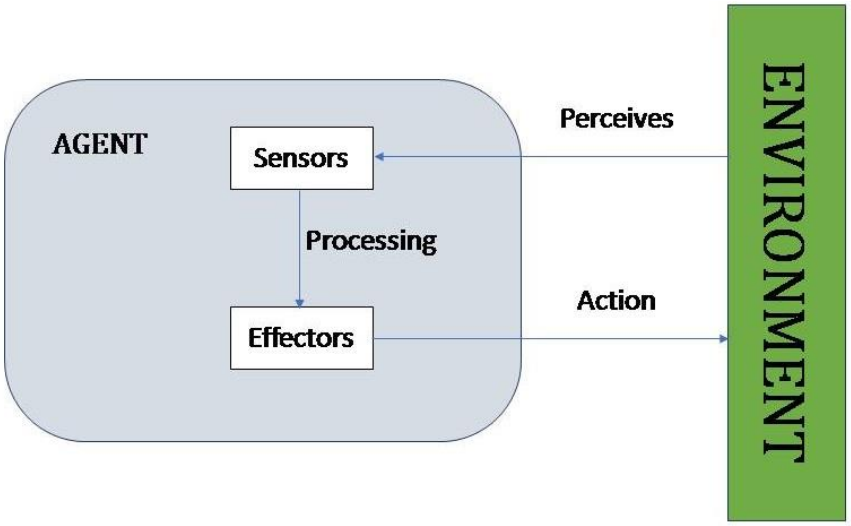
Solving Problems by Searching

Introduction to Problem-Solving Agents

an **agent** is anything that can be viewed as perceiving its environment through sensors and acting upon environment through effectors/actuators.



When an agent is faced with a situation where the correct action isn't immediately obvious, it must **plan ahead**. This process of looking for a sequence of actions that reaches a goal is called **Search**.

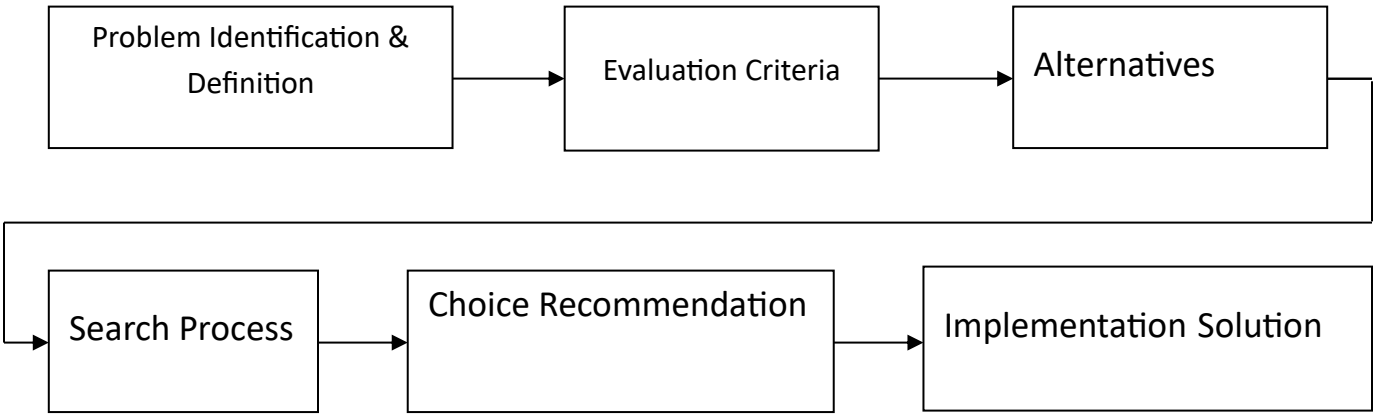
A **search algorithm** takes a problem as an input and returns a solution in the form of an action sequence.

Search is central in many AI systems: theorem proving, game playing, navigation, etc.

Key Characteristics of the Agent:

- **Atomic Representations:** The agent treats states as "wholes" (like a single city on a map) with no internal structure.
- **Environment Assumptions:** we assume the environment is:
    - **Fully Observable:** The agent knows exactly where it is.
    - **Deterministic:** Actions have predictable outcomes.
    - **Static:** The world doesn’t change while the agent is "thinking."
    - **Discrete:** There are a limited number of distinct actions and states.

Steps of Problem Solving



Problems Solving: There are two types of problems:

1. Problems solved by computation which is easy to program.

$$eg.: \mathrm{a} = 2*3 + 4$$
$$\mathrm{a} = 6 + 4$$
$$\mathrm{a} = 10$$

In this type, traditional numerical methods are utilized through linear search to reach the optimal solution. This involves moving toward the objective using specific mathematical steps that are easily programmed.

2. Problems solved by search space techniques to find the goal.

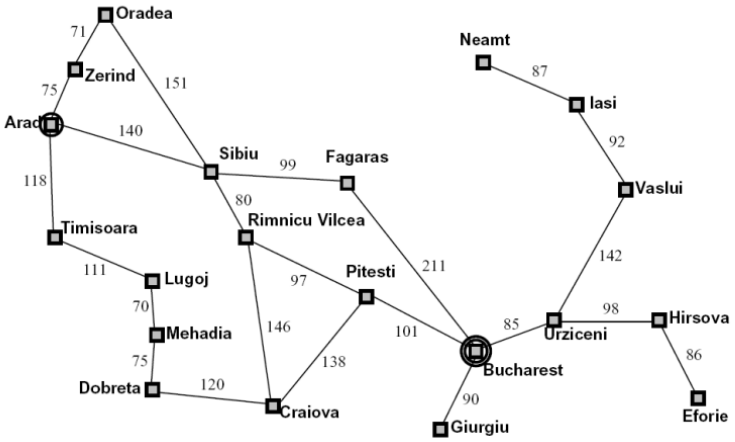
In this type, space search techniques are employed. These are intelligent problems that deal with structures and methods requiring spatial occupancy and possess a specific architecture that differs from linear search, which does not require significant storage or intelligent processing. Furthermore, space search requires more time and is capable of solving traditional linear problems, whereas the reverse is not true. Examples of this type include Games and Operations Research problems, which are real-world issues for which mathematical models are developed.

Formal Definition of a Search Problem

To write an algorithm to solve a problem, we must define it using these components:

- **State Space:** The set of all states reachable from the initial state by any sequence of actions.
- **Initial State:** The state in which the agent starts (e.g., "Chessboard in starting configuration").
- **Actions:** A description of the possible actions available to the agent. Given a state “s”, “Actions(s)” returns the set of actions that can be executed.
- **Transition Model:** A description of what each action does. It is typically defined by a function “Result(s, a)” that returns the state resulting from performing action “a” in state “s”.
- **Goal Test:** A condition that determines whether a given state is a goal state .
- **Path Cost:** A function that assigns a numeric cost to each path. The agent's objective is usually to find a solution with the lowest path cost.

example:



- State space:
  - Cities
- Successor function:
  - Roads: Go to adjacent city with cost = distance
- Start state:
  - Arad
- Goal test:
  - Is state == Bucharest?
- Solution?

Important Concepts:

- **Path:** A sequence of actions.
- **Solution:** A path from the initial state to a goal state.
- **Optimal Solution:** The solution with the lowest total path cost.

Problem Characteristics خواص المسألة:

a- Is the problem decomposable into a set of smaller easier.

Is the problem we have decomposable into smaller parts that are easier to solve, as in mathematical problems?

$$y = xdx + cos\, x$$

Here, the problem can be decomposed, as solving decomposed problems is easier than non-decomposable problems (such as the game of chess, which cannot be decomposed).

b- Can solution steps be ignored? Or at least undone if the prove unwise.

Do the rules of the problem allow for backtracking from a specific step or a legal move (or a permitted move)? That is, do the rules of the problem allow for the following cases:

1. To ignore the step.
2. To backtrack from the step.
3. We cannot do anything. These cases must be identified so that the method of programming the solution can be known.

c- Is the problem universe predictable (قابل للتنبؤ)?

Is the problem space predictable? That is, does the problem allow for the use of intuition techniques, reasoning, or both?

d- Is a good solution to the problem obvious (واضـــح ، جلي) without comparison to all other solution.

When there are problems we want to reach a good and acceptable solution for, the first solution method we reach is the appropriate method. However, if we want the best (optimal) solution, we must reach all cases or solutions to the problem, then compare them and choose the best solution (method).

e- Is the amount of knowledge required to solve the problem import (ضمني، فحوى) or is it only to constrain(توجيه) search?

The knowledge we need is either: knowledge within (implicit to) the problem (the solution) or knowledge outside the problem (the solution), which is guidance knowledge in solving the problem. Therefore, the type of knowledge must be determined.

Space Search: It is one of the axes of Artificial Intelligence. It involves transforming a problem into a **Graph** or a **Tree**, where every element in the problem that occupies space is represented. It is a deductive method where the required value (**the solution**) is reached using a **Knowledge Base** along with an appropriate technique. Any technique that moves away from traditional mathematics is called an "intelligent technique" and is defined by a specific **Structure**.

Graph: It consists of vertices(Nodes) connected by arcs(links), which can represent many real world entities including airline routes, electrical circuits and job scheduling.

The link between two nodes represented the way of the nodes related to the tree.

If there is a path direction associated(مرتبط) with the graph, then the graph is called directed graph.

A state search space can be represented by four – symbols:**[N, A, S, GD]**. where:

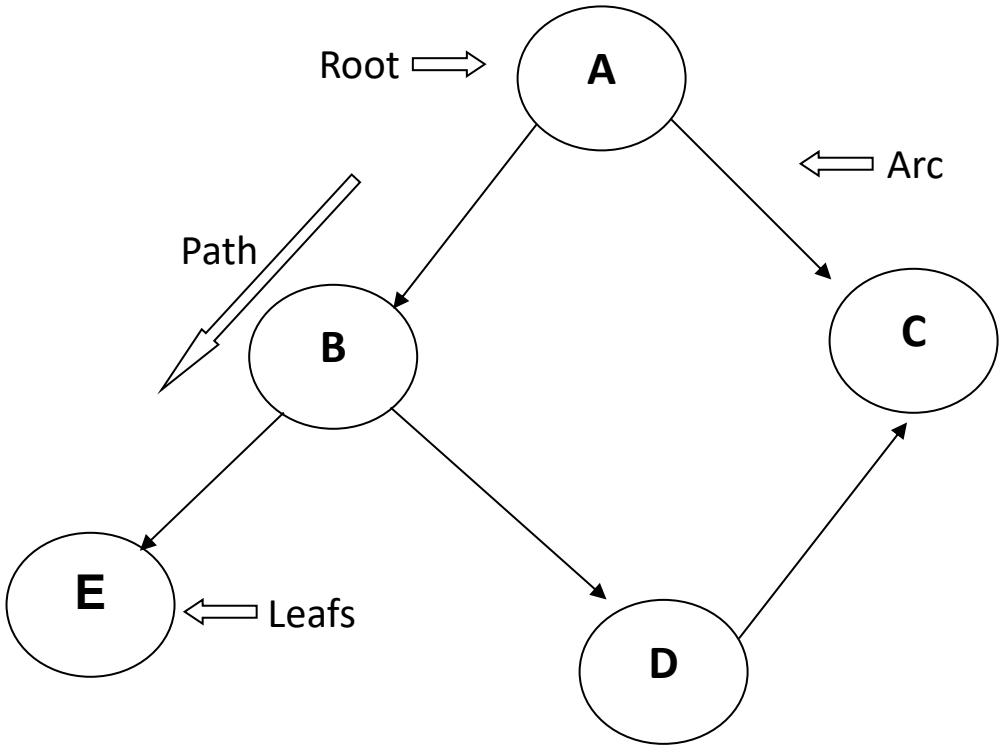
N: is a set of nodes of the graph. These correspond to the states in the problem.

A: is the set of links between nodes. These correspond to the steps in the problem.

S: is a non – empty subset of (N) contains the start state of the problem.

GD: is a non – empty subset of (N) contains the goal state of the problem.

Solution Path: Is the path through the graph from a node in(S) to a node in(GD).



Root(الجذر): is a node with no parent.

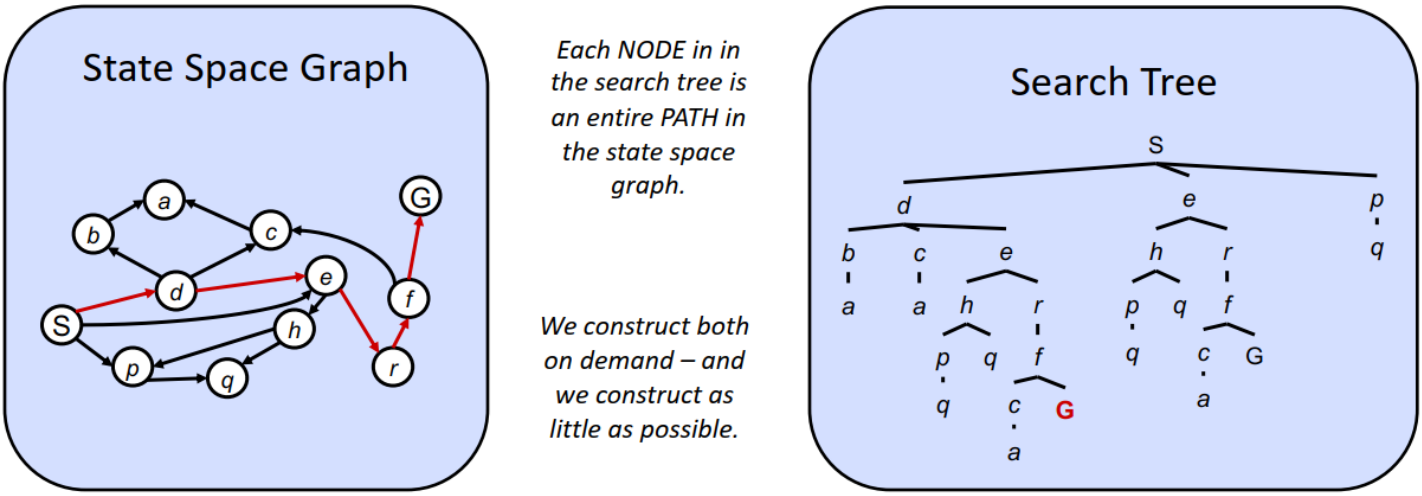
Leaf(الورقة): is a node with no child.

Path(المســار): is an ordered sequence of nodes[N1, N2, N3,…..Nn]. Where: Ni is a parent of Ni+1 and its called path of length n.

Cyclic Path(المسـار الدائري أو الحلقي): is a path that content the same state more than once (A, B, C, A).

Tree(الشجرة): is a graph with no cycle. Each node has a unique parent.

Note: from tree we can calculate the coast for each path and determine the minimum one.



General Tree Search Algorithm

function TREE-SEARCH(problem, strategy) returns a solution, or failure

initialize the frontier using the initial state of problem

loop do

if the frontier is empty then return failure

choose a leaf node and remove it from the frontier according to strategy

if the node contains a goal state then return the corresponding solution

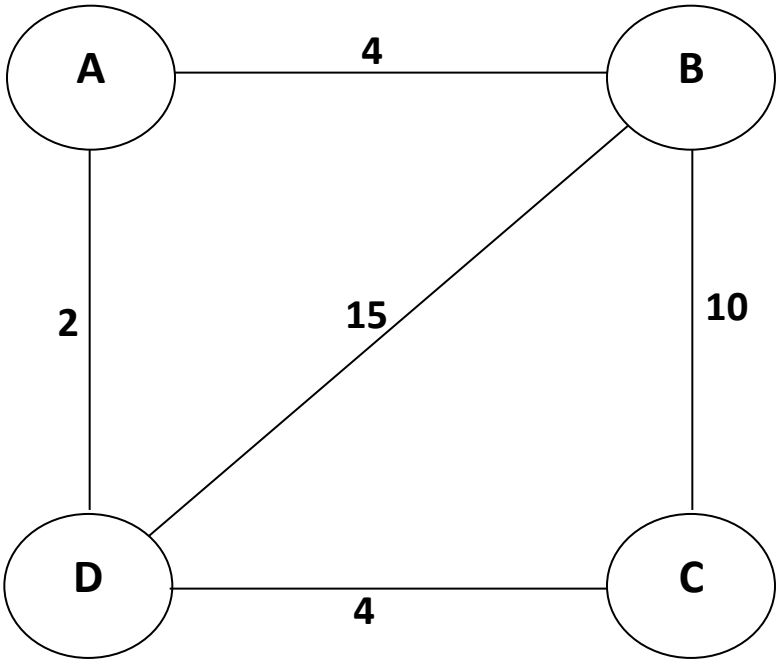
else expand the chosen node (generate its children) and

add the resulting child nodes to the frontier

end loop

Frontier (or Open List): The collection of nodes waiting to be expanded.

Example 1: In the graph shown below, the search algorithm allows you to visit each vertex(node) in the graph in a systematic way (بطريقة نظامية). The start node is A.



Solution of 1:

1- Convert graph to tree

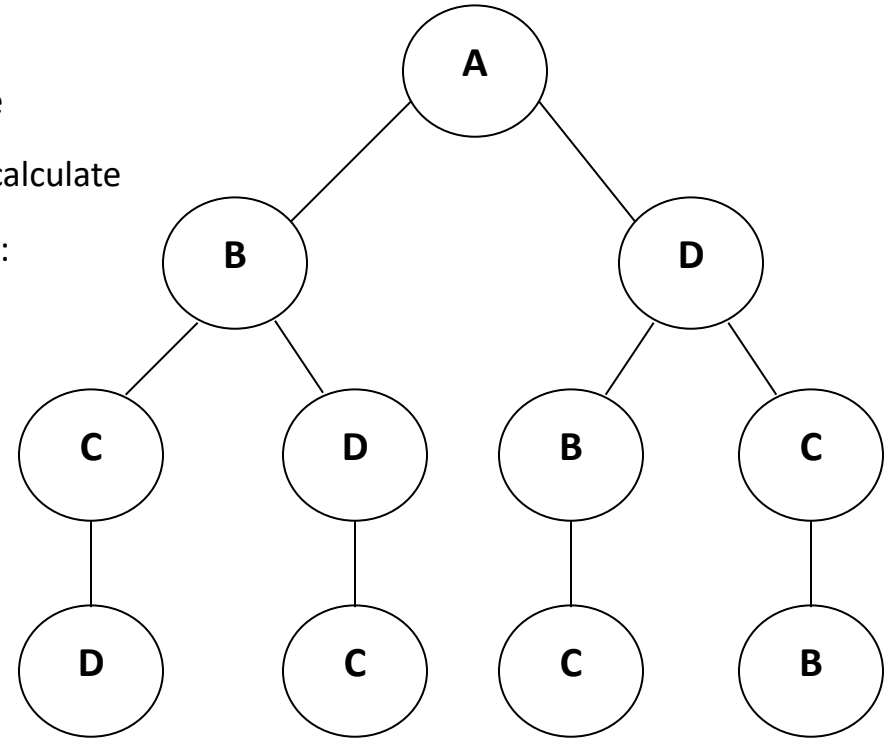
2- Write the paths and calculate the cost of each path:

A B C D " ? "

A B D C " ? "

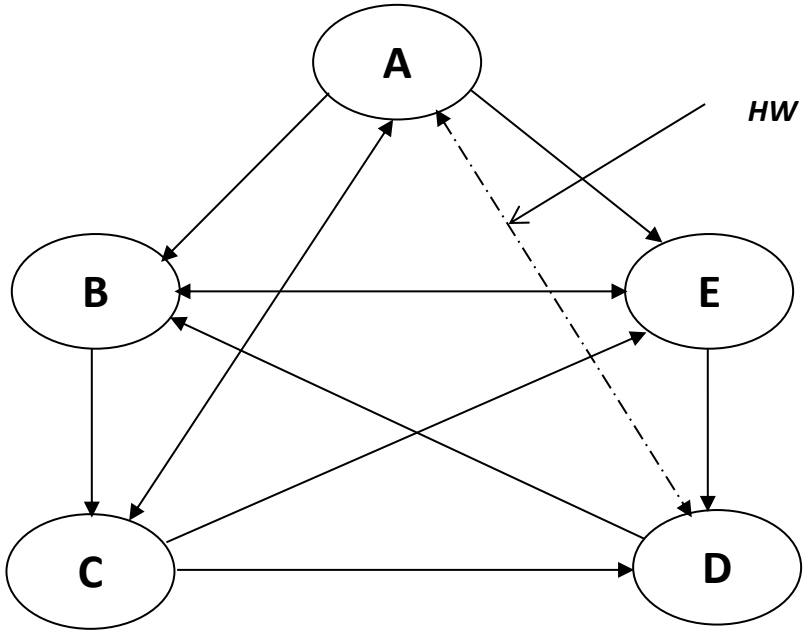
A D B C " ? "

A D C B " ? "

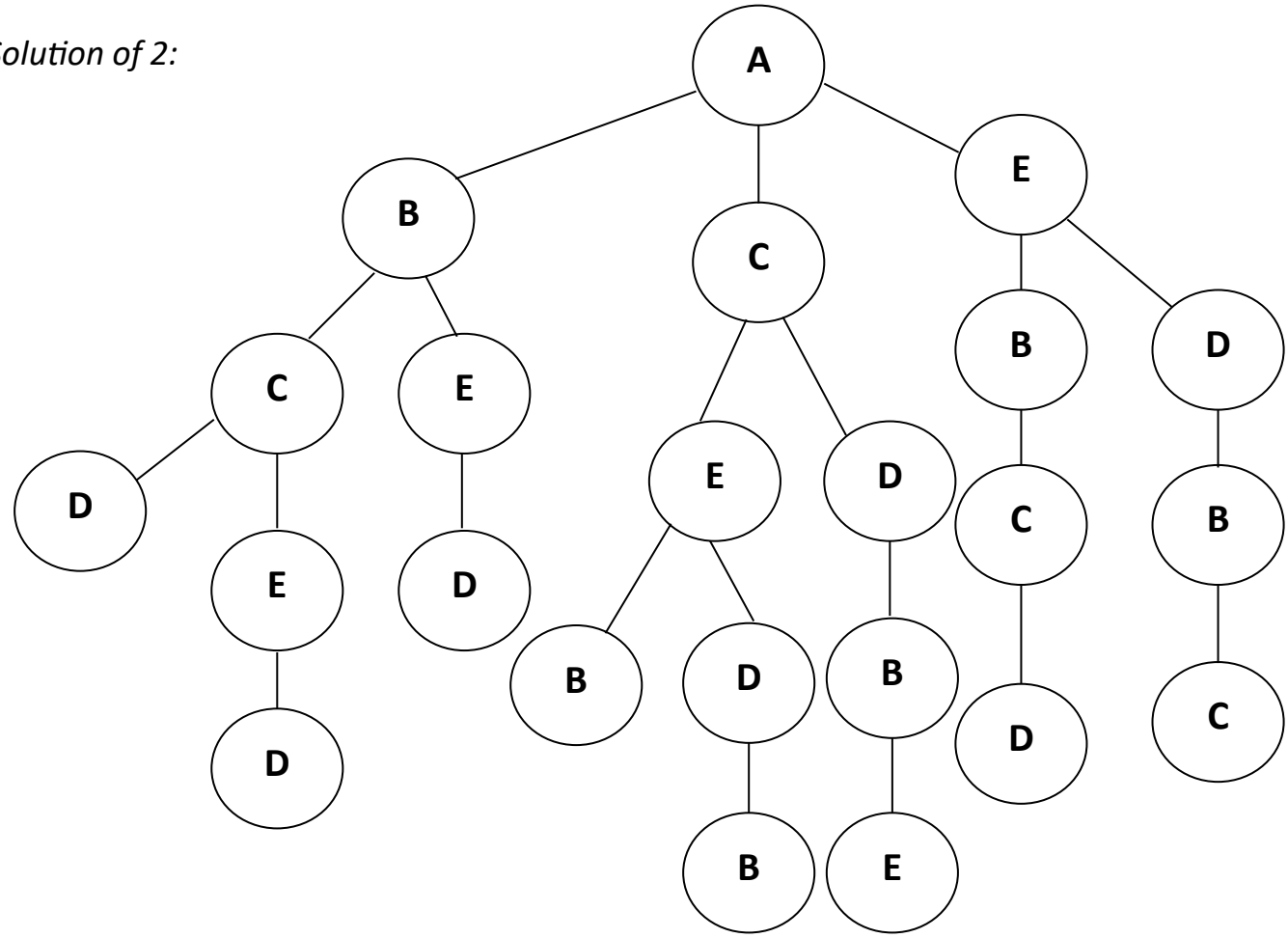


No. of nodes = 11 Links = 10 Paths = 4

Example 2: Find the tree which equivalent to the graph below, and then write all the paths of the tree.



Solution of 2:



No. of nodes = ? Links = ? Paths = ?

Example 3: *Coins Problem(العملة النقدية):* head(h) and tail(t)

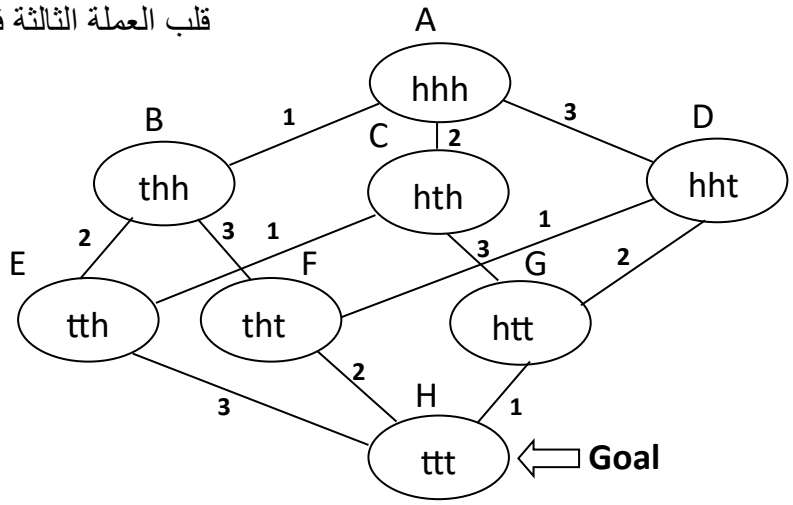
Initial state: hhh لو كان لدينا ثلاث عملات نقدية بالوجه الأول

Goal state: ttt والمطلوب قلب العملات النقدية الثلاثة إلى الوجه الثاني

- لا يجوز قلب أكثر من عملة نقدية في نفس الوقت.
- لا يجوز تكرار الـ(Nodes).

Solution of 3:

1. hAB → tAB قلب العملة الأولى فقط
2. AhB → AtB قلب العملة الثانية فقط
3. ABh → ABt قلب العملة الثالثة فقط



The figure above is the State Space for Coins Problem, where the Initial State(hhh) and the Goal State(ttt).