MODULE 2 SEARCH IN ARTIFICIAL INTELLIGENCE

UNIT 1 INTRODUCTION TO STATE SPACE SEARCH

CONTENTS

1.0 INTRODUCTION

In computer science, a search algorithm, broadly speaking, is an algorithm for finding an item with specified properties among a collection of items. The items may be stored individually as records in a database; or may be elements of a search space defined by a mathematical formula or procedure, such as the roots of an equation with integer variables; or a combination of the two, such as the Hamiltonian circuits of a graph.

Specifically, Searching falls under Artificial Intelligence (AI). A major goal of AI is to give computers the ability to think, or in other words, mimic human behaviour. The problem is, unfortunately, computers don't function in the same way our minds do. They require a series of *well-reasoned out* steps before finding a solution. Your goal, then, is to take a complicated task and convert it into simpler steps that your computer can handle. That conversion from something complex to something simple is what this unit is primarily about. Learning how to use two search algorithms is just a welcome side-effect. This unit will explain the background for AI search and some of the AI search techniques.

2.0 OBJECTIVES

After the end of this unit, you should be able to:

Describe the state space representation
Describe some algorithms
Formulate, when given a problem description, the terms of a state space search problem
Analyse the properties of some algorithms
Analyse a given problem and identify the most suitable search strategy for the problem
Solve some simple problems.

3.0 MAIN CONTENT

3.1 State Space Search

Let us begin by introducing certain terms.

An initial state is the description of the starting configuration of the agent.

An action or an operator takes the agent from one state to another state which is called a successor state. A state can have a number of successor states.

A plan is a sequence of actions. The cost of a plan is referred to as the path cost. The path cost is a positive number, and a common path cost may be the sum of the costs of the steps in the path. The goal state is the partial description of the solution

3.1.1 Goal Directed Agent

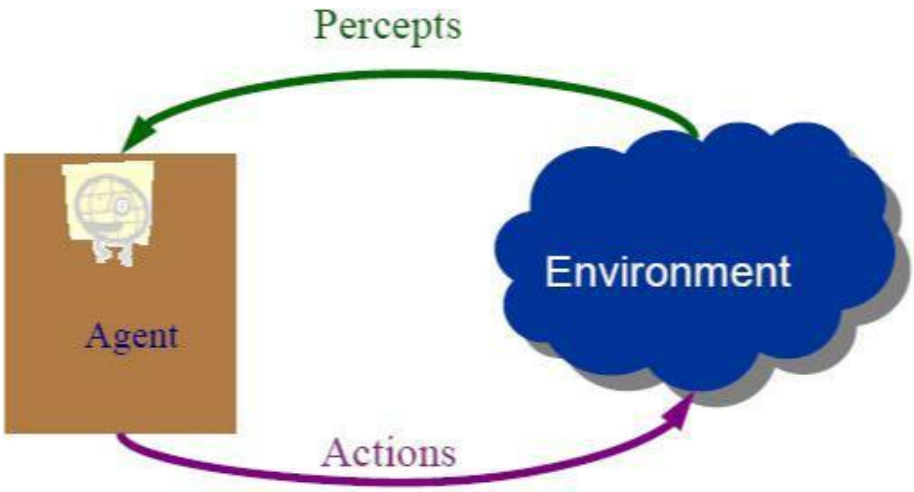


Figure 1: Goal Directed Agent

We have earlier discussed about an intelligent agent. In this unit we will study a type of intelligent agent which we will call a goal directed agent. A goal directed agent needs to achieve certain goals. Such an agent selects its actions based on the goal it has. Many problems can be represented as a set of states and a set of rules of how one state is transformed to another. Each state is an abstract representation of the agent's environment. It is an abstraction that denotes a configuration of the agent.

Let us look at a few examples of goal directed agents.

1. 15-puzzle: The goal of an agent working on a 15-puzzle problem may be to reach a configuration which satisfies the condition that the top row has the tiles 1, 2 and 3. The details of this problem will be described later.

2. The goal of an agent may be to navigate a maze and reach the HOME position.

The agent must choose a sequence of actions to achieve the desired goal.

3.1.2 State Space Search Notations

Now let us look at the concept of a search problem.
Problem formulation means choosing a relevant **set of states** to consider, and a feasible **set of operators** for moving from one state to another.

Search is the process of considering various possible sequences of operators applied to the initial state, and finding out a sequence which culminates in a goal state.

3.2 Problem Space

What is problem space?

A problem space is a set of states and a set of operators. The operators map from one state to another state. There will be one or more states that can be called initial states, one or more states which we need to reach what are known as goal states and there will be states in between initial states and goal states known as intermediate states. So what is the solution? The solution to the given problem is nothing but a sequence of operators that map an initial state to a goal state. This sequence forms a solution path. What is the best solution? Obviously the shortest path from the initial state to the goal state is the best one. Shortest path has only a few operations compared to all other possible solution paths. Solution path forms a tree structure where each node is a state. So searching is nothing but exploring the tree from the root node.

3.2.1 Search Problem

We are now ready to formally describe a search problem.

A search problem consists of the following:

S: the full set of states

s_0 : the initial state

$A: S \rightarrow S$ is a set of operators

G is the set of final states. Note that G ⊆S These are schematically depicted in Figure 2.

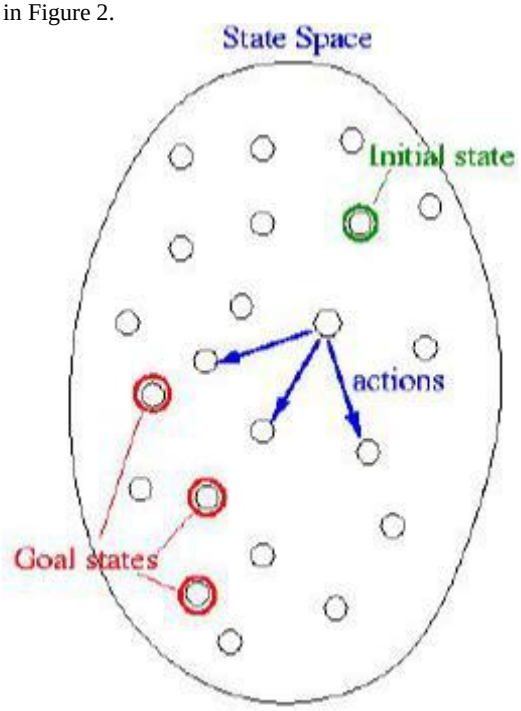


Figure 2

–

The search problem is to find a sequence of actions which transforms the agent from the initial state to a goal state g∈G. A search problem is represented by a 4-tuple {S, s , A, G}.
0

S: set of states

s ∈ S: initial state
0

A: S S operators/ actions that transform one state to another state

G: goal, a set of states. G ⊆ S

This sequence of actions is called a solution plan. It is a path from the initial state to a goal state. A *plan* P is a sequence of actions.

P = {a , a , a } which leads to traversing a number of states {s , s ,
0 1 N 0 1

Sn+ ∈G). A sequence of states is called a path. The cost of a path is a
1

positive number. In many cases the path cost is computed by taking the sum of the costs of each action.

Representation of search problems

A search problem is represented using a directed graph.

The states are represented as nodes.

The allowed actions are represented as arcs.

Searching process

The generic searching process can be very simply described in terms of the following steps:

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

3.3 Examples

3.3.1 Illustration of a search process

We will now illustrate the searching process with the help of an example. Consider the problem depicted in Figure 3.

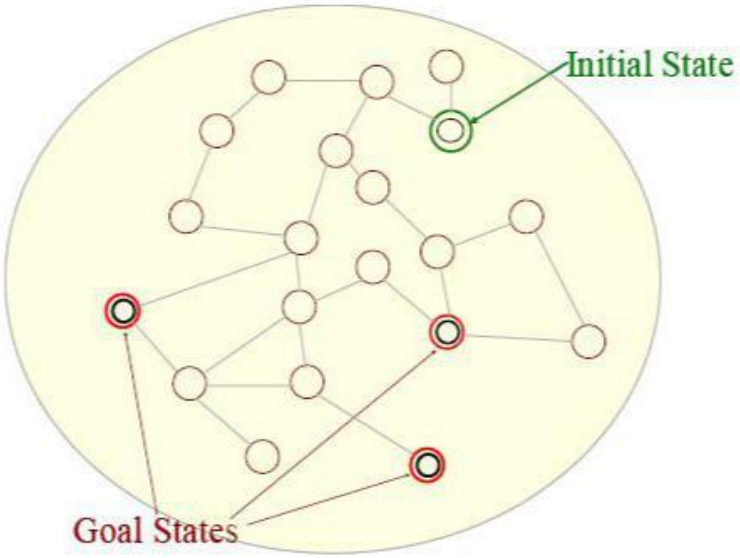


Figure 3

s is the initial state.
0
The successor states are the adjacent states in the graph.

There are three goal states.

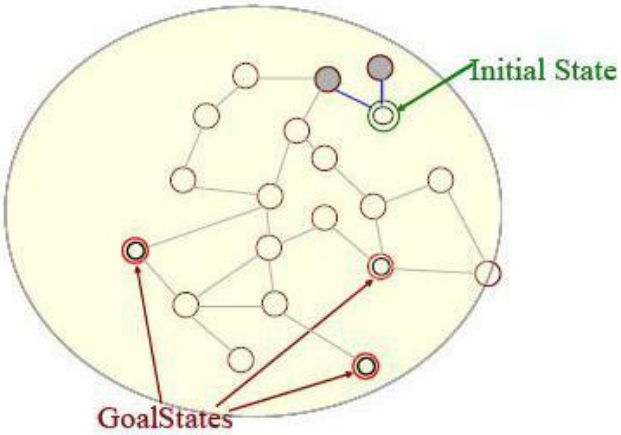


Figure 4

The two successor states of the initial state are generated.

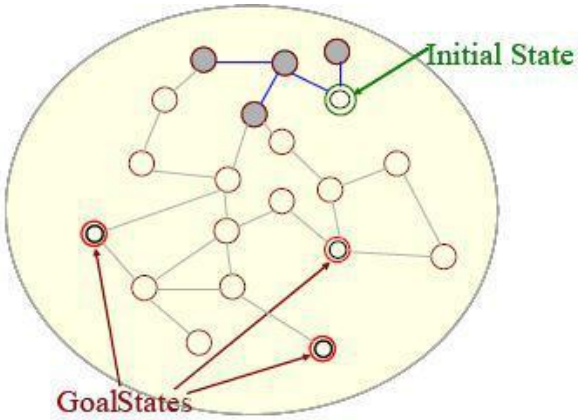


Figure 5

The successors of these states are picked and their successors are generated.

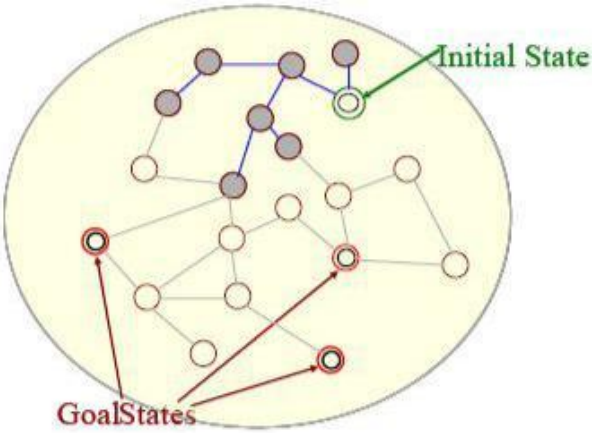


Figure 6

Successors of all these states are generated.

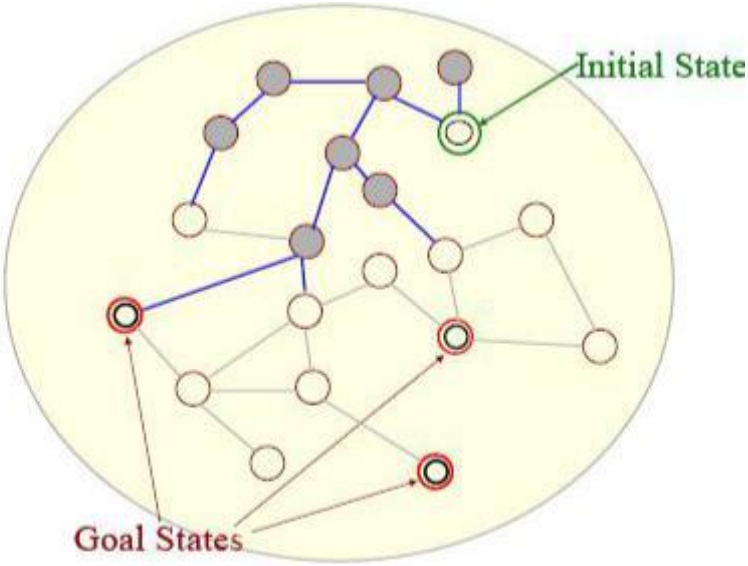


Figure 7

The successors are generated.

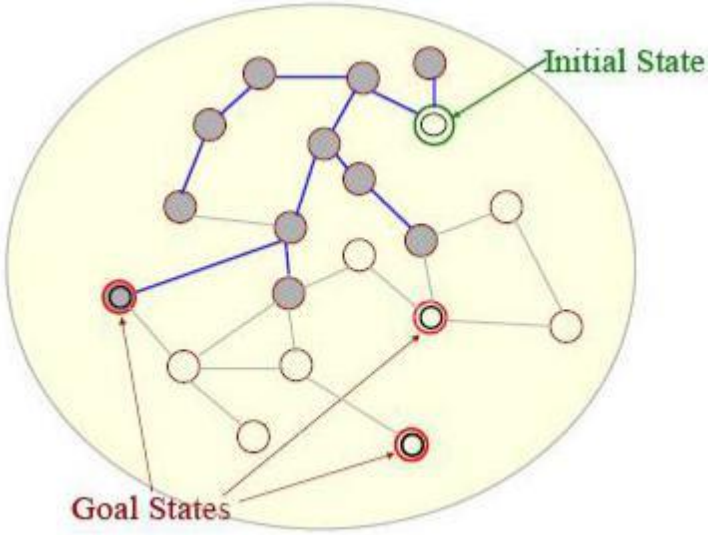


Figure 8

A goal state has been found.

The above example illustrates how we can start from a given state and follow the successors, and be able to find solution paths that lead to a goal state. The grey nodes define the search tree. Usually the search tree is extended one node at a time. The order in which the search tree is extended depends on the search strategy.

We will now illustrate state space search with one more example – the pegs and disks problem. We will illustrate a solution sequence which when applied to the initial state takes us to a goal state.

3.3.2 Example problem: Pegs and Disks problem

Consider the following problem. We have 3 pegs and 3 disks.

Operators: one may move the topmost disk on any needle to the topmost position to any other needle.

In the goal state all the pegs are in the needle B as shown in the figure below.

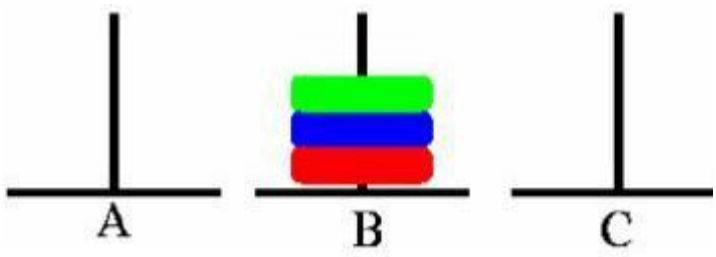


Figure 9

The initial state is illustrated below.

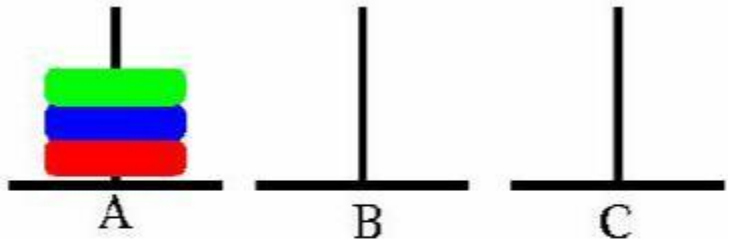


Figure 10

Now we will describe a sequence of actions that can be applied on the initial state.

Step 1: Move A → C

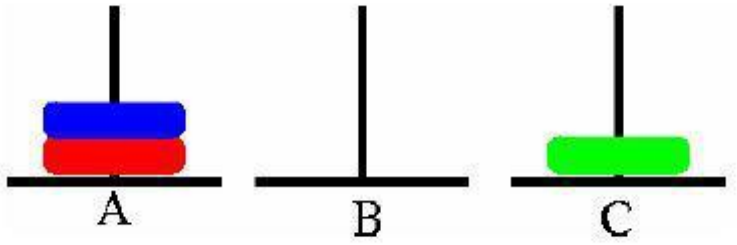


Figure 11

Step 2: Move A → B

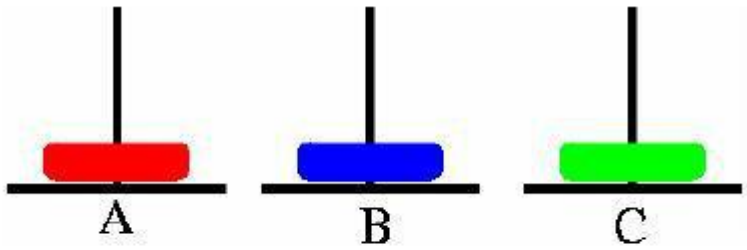


Figure 12

Step 3: Move A → C

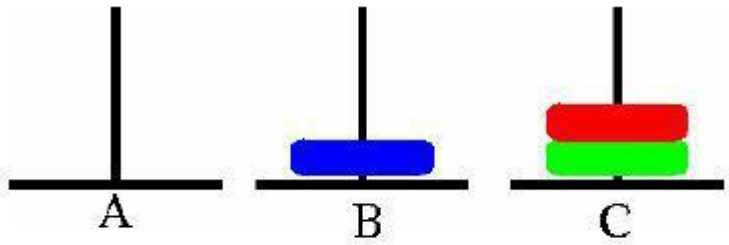


Figure 13

Step 4: Move B→ A

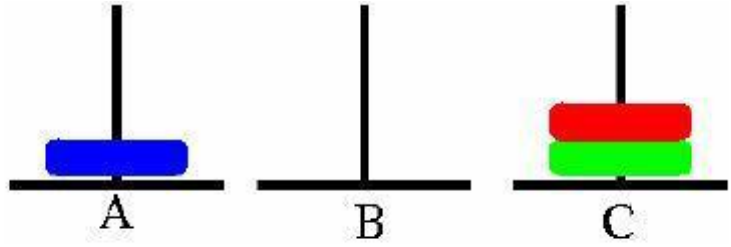


Figure 14

Step 5: Move C → B

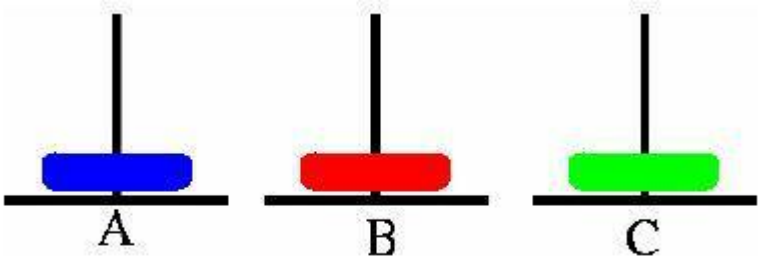


Figure 15

Step 6: Move A → B

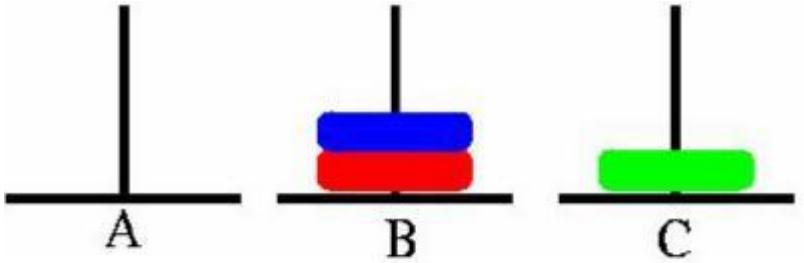


Figure 16

Step 7: Move C→ B

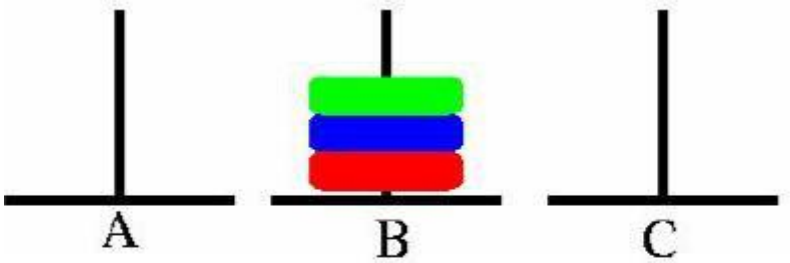


Figure 17

We will now look at another search problem – the 8-queens problem, which can be generalized to the N-queens problem.

3.3.3 Queens Problem

The problem is to place 8 queens on a chessboard so that no two queens are in the same row, column or diagonal.

The picture below on the left shows a solution of the 8-queens problem. The picture on the right is not a correct solution, because some of the queens are attacking each other.

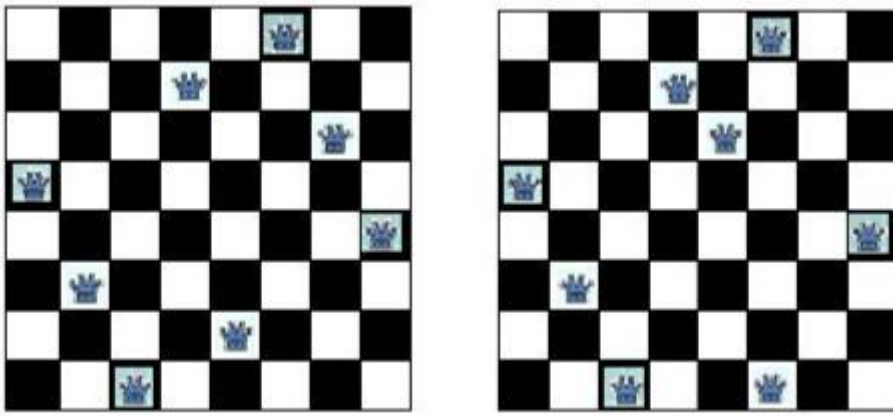


Figure 18: Queens Problem

How do we formulate this in terms of a state space search problem? The problem formulation involves deciding the representation of the states, selecting the initial state representation, the description of the operators, and the successor states. We will now show that we can formulate the search problem in several different ways for this problem.

N queens problem formulation 1

States: Any arrangement of 0 to 8 queens on the board
Initial state: 0 queens on the board
Successor function: Add a queen in any square
Goal test: 8 queens on the board, none are attacked

The initial state has 64 successors. Each of the states at the next level has 63 successors, and so on. We can restrict the search tree somewhat by considering only those successors where no queen is attacking each other. To do that, we have to check the new queen against all existing queens on the board. The solutions are found at a depth of 8.

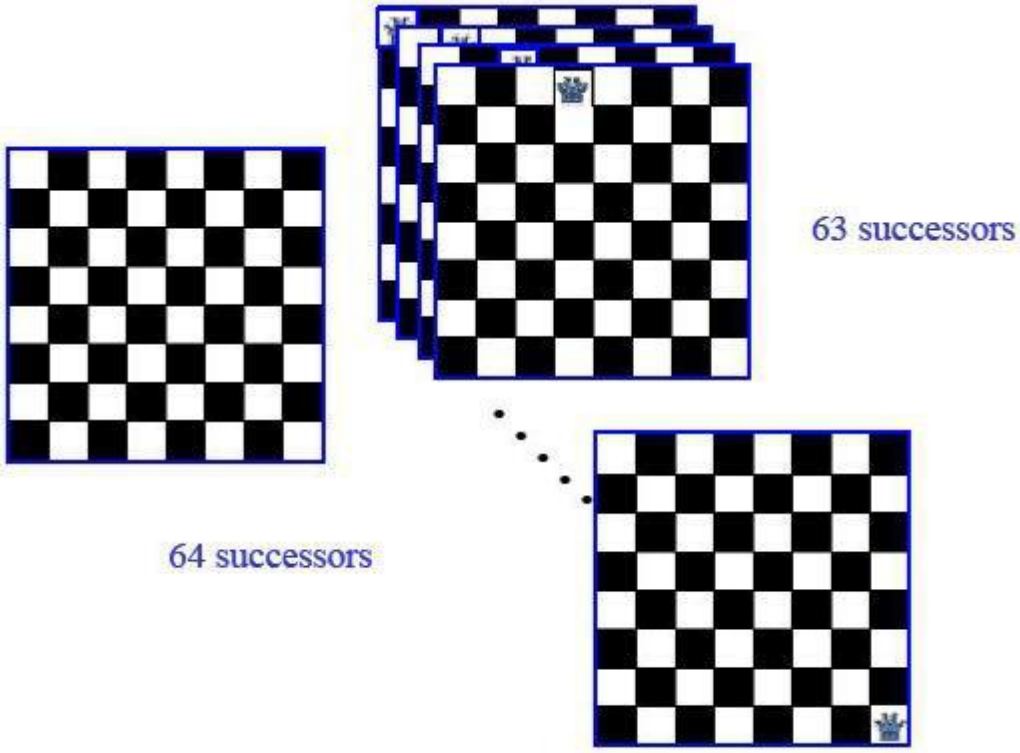


Figure 19

N queens problem formulation 2

States: Any arrangement of 8 queens on the board
Initial state: All queens are at column 1
Successor function: Change the position of any one queen
Goal test: 8 queens on the board, none are attacked

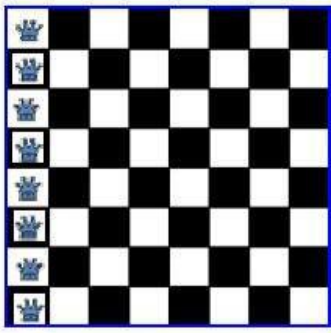


Figure 20

If we consider moving the queen at column 1, it may move to any of the seven remaining columns.

N queens problem formulation 3

States: Any arrangement of k queens in the first k rows such that none are attacked
Initial state: 0 queens on the board

Successor function: Add a queen to the (k+1) th row so that none are attacked.

Goal test : 8 queens on the board, none are attacked

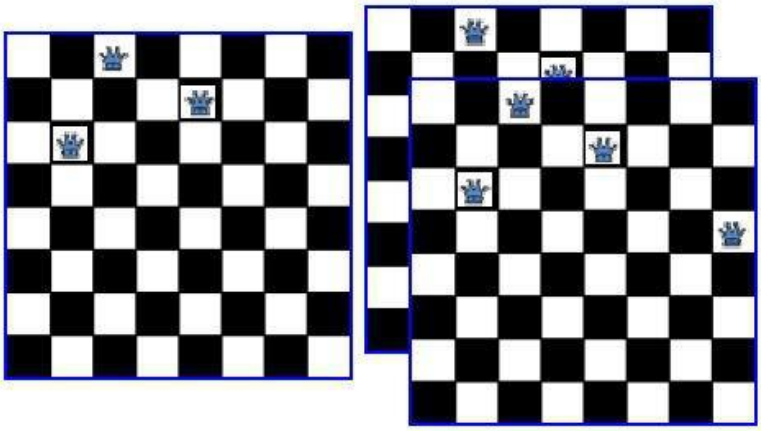


Figure 21

We will now take up yet another search problem, the 8 puzzle.

3.3.4 Problem Definition - Example, 8 puzzle

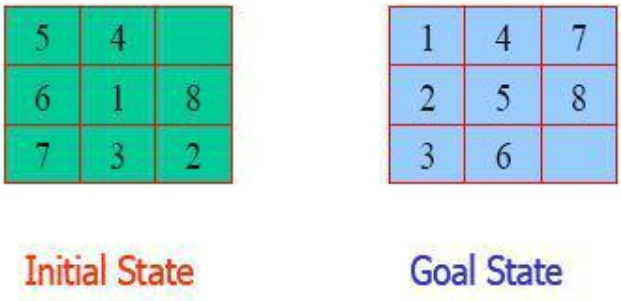


Figure 22

In the 8-puzzle problem we have a 3×3 square board and 8 numbered tiles. The board has one blank position. Bocks can be slid to adjacent blank positions. We can alternatively and equivalently look upon this as the movement of the blank position up, down, left or right. The objective of this puzzle is to move the tiles starting from an initial position and arrive at a given goal configuration.

The 15-puzzle problems is similar to the 8-puzzle. It has a 4×4 square board and 15 numbered tiles

The state space representation for this problem is summarized below: States: A state is a description of each of the eight tiles in each location that it can occupy.

Operators/Action: The blank moves left, right, up or down

Goal Test: The current state matches a certain state (e.g. one of the ones shown on previous slide)

Path Cost: Each move of the blank costs 1

A small portion of the state space of 8-puzzle is shown below. Note that we do not need to generate all the states before the search begins. The states can be generated when required.

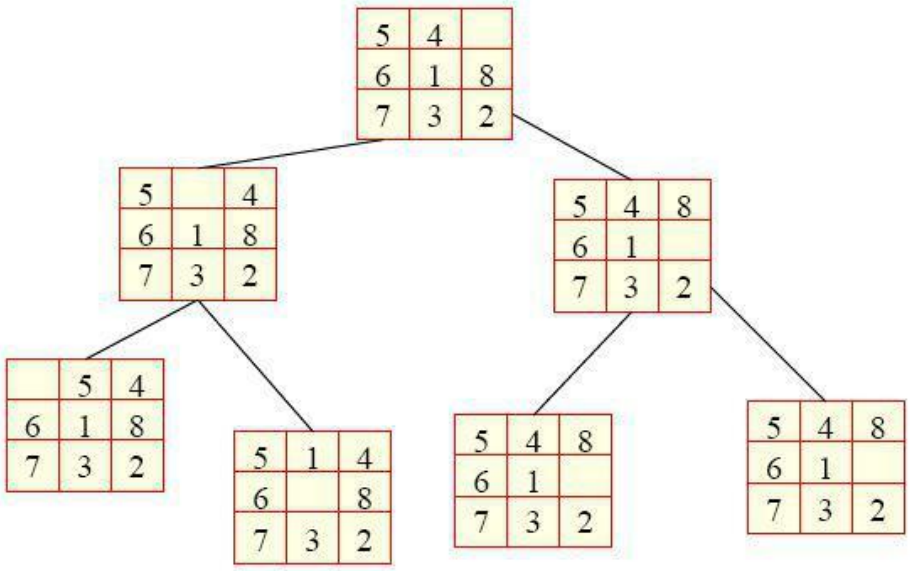


Figure 23

8-puzzle partial state space

3.4 Types of AI Search Techniques

Solution can be found with less information or with more information. It all depends on the problem we need to solve. Usually when we have more information it will be easy to solve the problem. The following are the types of AI search namely: Uninformed Search, List search, Tree search , Graph search, SQL search, Tradeoff Based search, Informed search, Adversarial search. This module will only deal with uninformed search, informed search and Tree search.

4.0 CONCLUSION

State space search is a process used in the field of computer science, including artificial intelligence (AI), in which successive configurations or *states* of an instance are considered, with the goal of finding a *goal state* with a desired property.

Problems are often modelled as a state space, a set of *states* that a problem can be in. The set of states forms a graph where two states are connected if there is an *operation* that can be performed to transform the first state into the second.

State space search often differs from traditional computer science search methods because the state space is *implicit*: the typical state space graph is much too large to generate and store in memory. Instead, nodes are generated as they are explored, and typically discarded thereafter. A solution to a combinatorial search instance may consist of the goal state itself, or of a path from some *initial state* to the goal state.

5.0 SUMMARY

In this unit, you have learnt that:

State space search is a process used in the field of computer science, including artificial intelligence (AI), in which successive configurations or *states* of an instance are considered, with the goal of finding a *goal state* with a desired property

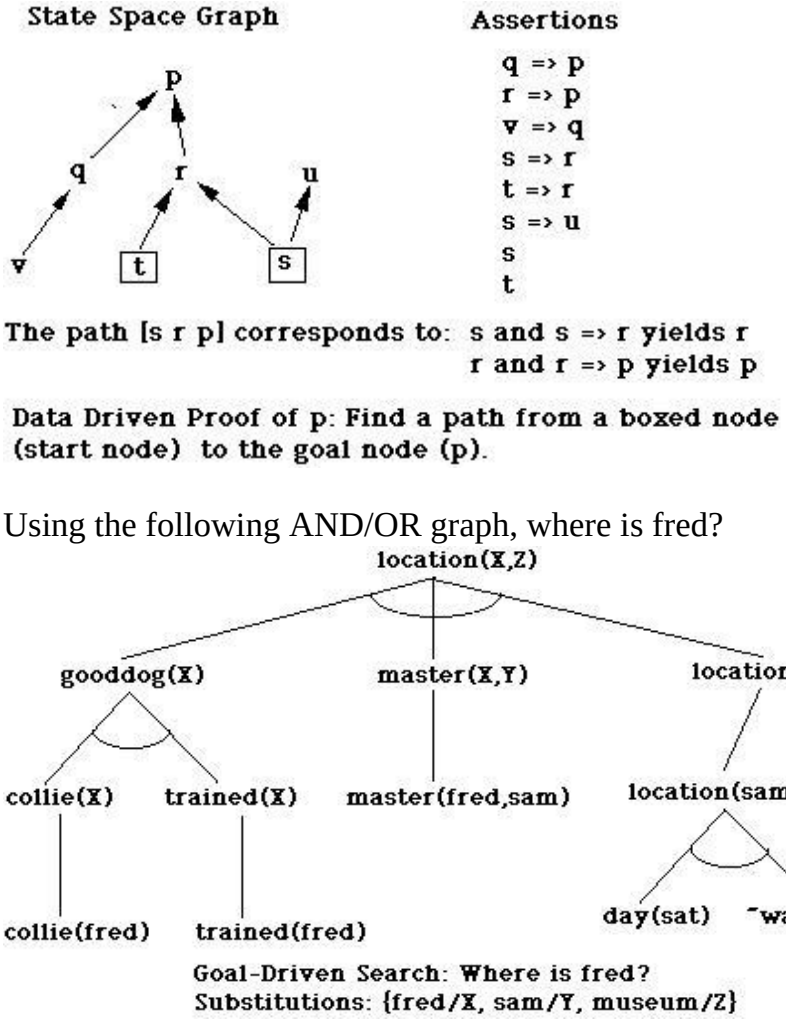
The search problem is to find a sequence of actions which transforms the agent from the initial state to a goal state g∈G. A search problem is represented by a 4-tuple {S, s , A, G}.

0

Solution can be found with less information or with more information. It all depends on the problem we need to solve

6.0 TUTOR-MARKED ASSIGNMENT

1. Find a path from a boxed node to the goal node (p).



2. Using the following AND/OR graph, where is fred?

7.0 REFERENCES/FURTHER READING

Dechter, R. & Judea, P. (1985). "Generalized Best-First Search Strategies and the Optimality of A*". *Journal of the ACM* 32 (3): 505–536. doi:10.1145/3828.3830.

Koenig, S.; Maxim, L.; Yaxin, L.; David, F. (2004). "Incremental Heuristic Search in AI". *AI Magazine* 25 (2): 99–112. http://portal.acm.org/citation.cfm?id=1017140.

Nilsson, N. J. (1980). *Principles of Artificial Intelligence*. Palo Alto, California: Tioga Publishing Company. ISBN 0-935382-01-1.

Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc. ISBN 0-201-05594-5.

Russell, S. J. & Norvig, P. (2003). *Artificial Intelligence. A Modern Approach*. Upper Saddle River, N.J.: Prentice Hall. pp. 97–104. ISBN 0-13-790395-2.