Artificial Intelligence

Lecture1:Introduction to PROLOG Programming Language

Asst. Lect.: Hazim N. Abed

PROLOG PL

- Prolog: is language derived from (Programming in Logic), it one of the most widely used programming languages in artificial intelligence research:

- Programming languages are two kinds:

1. **Procedural Programming** : in which we tell the computer how to solve a problem, such as BASIC, C++, Java, Pascal)

2. **Declarative Programming**: in which we tell the computer what problem we want solved, such as(LISP, Prolog, ML)

PROLOG PL

Basic Elements of Prolog

- There are only three basic constructs in Prolog: **Fact**, **Rule** and **Queries**.

- A collection of facts and rules is called a knowledge base(or a database) and Prolog Programming is all about writing knowledge bases.

- That is, Prolog programs simply are knowledge bases, collections of facts and rules which describe some collection of relationships that we find interesting**.**

PROLOG PL

- **Facts**: are statements that describe object properties or relations between objects. ”Some are always true”

  e.g. `father(jane,alan).`

  = Can be read as “Jane is the father of Alan.”

- **Rules**: rule enable to derive a new property or relation from a set of existing ones.”Some are dependent on others being true”

  e.g. `parent(X,Y) :-  father(X,Y).`

  = “Person X is the parent of person Y if X is Y’s father.”

PROLOG PL

- **Queries or Questions**: is a request to prove or retrieve information from the database.

```
-Example: ‟ who is Jim′s father”??-father(who,Jim)
```

- Both facts and rules are **predicate definitions**.

- ‘**Predicate**’ is the name given to the word occurring before the bracket in a fact or rule:

```
father(jane,alan).
```

Predicate name

PROLOG PL

Clauses

- Predicate definitions consist of *clauses.*
  - = An individual definition (whether it be a fact or rule).

e.g. `mother(jane,alan).` = Fact

`parent(P1,P2):- mother(P1,P2).` = Rule

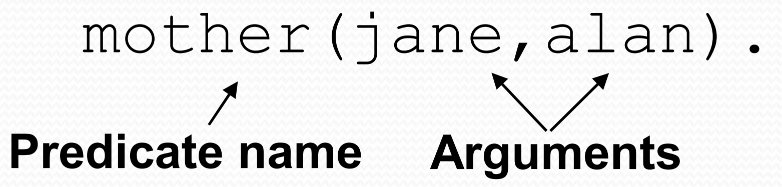
head **body**

- A clause consists of a head and sometimes a body. Facts don't have a body because they are always true.

PROLOG PL

Arguments:

- A predicate <u>head</u> consists of a *predicate name* and sometimes some *arguments* contained within brackets and separated by commas.



```
mother(jane,alan).
```

- A body can be made up of any number of subgoals (calls to other predicates) and terms.
- Arguments also consist of terms, which can be:
  - Constants e.g. jane,
  - Variables e.g. Person1, or
  - Compound terms.

PROLOG PL Terms: Constants

Constants can either be:

- Numbers:
  - integers are the usual form (e.g. 1, 0, -1, etc), but
  - floating-point numbers can also be used (e.g. 3.0E7)
- Symbolic (non-numeric) constants:
  - always start with a lower case alphabetic character and contain any mixture of letters, digits, and underscores (but no spaces, punctuation, or an initial capital).
    - e.g. abc, big_long_constant, x4_3t).
- String constants:
  - are anything between single quotes e.g. ‘Like this’.

PROLOG PL Terms: Variables

- Variables: always start with an upper case alphabetic character or an underscore.

- Other than the first character they can be made up of any mixture of letters, digits, and underscores.

  e.g. `X, ABC, _89two5, _very_long_variable`

- There are no "types" for variables (or constants) – a variable can take any value.
- All Prolog variables have a "local" scope:
  - they only keep the same value within a clause; the same variable used outside of a clause does not inherit the value (this would be a "global" scope).

PROLOG PL Terms: Compound Terms

Compound Terms

- The structured data objects of the language are the compound terms.
- A compound term comprises a functor (called the principal functor of the term) and a sequence of one or more terms called arguments. A functor is characterized by its name, which is an atom, and its arity or number of arguments.
- For example the compound term whose functor is named point of arity 3, with arguments X, Y and Z, is written: point(X, Y, Z)

PROLOG PL : Data type.

Data type:

Prolog supports the following data type to define program entries.

1. **Integer**: to define numerical value like 1, 20, 0,-3,-50, ect.

2. **Real**: to define the decimal value like 2.4, 3.0, 5,-2.67, ect.

3. **Char**: to define single character, the character can be of type small letter or capital letter or even of type integer under one condition it must be surrounded by single quota. For example, ‘a’,’C’,’123’.

PROLOG PL : Data type.

4. **string** : to define a sequence of character like “good” i.e define word or statement entries the string must be surrounded by double quota for example “computer”, “134”, “a”. The string can be of any length and type.

5. **Symbol**: another type of data type to define single character or sequence of character but it must begin with small letter and don’t surround with single quota or double quota.

PROLOG PL : Program Structure

Program structure: Prolog program structure consists of five segments, not all of them must appear in each program. The following segment must be included in each program predicates, clauses, and goal.

1. **Domains**: define global parameter used in the program.

```
Domains
    I= integer
    C= char
    S = string
    R = real
```

PROLOG PL : Program Structure

2. **Data base**: define internal data base generated by the program

    Database

        Greater (integer)

3. **Predicates**: define rule and fact used in the program.

    Predicates

        Mark(symbol,integer).

4. **Clauses**: define the body of the program.. For the above predicates the clauses portion may contain Mark (a, 20).

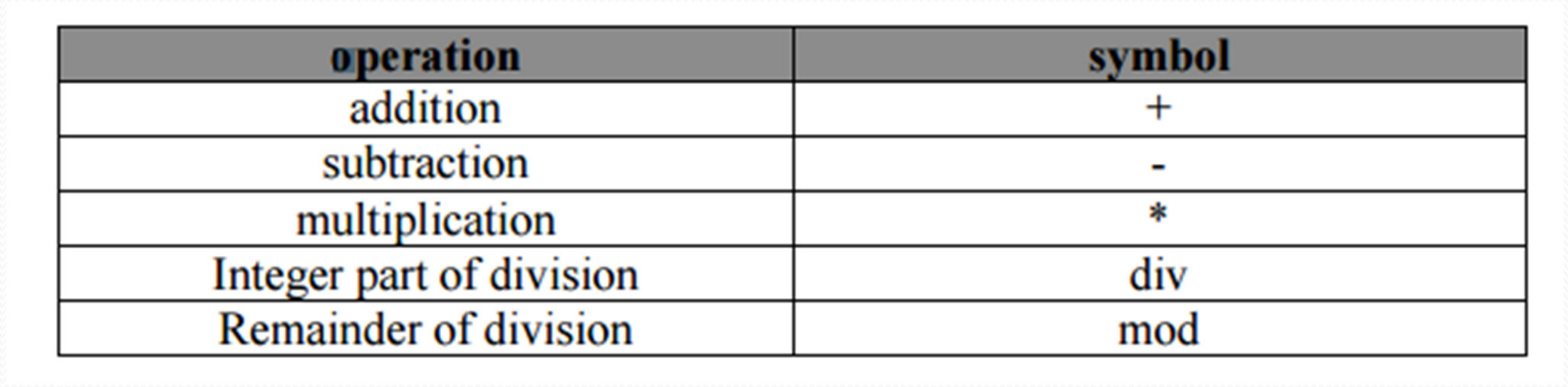
PROLOG PL : Program Structure

5. Goal: can be internal or external, internal goal written after clauses portion , external goal supported by the prolog compiler if the program syntax is correct

- This portion contains the rule that drive the program execution.

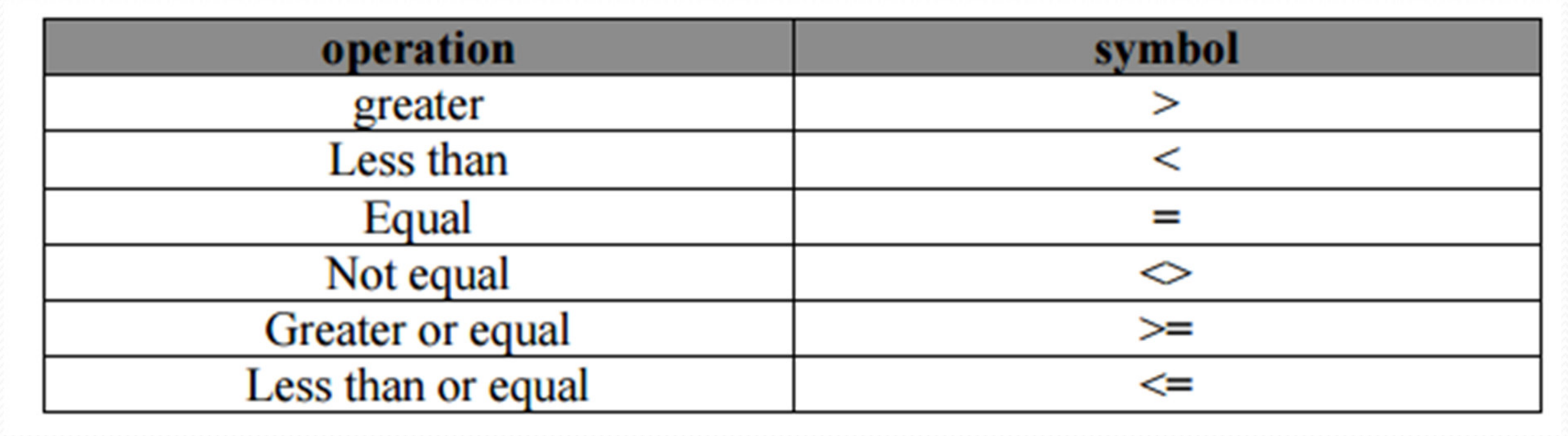
PROLOG PL: Mathematical and logical operation

A .mathematical operation:



operation	symbol
addition	+
subtraction	-
multiplication	*
Integer part of division	div
Remainder of division	mod

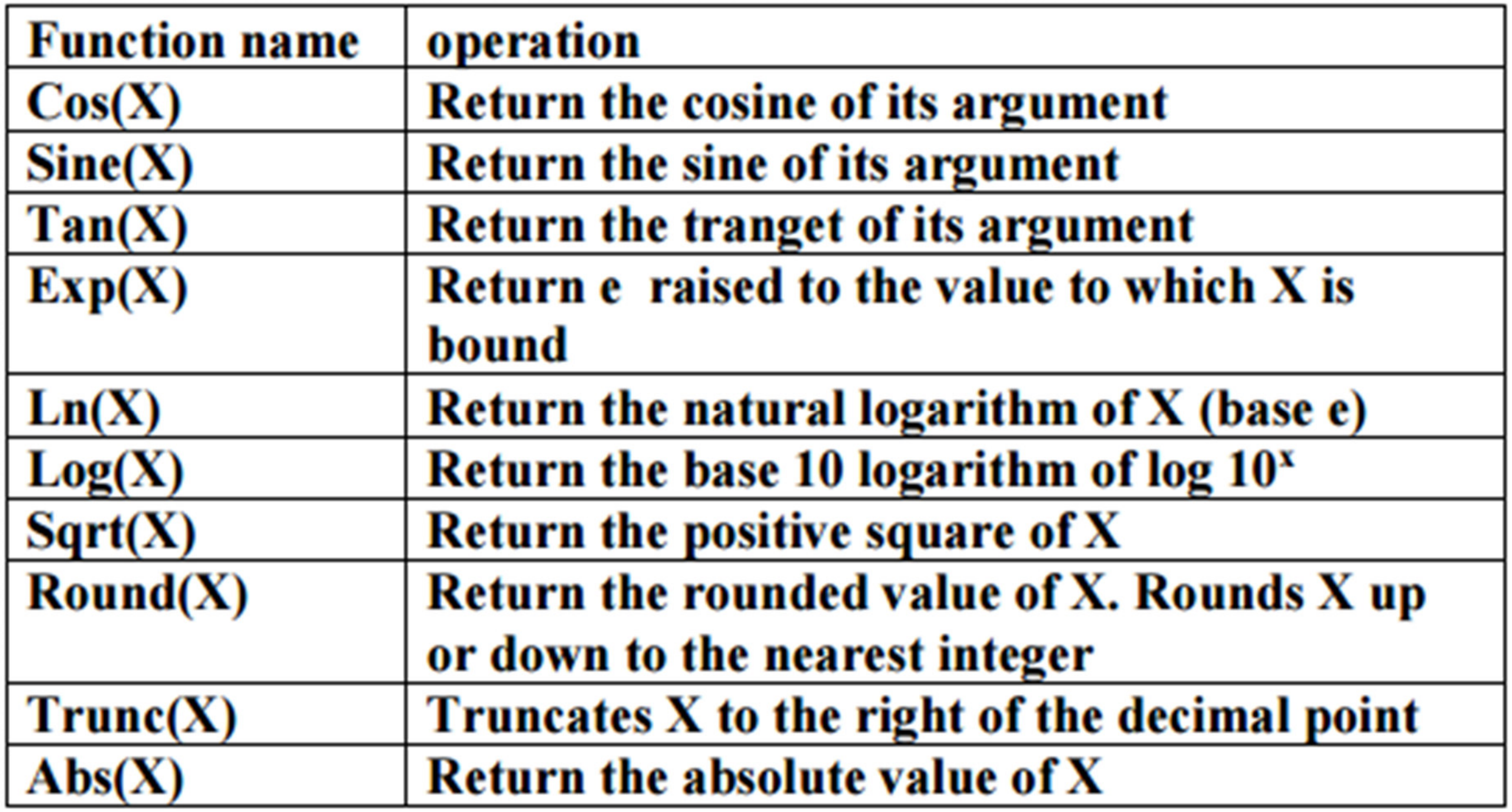
B .logical operation:



operation	symbol
greater	>
Less than	<
Equal	=
Not equal	<>
Greater or equal	>=
Less than or equal	<=

PROLOG PL: Other mathematical function

- **Other mathematical function**



Function name	operation
Cos(X)	Return the cosine of its argument
Sine(X)	Return the sine of its argument
Tan(X)	Return the tranget of its argument
Exp(X)	Return e raised to the value to which X is bound
Ln(X)	Return the natural logarithm of X (base e)
Log(X)	Return the base 10 logarithm of log 10^x
Sqrt(X)	Return the positive square of X
Round(X)	Return the rounded value of X. Rounds X up or down to the nearest integer
Trunc(X)	Truncates X to the right of the decimal point
Abs(X)	Return the absolute value of X

PROLOG PL: Read and Write function

Read and Write function

1. Read function:

Readint(Var) : read integer variable.

Readchar(Var) : read character variable.

Readreal(Var) : read (decimal) variable.

Readln(Var) : read string.

2. Write function

Write(Var) : write variable of any type.

PROLOG PL: Read and write function

Example : Write prolog program to read integer value and print it.

```
Domains
        I = integer
Predicates
         print.
Clauses
        Print:- write ("please read integer number"), readint(X),
        write("you read",X).
Goal
         Print.
Output:
        Please read integer number 4
        You read 4
```

PROLOG PL: Pattern Matching

- Prolog uses unification to match variables to values. An expression tat contains variables like X+Y*Z describes a pattern where there are three blank spaces to fill in named X, Y, and Z.
- The expression 1+2*3 as the same structure (pattern) but no variables. If we input this query X+Y*Z=1+2*3.
- Then prolog will respond that X=1, Y=2, and Z=3. the pattern matching is very powerful because you can match variables to expression like this X+Y=1+2*3, and get X+1 and Y=2*3.
- You can match variable to variable: X+1+Y=Y+Z+2. This sets X=Y=2 and Z=1.

PROLOG PL: Cut and Fail function

1. Cut

Represented as “!” is a built in function always True , used to stop backtracking and can be placed any where in the rule, we list the cases

where “!” can be inserted in the rule:

1 .R:-f1, f2,!. “f1, f2 will be deterministic to one solution.

2. R:-f1,!,f2. “ f1 will be deterministic to one solution while f2 to all .

3. R:- !,f1,f2. “R will be deterministic to one solution.

PROLOG PL: Cut and Fail function

Example1 : program without use cut.

```
Domains
        I= integer
Predicates
        No( I )
Clauses
        No (5).
        No (7).
        No (10).
Goal
        No (X).
Output:
        X=5
        X=7
        X=10
```

Example 2: program using cut.

```
Domains
        I= integer
Predicates
        No( I )
Clauses
        No (5):-!.
        No (7).
        No (10).
Goal
        No (X).
Output:
        X=5.
```

PROLOG PL: Cut and Fail function

Example 3: using cut in the end of the rule.

```
Domains
        I =integer
        S = symbol
Predicates
        a(I )
        b (s )
        c (I, s )
Clauses
        a(10).
        a(20)
        b(a)
        b(c)
        c (X, Y):- a (X), b (Y),!.
Goal
        c(X,Y).
Output:
        X= 10 Y=a
```

Example 5: using cut in the middle of the rule.

```
Domains
        I =integer
        S = symbol
Predicates
        a(I )
        b (s )
        c ( I, s )
Clauses
        a(10).
        a(20)
        b(a)
        b(c)
        c (X, Y):- a (X),!, b (Y).
Goal
        c(X,Y).
Output:
        X= 10 Y=a
              Y=c
```

PROLOG PL: Cut and fail function

2- fail: the fail predicate is provided by prolog. When it is called, it causes the failure of the rule. And this will be forever, nothing can change the statement of this predicate.

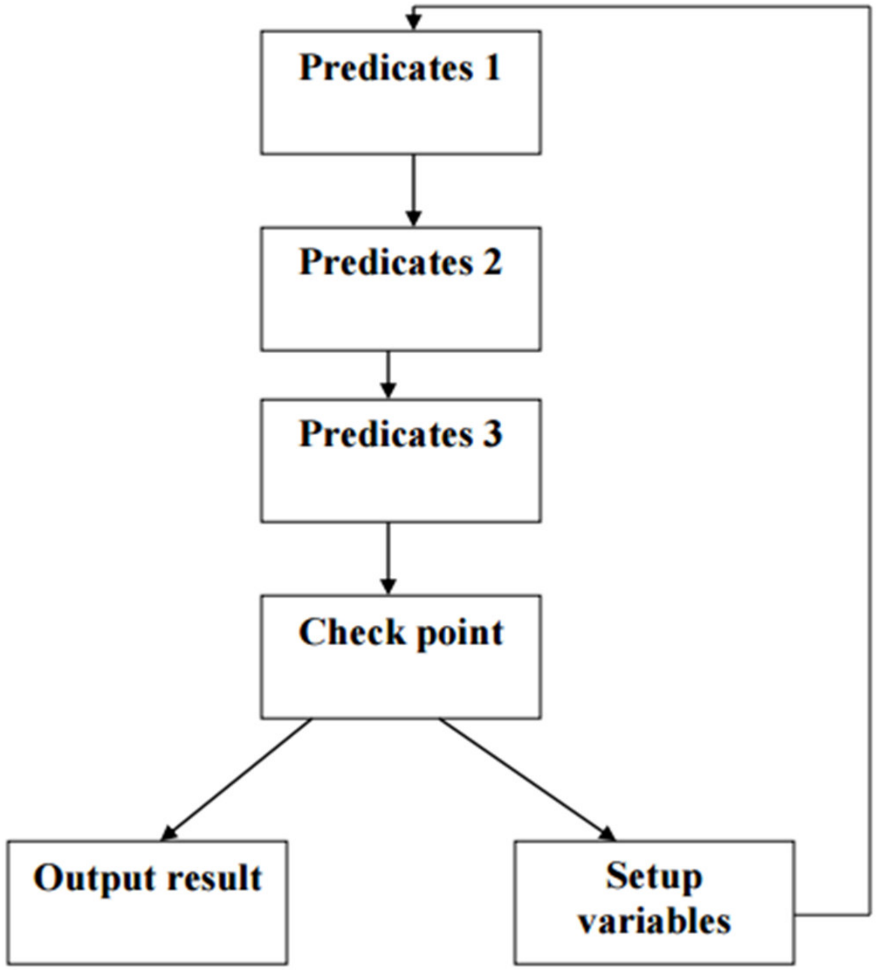
PROLOG PL: Recursion

- In addition to have rules that use other rules as part of their requirements, we can have rules that use themselves as part of their requirements.
- This kind of rule called “recursive “because the relation ship in the conclusion appears again in the body of the rule, where the requirements are specified.
- A recursive rule is a way of generating a chain of relationship for a recursive rule to be effective. However, there must be some place in the chain of relationship where the recursion stops.

PROLOG PL: Recursion

1. Tail Recursion

- We place the predicate that cause the recursion in the tail of the rule as shown below:



PROLOG PL: Recursion

Example: Program to find factorial.

5! = 5*4*3*2*1

Predicates

```
Fact ( integer, integer, integer)
```

Clauses

```
Fact(1, F, F):-!.
Fact(N,F,R):- F1=F*N , N1=N-1, fact(N1,F1,R).
```

Goal

```
Fact (5,1,F).
```

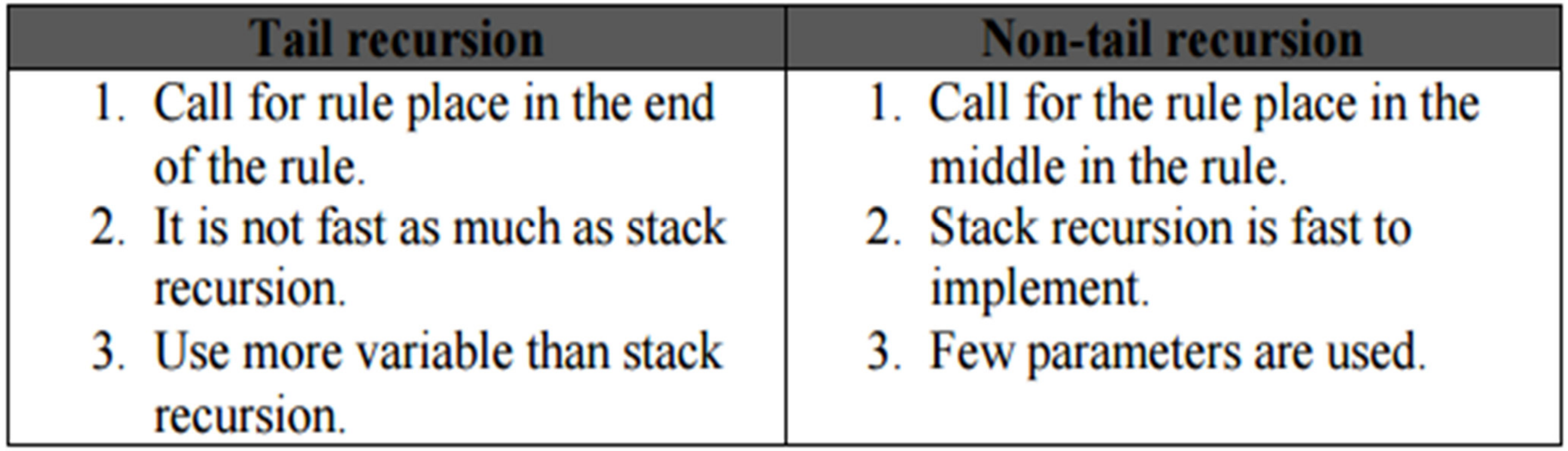
Output:

```
F = 120
```

PROLOG PL: Recursion

2. Non –Tail Recursion (Stack Recursion)

- This type of recursion us the stack to hold the value of the variables till the recursion is complete. The statement is self – repeated as many times as the number of items in the stack.. Below a simple comparison between tail and non-tail recursion.



Tail recursion	Non-tail recursion
1. Call for rule place in the end of the rule. 2. It is not fast as much as stack recursion. 3. Use more variable than stack recursion.	1. Call for the rule place in the middle in the rule. 2. Stack recursion is fast to implement. 3. Few parameters are used.

PROLOG PL: Recursion

Example: factorial program using non-tail recursion.

Predicates

```
    fact(integer, integer).
```

Clauses

```
        fact(1,1).
        fact(N,F):- N>1,N1=N-1,fact(N1,F1),F=N*F1.
```

Goal

```
        Fact (4,Y)
```

Output:

```
        Y =24.
```

PROLOG PL: List

1. List in prolog:

- In prolog, a list is an object that contains an arbitrary number of other objects within it.

- Lists correspond roughly to array in other languages but unlike array, a list dose not require you to how big it will be before use it.

2. syntax of list

- List always defined in the domains section of the program as follow:

Domains

list = integer*

- '*' refer to list object which can be of length zero or un defined.

PROLOG PL: List

- The type of element list can be of any standard defined data type like integer, char … ect

- List element surrounded with square brackets and separated by comma as follow: l = [1, 2, 3, 4].
- List consist of two parts **head** and **tail** , the head represent the first element in the list and the tail represent the remainder (i.e head is an element but tail is a list) . for the following list :

```
L = [1,2,3]
        H = 1   T =[2,3]
                H =2    T =[3]
                        H =3 T=[ ]
```

PROLOG PL: List

- [ ] refer to empty list.
- List can be written as [H|T] in the program, if the list is non empty then this statement decompose the list into Head and tail otherwise (if the list is empty) this statement add element to the list.

3. **List and Recursion**

- As maintained previous list consist of many element, therefore to manipulate each element in the list we need recursive call to the list until it become empty.

PROLOG PL: List

Example: program to print list element in one line.

Domains

```
L = integer*
```

Predicates

```
Print (L)
```

Clauses

```
Print ([ ]):-!.
Print ( [ H|T]):- write (H) , print (T).
```

Goal

```
Print ([1,4,6,8]).
```

Output:

```
1468
```

PROLOG PL: List

Example: program to find sum of integer list.

Domains

```
I= integer
L=i*
```

Predicates

```
Sum ( L I, I)
```

Clauses

```
Sum ( [ ],S,S):-!.
Sum( [H| T],S1,S):- S2 = S1+H , Sum (T,S2,S).
```

Goal

```
Sum ( [ 1,4,6,9],0 ,S).
```

Output

```
S = 20
```

Artificial Intelligence

Lecture 2: Introduction to AI

Instructor : MSc. Hazim N. Abed

Introduction: Intelligence

What is intelligence?

- "Intelligence denotes the ability of an individual to adapt his thinking to new demands; it is the common mental adaptability to new tasks and conditions of life" (William Stern, 1912)

- Being "intelligent" means to be able to cognitively grasp phenomena, being able to judge, to trade of between different possibilities, or to be able to learn.

Introduction: Intelligence

- An important aspect of "Intelligence" is the way and efficiency how humans are able to adapt to their environment or assimilate their environment for solving problems.

- Intelligence manifests itself in logical thinking, computations, the memory capabilities of the brain, through the application of words and language rules or through the recognition of things and events.

Introduction: Intelligence

- The combination of information, creativity, and new problem solutions is crucial for acting "intelligent".

- Intelligence: - “the capacity to learn and solve problems”
  - in particular,
    - the ability to solve novel problems
    - the ability to act rationally
    - the ability to act like humans

Introduction: Intelligence System

What is Intelligent Systems?

- Any formal or informal system to manage data gathering, to obtain and process the data, to interpret the data, and to provide reasoned judgments to decision makers as a basis for action.

- The term is not limited to intelligence organizations or services but includes any system, in all its parts, that accomplishes the listed tasks.

Introduction: Intelligence System

- The benefits of intelligent systems are:

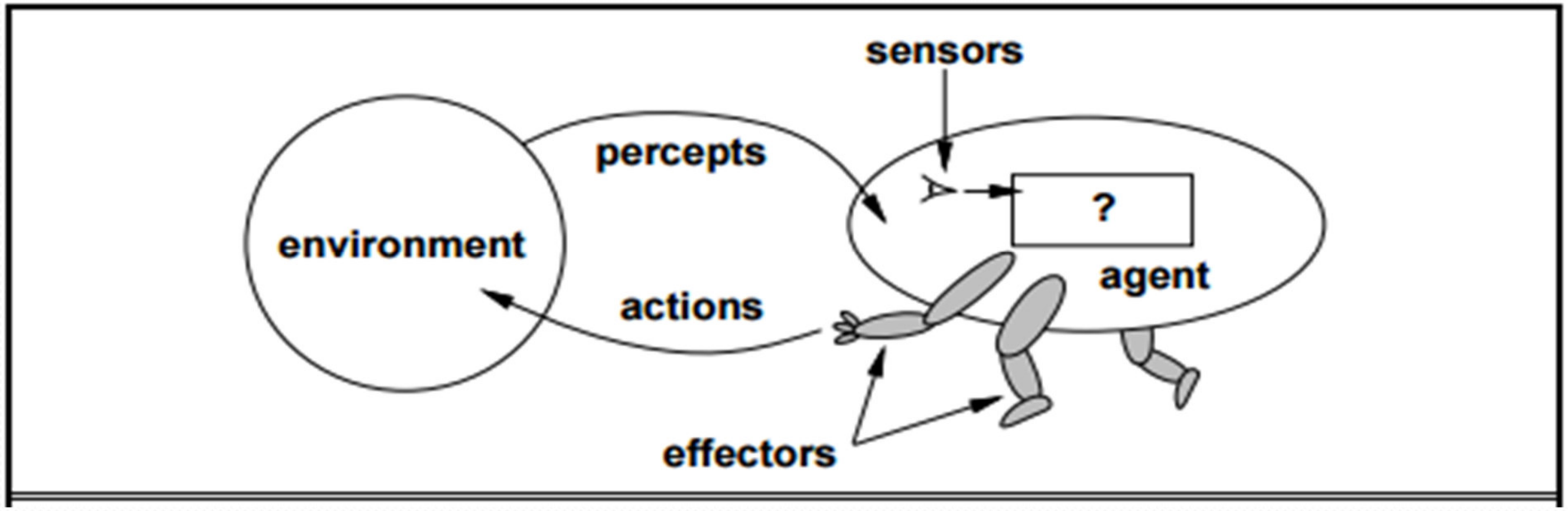
1. Enhanced problem-solving.
2. Improved decision quality.
3. Ability to solve complex problems.
4. Consistent decisions.

- The major difficulty is developing theses system is extracting the expertise needed to develop the knowledge base. It is difficult to extract an expert’s knowledge and codify it into a format that can be used in an automated application.

Introduction: Intelligence Agents

- **Agents:** Software that gathers information about an environment and takes actions based on that.

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. Figure 1.1 Agents interact with environment through sensors and effectors



Introduction: Intelligence Agents

- **There are different types of agent:**

1. **Human Agent**: A human agent has eyes, ears, and other organs for sensors, and hands, legs, mouth, and other body parts for effectors.

2. **Robotic Agents**: A robotic agent has cameras and infrared range finders for the sensors and various motors for the effectors.

3. **Software Agent**: A software agent has encoded bit strings as its percepts and actions.

4. **Generic agent**: A general structure of an agent who interacts with the environment

Introduction: Intelligence Agents

- The **agent function** for an agent specification the action taken by the agent in response to any percept sequence. Internally, the agent function for an artificial agent will be implemented ba an agent program.

- An **agent program** is a function that implements the agent mapping from percepts to actions. it is a concrete implementation, running on the agent architecture.

Introduction: Intelligence Agents

Structure of Agents:

- An intelligent agent is a combination of Agent Program and Architecture.

  Intelligent Agent= Agent Program + Architecture

- **Agent Program:** is a function that implements the agent mapping from percepts to action. The design of the agent program depends on the nature of the environment.

- **Architecture**: is a computing device used to run the agent program.

Introduction: Artificial Intelligence (AI)

What is AI?

- It is often difficult to construct a definition of a discipline that is satisfying to all of its practitioners. AI research encompasses a spectrum of related topics.

- Broadly, AI is the computer-based exploration of methods for solving challenging tasks that have traditionally depended on people for solution. Such tasks include complex logical inference, diagnosis, visual recognition, comprehension of natural language, game playing, explanation, and planning.

- There are many definitions of AI:

Introduction: Artificial Intelligence (AI)

- Artificial intelligence is the branch of computer science concerned with making computers behave like humans.

- AI is a branch of the field of computer and information science. It focuses on developing hardware and software system that solve problems and accomplish tasks if accomplished by humans would be considered a display of intelligence. The field of AI includes studying and developing machines such as robots, Automatic pilots foe airplane and space ships, and “smart” military weapons.

Introduction: Artificial Intelligence (AI)

- AI is the field as “ the study and design of intelligence agents”, where an intelligent agent is a system that perceive its environment and takes actions that maximize its chances of success.
- John McCarthy, who coined the term in 1956, defines AI ass “the science and engineering of making intelligence machines.
- AI is the intelligence of machines and branch of computer science that aim to create it.
- AI is the study of how to make computers do thinks which, at the moment, people do better.

Introduction: Artificial Intelligence (AI)

- As the result, AI is concerned with developing computer system that can store knowledge and effectively use the knowledge to help solve problems and accomplish tasks.

- The above definitions give us four possible goals to pursue in artificial intelligence:
  - Systems that think like humans
  - Systems that act like humans
  - Systems that think rationally.
  - Systems that act rationally

Introduction: Artificial Intelligence (AI)

1. **Acting Humanly: Turing Test approach**

- The Turing Test, proposed by Alan Turing (Turing, 1950), was designed to provide a satisfactory operational definition of intelligence.

- He considered the question, “Can machine think?” Rather than define the terms “machine or “think”. Turing proposed the test that begins with to people and the machine to be evaluated. One person plays the role of the interrogator, who is in a separate room from the computer and the other person (Fig. 1.2).

Introduction: Artificial Intelligence (AI)

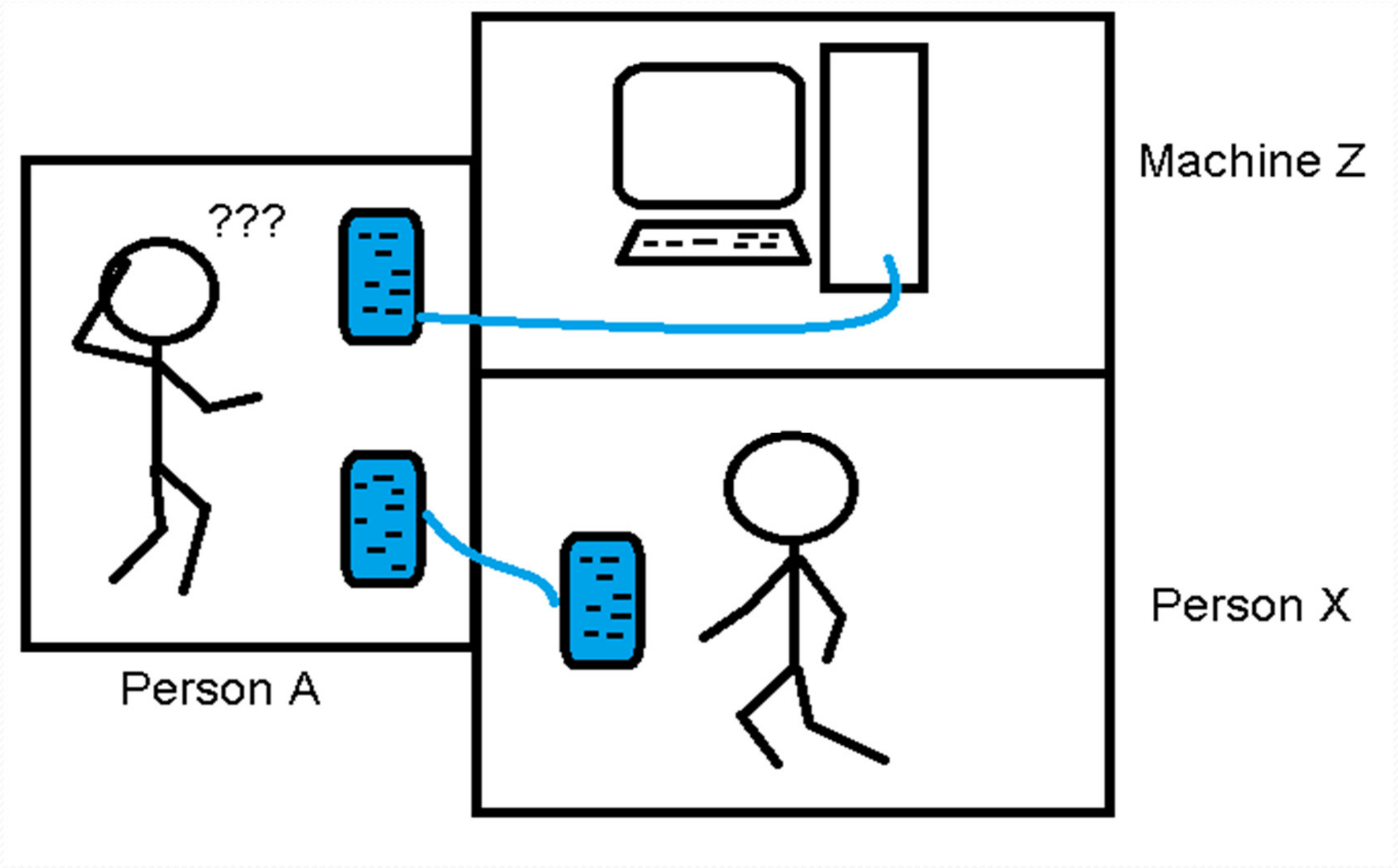


Figure 1.2: The Turing Test

Introduction: Artificial Intelligence (AI)

- The interrogator can ask questions of either the person or the computer by typing questions and receiving typed responses. However, the interrogator knows them only as A and B and aim to determine which the person is and which are the machine.

- The goal of the machine is to fool the interrogator into believing that is the person.

- If the machine succeeds at this, then we will conclude that the machine is acting humanly.

Introduction: Artificial Intelligence (AI)

- Programming a computer to pass the test provides plenty to work on. The computer would need to possess the following capabilities:

1. **Natural language processing:** to enable it to communicate successfully in English (or some other human language);
2. **Knowledge representation:** to store information provided before or during the interrogation;
3. **Automated reasoning:** to use the stored information to answer questions and to draw new conclusions;
4. **Machine learning:** to adapt to new circumstances and to detect and extrapolate patterns.

Introduction: Artificial Intelligence (AI)

- Total Turing Test includes a video signal so that the interrogator can test the subject's perceptual abilities, as well as the opportunity for the interrogator to pass physical objects ``through the hatch." To pass the total Turing Test, the computer will need:

- Computer vision to perceive objects, and
- Robotics to move them about.

Introduction: Artificial Intelligence (AI)

2. **Thinking Humanly: The Cognitive modeling approach**

- To construct a machine program to think like a human, first it requires the knowledge about the actual workings of human mind.
- After completing the study about human mind it is possible to express the theory as a computer program. It the program’s input/output and timing behavior matching with the human behavior then we can say that the program’s mechanism is working like a human mind.
- Example: General Problem Solver(GPS)

Introduction: Artificial Intelligence (AI)

3. **Thinking rationally: The laws of thought approach**

- The right thinking introduced the concept of logic.

- **Example**: - Rami is a student of III year CS.
  - All students are good in III year in CS.
  - Rami is a good student.

Introduction: Artificial Intelligence (AI)

3. **Acting rationally: The rational agent approach**

- Acting rationally means, to achieve one’s goal given one’s beliefs.
- In the previous topic laws of thought approach, correct inference is selected, conclusion is derived, but the agent acts on the conclusion defined the task of acting rationally.
- The study of rational agent has Tow advantages:

1. Correct inference is selected and applied.
2. It concentrates on scientific development rather than other methods

Artificial Intelligence

Lecture 3: Applications of AI, Search Algorithms

Instructor : MSc. Hazim N. Abed

Introduction: Applications Of AI

1. Problem Solving:-

- This is the first application area of AI research., the objective of this particular area of research is how to implement the procedures on AI systems to solve the problems like Human Beings.

2. Game Playing:-

- Much of early research in state space search was done using common board games such as checkers, chess and 8 puzzle. Most games are played using a well defined set of rules.

Introduction: Applications Of AI

- This makes it easy to generate the search space and frees the researcher from many of the ambiguities and complexities inherent in less structured problems. The board Configurations used in playing these games are easily represented in computer, requiring none of complex formalisms.
- For solving large and complex AI problems it requires lots of techniques like Heuristics. We commonly used the term intelligence seems to reside in the heuristics used by Human beings to solve the problems.

Introduction: Applications Of AI

3. Natural Language understanding:-

- The main goal of this problem is we can ask the question to the computer in our mother tongue the computer can receive that particular language and the system gave the response with in the same language. The effective use of a Computer has involved the use of a Programming Language that use a set of Commands that we must use to Communicate with the Computer. The goal of natural language processing is to enable people and language such as English, rather than in a computer language.

Introduction: Applications Of AI

It can be divided in to Two sub fields.

A. Natural Language Understanding : Which investigates methods of allowing the Computer to improve instructions given in ordinary English so that Computers can understand people more easily.

B. Natural Language Generation : This aims to have Computers produce ordinary English language so that people an understand Computers more easily.

Introduction: Applications Of AI

4. Perception:-

- The process of perception is usually involves that the set of operations i.e. Touching , Smelling Listening , Tasting , and Eating. These Perceptual activities incorporation into Intelligent Computer System is concerned with the areas of Natural language Understanding, Processing and Computer Vision mainly.
- The are two major Challenges in the application area of Perception.

Introduction: Applications Of AI

A. Speech Reorganization:-

- The main goal of this problem is how the Computer System can recognize our Speeches. (Next process is to understand those Speeches and process them i.e. Encoding & Decoding i.e producing the result in the same language.) Its one is very difficult; Speech Reorganization can be described in two ways.

Introduction: Applications Of AI

1. Discrete Speech Reorganization

Means People can interact with the Computer in their mother tongue. In such interaction whether they can insert time gap in between the two words or two sentences (In this type of Speech Reorganization the computer takes some time for searching the database).

Introduction: Applications Of AI

2. Continues Speech Reorganization

- Means when we interact with the computer in our mother tongue we can not insert the time gap in between the two words or sentences , i.e. we can talk continuously with the Computer (For this purpose we can increase speed of the computer).

Introduction: Applications Of AI

B. Pattern Reorganization:-

- This the computer can identify the real world objects with the help of "Camera". Its one is also very difficult , because

- To identify the regular shape objects, we can see that object from any angle; we can imagine the actual shape of the object (means to pictures which part is light fallen) through this we can identify the total structure of that particular object.

Introduction: Applications Of AI

- To identify the irregular shape things, we can see that particular thing from any angle; through this we cannot imagine the actual structure. With help of that we can attach the Camera to the computer and pictures certain part of the light fallen image with the help of that whether the AI system can recognize the actual structure of the image or not? It is some what difficult compare to the regular shape things, till now the research is going on. This is related the application area of Computer Vision.

Introduction: Applications Of AI

5. Image Processing:-

- Where as in pattern reorganization we can catch the image of real world things with the help of Camera. The goal of Image Processing is to identify the relations between the parts of image.

- It is a simple task to attach a Camera to a computer so that the computer can receive visual images. People generally use Vision as their primary means of sensing their environment.

Introduction: Applications Of AI

- We generally see more than we here. i.e. how can we provide such perceptual facilities touch, smell, taste, listen, and eat to the AI System.
- The goal of Computer Vision research is to give computers this powerful facility for understanding their surroundings. Currently, one of the primary uses of Computer Vision is in the area of Robotics.

Introduction: Applications Of AI

6. Expert system:-

- Expert means the person who had complete knowledge in particular field, i.e is called as an expert. The main aim of this problem is with the help of experts, to load their tricks on to the compute and make available those tricks to the other users. The expert can solve the problems with in the time.

- The goal of this problem is how to load the tricks and ideas of an expert on to the computer, till now the research will be going on.

Introduction: Applications Of AI

7. Computer Vision:-

- It is a simple task to attach a camera to a computer so that the computer can receive visual images. People generally use vision as their primary means of sensing their environment. We generally see more than we here, feel, smell, or taste.
- The goal of computer vision research is to give computers this powerful facility for understanding their surroundings. Currently, one of the primary uses of computer vision is in the area of Robotics.

Introduction: Applications Of AI

8. Robotics:-

- A robot is an electro – mechanical device that can be programmed to perfume manual tasks. The robotics industries association formally defines to move a Robot as a " Programmable multi-functional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of variety of tasks".

Introduction: Applications Of AI

- Not all robotics is considered to be part of AI. A Robot that perform sonly the actions that it is has been pre-programmed to perform is considered to be a “dumb” robot, includes some kind of sensory apparatus, such as a camera , that allows it to respond to changes in its environment , rather than just to follow instructions “mindlessly”.

Introduction: Applications Of AI

9. Intelligent Computer – Assisted Instruction:-

- Computer - Assisted Instruction (CAI) has been used in bringing the power of the computer to bear on the educational process. Now AI methods are being applied to the development of intelligent computerized “ Tutors” that shape their teaching techniques to fit the leaning patterns of individual students.

Introduction: Applications Of AI

10. Heuristic Classification:-

- The term Heuristic means to Find & Discover., find the problem and discover the solution. For solving complex AI problems it’s requires lots of knowledge and some represented mechanisms in form of Heuristic Search Techniques., i.e referred to known as Heuristic Classification.

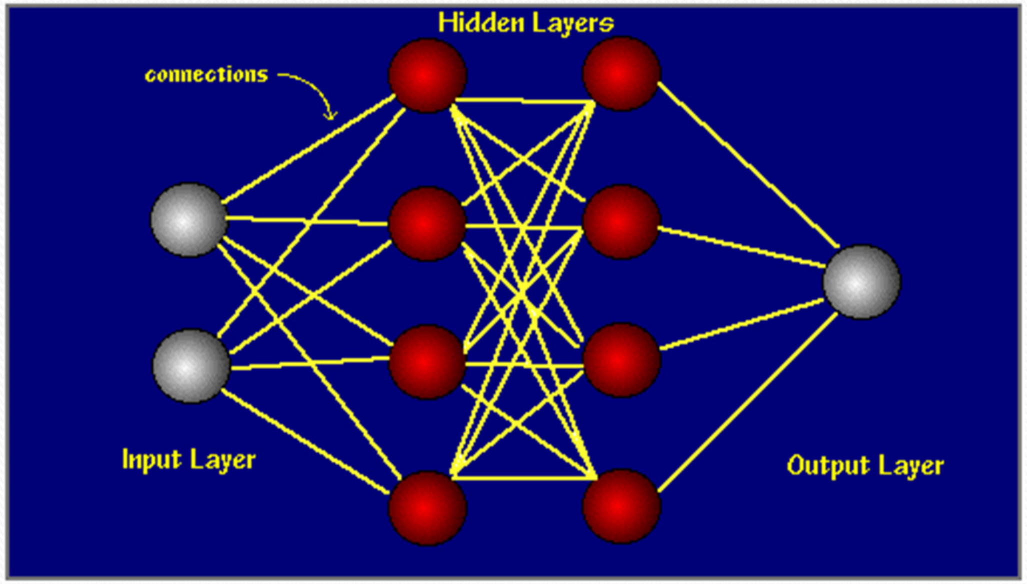
Introduction: Applications Of AI

11. Neural Network:-

- An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example.

Introduction: Applications Of AI

- An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.



Introduction: Search Algorithms

What is problem reduction meaning?

- Problem Reduction means that there is a hard problem may be one that can be reduced to a number of simple problems. Once each of the simple problems is solved, then the hard problem has been solved.

Search Algorithms:

- To successfully design and implement search algorithms, a programmer must be able to analyze and predict their behavior.

Introduction: Search Algorithms

Many questions needed to be answered by the algorithm these include:

- Is the problem solver guaranteed to find a solution?
- Will the problem solver always terminate , or can it become caught in an infinite loop?
- When a solution is found, is it guaranteed to be optimal?
- What is the complexity of the search process in terms of time usage? Space search?
- How can the interpreter be designed to most effectively utilize a representation language?

Introduction: Search Algorithms

State Space Search:-

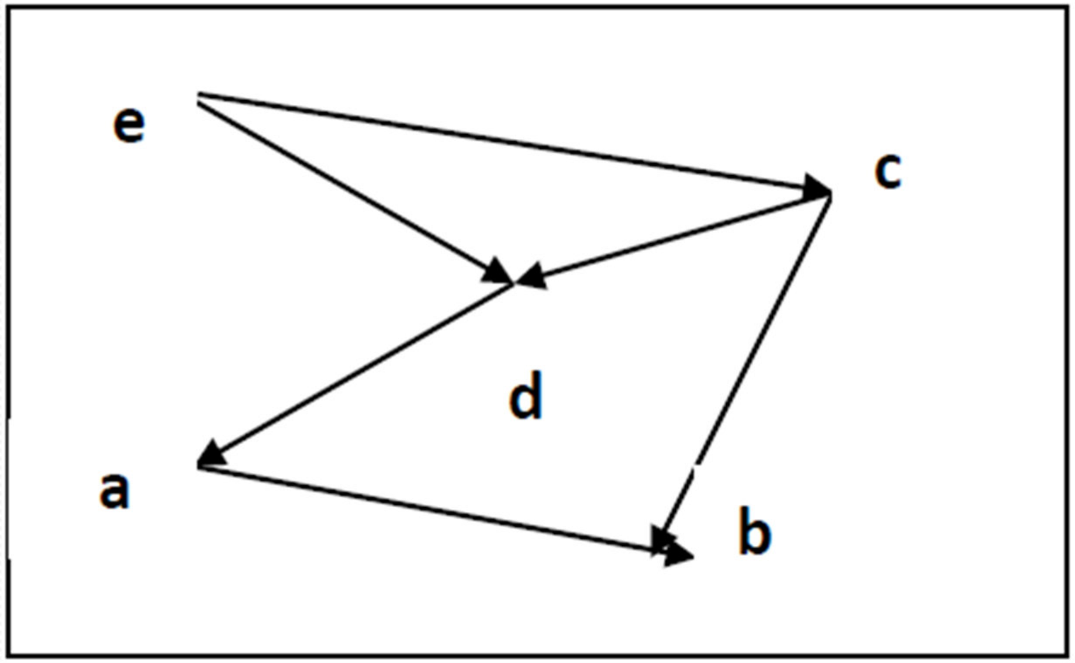
- The theory of state space search is our primary tool for answering these questions, by representing a problem as state space graph, we can use graph theory to analyze the structure and complexity of both the problem and procedures used to solve it.

Graph Theory:-

- A graph consists of a set of a nodes and a set of arcs or links connecting pairs of nodes. The domain of state space search, the nodes are interpreted to be stated in problem solving process, and the arcs are taken to be transitions between states.

Introduction: Search Algorithms

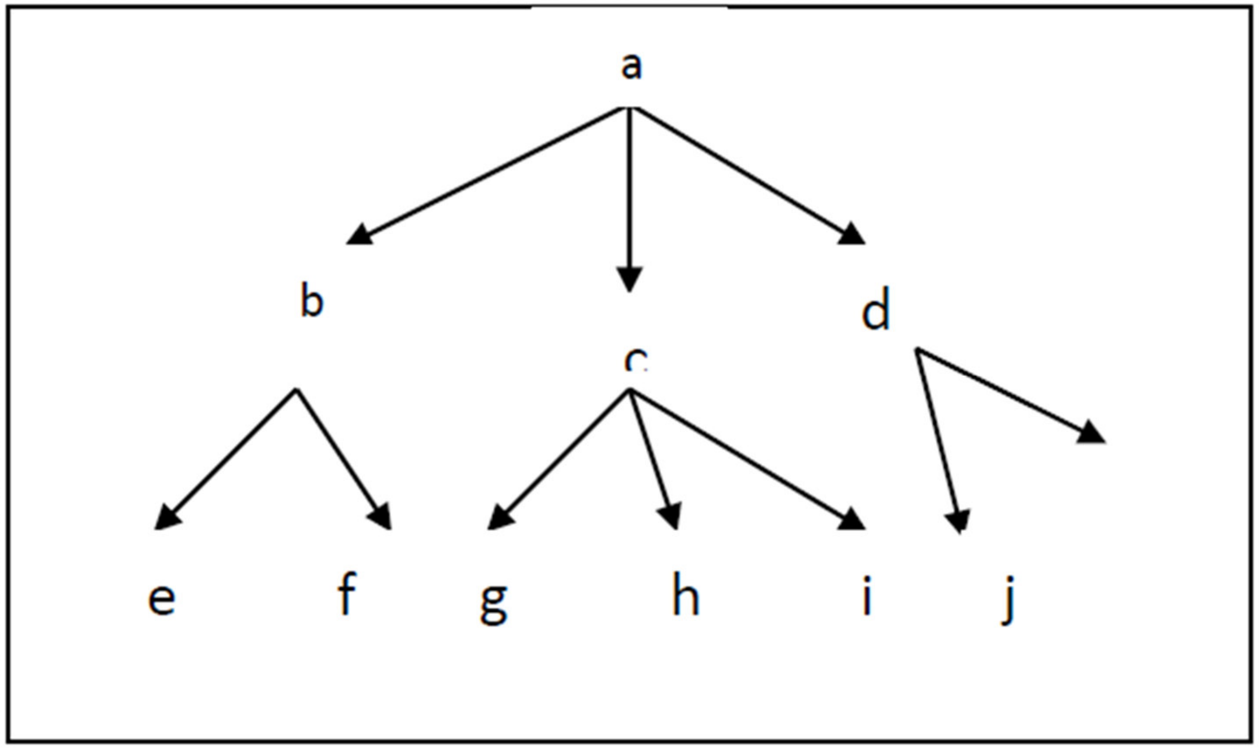
Graph theory is our best tool for reasoning about the structure of objects and relations:



Nodes={a,b,c,d,e}

Arcs={(a,b), (a,d),(b,c),(c,b),(d,e),(e,c),(e,d)}

Introduction: Search Algorithms



Nodes=={a,b,c,d,e,f,g,h,i}

Arcs={(a,b),(a,c),(a,d),(b,e),(b,f),(c,f),(c,g),(c,h),(c,i),(d,j)}

Introduction: Search Algorithms

State Space Representation of Problems:-

A state space: is represented by four items [N,A,S,G], where:-

- **N** is a set of nodes or states of the graph. These correspond to the states in a problem –solving process.
- **A** is the set of arcs between the nodes. These correspond to the steps in a problem –solving process.
- **S** a nonempty subset of N , contains the start state of the problem.
- **G** a nonempty subset of N contains the goal state of the problem.

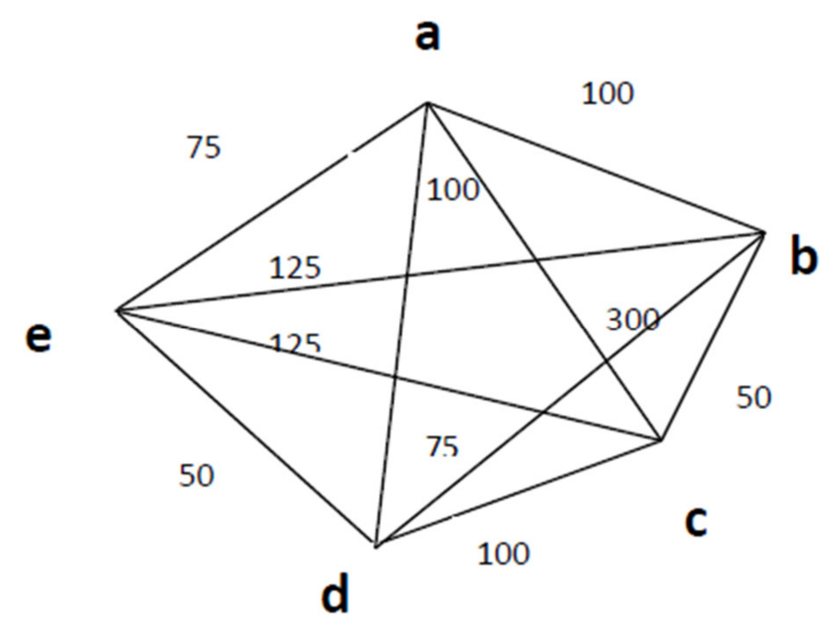
Introduction: Search Algorithms

A solution path:- Is a path through this graph from a node S to a node in G.

Example:- Traveling Salesman Problem

- Starting at A , find the shortest path through all the cities , visiting each city exactly once returning to A.

Introduction: Search Algorithms



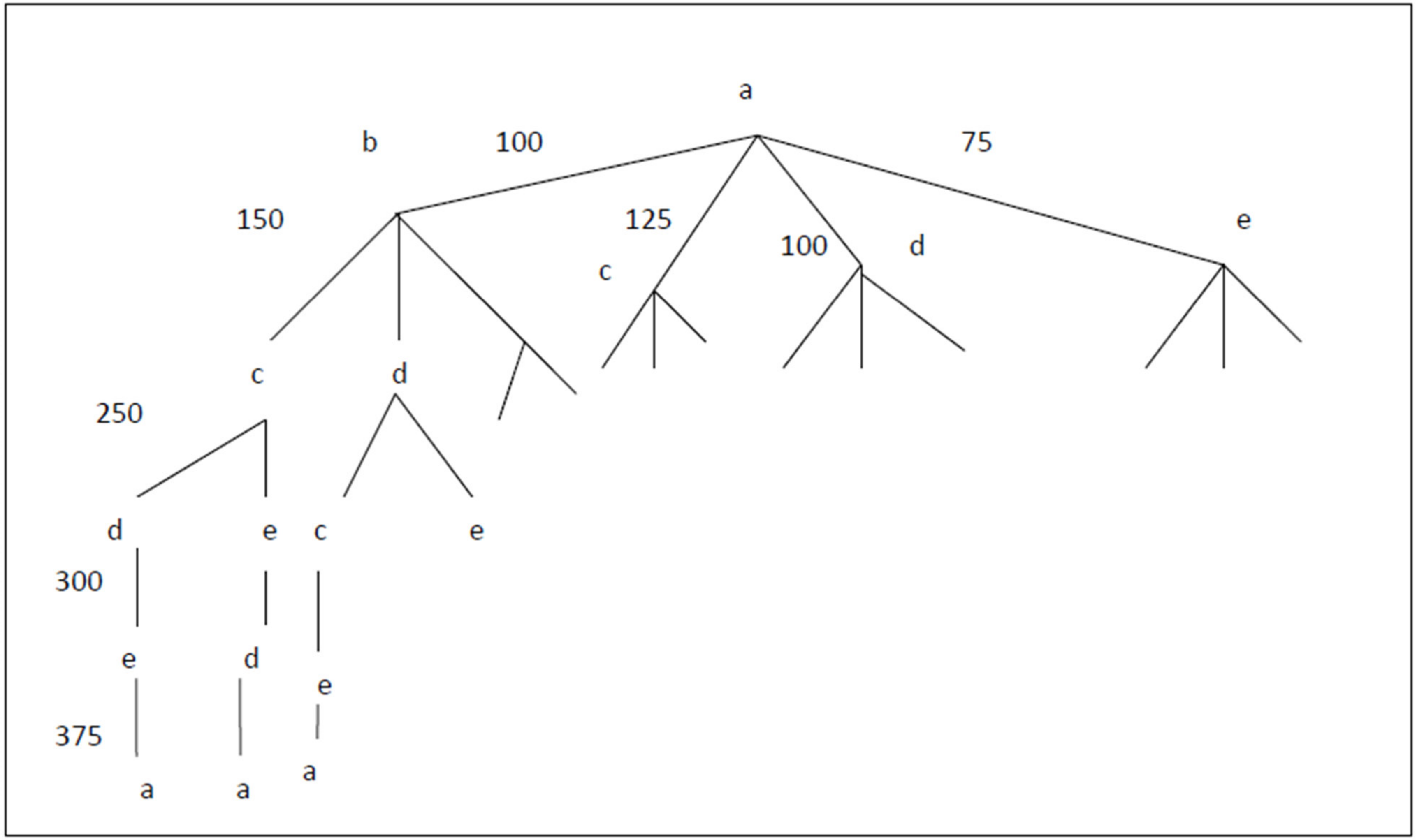
- *“An instance of traveling Salesman Problem”*

Introduction: Search Algorithms

- The complexity of exhaustive search in the traveling Salesman is **(N-1)!**, where N is the No. of cities in the graph. There are several technique that reduce the search complexity.

1. **Branch and Bound Algorithm**:-Generate one path at a time, keeping track of the best circuit so far. Use the best circuit so far as a bound of future branches of the search. Figure below illustrate branch and bound algorithm.

Introduction: Search Algorithms

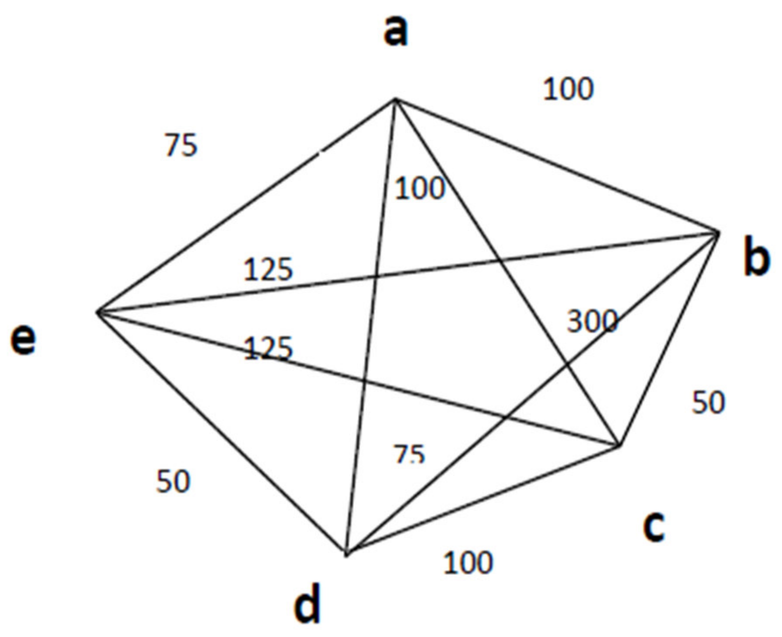


a b c d e a=375   a b c e d a =425   a b d c e a=474 ....

Introduction: Search Algorithms

2. **Nearest Neighbor Heuristic:** At each stage of the circuit, go to the nearest unvisited city. This strategy reduces the complexity to N, so it highly efficient, but it is not guaranteed to find the shortest path, as the following example:

- Cost of Nearest neighbor path is **a e d b c a=550**
- Is not the shortest path , the comparatively high cost of arc (C,A) defeated the heuristic.



Artificial Intelligence

Lecture 4: Search Algorithms

Instructor : MSc. Hazim N. Abed

Introduction: Search Algorithms

- In general the Search Techniques or Algorithms are classified into **Uniformed Search** or **Blind Search** and **Informed Search** or **Heuristic Search.**

Uniformed Search	Informed Search
No information about the number of steps or path cost from the current state to goal state	The path cost from the current state is calculated, to select the minimum path cost as the next state
Less effective in search method	More effective
Problem to be solved with the given information	Additional information can be added as assumption to solve the problem
Depth First Search, Breadth First Search	Hill Climbing, Best First Search, A* Search

Uniformed Search

1. Uninformed Search (Blind Search)

- In this search, we generate a potential solution and then check it against the solution. If we’ve found the solution, we’re done, otherwise, we repeat by trying another potential solution. This is called “Generate and Test” because we generate a potential solution, and then test it. Without a proper solution, we try again.

Uniformed Search

<u>Procedure Generate & Test Algorithm</u>

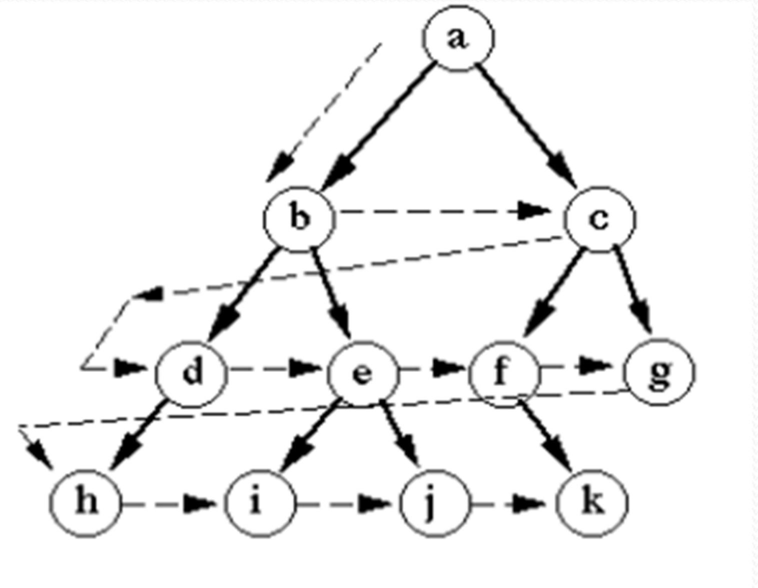
```
Begin
Repeat
Generate a new state and call it current-state;
Until current-state = Goal;
End.
```

- This type of search takes all nodes of tree in specific order until it reaches to goal. The order can be in breath and the strategy will be called breadth – first – search, or in depth and the strategy will be called depth first search.

Uniformed Search

A-Breadth – First – Search

- In breadth –first search, when a state is examined, all of its siblings are examined before any of its children. The space is searched level-by-level, proceeding all the way across one level before doing down to the next level.



Breadth-first search

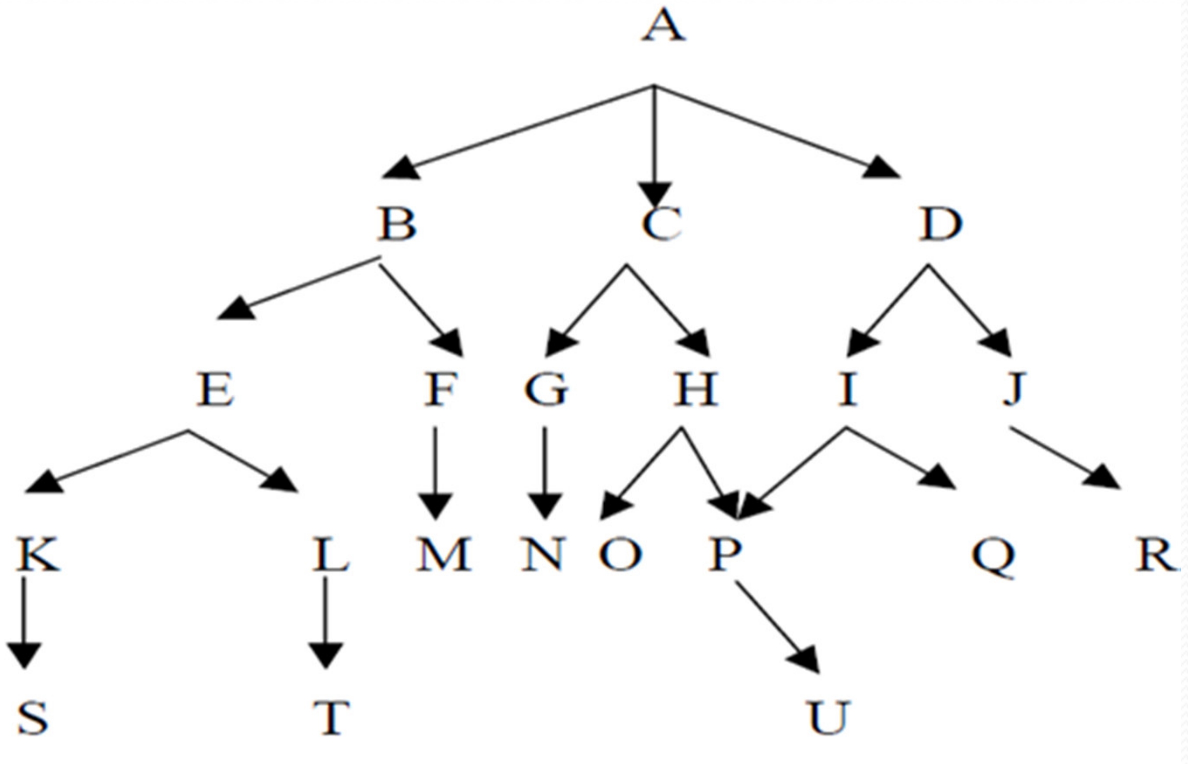
Uniformed Search

Breadth – first – search Algorithm

```
Begin
Open: = [start];        Closed: = [ ];
While open ≠ [ ] do
Begin
Remove left most state from open, call it x;
If x is a goal the return (success)
Else Begin
Generate children of x;
Put x on closed;
Eliminate children of x on open or closed;
Put remaining children on right end of open
End
End
Return (failure)
End.
```

Uniformed Search

- **Example:** Assume **M** is the Goal State



Breadth – first – search

Uniformed Search

Iteration	Current State	Open	Closed
1	A	[A]	[ ]
2	B	[B, C, D]	[A]
3	C	[C, D, E,F]	[B,A]
4	D	[D, E, F, G, H]	[C,B,A]
5	E	[E, F, G, H, I, J]	[D,C,B,A]
6	F	[F,G,H, I, J, K, L]	[E,D,C,B,A]
7	G	[G,H, I, J, K, L, M]	[F,E,D,C,B,A]
8	H	[H, I, J, K, L, M, N]	[G,F,E,D,C,B,A]
9	I	[I, J, K, L, M, N,O,P]	[H,G,F,E,D,C,B,A]
10	J	[J, K, L, M, N,O,P,Q]	[I,H,G,F,E,D,C,B,A]
11	K	[K, L, M, N,O,P,Q, R]	[J,I,H,G,F,E,D,C,B,A]
12	L	[L, M, N,O,P,Q, R, S]	[K,J,I,H,G,F,E,D,C,B,A]
13	M	[M, N,O,P,Q, R, S,T]	[L,K,J,I,H,G,F,E,D,C,B,A]

Uniformed Search

Advantages:-

- Guaranteed to find the single solution at the shallowest depth level.

Disadvantages:-

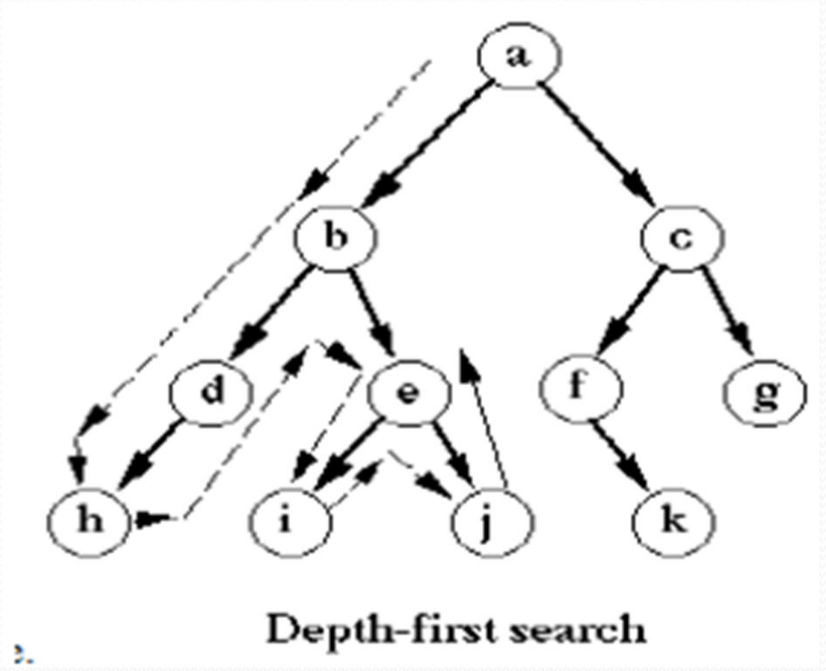
- The memory requirements are a bigger problem for BFS than is the execution time.

- Exponential-complexity search problems cannot be solved by uniformed methods for any but only suitable for smallest instances problem (i.e) (number of levels to be minimum (or) branching factor to be minimum).

Uniformed Search

B- Depth – first – search

In depth – first – search, when a state is examined, all of its children and their descendants are examined before any of its siblings. Depth – first search goes deeper in to the search space whenever this is possible only when no further descendants of a state cam found owe its.



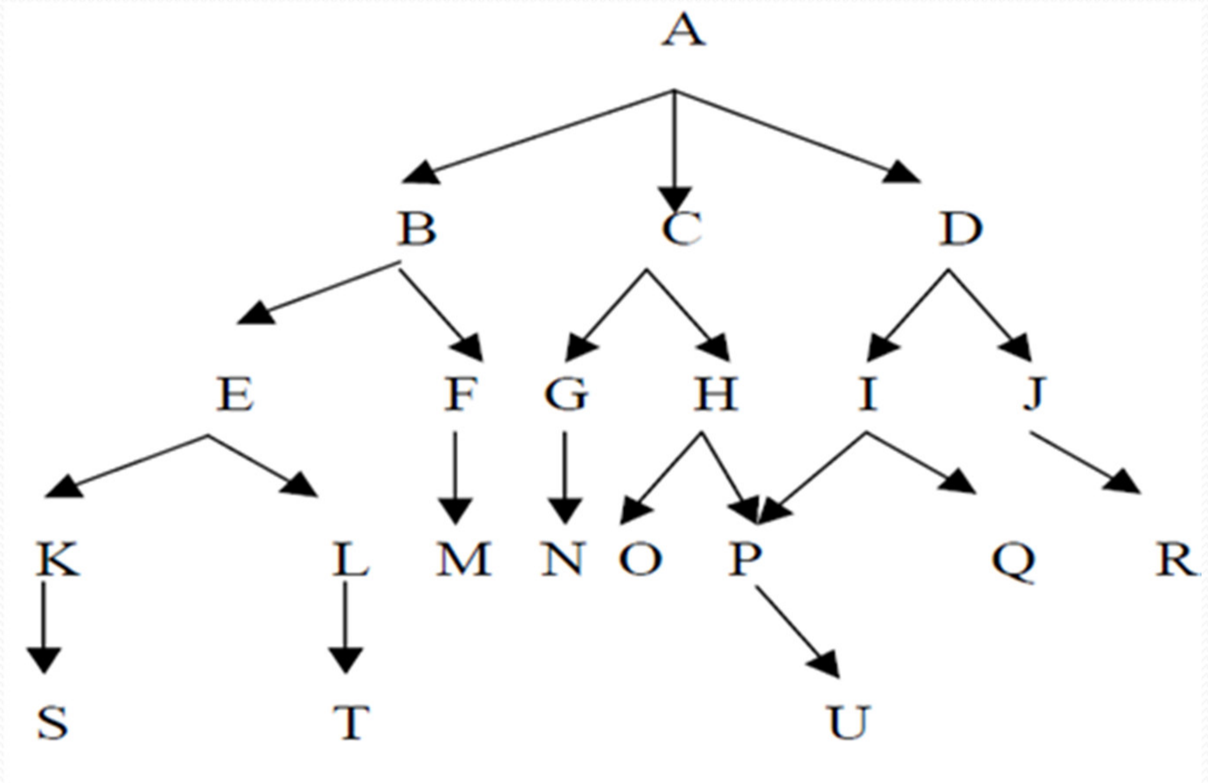
Uniformed Search

Depth – first – search Algorithm

```
Begin
Open: = [start]; Closed: = [ ];
While open ≠ [ ] do
Remove leftmost state from open, call it x;
If x is a goal then return (success)
Else begin
Generate children of x;
Put x on closed;
Eliminate children of x on open or closed; put remaining children on left end
of open end
End;
Return (failure)
End.
```

Uniformed Search

- **Example:** Assume **M** is the Goal State



Breadth – first – search

Uniformed Search

Iteration	Current State	Open	Closed
1	A	[A]	[ ]
2	B	[B,C,D]	[A]
3	E	[E,F,C,D]	[B,A]
4	K	[K,L,F,C,D]	[E,B,A]
5	S	[S,L,F,C,D]	[K,E,B,A]
6	L	[L,F,C,D]	[S,K,E,B,A]
7	T	[T,F,C,D]	[L,S,K,E,B,A]
8	F	[F,C,D]	[T,L,S,K,E,B,A]
9	M	[M,C,D]	[F,T,L,S,K,E,B,A]

Uniformed Search

Advantages:

- If more than one solution exists (or) number of levels is high then DFS is best because exploration is done in a small portion of the whole space.

Disadvantages:

- Not guaranteed to find a solution

Artificial Intelligence

Lecture 5
Informed Search (Heuristic Search)

Instructor : MSc. Hazim N. Abed

Informed Search (Heuristic Search)

2- Informed Search (Heuristic Search)

- A heuristic is a method that might not always find the best solution but is guaranteed to find a good solution in reasonable time. By sacrificing completeness it increases efficiency. Heuristic search is useful in solving problems which:-

  ✓Could not be solved any other way.

  ✓Solution takes an infinite time or very long time to compute.

Informed Search (Heuristic Search)

- Heuristic search methods generate and test algorithms, from these methods are:-

1- Hill Climbing.

2- Best-First Search.

3- A and A* algorithm.

Informed Search (Heuristic Search)

1- Hill Climbing.

- The idea here is that, you don’t keep the big list of states around you just keep track of the one state you are considering, and the path that got you there from the initial state. At every state you choose the state leads you closer to the goal (according to the heuristic estimate), and continue from there.

Informed Search (Heuristic Search)

- The name “Hill Climbing” comes from the idea that you are trying to find the top of a hill, and you go in the direction that is up from wherever you are. This technique often works, but since it only uses local information.

Informed Search (Heuristic Search)

- Hill Climbing Algorithm

```
Begin
Cs=start state;  Open=[start]; Stop=false;
Path=[start];
While (not stop) do
{
if (cs=goal) then
return (path);
generate all children of cs and put it into open
```

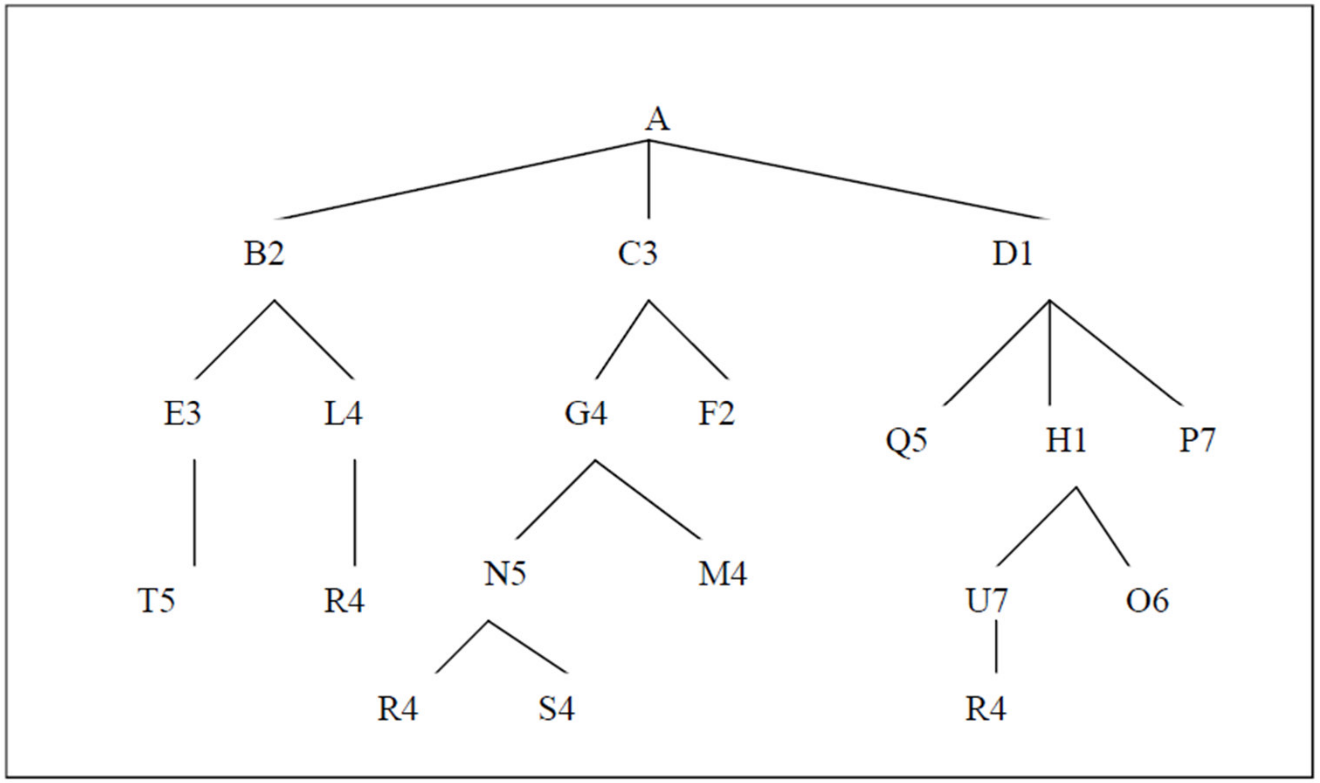
Informed Search (Heuristic Search)

```
if (open=[]) then
stop=true
else
{
x:= cs;
for each state in open do
{
compute the heuristic value of y (h(y));
if y is better than x then
x=y
}
if x is better than cs then
cs=x
else
stop =true;
}
}
return failure;
}
```

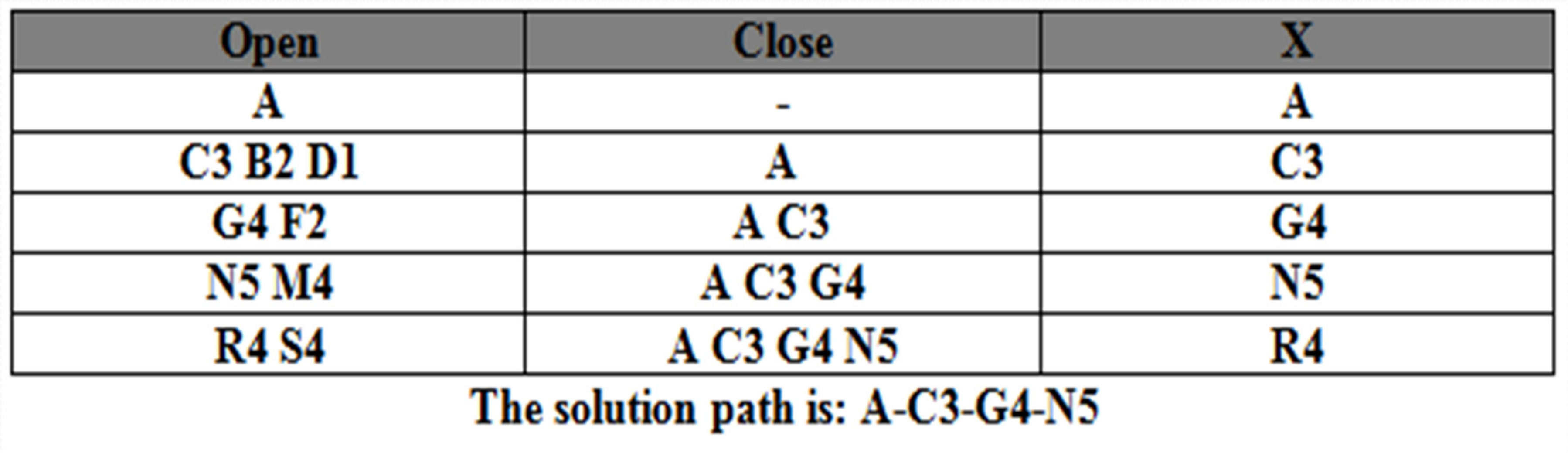
Informed Search (Heuristic Search)

Example:

- A trace of hill climbing searches for R4 of Figure below:



Informed Search (Heuristic Search)



Open	Close	X
A	-	A
C3 B2 D1	A	C3
G4 F2	A C3	G4
N5 M4	A C3 G4	N5
R4 S4	A C3 G4 N5	R4

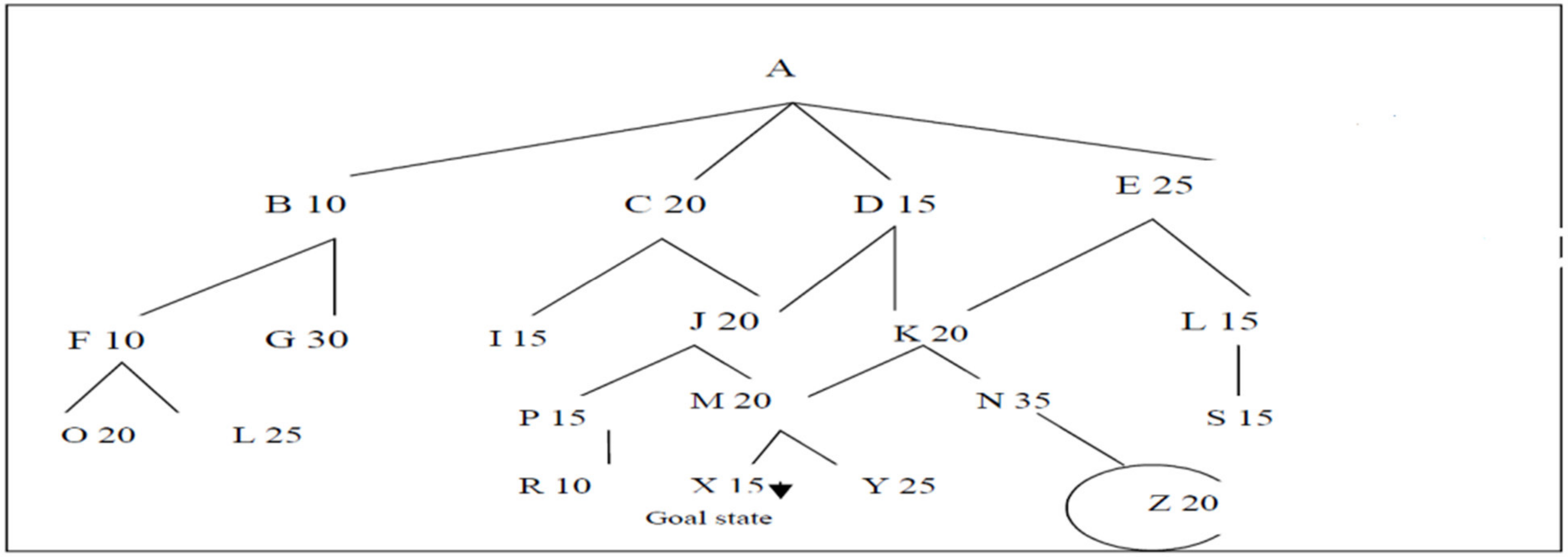
The solution path is: A-C3-G4-N5

Informed Search (Heuristic Search)

Hill climbing Problems:-

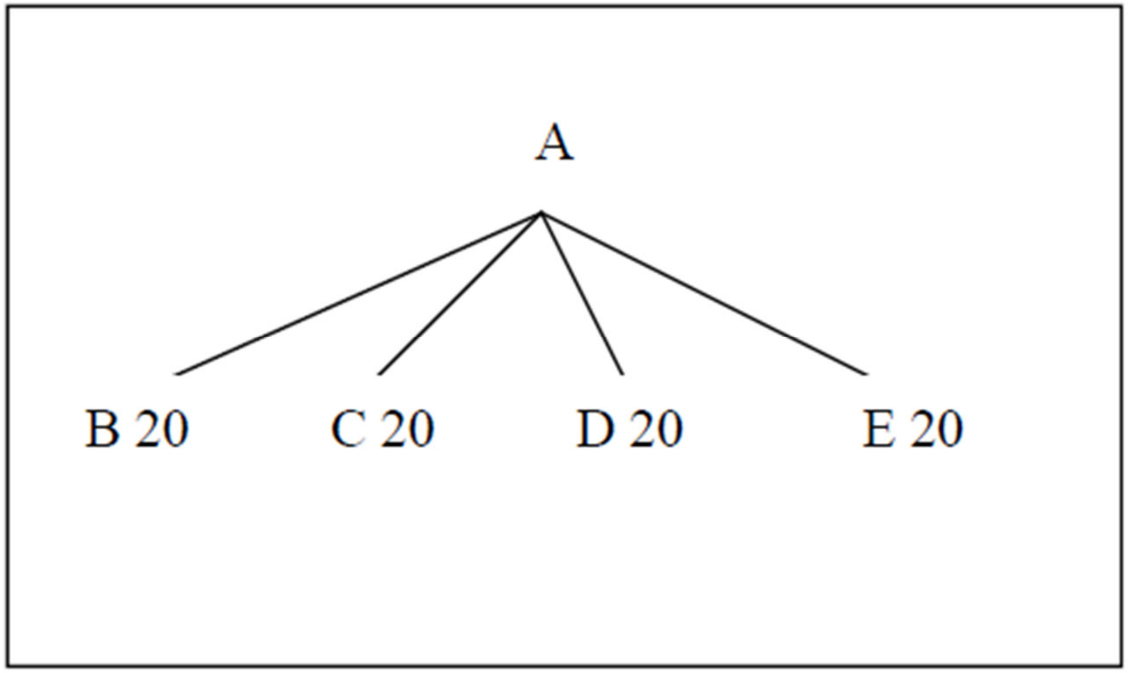
Hill climbing may fail due to one or more of the following reasons:-

1- a local maxima: Is a state that is better than all of its neighbors but is not better than some other states.



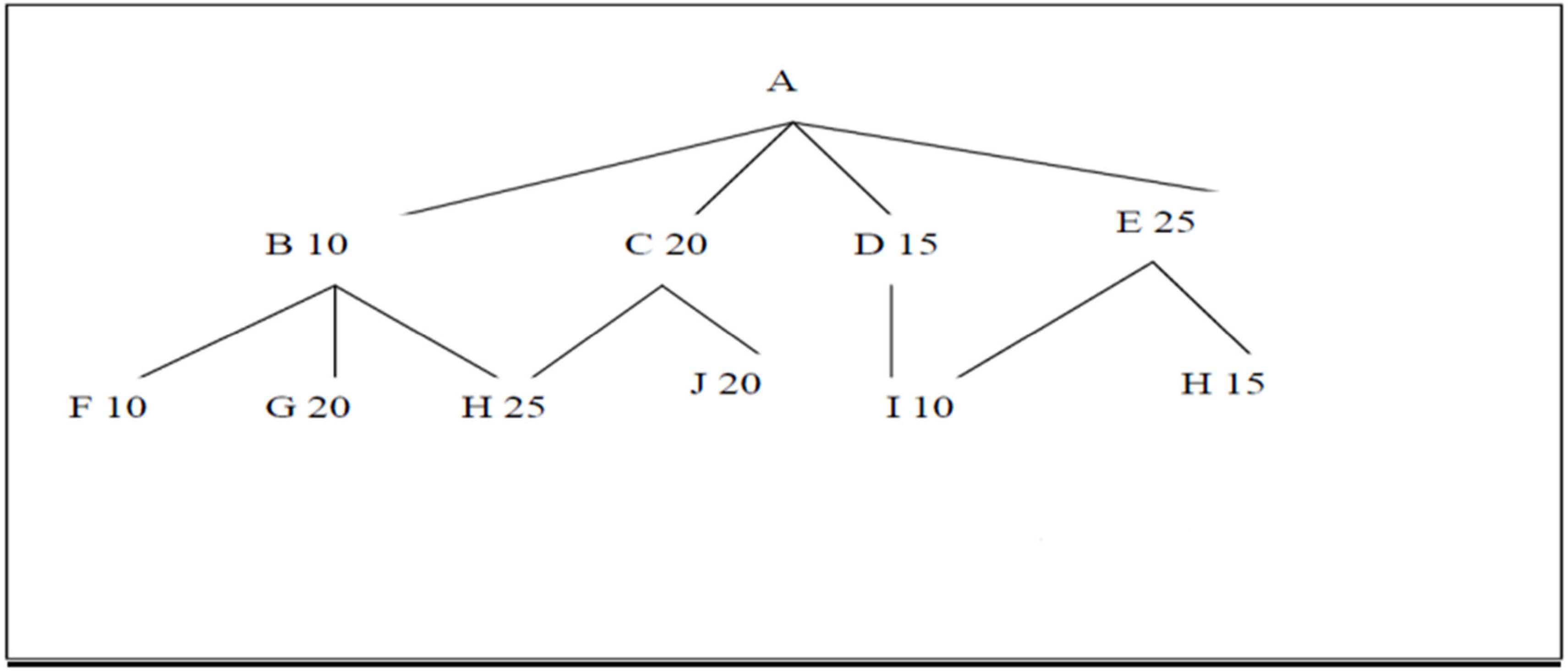
Informed Search (Heuristic Search)

2- A Plateau: Is a flat area of the search space in which a number of states have the same best value, on plateau it's not possible to determine the best direction in which to move.



Informed Search (Heuristic Search)

3- A ridge: Is an area of the search space that is higher than surrounding areas, but that cannot be traversed by a single move in any one direction.



Informed Search (Heuristic Search)

2- Best-First-Search

- Best First search is away of combining the advantages of both depth-first and breadth-first search into a single method. The actual operation of the algorithm is very simple. It proceeds in steps, expanding one node at each step, until it generates a node that corresponds to a goal state. At each step, it picks the most promising of the nodes that have so far been generated but not expanded.

Informed Search (Heuristic Search)

- It generates the successors of the chosen node, applies the heuristic function to them, and adds them to the list of open nodes, after checking to see if any of them have been generated before. By doing this check, we can guarantee that each node only appears once in the graph, although many nodes may point to it as a successors. Then the next step begins.
- In Best-First search, the search space is evaluated according to a heuristic function. Nodes yet to be evaluated are kept on an OPEN list and those that have already been evaluated are stored on a CLOSED list.

Informed Search (Heuristic Search)

- The OPEN list is represented as a priority queue, such that unvisited nodes can be queued in order of their evaluation function. The evaluation function f(n) is made from only the heuristic function (h(n)) as: $f(n) = h(n)$.

<u>Best-First-Search Algorithm</u>

```
{
Open:=[start];
Closed:=[];
While open ≠ [] do
{
Remove the leftmost from open, call it x;
```

Informed Search (Heuristic Search)

```
If x= goal then
Return the path from start to x
Else
{
Generate children of x;
For each child of x do
Do case
The child is not already on open or closed;
{
 assign a heuristic value to the child state ;
Add the child state to open;
}
```

Informed Search (Heuristic Search)

```
The child is already on open:
If the child was reached along a shorter path than the state currently on open then give the state on open this shorter path value.
The child is already on closed:
If the child was reached along a shorter path than the state currently on open then
{
Give the state on closed this shorter path value
Move this state from closed to open
}
}
Put x on closed;
Re-order state on open according to heuristic (best value first)
}
Return (failure);
}
```

Informed Search (Heuristic Search)

Example:

Informed Search (Heuristic Search)

Iteration	Current State	Open	Closed
1	A	[A5]	[]
2	D	[D3,B4,C5]	[A5]
3	C	[C2, B4, I5]	[A5,D3]
4	F	[F3,B4,I5]	[A5,D3,C2]
5	B	[B4,I5]	[A5,D3,C2,F3]
6	C	[C1,E3,I5]	[A5,D3,C2,F3,B4]
7	E	[E3, I5]	[A5,D3,F3,B4,C1]
8	G	[G0,I5]	[A5,D3,F3,B4,C1,E3]
		[I5]	[A5,D3,F3,B4,C1,E3,G0]
The goal is found and the Path is = A5,D3,F3,B4,C1,E3,G0			

A-Star search algorithm

3- A Star search algorithm

- A* algorithm is simply define as a best first search plus **specific function**. This **specific function** represents the actual distance (levels) between the current state and the goal state and is denoted by **f(n)**. It evaluates nodes by combining **g(n)**, the cost to reach the node, and **h(n)**, the cost to get from the node to the goal:

$$\mathbf{f(n) = g(n) + h(n).}$$

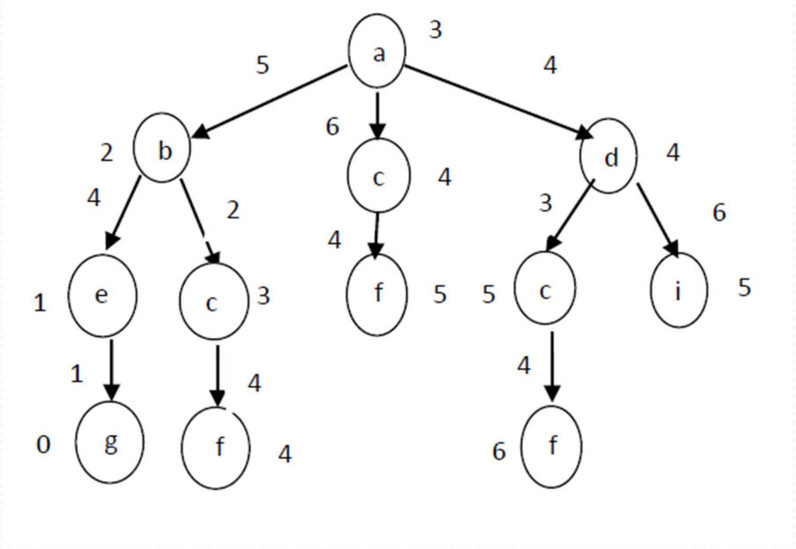
A-Star search algorithm

- Since g(n) gives the path cost from the start node to node n, and h(n) is the estimated cost of the cheapest path from n to the goal, we have f (n) = estimated cost of the cheapest solution through n.

- Thus, if we are trying to find the cheapest solution, a reasonable thing to try first is the node with the lowest value of g(n) + h(n). It turns out that this strategy is more than just reasonable: provided that the heuristic function h(n) satisfies certain conditions, A* search is both complete and optimal.

A-Star search algorithm

Example:



Iteration	Current State	Open	Closed
1	A	[A3]	[]
2	B	[B7,D8,C10]	[A3]
3	D	[D8,E10,C10]	[A3,B7]
4	E	[E10,C10,I15]	[A3,B7,D8]
5	G	[G10,C10,I15]	[A3,B7,D8,E10]
6		The path is =	[A3,B7,D8,E10,G10]

Behavior of Algorithms

1. **Completeness**: An algorithm is said to be complete, if it terminates with a solution, when one exists.
2. **Admissibility**: An algorithm is called admissible if it is guaranteed to return an optimal solution, whenever a solution exists.
3. **Dominance:** An algorithm A1 is said to dominate A2, if every node expanded by A1 is also expanded by A2.
4. **Optimality:** an algorithm is said to be optimal over a class of algorithms, if it dominates all members of the class.

Behavior of Algorithms

- For example, breadth first search is an **admissible** search strategy, because it look at every state at level n of the graph before considering any state at level n+1. So, goal nodes are found along the shortest possible path. Breadth first search is an A* algorithm in which F[n]=g[n]+0. In other words, breadth-first search uses a trivial estimate of the distance to the goal.

Measuring problem-solving performance

- The output of a problem solving is either failure or a solution when will evaluate an algorithm's performance in four ways:

  A. Completeness: The strategy guaranteed to find a solution when there is one.

  B. Optimality: If more than one way exists to derive the solution then then the best one is selected.

  C. Time complexity: Time taken to run a solution.

  D. Space complexity: Memory needed to perform the search.

Artificial Intelligence

Lecture 6
Problem Solving (Games)
Instructor : MSc. Hazim N. Abed

Problem Definition

- In order to solve the problem play a game, which is restricted to two person table or board games, a topic which was fully discussed in the last year's course, we require the rules of the game and the targets for winning as well as a means of representing positions in the game.
- The opening position can be defined as the **initial state** and a winning position as a **goal state**, there can be more than one. Legal moves allow for transfer from initial state to other states leading to the goal state.

Problem Definition

- However the rules are far to copious in most games especially chess where they exceed the number of particles in the universe.
- Thus the rules cannot in general be supplied accurately and computer programs cannot easily handle them. The storage also presents another problem but searching can be achieved by hashing.
- The number of rules that are used must be minimized and the set can be produced by expressing each rule in as general a form as possible.

Problem Definition

- The representation of games in this way leads to a state space representation and it is natural for well organized games with some structure. This representation allows for the formal definition of a problem which necessitates the movement from a set of initial positions to one of a set of target positions.

- It means that the solution involves using **known techniques** and a **systematic search**. This is quite a common method in AI.

Problem Definition

Well organized problems (e.g. games) can be described as a set of rules.

- Rules can be generalized and represented as a state space representation:
    - ✓ Formal definition.
    - ✓ Move from initial states to one of a set of target positions.
    - ✓ Move is achieved via a systematic search.

Problem space and search

To build a system to solve a practical problem, need to do four thinks:

1. Define the problem precisely.

- This define must include specifications of what the initial state(S) we well be as well as final state (F) conditions to acceptable solution to the problem.

Analyze the problem.

2. Find the optimal techniques for solving the problem.

3. Isolate and represent the task knowledge that is necessary to solve the problem.

4. Choose the best problem-solving techniques and apply it to practical problem.

Problem space and search

- **That means an problem in A.I must consist of features:**

1. Two point (**Initial** and **Goal State**).
2. Final path between the **start state** and the **goal state**.
3. Problem state space.
4. To solve the problem in A.I means to search about the optimal solution between **START** and **GOAL** state.

Define the problem and or as a space (Start Space)

- **Define the problem as a start space search.**

A-By using Problem Characteristics:

- Is the problem decomposable into a set of (nearly) independence smaller or easier sub problem?
- Can solution steps be ignored or at last undone if they prove unwise?
- Is the problems universe predictable?
- Is a good solution to the obvious comparison to all other possible solution?
- Is the described solution a state of the world or a path to a state?

Define the problem and or as a space (Start Space)

- **Problem Characteristics: ALGORITHM**

  Step (1): Decomposition a problem into a set of smaller problem.

  Step (2): Repeat the Step (1) into a set of problem or Step (3).

  Step (3): Ignored at last undone if they prove unwise.

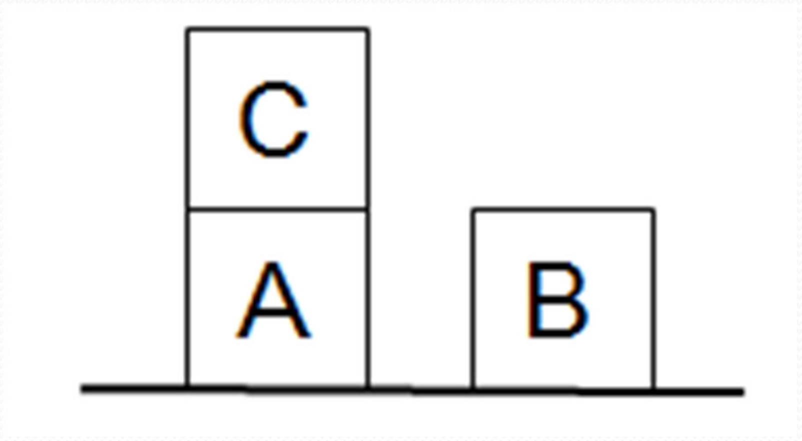
  Step (4): Problem universe predictable.

  Step (5): Find a good solution to the problem.

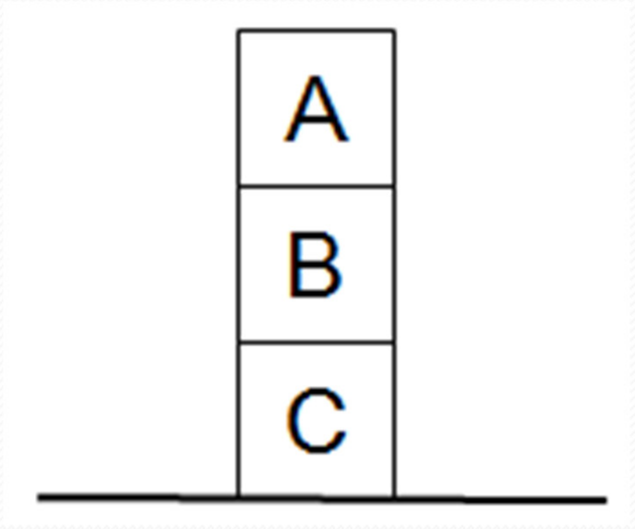
  Step (6): Described the solution a state of the world or a path to a state.

Define the problem and or as a space (Start Space)

Problem (1): Decomposition the problem and find the problem state space?

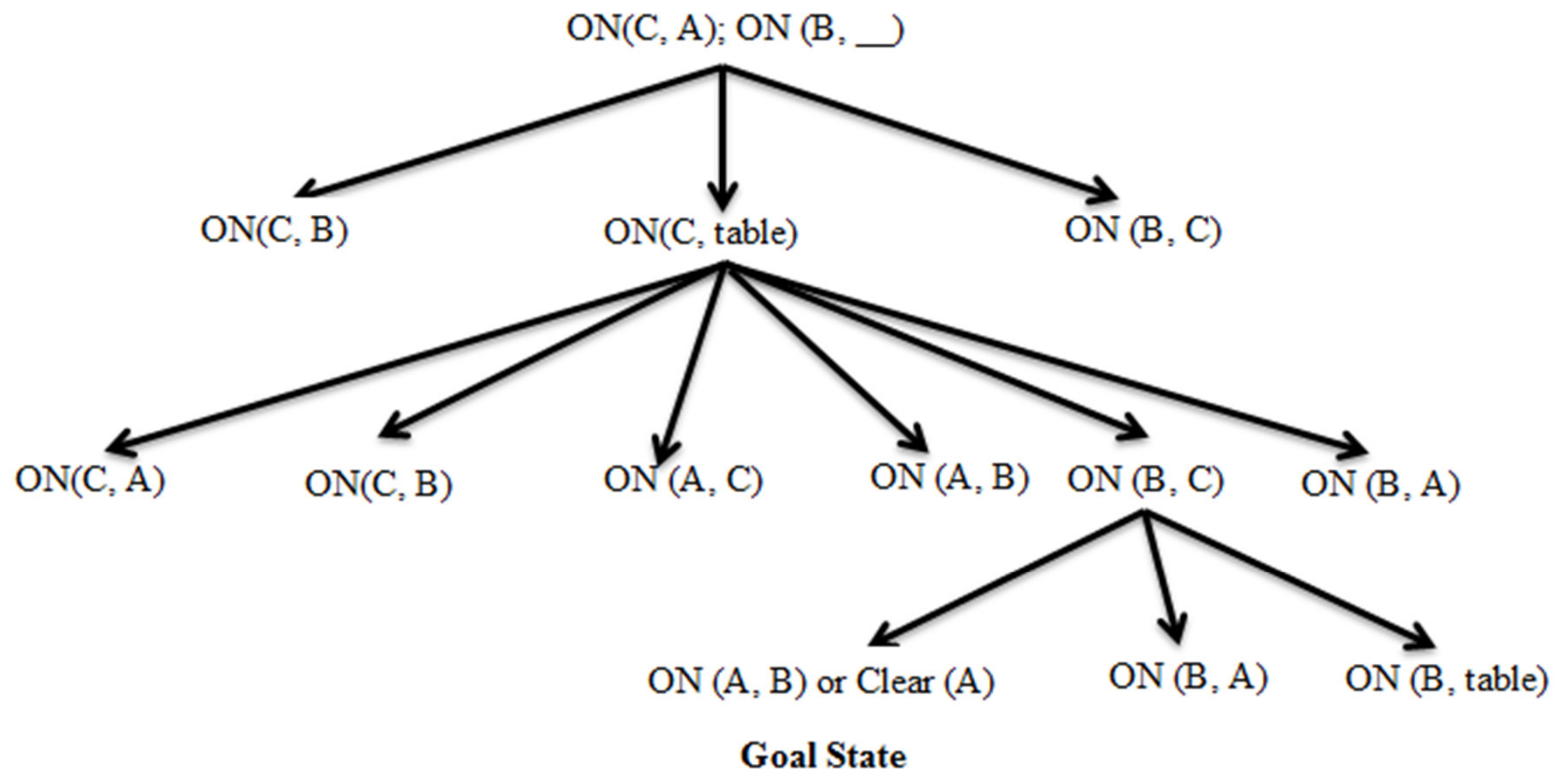


Initial State:
ON(C,A); ON(B,__) or ON(B, Table)



Goal State:
ON(B,C) and ON(A,B)

Define the problem and or as a space (Start Space)



Define the problem and or as a space (Start Space)

B-By using Problem Solving:

- In this section we can use the knowledge base by using the set of steps that must be use this to find the solve the problem, then we can found a multi problems that indicates a sum of steps that use to find the solve of these problem.

- Knowledge base. Find the INITIAL and GOAL (Final State). Of the problem. Then the generat algorithm that uses to solve the problem in A.I is the knowledge base.

Define the problem and or as a space (Start Space)

Problem State Space Search: ALGORITHM

Step (1): Find all the PARAMETERS parameter in the problem.

Step (2): State all the VALUE of these parameters**.**

Step (3): State the problem

1- INITIAL state.

2- GOAL state

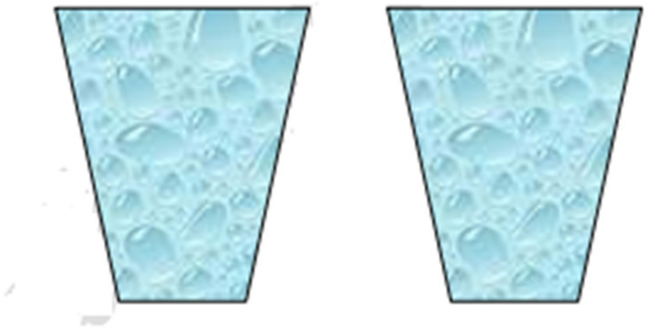
Step (4): Find the entire RULE to solve these problem.

Step (5): Find the TREE that represent these solution**.**

Step (6): Described the solution a state of the world or a path to a state.

Define the problem and or as a space (Start Space)

Problem (2): You have two empty Jugs a 4-litre one and a 3-litre one. You are asked to provide a Jug with a specified a mount of water, the Jugs have no "grading marks" and the only think you can do are fill to Jugs form the top, empty the Jugs into the sink or pour the water form one Jug to the another. Find the problem state space by using problem solving.



Jug 1 of 4 liters   Jug 2 of 3 liters

Define the problem and or as a space (Start Space)

Solve:

- The capacities of each Jug are:

  Jug is full on 4 liters.

  Jug is full on 3 liters

Step (1): The parameters of the problem are: X and Y.

That mean:

X is the Jug 1.

Y is the Jug 2.

Step (2): The value of the parameters is:

X=0 for " Empty Jug ".

X=1,2,3 and 4 for " Full Jug ".

Define the problem and or as a space (Start Space)

Y=0 for " Empty Jug ".

Y=1, 2 and for " Full Jug ".

Step (3): The Initial state for each parameters

Initial state: if empty Jug (0, 0) or Full Jug (4, 3)

Goal state: (2,y) or (x,2)

Step (4): The Rule of the problem.

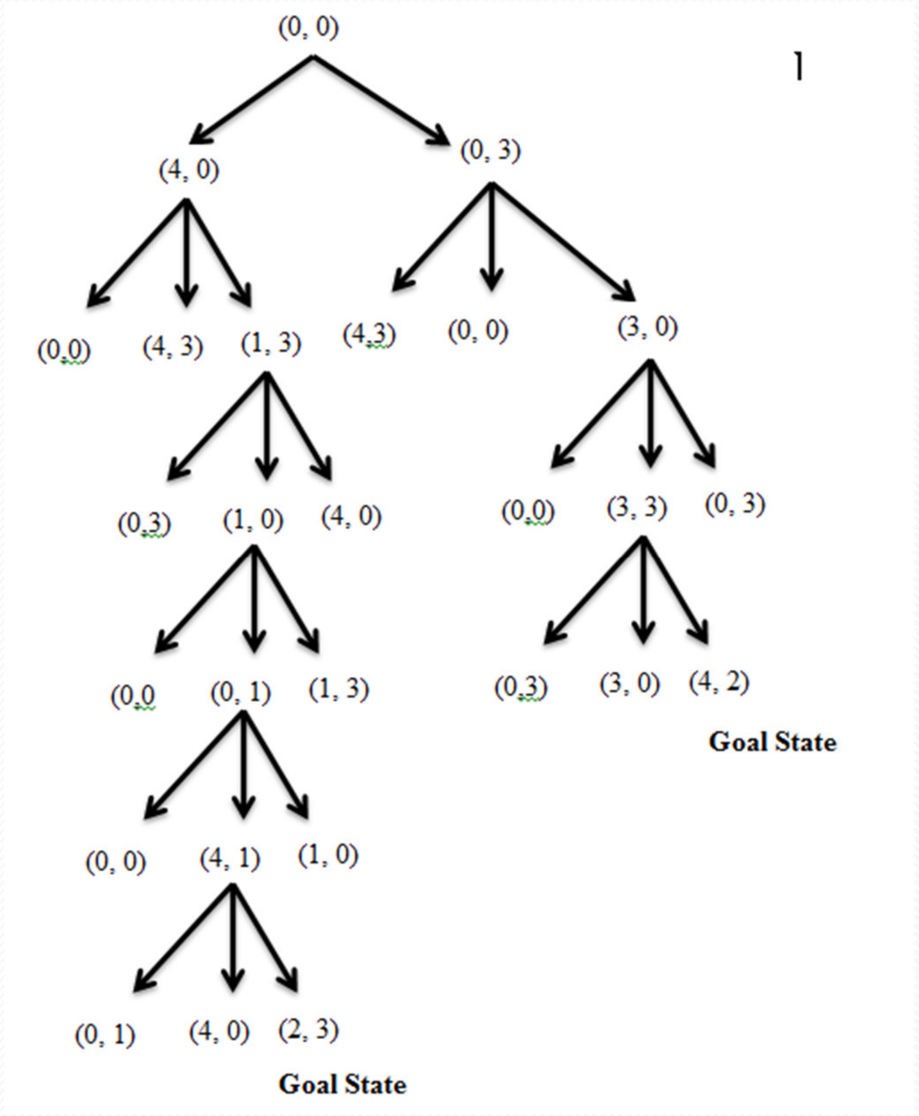
1. (x,y) →(4, y) full x
2. (x,y) →(x, 3) full y
3. (x,y) → (0, y) empty x Jug
4. (x,y) → (x,0) empty y Jag

Define the problem and or as a space (Start Space)

5. (x, y) →(x, y-(4-x)) pour water from y into x until x is full
6. (x, y) →(x-(3-y), y) pour water from x into y until y is full
7. (x, y) → (x+y, 0) pour all y Jug to x Jug
8. (x, y) → (0, y+x) pour all x Jug to y Jug
9. (0,2) → (2,0) pour 2 liters from y to x
10. (2,0) → (0,2) pour 2 liters from x to y

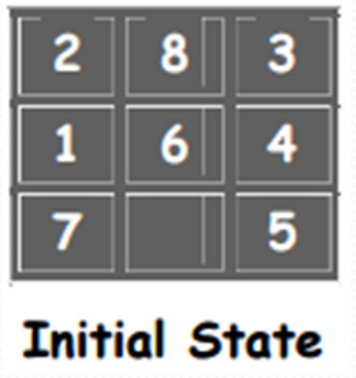
Step (5): the tree of the solution

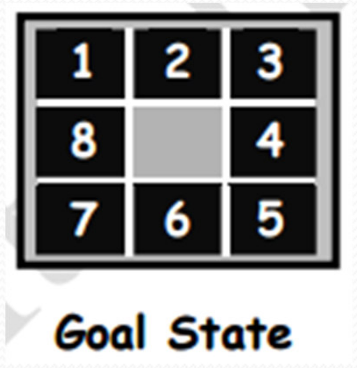
Define the problem and or as a space (Start Space)



Define the problem and or as a space (Start Space)

Problem (3): solve the 8-puzzle problem





Solve:

Step (1): The parameters of the problem are: X1, X2, X3… X9.

Step (2): The value of the parameters is:

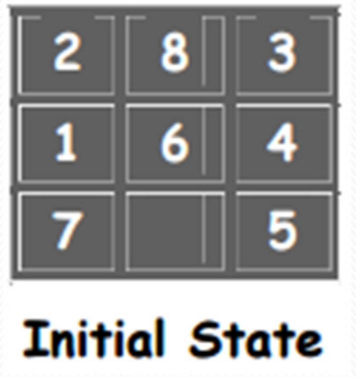
X1=0 or 1.

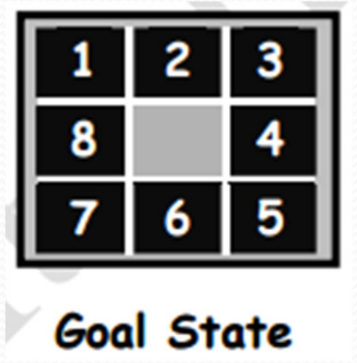
1 that means X=1 if X is translated or moving.

0 that means X=0 if X is not translates or not moving

Define the problem and or as a space (Start Space)

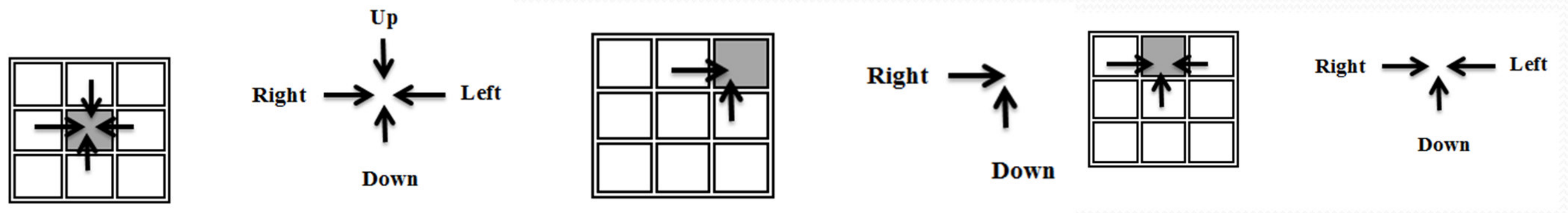
Step (3): The **Initial state** and **Goal state**





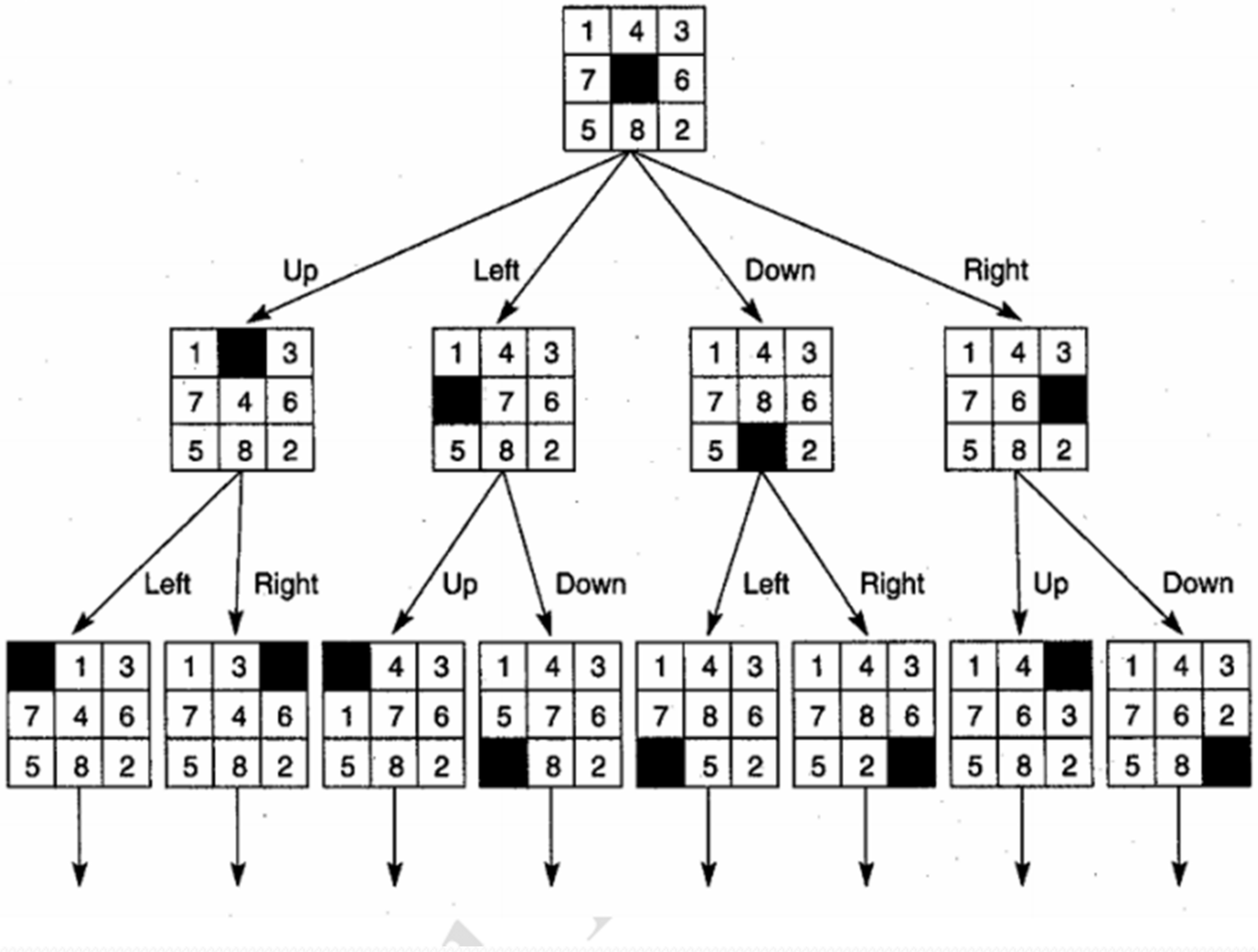
Step (4): The Rule of the problem solve in the first translation is (X, Y, Z).

Rule : move blank left, right, up, down

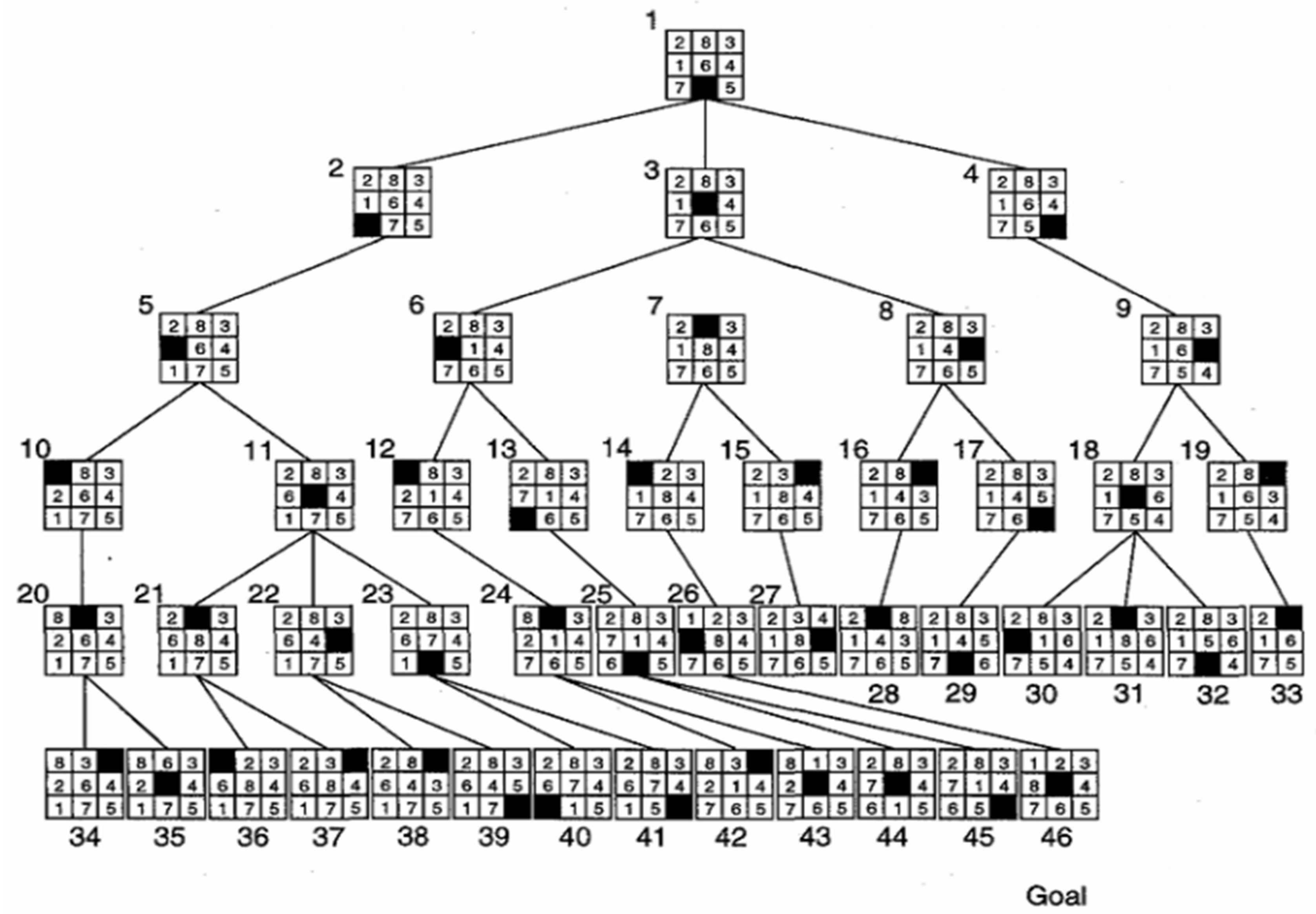


Define the problem and or as a space (Start Space)

Step (5): The **TREE** of the solution is:



Define the problem and or as a space (Start Space)



Define the problem and or as a space (Start Space)

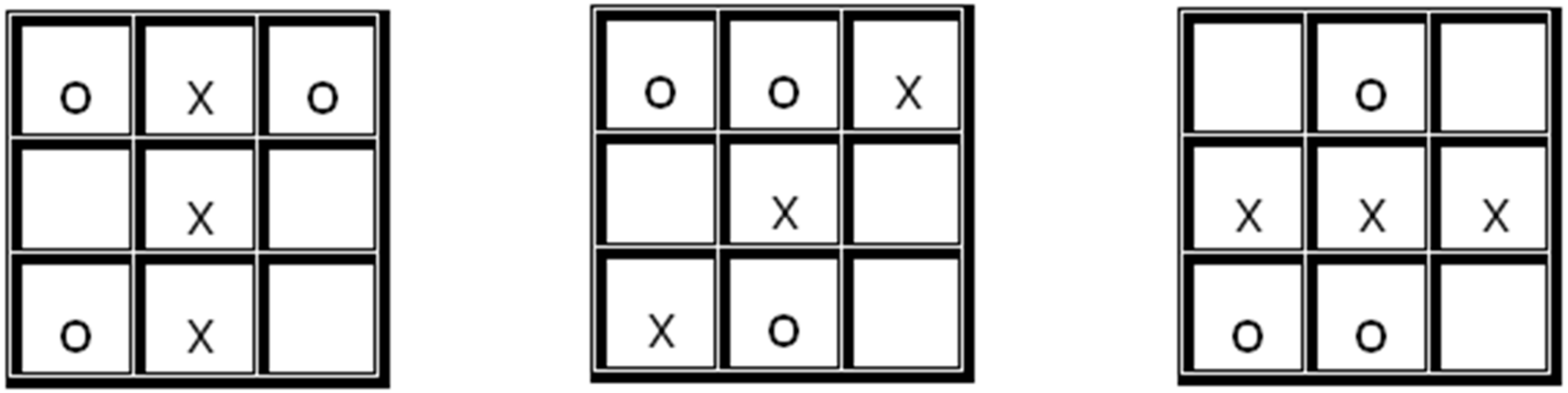
TWO-PLAYER GAMES (Tic-Tac-Toe)

- Two-player games are games in which two players compete against each other. These are also known as zero-sum games. The goal then in playing a two-player game is choosing a move that maximizes the score of the player and/or minimizes the score of the competing player.

- Consider the two-player game **Tic-Tac-Toe.** Players alternate moves, and as each move is made, the possible moves are constrained (see the partial Tic-Tac-Toe game tree in next Figure). In this simple game, a move can be selected based on the move leading to a win by traversing all moves that are constrained by this move.

Define the problem and or as a space (Start Space)

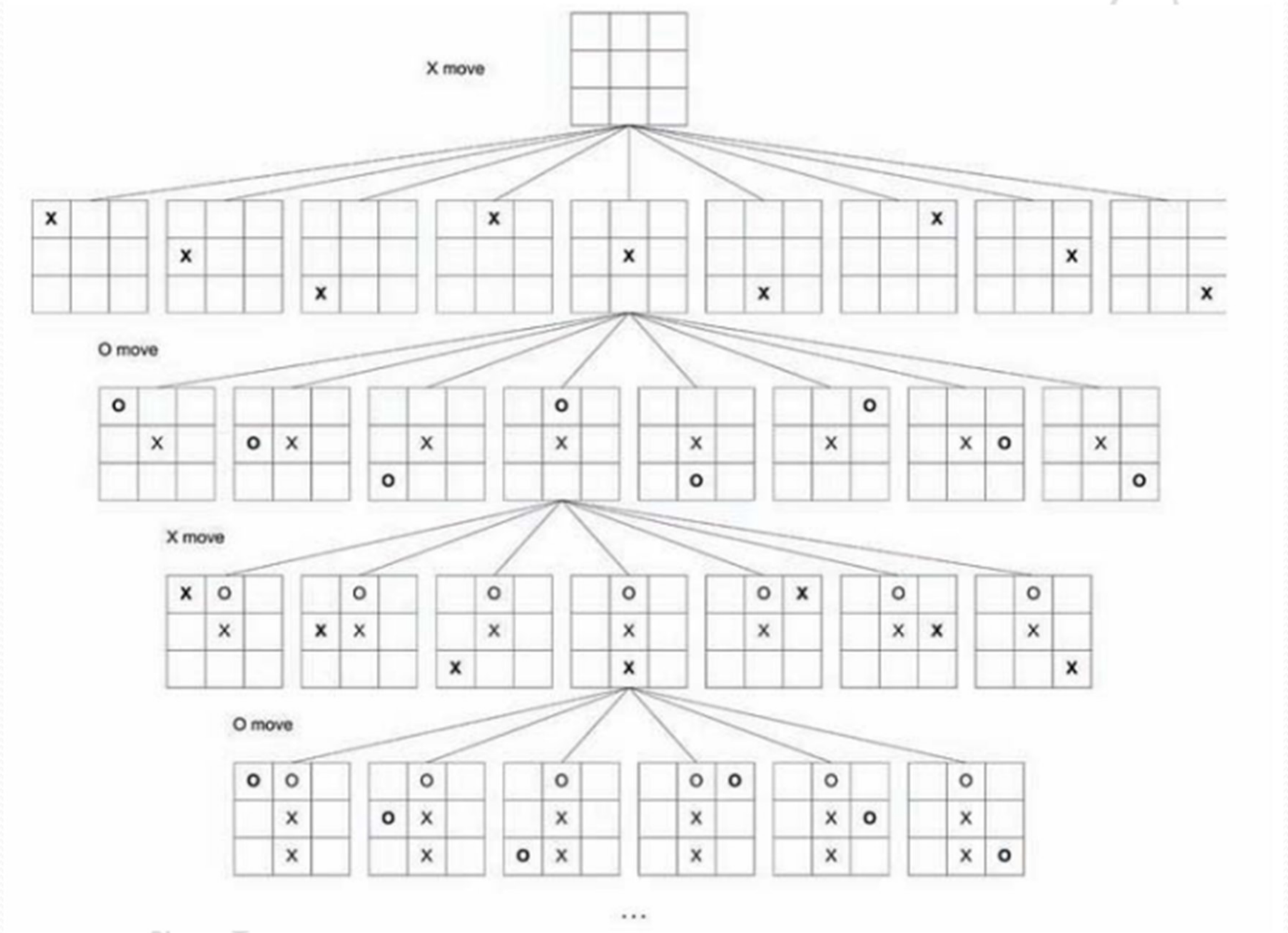
- Also, by traversing the tree for a given move, we can choose the move that leads to the win in the shallowest depth (minimal number of moves).



Define the problem and or as a space (Start Space)

- **Tic-Tac-Toe** is an interesting case because the maximum number of moves is tiny when compared to more complex games such as Checkers or Chess. Tic-Tac-Toe is also open to numerous optimizations. Consider, for example, the first X move in below figure . If the board is rotated, only three unique moves are actually possible. Without optimization, there exist 362,880 nodes within the complete game tree.

Define the problem and or as a space (Start Space)


X move
O move
X move
O move
...