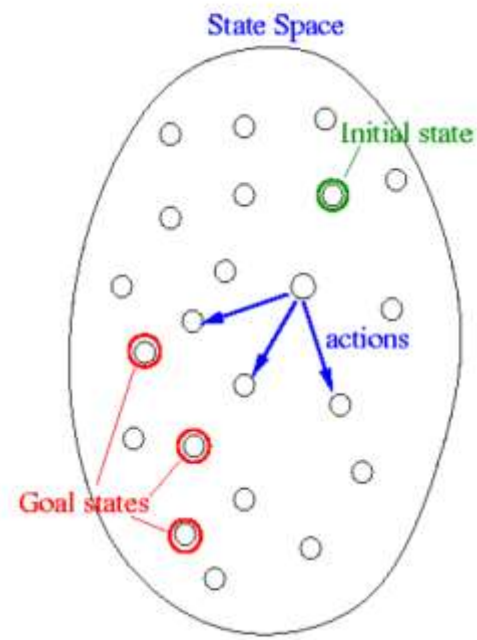
- Search for a sequence of steps (actions) that lead from an initial state to a desired goal state (either a specific state or any state satisfying given conditions).
- **Ex:** To find a sequence of moving, picking up and placing actions that a robot can execute to lay a table.

Type 3) Find the best way to reach a solution

- Search for an optimal sequence of steps (actions) that lead from an initial state to a goal state.
- An optimal sequence is one that has the lowest ‘cost’ (i.e. takes the least time or resources).

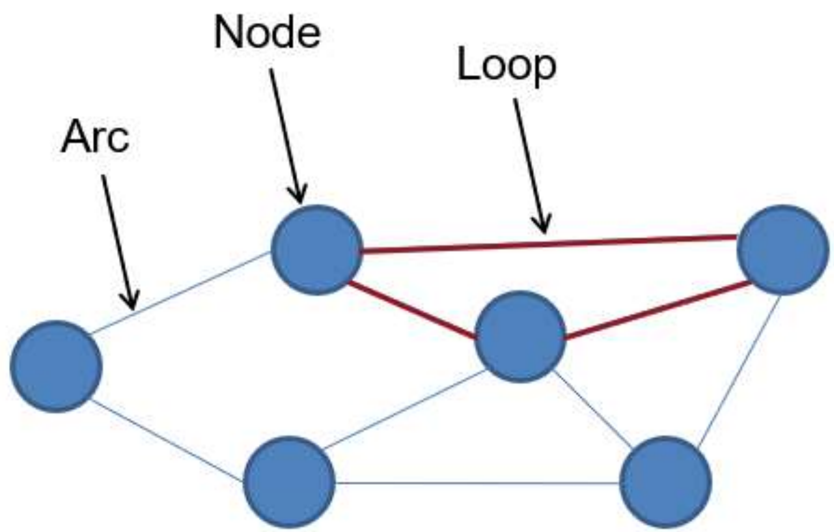
State Space Representation

- To formally describe, a **search problem** consists of the following:
  - S: the full **set of states**
  - s0 : the **initial state**, s0 ∈ S
  - A: S→S is a set of **operators**
  - G is the set of **final states**. Note that G ⊆ S
- **Solution plan** - a sequence of actions - a path from the initial state to a goal state
- A ***plan*** P is a sequence of actions
- P = {a**0**, a**1**, … , a**N**} which leads to traversing a number of states {s**0**, s**1**, … , s**N+1**}
- A **path** – is a sequence of states
- The **cost of a path** is a positive number - computed by taking the sum of the costs of each action



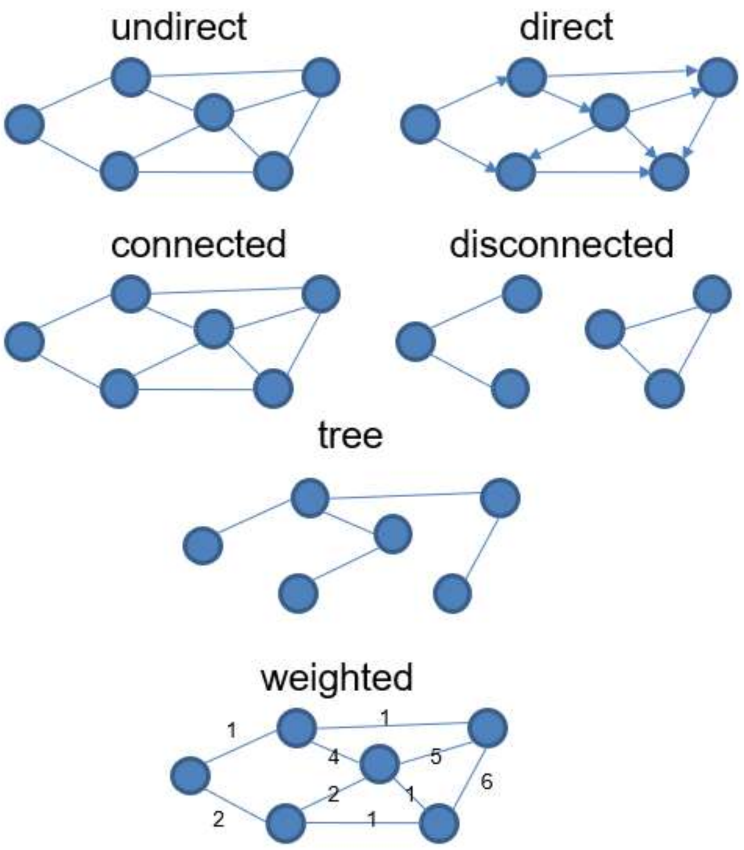
State Space Representation

- Representing the search space - is the first step to enable the problem resolution
- Search space - is mostly represented through graphs
- A **graph** is a finite set of *nodes* that are connected by *arcs*
- A ***loop*** may exist in a graph, where an arc lead back to the original node
- Search space is constructed during search



State Space Representation

- A graph is ***undirected*** if arcs do not imply a direction, *direct* otherwise
- A graph is ***connected*** if every pair of nodes is connected by a path
- A connected graph with no loop is called ***tree***
- A ***weighted graph***, is a graph for which a value is associated to each arc



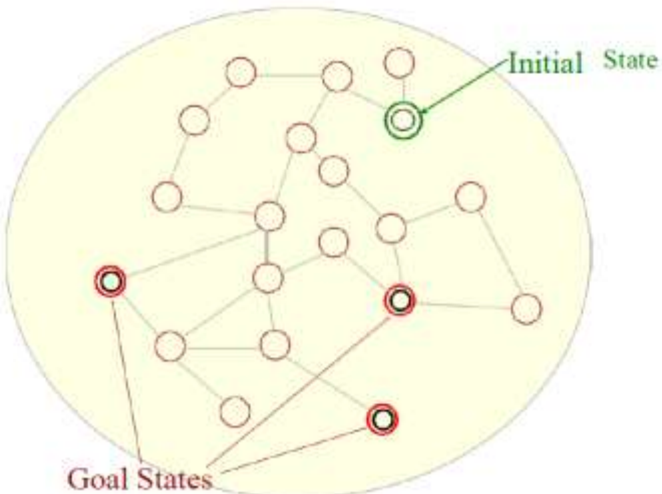
Searching Process

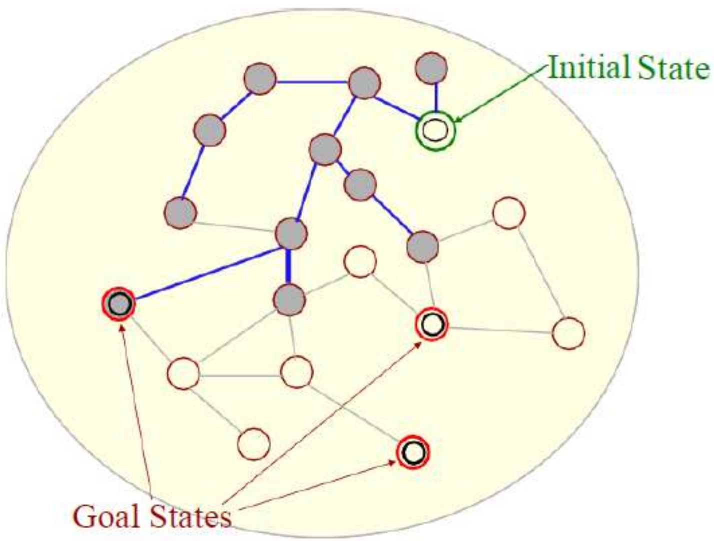
- Generic searching process – simply described as follows:

  Do until a solution is found or the state space is exhausted.

  1. Check the current state
  2. Execute allowable actions to find the successor states.
  3. Pick one of the new states.
  4. Check if the new state is a solution state

     If it is not, the new state becomes the current state and the process is repeated





Search Trees

- State space associated with a problem - can be an arbitrary graph
- In devising a search algorithm, treat the search space as if it were a tree

Search tree

- A “what if” tree of plans and their outcomes
- Start state - is the root node; Children - correspond to successors
- Nodes show states, but correspond to PLANS that achieve those states
- For most problems, the whole tree can never be actually built

State space graph

- A mathematical representation of a search problem
- Nodes - are (abstracted) world configurations
- Arcs - represent successors (action results)
- The goal test - is a set of goal nodes (maybe only one)
- In a state space graph, each state occurs only once!
- The full graph can rarely be built in memory (it’s too big), but it’s a useful idea

State Space Graphs Vs. Search Trees

The *Structures* of *state space* are *trees* and *graphs*.

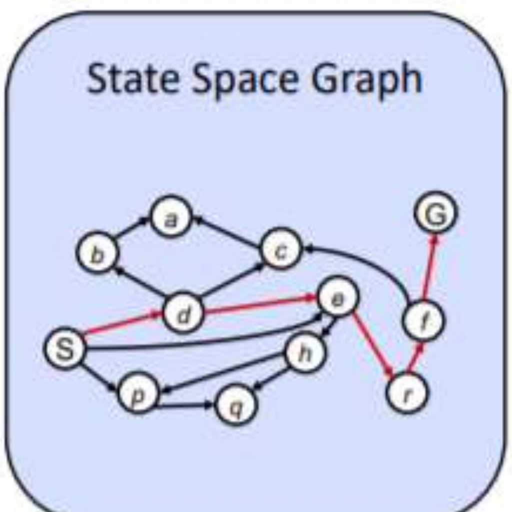
- Tree is a hierarchical structure in a graphical form; and
- Graph is a non-hierarchical structure.

◇ **Tree** has only one path to a given node;

i.e., a *tree* has one and only one path from any point to any other point.

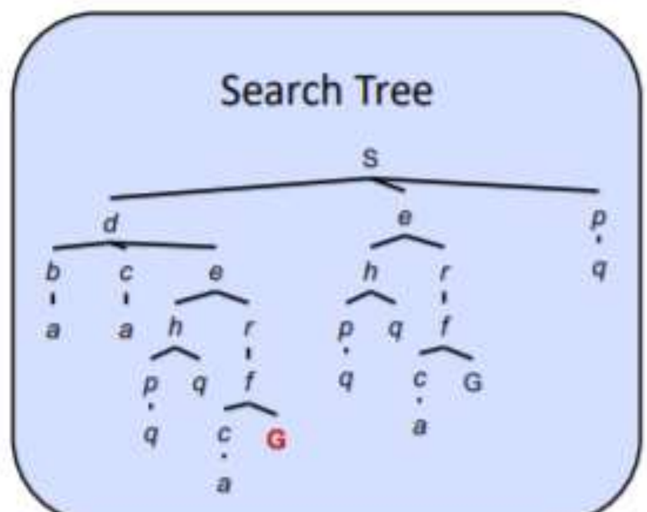
◇ **Graph** consists of a set of nodes (vertices) and a set of edges (arcs). Arcs establish relationships (connections) between the nodes;

i.e., a graph has several paths to a given node.



Each NODE in in the search tree is an entire PATH in the state space graph.

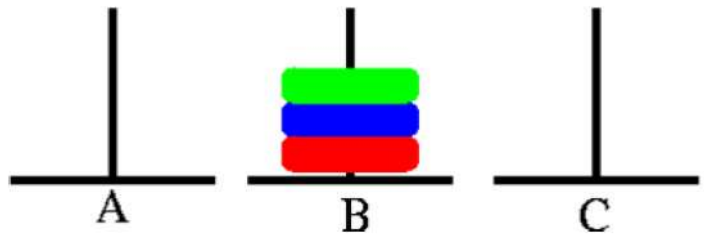
We construct both on demand – and we construct as little as possible.



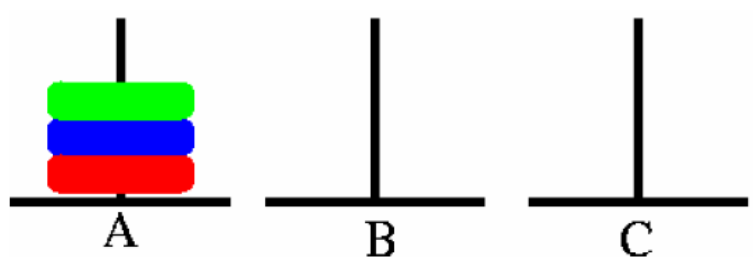
State Space Search - Example

Pegs and Disks problem

- There are 3 pegs and 3 disks
- **Operators**: one may move the topmost disk on any needle to the topmost position to any other needle
- The **initial state** is as shown:

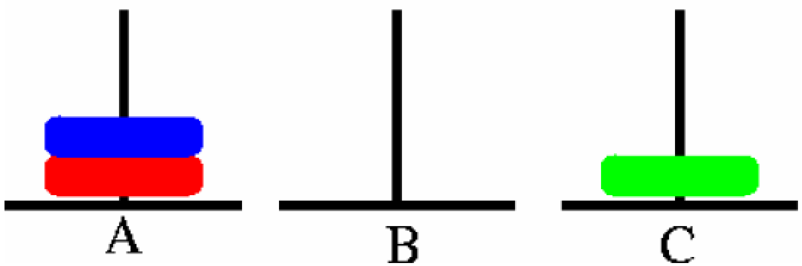


- In the **goal state** all the pegs are in the needle B as shown in the figure below:

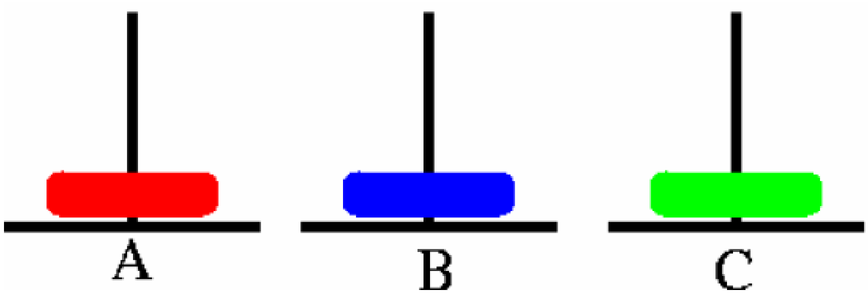


State Space Search - Example

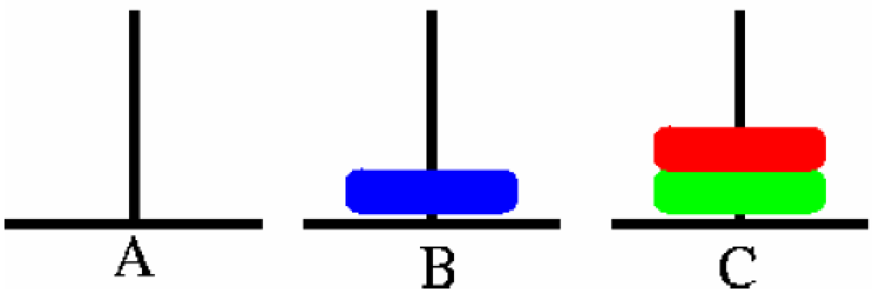
- Step-1: Move A to C



- Step-2: Move A to B

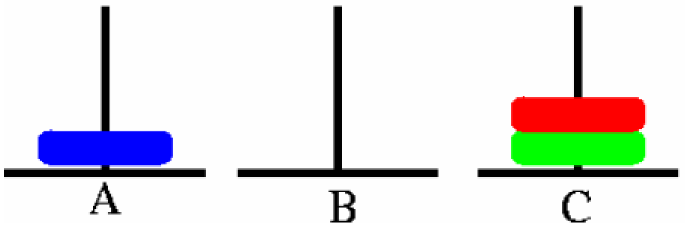


- Step-3: Move A to C

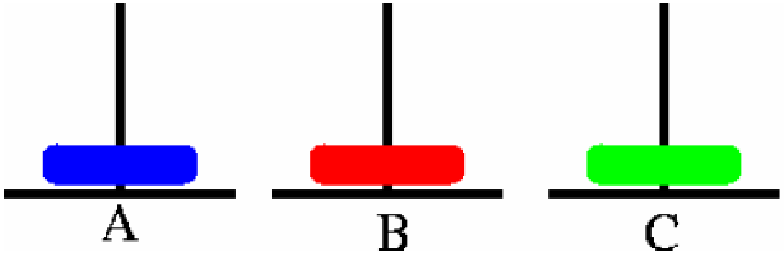


State Space Search - Example

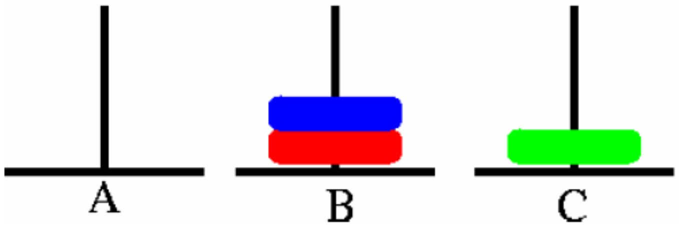
- Step-4: Move B to A



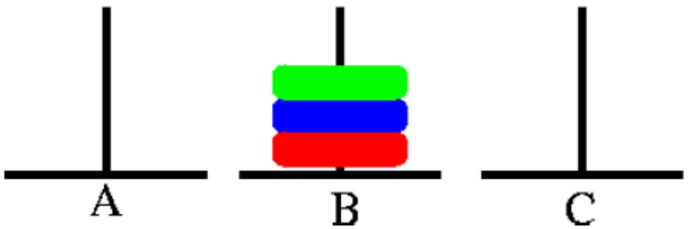
- Step-5: Move C to B



- Step-6: Move A to B



- Step-7: Move C to B



Search Algorithms

- A **search strategy** is defined by picking the **order of node expansion**
- **Strategies** are evaluated along the following **dimensions**:
  - **completeness:** does it always find a solution if one exists?
  - **time complexity:** number of nodes generated
  - **space complexity:** maximum number of nodes in memory
  - **optimality:** does it always find a least-cost solution?
- Time and space complexity are measured in terms of
  - b: **maximum branching factor** of the search tree
  - d: **depth** of the least-cost solution
  - m: **maximum depth** of the state space (may be ∞)

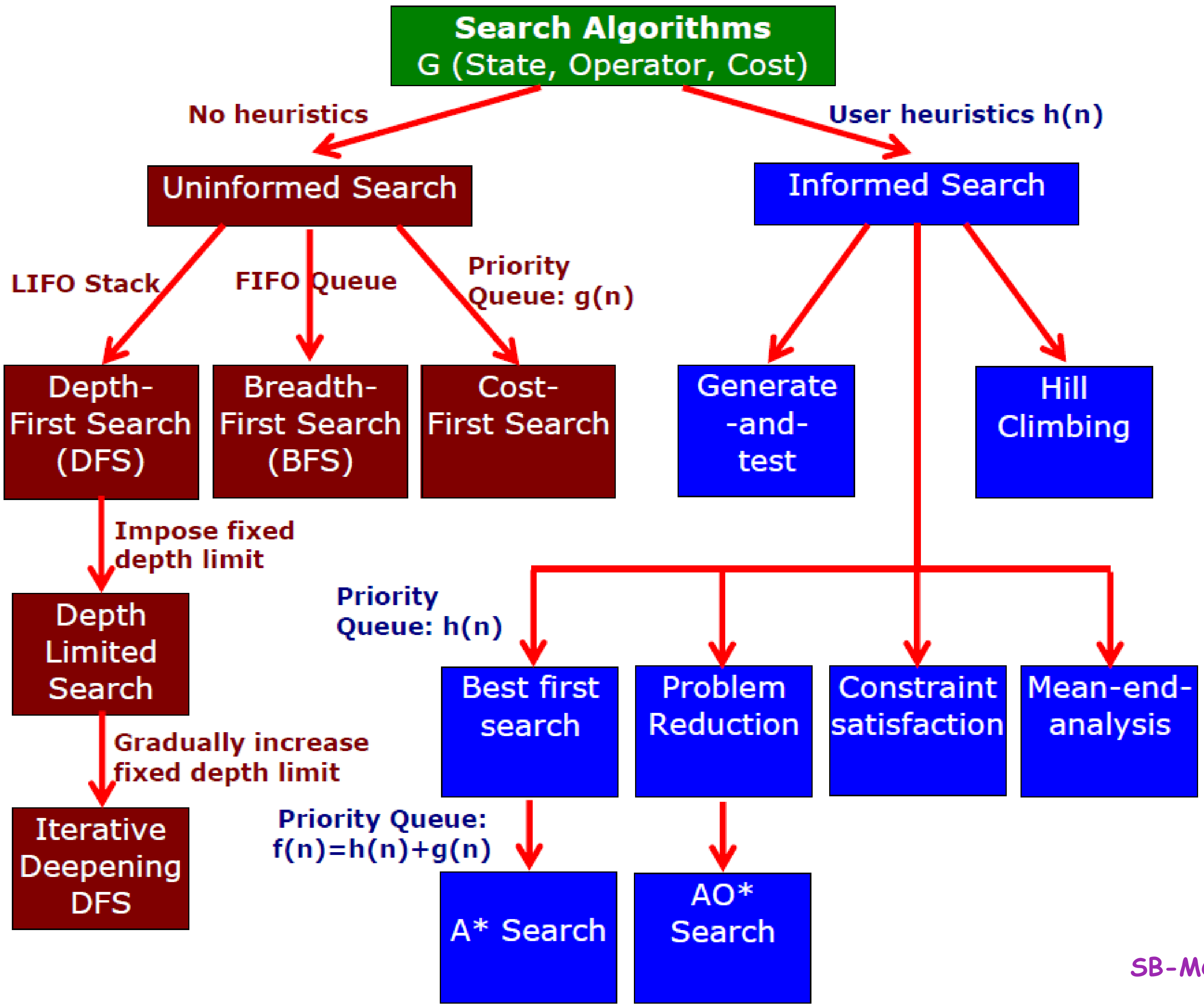
Search Algorithms

A representation of most search algorithms is illustrated below. It begins with two types of search - Uninformed and Informed.

Uninformed Search : Also called *blind*, *exhaustive or brute-force* search, uses no information about the problem to guide the search and therefore may not be very efficient.

Informed Search : Also called *heuristic* or *intelligent* search, uses information about the problem to guide the search, usually guesses the distance to a goal state and therefore efficient, but the search may not be always possible.

Search Algorithms



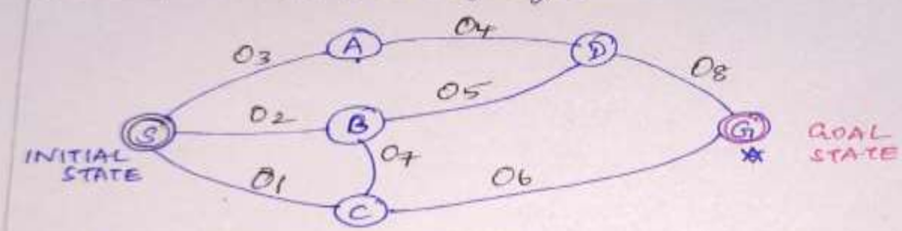
SB-MCA-AI

Trial and Error Search

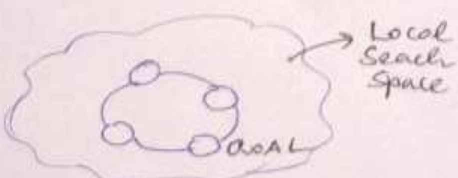
TRIAL AND ERROR SEARCH

procedure TRIAL AND ERROR SEARCH
    set INITIAL STATE as STATE ;
    while STATE $\neq$ GOAL STATE do
        select OPERATOR applicable to STATE
        and make this as the next STATE
        (ie) STATE = OPERATOR (STATE)
    end
end TRIAL AND ERROR SEARCH

Example: (Visualizing)



- From S, just a random move B – C – G (O2 – O7 – O6)
- Next time – may start from O1, next from O3, ...
- This random wandering – may sometimes arrive at a local neighbourhood ("a local search space")



Questions?

- Can there be more than one agent program that implements a given agent function?
- Identify the drawbacks of table-driven agent program.
- Develop PEAS description for the following task environment:
  - Movie recommendation agent
  - Robot soccer player
  - Shopping for used AI books on the Internet
- Select a suitable agent design for:
  - Robot soccer player
  - Shopping for used AI books on the Internet
- **Find 3 more examples of state space search problems, with descriptions of initial and final states and operators.**