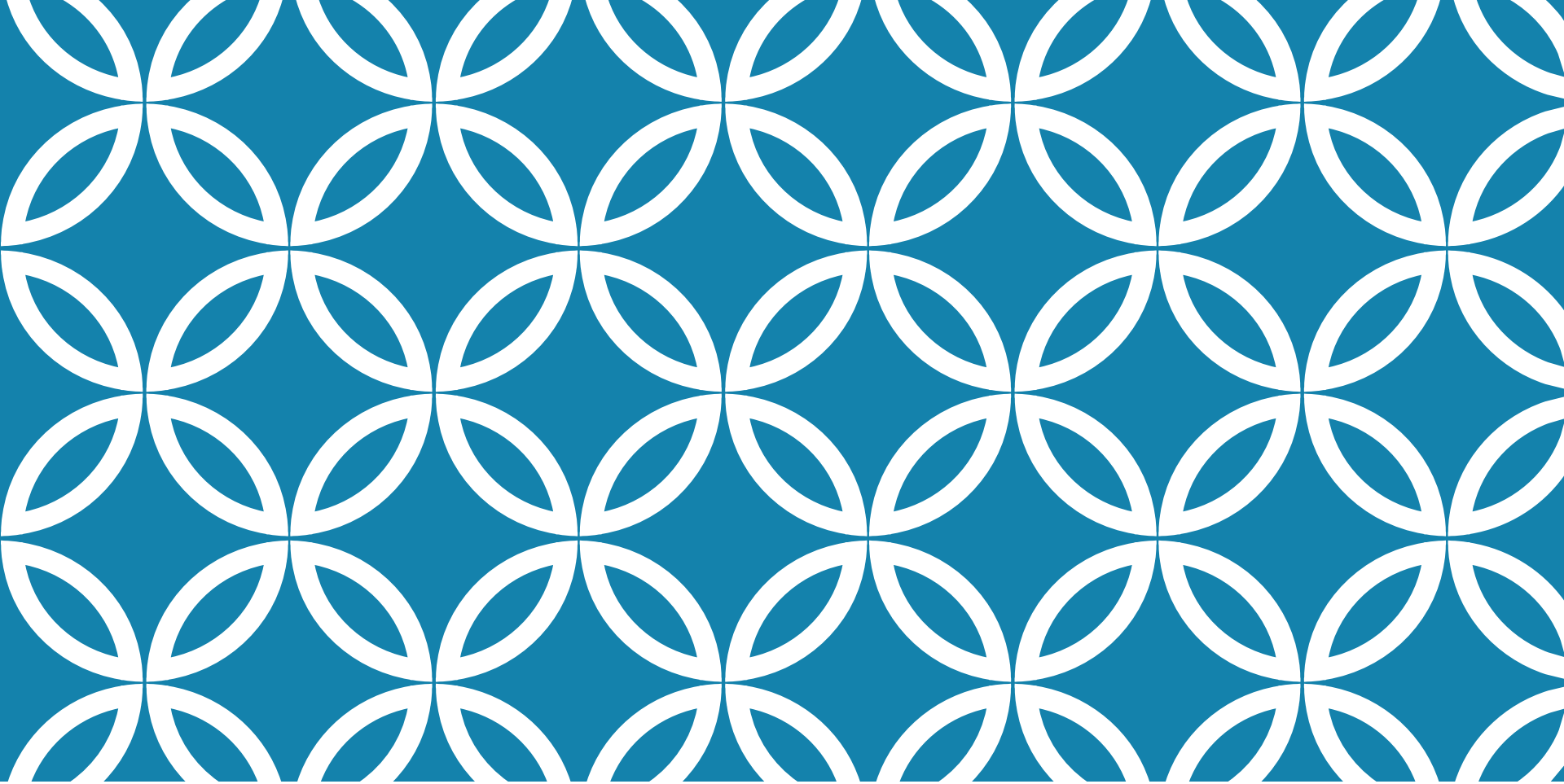


ARTIFICIAL INTELLIGENCE

“Artificial Intelligence is the part of computer science with designing intelligent computer systems, that is, systems that exhibit the characteristics we associate with intelligence in human behavior.”


Fingent
AI

IN WHAT WAY COMPUTER & HUMAN BEING ARE BETTER?

Computers	Human Being
1. Numerical Computation is fast	1. Numerical Computation is slow
2. Large Information Storage Area	2. Small Information Storage Area
3. Fast Repetitive Operations	3. Slow Repetitive Operations
4. Numeric Processing	5. Symbolic Processing
5.Computers are just Machine (Performed Mechanical "Mindless" Activities)	4. Human Being is intelligent (make sense from environment)

Artificial Intelligence (AI) is the area of computer science focusing on creating machines that can engage on behaviors that humans consider intelligent.

According to **Avron Barr** and **Edward A. Feigenbaum**, " The Handbook of Artificial Intelligence", the goal of AI is to develop intelligent computers. Here intelligent computers means that emulates intelligent behavior in humans.

HISTORY OF AI/TIME MACHINE OF AI

1943 McCulloch & Pitts: Boolean circuit model of brain

1950 Turing's "Computing Machinery and Intelligence"

1956 Dartmouth meeting: "Artificial Intelligence" adopted

1952—69 Look, Ma, no hands!

1950s Early AI programs, including Samuel's checkers program, Newell & Simon's Logic Theorist, Gelernter's Geometry Engine

1965 Robinson's complete algorithm for logical reasoning

1966—73 AI discovers computational complexity
Neural network research almost disappears

1969—79 Early development of knowledge-based systems

1980-- AI becomes an industry

1986-- Neural networks return to popularity

1987-- AI becomes a science

1995-- The emergence of intelligent agents

Achievements in the area of AI

❖In the early 1950's Claude Shannon and Alan Turing developed chess programs for Von-Neumann-style computers.

❖ McCarthy in 1958 wrote a programming language LISP.

❖ He developed a program called '**Advice Taker**'(Programs with common sense) . This system used domain knowledge to generate plans.

❖The first computer program designed to solve problems in a human way is **General Problem Solver' (GPS)**by Allen Newell and Herbert Simon.

❖Within a very short time a number of Knowledge representation languages developed such as predicate calculus, semantic networks, frames and objects.

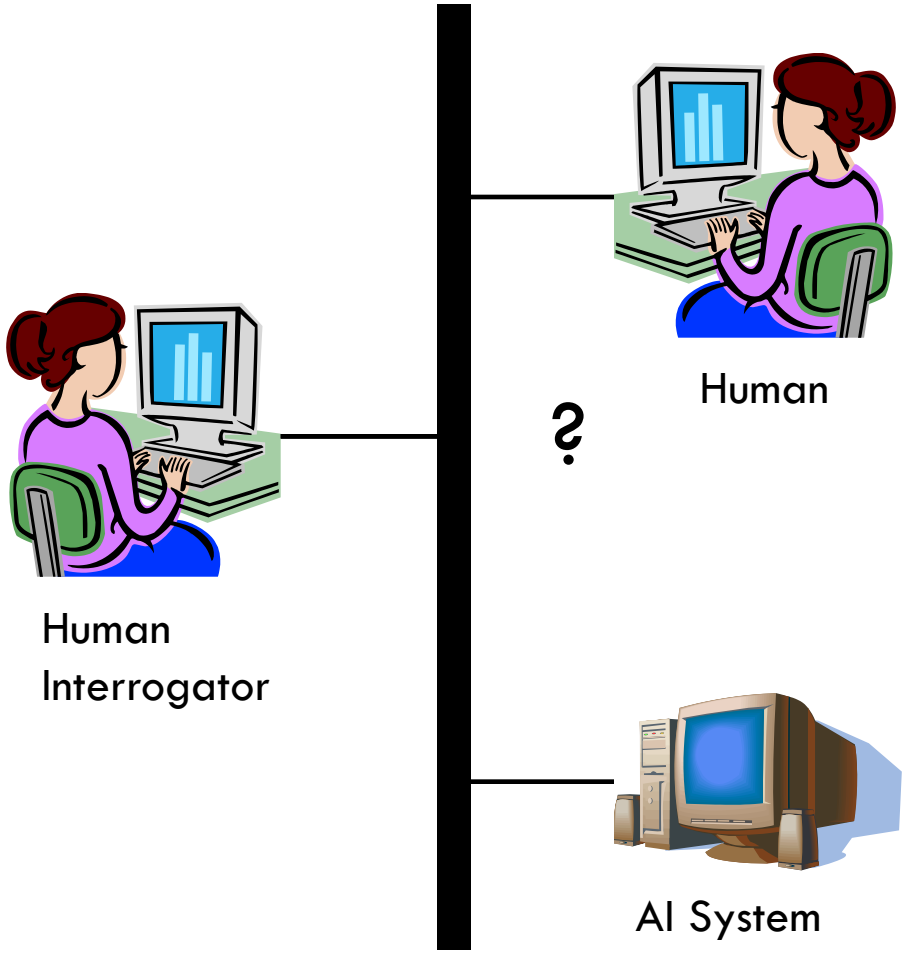
ACTING HUMANLY

The Turing Test (1950)

- Can machines think?
- Can machines behave intelligently?

Operational test for intelligent behavior

- The Imitation Game



CHALLENGES BEFORE AI

In order to achieve the task of imitating human behavior or acquiring human intelligence, a machine (a computer in our case) must reflect the following capabilities which are commonly inherited by an intelligent person:

Natural language processing : Like a human, a machine should understand the spirit or the meaning of sentences spoken or written freely in natural language by humans.

knowledge representation : Knowledge which can be presented in mathematical or some logical format. Ultimate goal to get a work done by a computer will be to translate the informal sentences into formal ones which could be well interpreted by a computer. We use production systems, semantic nets, frames like structures to express knowledge.

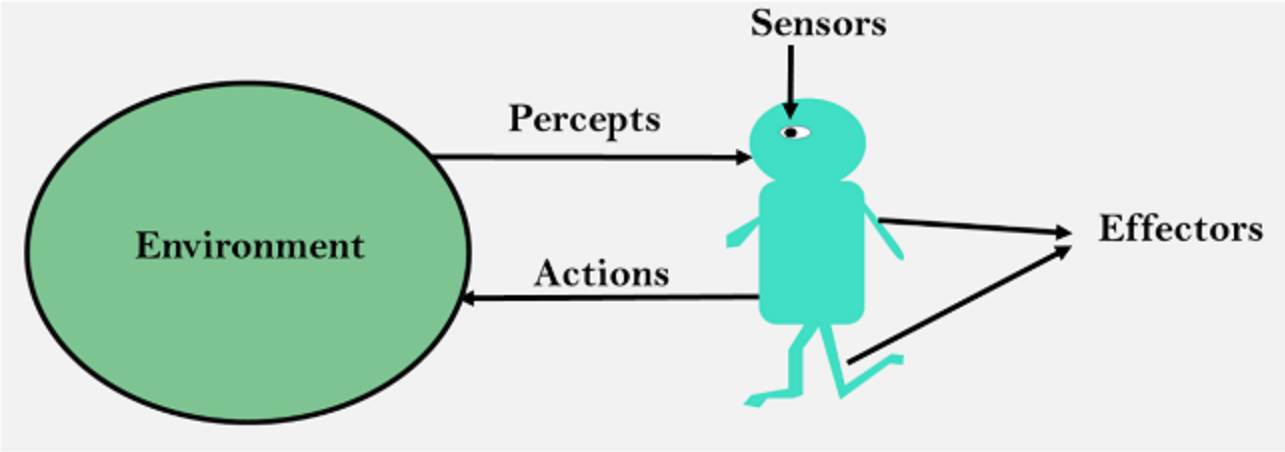
automated reasoning : The capability to use the stored information to answer questions and to draw new conclusions.

machine learning : Learning is an important property of humans. Whatever we wish to act, we learn first then exercise for perfection. A machine should also be able to learn to adapt to new circumstances and to detect and extrapolate patterns.

APPLICATIONS OF ARTIFICIAL INTELLIGENCE

- Finance
- Medical
- Industries
- Telecom
- Transport
- Entertainment
- Pattern Recognition
- Robotics
- Data Mining
- E-Commerce
- Etc….

INTELLIGENT AGENTS

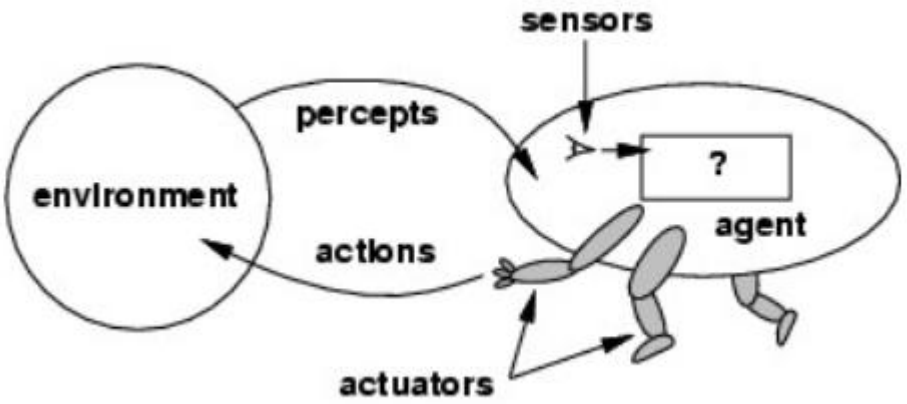


INTELLIGENT AGENTS

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators

Examples:

- Human agent
- Robotic agent
- Software agent

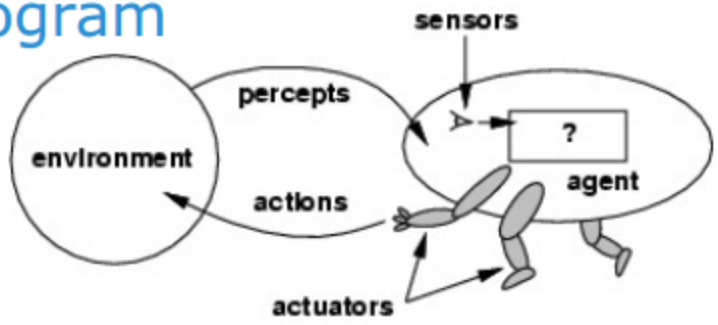


EXAMPLES

- **Human-Agent:** A human agent has eyes, ears, and other organs which work for sensors and hand, legs, vocal tract work for actuators.
- **Robotic Agent:** A robotic agent can have cameras, infrared range finder, NLP for sensors and various motors for actuators.
- **Software Agent:** Software agent can have keystrokes, file contents as sensory input and act on those inputs and display output on the screen.

TERMINOLOGIES

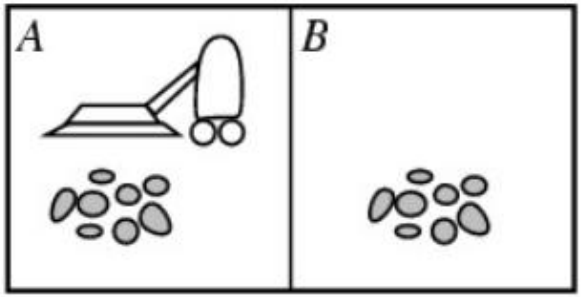
- Percept: the agent's perceptual inputs
- Percept sequence: the complete history of everything the agent has perceived
- Agent function maps any given percept sequence to an action [$f: p^* \rightarrow A$]
- The agent program runs on the physical architecture to produce f
- Agent = architecture + program



GOALS -AGENTS

- High permormance
- Optimized result
- Rational action

EX: VACUUM CLEANER

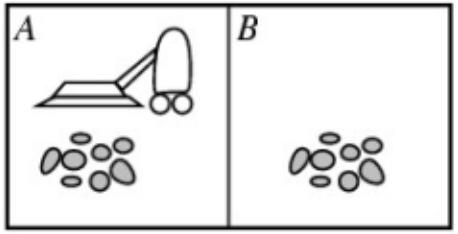


- Percepts: location and contents, e.g., [A, dirty]
- Actions: Left, Right, Suck, NoOp

Percept sequence	Action
[*A*, *Clean*]	*Right*
[*A*, *Dirty*]	*Suck*
[*B*, *Clean*]	*Left*
[*B*, *Dirty*]	*Suck*

SIMPLE AGENT FUNCTION

Percept sequence	Action
[A, Clean]	?
[A, Dirty]	
[B, Clean]	
[B, Dirty]	
[A, Clean], [A, Clean]	
[A, Clean], [A, Dirty]	
…	
[A, Clean], [A, Clean], [A, Clean]	
[A, Clean], [A, Clean], [A, Dirty]	
…	



AGENTS- NATURE OF ENVIRONMENT(STRUCTURE OF AGENTS)

agent = architecture + program

AGENT PROGRAM /(PSEUDOCODE)

```
function SKELETON-AGENT(percept) returns action
  static: memory, the agent's memory of the world

  memory ← UPDATE-MEMORY(memory, percept)
  action ← CHOOSE-BEST-ACTION(memory)
  memory ← UPDATE-MEMORY(memory, action)
  return action
```

ENVIRONMENT

Actions performed by the agent on the environment, which in turn provides percepts to the agent

Properties of ENVIRONMENT:

- Accessible vs inaccessible
- Deterministic vs nondeterministic
- Episodic vs nonepisodic
- Static vs dynamic
- Discrete vs continuous

EXAMPLES OF AGENT TYPES WITH PAGE DESCRIPTION

Agent Type	Percepts	Actions	Goals	Environment
Medical diagnosis system	Symptoms, findings, patient's answers	Questions, tests, treatments	Healthy patient, minimize costs	Patient, hospital
Satellite image analysis system	Pixels of varying intensity, color	Print a categorization of scene	Correct categorization	Images from orbiting satellite
Part-picking robot	Pixels of varying intensity	Pick up parts and sort into bins	Place parts in correct bins	Conveyor belt with parts
Refinery controller	Temperature, pressure readings	Open, close valves; adjust temperature	Maximize purity, yield, safety	Refinery
Interactive English tutor	Typed words	Print exercises, suggestions, corrections	Maximize student's score on test	Set of students

RATIONALITY

- An agent should "do the right thing", based on what it can perceive and the actions it can perform. The right action is the one that will cause the agent to be most successful
- Performance measure: An objective criterion for success of an agent's behavior
- Back to the vacuum-cleaner example
  - Amount of dirt cleaned within certain time
  - +1 credit for each clean square per unit time
- General rule: measure what one wants rather than how one thinks the agent should behave

RATIONAL AGENT- DEFINITION

- For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

EX: VACUUM CLEANER

- Ȧ simple agent that cleans a square if it is dirty and moves to the other square if not
- Is it rational?
- Assumption:
  - performance measure: 1 point for each clean square at each time step
  - environment is known a priori
  - actions = {left, right, suck, no-op}
  - agent is able to perceive the location and dirt in that location
- Given different assumption, it might not be rational anymore

PROBLEM SOLVING AGENTS

- An **agent** that tries to come up with a sequence of actions that **will** bring the environment into a desired state. Search. The process of looking for such a sequence, involving a systematic exploration of alternative actions

- Problem-solving agents decide what to do by finding sequences of actions that lead to desirable states.

```
function SIMPLE-PROBLEM-SOLVING-AGENT(p) returns an action
  inputs: p, a percept
  static: s, an action sequence, initially empty
          state, some description of the current world state
          g, a goal, initially null
          problem, a problem formulation

  state ← UPDATE-STATE(state,p)
  if s is empty then
      g — FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, g)
      s ← SEARCH(problem)
  action — RECOMMENDATION(s, state)
  s ← REMAINDER(s, state)
  return action
```

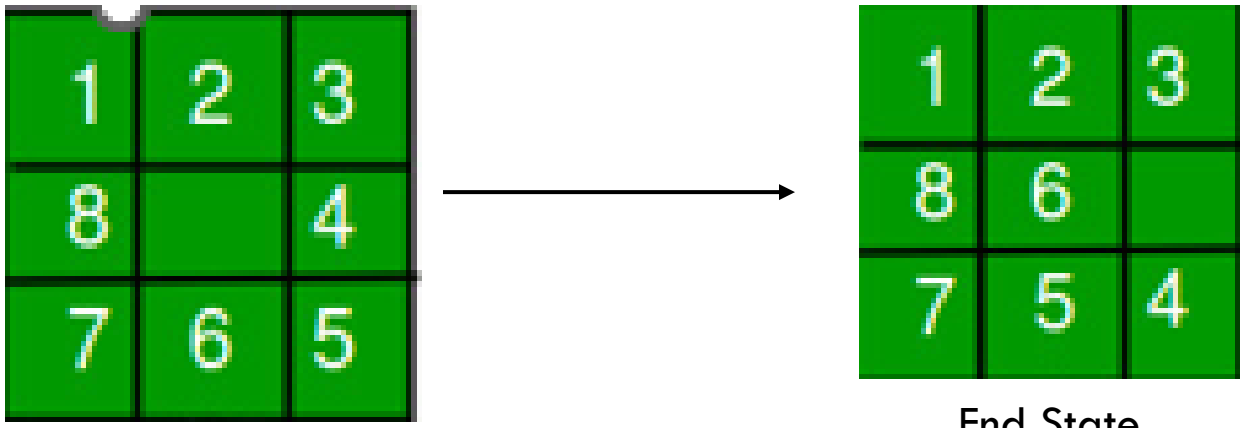
A simple problem-solving agent.

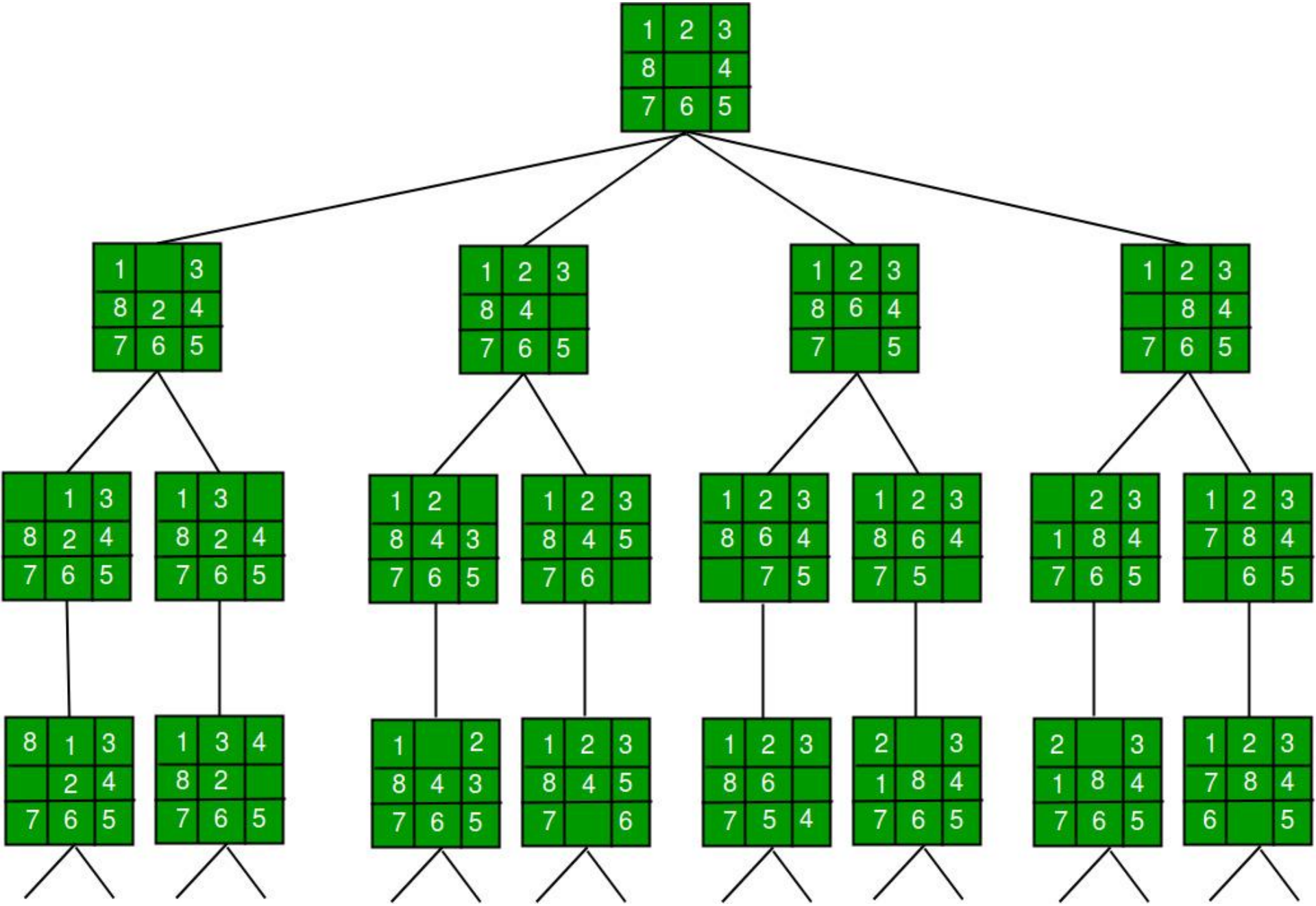
EXAMPLE PROBLEMS-TOY PROBLEMS

- **States**
- **Operators**
- **Goal test**
- **Path cost**

THE 8-PUZZIE

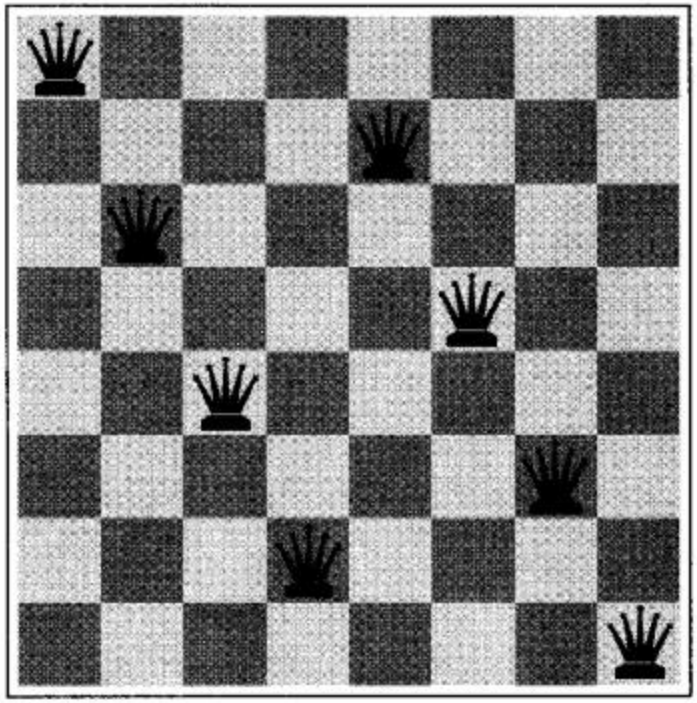
- **States:** a state description specifies the location of each of the eight tiles in one of the nine squares. For efficiency, it is useful to include the location of the blank.
- **Operators:** blank moves left, right, up, or down.
- **Goal test:** state matches the goal configuration
- **Path cost:** each step costs 1, so the path cost is just the length of the path.




1 2 3
8 4
7 6 5
1 3
8 2 4
7 6 5
1 2 3
8 4
7 6 5
1 2 3
8 6 4
7 5
1 2 3
8 4
7 6 5
1 3
8 2 4
7 6 5
1 3
8 2 4
7 6 5
1 2
8 4 3
7 6 5
1 2 3
8 4 5
7 6
1 2 3
8 6 4
7 5
1 2 3
8 6 4
7 5
2 3
1 8 4
7 6 5
1 2 3
7 8 4
6 5
8 1 3
2 4
7 6 5
1 3 4
8 2
7 6 5
1 2
8 4 3
7 6 5
1 2 3
8 4 5
7 6
1 2 3
8 6
7 5 4
2 3
1 8 4
7 6 5
2 3
1 8 4
7 6 5
1 2 3
7 8 4
6 5

THE 8-QUEENS PROBLEM

- **Goal test**: 8 queens on board, none attacked.
- **Path** cost: zero.
- **Operators :**There are also different possible states and operators. Consider the following simple-minded formulation
- **States:** any arrangement of 0 to 8 queen



REAL WORLD PROBLEMS

- States
- Initial state
- Actions
- Transition model
- Goal test
- Path cost

REAL WORLD PROBLEMS

- **Route-finding problem** –computer networks, military operations planning, airline travel planning
- **Touring problems**-TSP problem
- **VLSI Layout problems**-positioning millions of components and connections on a chip to minimize area, minimize circuit delays and maximize manufacturing yield.
- **Robot navigation**
- **Automatic assembly sequencing**

CONSTRAINT SATISFACTION

In this type of search set of constraints are available. As compared with a straightforward search procedure, viewing a problem as one of constraint satisfaction can reduce substantially the amount of search.

Applications:

- ✓ Design problems
- ✓ Labelling graphs
- ✓ robot path planning
- ✓ Cryptarithmetic puzzles

CRYPTARITHMETIC PUZZLES

Cryptarithmetic puzzle:

If **TO** + **GO** = **OUT**, find the value of **TO**

```
  SEND
+ MORE
------
 MONEY
```

CRYPTARITHMETIC PUZZLES

- **States:** a cryptarithmetic puzzle with some letters replaced by digits.
- **Operators:** replace all occurrences of a letter with a digit not already appearing in the puzzle.
- **Goal test:** puzzle contains only digits and represents a correct sum.
- **Path cost:** zero. All solutions equally valid.

Search Algorithms in Artificial Intelligence

- Search forms the core component of many intelligent processes
- Search is the systematic examination of states to find path from the start/root state to the goal state.
- Many traditional search algorithms are used in AI applications.
- For complex problems, the traditional algorithms are unable to find the solution within some practical time and space limits.
- Consequently, many special techniques are developed; using heuristic functions.

Search Algorithms in Artificial Intelligence

Requirement of Search Algorithms / Techniques

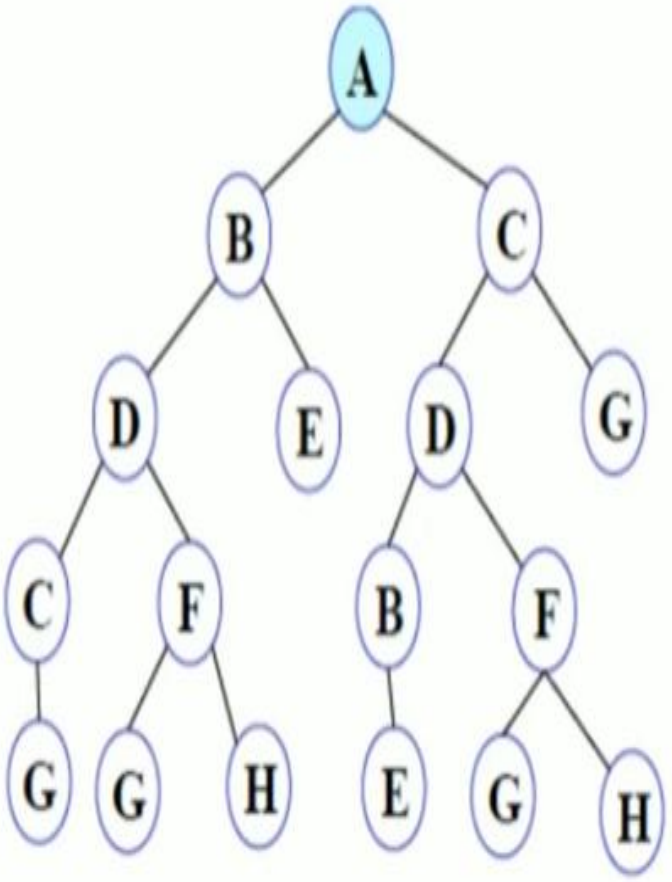
1. *The first requirement is that it should causes* ***motion***

2. *The second requirement is that it should be* ***systematic***

Types of Search Algorithms in Artificial Intelligence

1. **Uninformed search** (blind search)

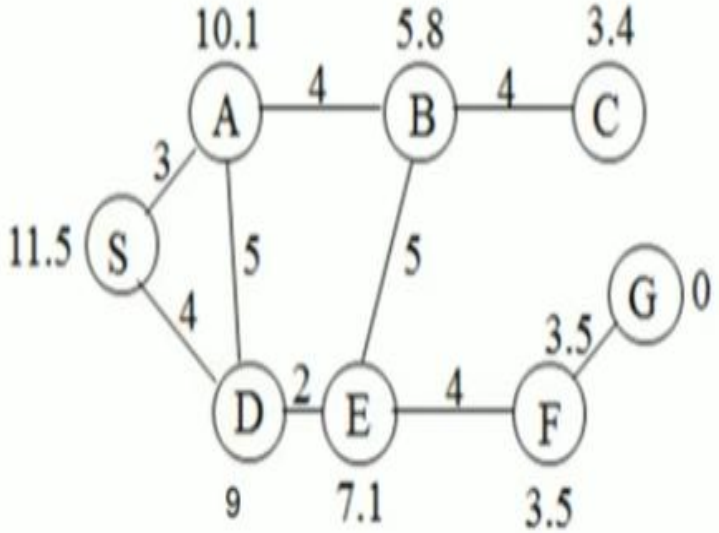
   – Also called blind, exhaustive or brute-force search, uses no information about the problem to guide the search and therefore may not be very efficient.



Types of Search Algorithms in Artificial Intelligence

2. **Informed search** (heuristic search)

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



INFORMED SEARCH STRATEGIES(HEURISTIC SEARCH)

- **Best first search**

Combines the advantages of **Breadth-First** and **Depth-First** searches.

– **DFS**: follows a single path, don't need to generate all competing paths.

– **BFS**: doesn't get caught in loops or dead-end paths.

Best First Search: explore the most promising path

UNINFORMED SEARCH STRATEGIES(BLIND SEARCH)

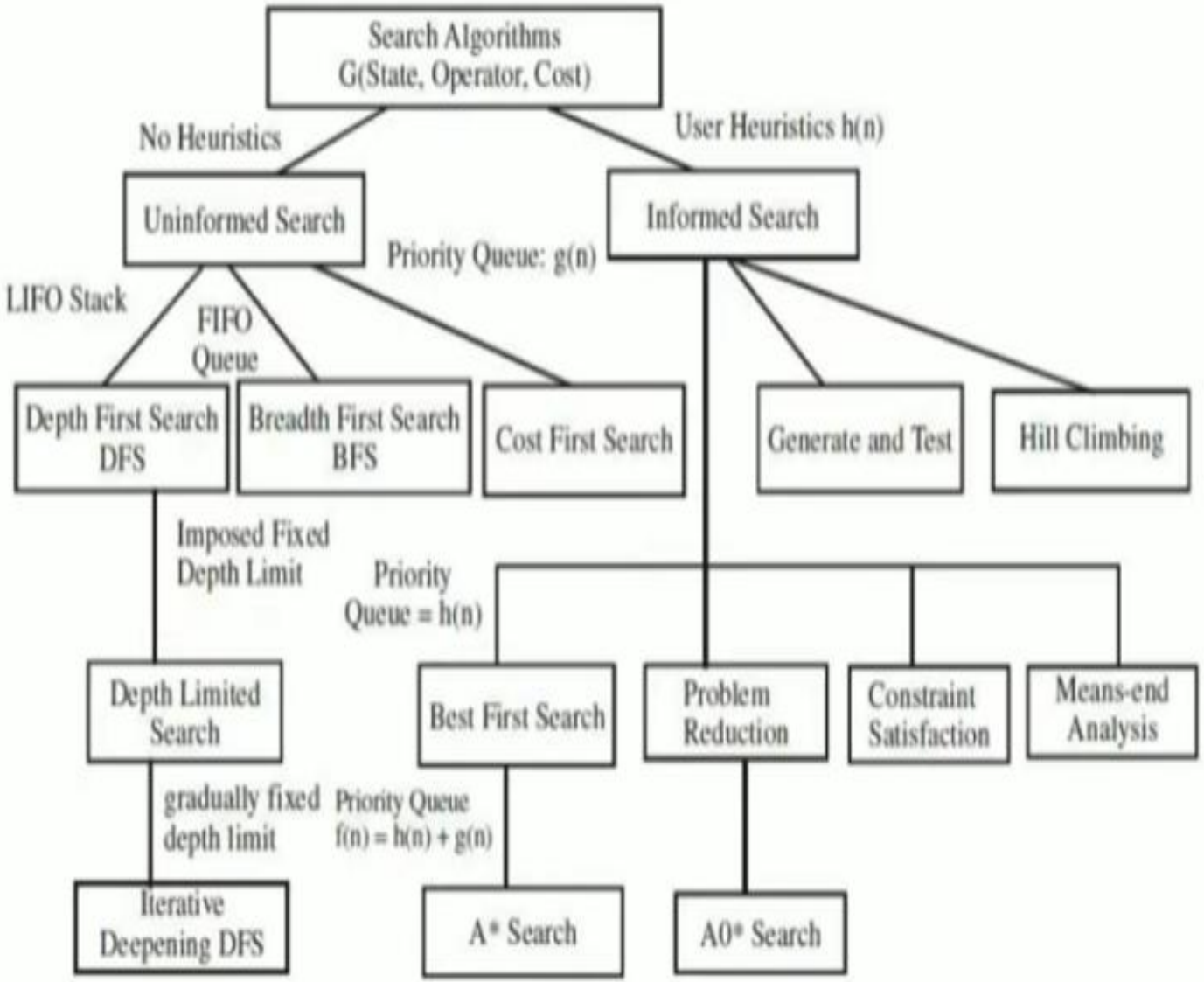
- BFS
- DFS
- Bidirectional search

UNINFORMED AND INFORMED SEARCH

Algorithms that are given no information about the problem other than its definition are **uninformed search algorithms**

Informed search algorithms are given with some guidance on where to look for solutions.

Types of Search Algorithms in Artificial Intelligence



Best-First Search in Artificial Intelligence

- Best First Search algorithm combines the advantages of both DFS and BFS into a single method.
- At each step of the BFS search process, we select the most promising of the nodes we have generated so far.
- This is done by applying an appropriate heuristic function to each of them.
- We then expand the chosen node by using the rules to generate its successors

Best-First Search in Artificial Intelligence

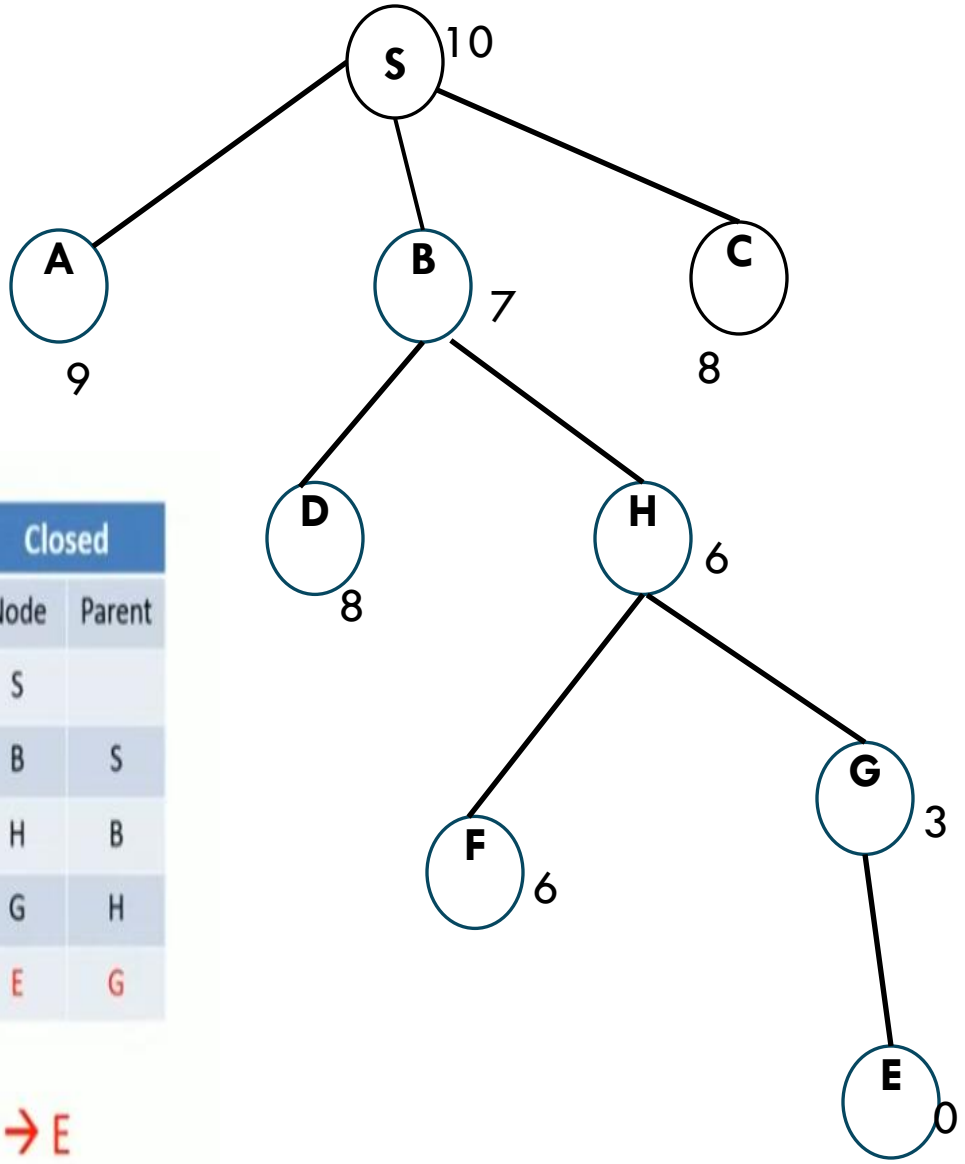
- To implement such a graph-search procedure, we will need to use two lists of nodes:
  - **OPEN** — nodes that have been generated and have had the heuristic function applied to them, but which have not yet been examined (i.e., had their successors generated).
  - **CLOSED** — nodes that have already been examined. We need to keep these nodes in memory if we want to search a graph rather than a tree, since whenever a new node is generated, we need to check whether it has been generated before.

Heuristic Search – Best First Search Algorithm

- Create 2 empty lists : OPEN and CLOSED
- Start from the initial node(say N) and put it in the ordered OPEN list
- Repeat the next steps until the GOAL node is reached OR OPEN list is empty

1. Select the best node(say N) in the OPEN list and move it to the CLOSED list. Also,capture the information of the parent node.
2. If N is a GOAL node, then exit the loop returning 'True'. The solution can be found by backtracking the path.
3. If N is not the GOAL node, expand node N to generate the 'immediate' next nodes linked to node N and add all these to the OPEN list.

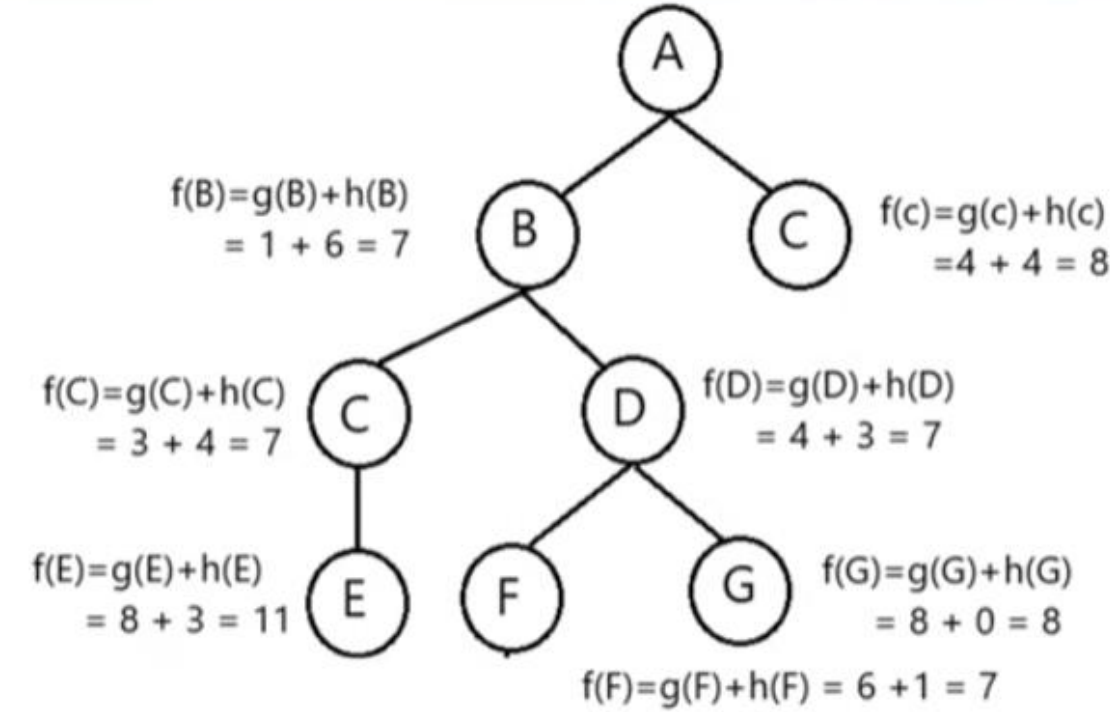
- Reorder the nodes in the OPEN list in ascending order.



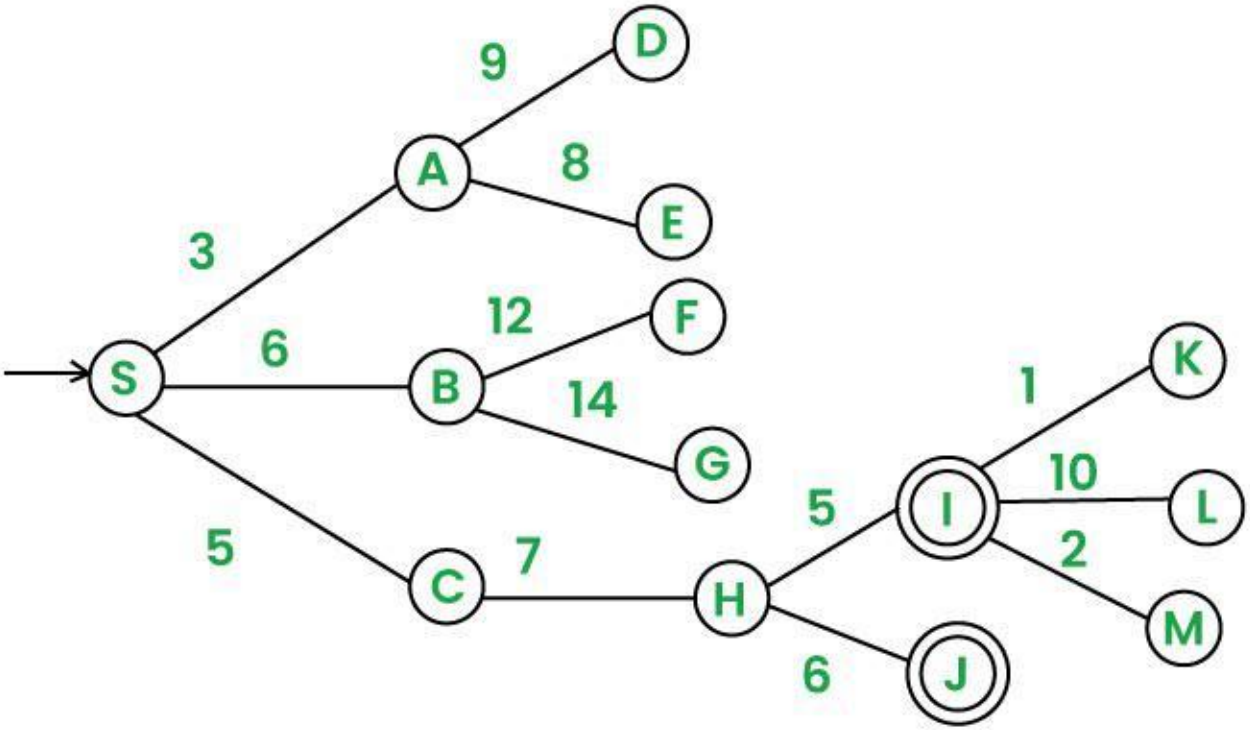
Open			Closed	
Node	H(n)		Node	Parent
F	6		S	
C	8		B	S
D	8		H	B
A	9		G	H
			E	G

S → B → H → G → E

Path: A→B→D→F→G
Path Cost: 7



SOLVE USING BEST FIRST SEARCH



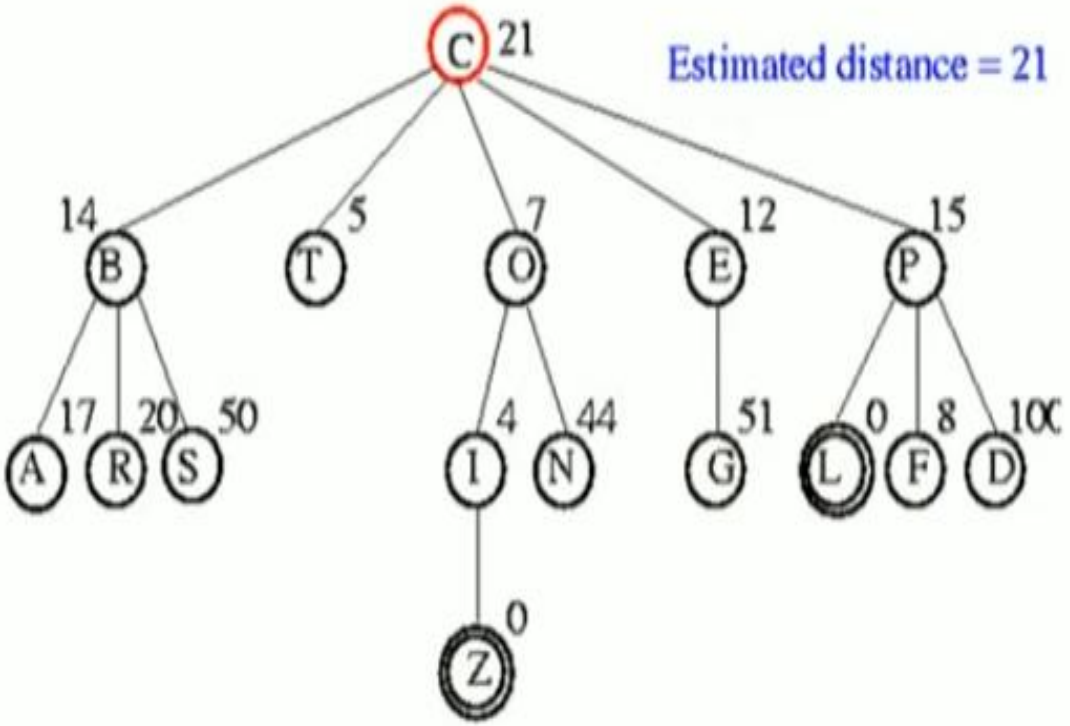
Best-First Search – Solved Example

Here **C** is the initial or source node and **L** and **Z** are goal nodes.

Open: O, E, B, P
Closed: C, T

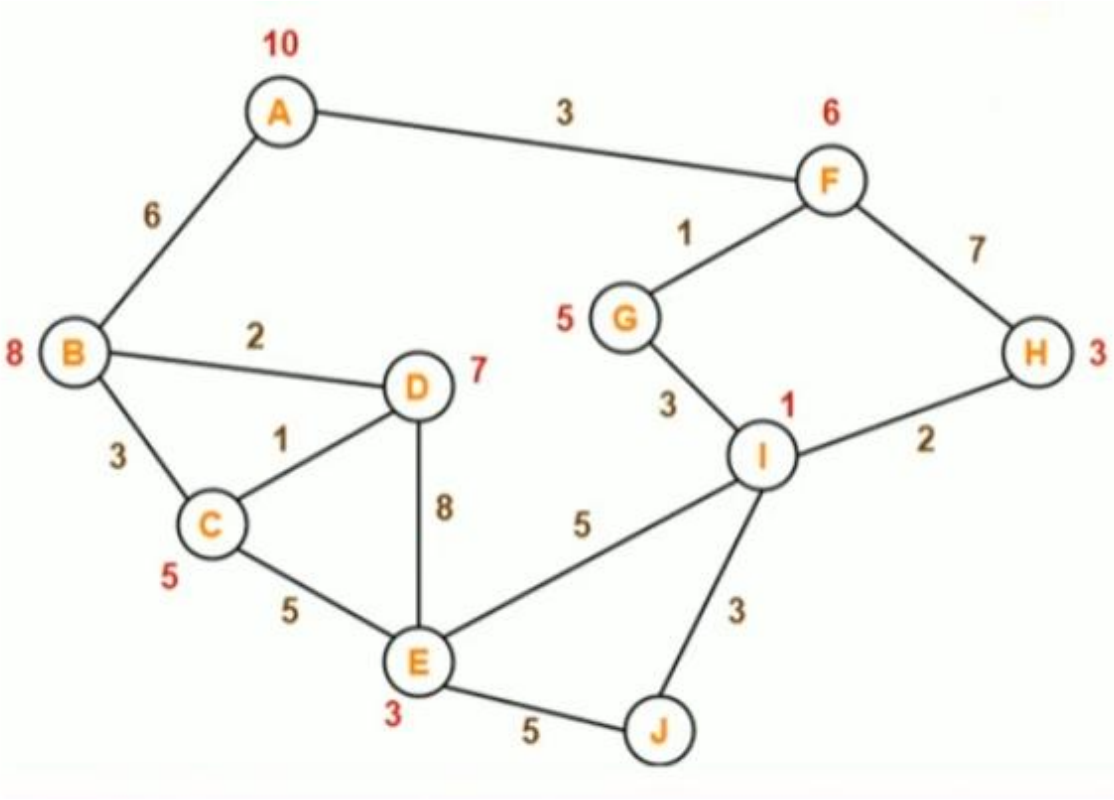
Open: E, B, P
Closed: C, T, O

Open: E, B, P, N
Closed: C, T, O, I



The Goal is found. The final path is C – O – I – Z

A* Search Algorithm



- The numbers written on edges represent the distance between the nodes.
- The numbers written on nodes represent the heuristic value.
- Find the most cost-effective path to reach from start state A to final state J using A* Algorithm.

Step-01:

- We start with node A.
- Node B and Node F can be reached from node A.
- A* Algorithm calculates

$$f(n) = g(n) + h(n)$$

- $f(B) = g(B) + h(B) = 6 + 8 = 14$
- $f(F) = g(F) + h(F) = 3 + 6 = 9$
- Since $f(F) < f(B)$, so it decides to go to node F.
- Path- A → F

Step-02:

- Node G and Node H can be reached from node F.
- A* Algorithm calculates

$$f(n) = g(n) + h(n)$$

- $f(G) = g(G) + h(G) = (3+1) + 5 = 9$
- $f(H) = g(H) + h(H) = (3+7) + 3 = 13$

Step-03:

- Node I can be reached from node G.
- A* Algorithm calculates

$$f(n) = g(n) + h(n)$$

- $f(I) = g(I) + h(I)$

  $= (3+1+3) + 1 = 8$

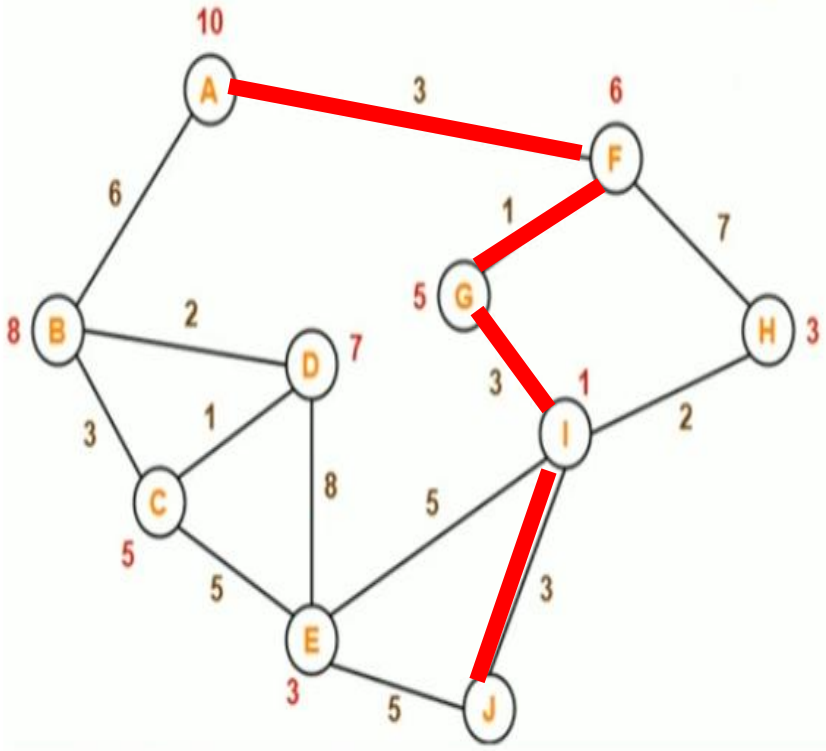
- So it decides to go to node I.
- Path - A → F → G → I

Step-04:

- Node E, Node H and Node J can be reached from node I.
- A* Algorithm calculates

$$f(n) = g(n) + h(n)$$

- f(E) = (3+1+3+5) + 3 = 15
- f(H) = (3+1+3+2) + 3 = 12
- f(J) = (3+1+3+3) + 0 = 10
- Since f(J) is least, so it decides to go to node J.
- Path - A → F → G → I → J



AO* Search Algorithm in Artificial Intelligence

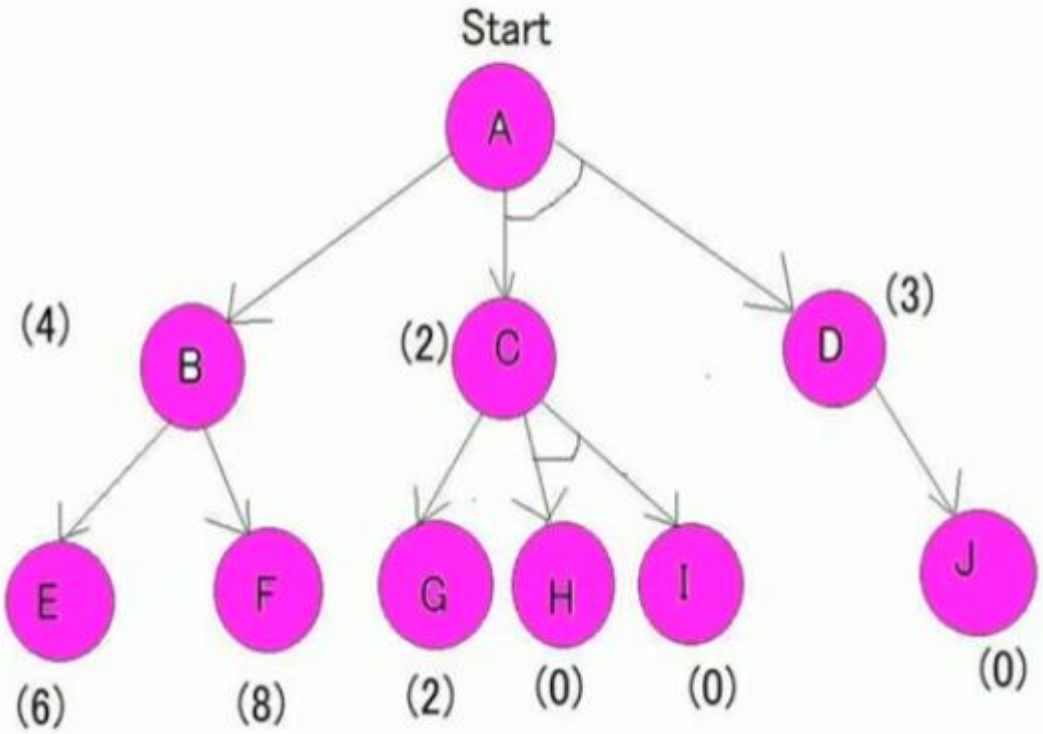
- AO* algorithm is a heuristic search algorithm in AI.
- AO* algorithm uses the concept of AND-OR graphs to decompose any complex problem given into smaller set of problems which are further solved.
- **Working of AO* algorithm:**
- The AO* algorithm works on the formula given below :

$$f(n)=g(n)+h(n)$$

- where,
- $g(n)$: The actual cost of traversal from initial state to the current state.
- $h(n)$: The estimated cost of traversal from the current state to the goal state.
- $f(n)$: The actual cost of traversal from the initial state to the goal state.

AO* Search Algorithm in Artificial Intelligence

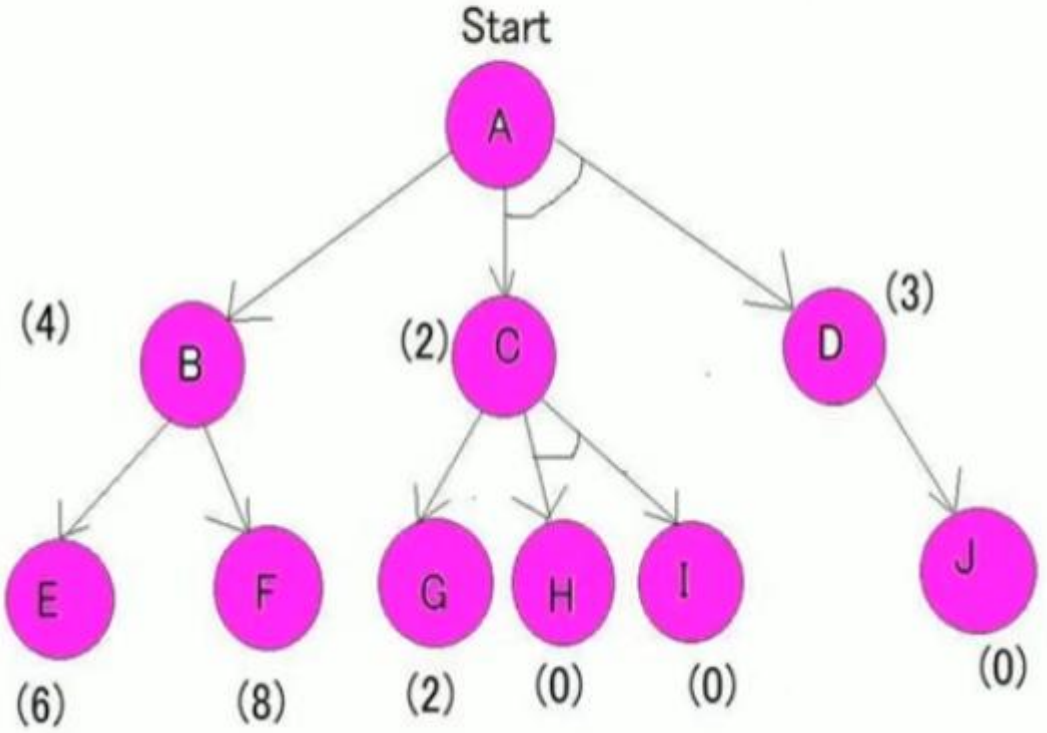
- Here, in the above example all numbers in brackets are the heuristic value i.e h(n).
- Each edge is considered to have a value of 1 by default.



AO* Search Algorithm in Artificial Intelligence

Step-1

- Starting from node A, we first calculate the best path.
- **f(A-B) = g(B) + h(B) = 1+4= 5** , where 1 is the default cost value of travelling from A to B and 4 is the estimated cost from B to Goal state.
- **f(A-C-D) = g(C) + h(C) + g(D) + h(D) = 1+2+1+3 = 7**, here we are calculating the path cost as both C and D because they have the AND-Arc.
- The default cost value of travelling from A-C is 1, and from A-D is 1, but the heuristic value given for C and D are 2 and 3 respectively hence making the cost as 7.



AO* Search Algorithm in Artificial Intelligence

Step-2

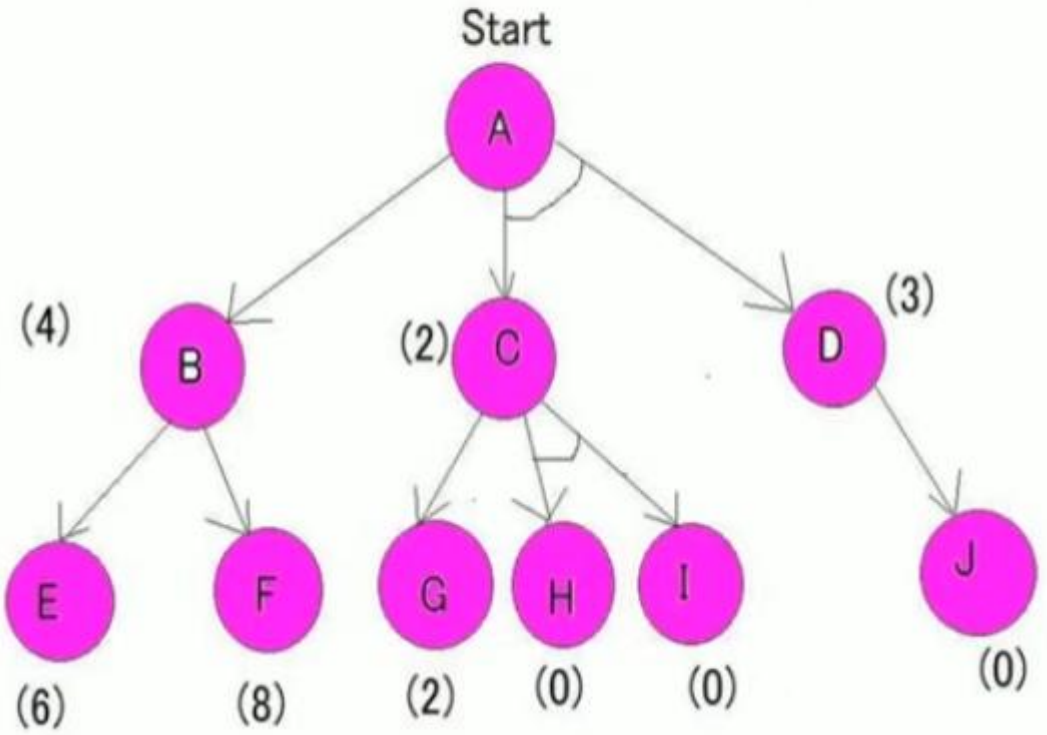
- From the B node,

  f(B-E) = 1 + 6 = 7

  f(B-F) = 1 + 8 = 9

- Hence, the B-E path has lesser cost. Now the heuristics have to be updated since there is a difference between actual and heuristic value of B.
- The minimum cost path is chosen and is updated as the heuristic , in our case the value is 7.
- And because of change in heuristic of B there is also change in heuristic of A which is to be calculated again.

  f(A-B) = g(B) + updated((h(B)) = 1+7=8



AO* Search Algorithm in Artificial Intelligence

Step-3

- Now the current node becomes C node and the cost of the path is calculated,

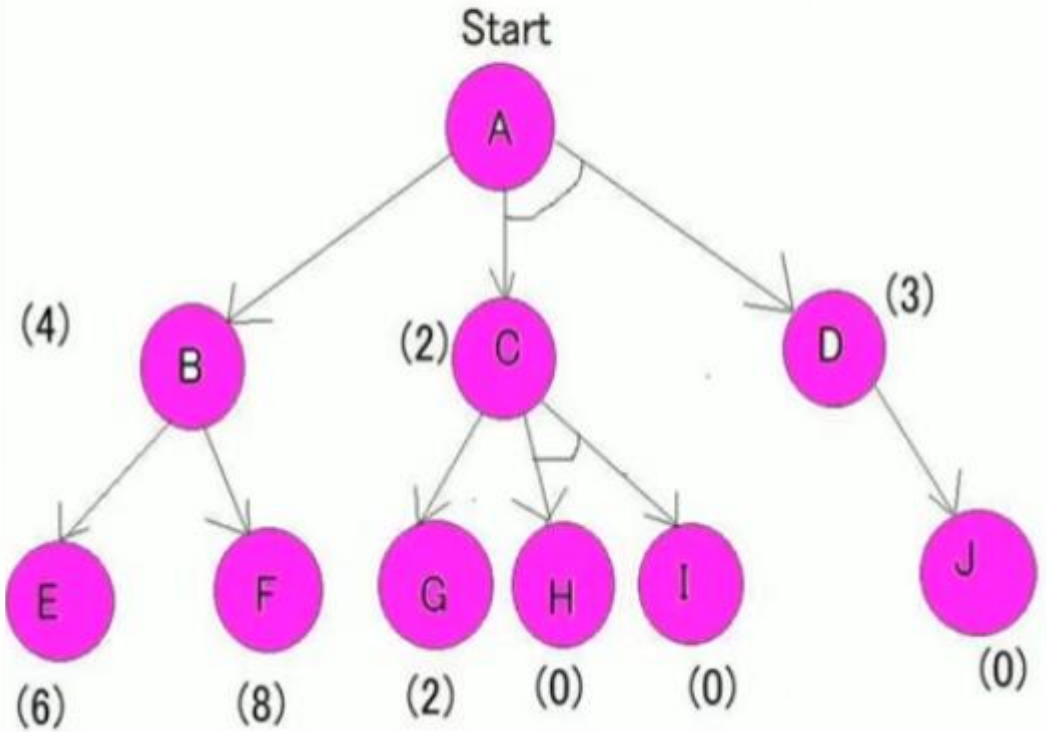
  f(C-G) = 1+2 = 3

  f(C-H-I) = 1+0+1+0 = 2

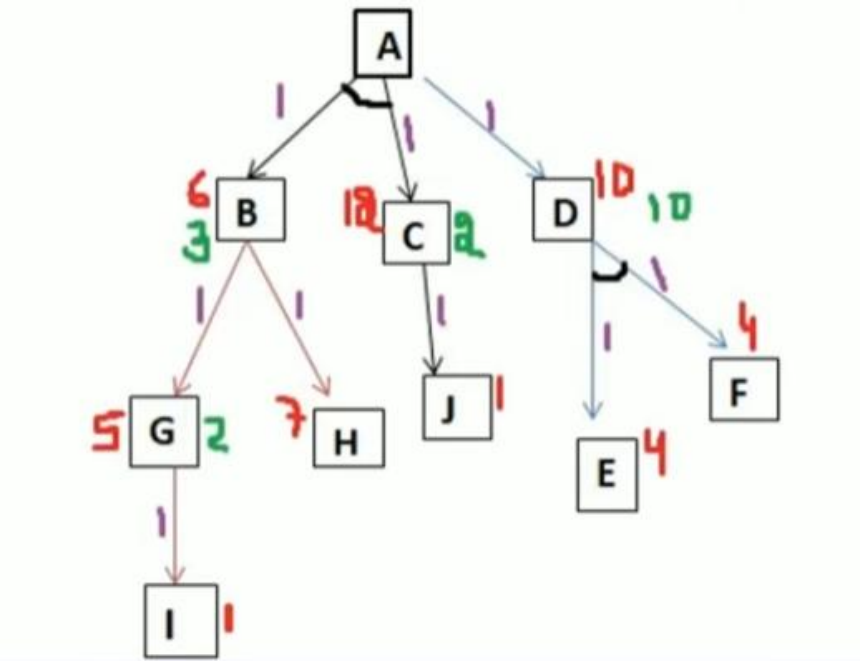
  f(C-H-I) is chosen as minimum cost path.
- Heuristic of path of H and I are 0 and hence they are solved,
- But Path A-D also needs to be calculated , since it has an AND-arc.
- **f(D-J) = 1+0 = 1**, hence heuristic of D needs to be updated to 1.
- And finally the f(A-C-D) needs to be updated.

  f(A-C-D) = g(C) + h(C) + g(D) + updated((h(D))

  = 1+2+1+1 =5.



PROBLEM 2:SOLVE USING AO* SEARCH ALGORITM.



- In the diagram we have two ways from **A to D** or **A to B-C** (because of and condition).
- Calculate cost to select a path
- **F(A-D) = 1+10 = 11**
- **F(A-BC) = 1 + 1 + 6 +12 = 20**
- As we can see **F(A-D)** is less than **F(A-BC)** then the algorithm choose the path **F(A-D).**

- Form D we have one choice that is **F-E.**
- **F(A-D-FE) = 1+1+ 4 +4 =10**
- Basically **10** is the cost of reaching **FE from D.**
- And **Heuristic value of node D** also denote the cost of reaching **FE from D**.
- So, the new Heuristic value of D is 10.
- And the Cost from A-D remain same that is **11**.

- Let's take a look at F(A-BC)
- Now from B we have two path G and H , let's calculate the cost
- F(B-G)= 5+1 =6 and F(B-H)= 7 + 1 = 8
- So, cost from F(B-H) is more than F(B-G) we will take the path B-G.

1. The Heuristic value from **G to I is 1 but let's calculate the cost form G to I.**
2. **F(G-I) = 1 +1 = 2.** which is less than **Heuristic value 5**. So, the new **Heuristic value form G to I is 2.**

1. **But A is associated with both B and C .**
2. As we can see from the diagram **C only have one choice or one node to explore that is J.** The Heuristic value of C is 12.
3. Cost form C to **J= F(C-J) = 1+1= 2** Which is less than Heuristic value
4. No**w the New Heuristic value of C is 2.**

1. **And the New Cost from A- BC that is**
2. **F(A-BC) = 1+1+2+3 = 7**
3. **which is less than F(A-D)=11.**
4. In this case Choosing path A-BC is more cost effective and good than that of A-D.