Problem
Problem State space
Search

Prof. Deepak C Vegda
email.:deepakvegda.it@ddu.ac.in

To build a system to solve a problem :

1. Define the problem precisely.
2. Analyze the problem.
3. Isolate and represent the task knowledge that is necessary to solve the problem.
4. Choose the best problem solving technique(s) and apply it (them) to particular problem.

Define the problem

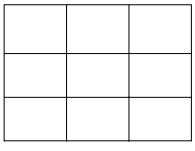
- Start State (Initial state)
- Rules that define the legal moves.
- Board position that represent Win for one or the other.
- Goal State (Final State).

What is state?

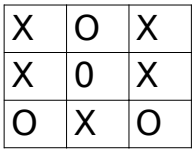
What is state space?

Tic Tac Toe

- The game is played on a grid that's 3 squares by 3 squares.
- You are X, your friend (or the computer in this case) is O. **Players** take turns putting their marks in empty squares.
- The first **player** to get 3 of her marks in a row (up, down, across, or diagonally) is the winner.
- When all 9 squares are full, the game is over.



Start State



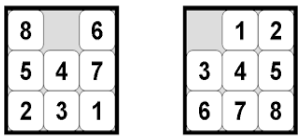
Goal state

Water Jug Problem

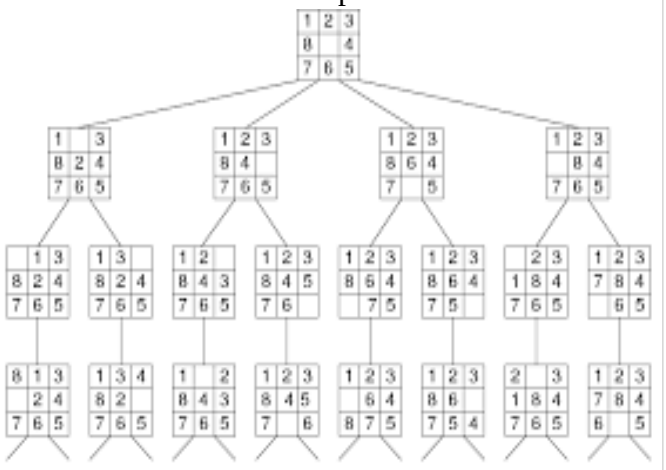
- You are given two jugs, a 4-gallon one and a 3-gallon one. Neither has any measuring mark on it. There is a pump that can be used to fill the jugs with **water**.

Eight tile puzzle

Given a 3×3 board with **8 tiles** (every **tile** has one number from 1 to **8**) and one empty space. The objective is to place the numbers on **tiles** to match final configuration using the empty space. We can slide four adjacent (left, right, above and below) **tiles** into the empty space.



State Space



State space representation

- Before a solution can be found, the prime condition is that the problem must be very precisely defined. By defining it properly, one converts the abstract problem into real workable states that are really understood.
- A set of all possible states for a given problem is known as the **state space of the problem. State space representations are highly beneficial in AI because they provide all possible states, operations and goals.**
- If the entire state space representations for a problem is given, it is possible to trace the path from the initial state to the goal state and identify the sequence of operators necessary for doing it.
- The major deficiency of this method is that it is not possible to visualize all states for a given problem. Moreover, the resources of the computer system are limited to handle huge state-space representation.

Production system

Since search forms the core of many intelligent processes, it is useful to structure AI programs in a way that facilitates describing and performing the search process. Production systems provide such structures.

A production system consist of:-

1. A set of rules, each consisting of a left side (pattern) that determines the applicability of the rule and a right side describing the operation to be performed.
2. One or more knowledge/databases that contain whatever information is appropriate for the particular task.
3. A control strategy that specifies the order in which the rules will be compared to the database and a way of resolving the conflicts that arise when several rules match at once.
4. A rule applier which is the computational system that implements the control strategy and applies the rules

Control Strategies

- How to decide which rule to apply next during the process of searching for a solution to a goal?
- Control strategies help us to overcome the abnormal situations, when there are more than one rules or fewer than one rule will have its left sides match the current state.
- Requirement for control strategy
  1. **A good control strategy causes motion**
  2. **A good control strategy is systematic**

Problem Characteristics

To choose an appropriate method for a particular problem:

1. Is the problem decomposable?
2. Can solution steps be ignored or undone?
3. Is the universe predictable?
4. Is a good solution absolute or relative?
5. Is the solution a state or a path?
6. What is the role of knowledge?
7. Does the task require human-interaction?

1. Is the problem decomposable?

- Can the problem be broken down to smaller problems to be solved independently?
- Decomposable problem can be solved easily.

1. Is the problem decomposable?

Is the problem decomposable?

1. Is the problem decomposable?

- 8-tile puzzle
- Tic-tac-toe
- Block word problem
- Water Jug problem

2. Can solution steps be ignored or undone?

8-tile puzzle

1	4	6
2		3
8	7	5

1		6
2	4	3
8	7	5

2. Can solution steps be ignored or undone?

8-tile puzzle

1	4	6
2		3
8	7	5

1		6
2	4	3
8	7	5

Recoverable!!

2. Can solution steps be ignored or undone?

Theorem Proving

A lemma that has been proved can be ignored for next steps.

Ignorable!

2. Can solution steps be ignored or undone?

Chess???

Irrecoverable !!

2. Can solution steps be ignored or undone?

Ignorable problems can be solved using a simple control structure that never backtracks.

Recoverable problems can be solved using backtracking.

Irrecoverable problems can be solved by recoverable style methods via planning.

3. Is the universe predictable?

The 8-Puzzle

Every time we make a move, we know exactly what will happen.

Certain outcome!

3. Is the universe predictable?

For certain-outcome problems, planning can used to generate a sequence of operators that is guaranteed to lead to a solution.

For uncertain-outcome problems, a sequence of generated operators can only have a good probability of leading to a solution.

Plan revision is made as the plan is carried out and the necessary feedback is provided.

4. Is a good solution absolute or relative?

1. Marcus was a man.
2. Marcus was a Pompeian.
3. Marcus was born in 40 A.D.
4. All men are mortal.
5. All Pompeians died when the volcano erupted in 79 A.D.
6. No mortal lives longer than 150 years.
7. It is now 2004 A.D.

4. Is a good solution absolute or relative?

1. Marcus was a man.
2. Marcus was a Pompeian.
3. Marcus was born in 40 A.D.
4. All men are mortal.
5. All Pompeians died when the volcano erupted in 79 A.D.
6. No mortal lives longer than 150 years.
7. It is now 2004 A.D.

Is Marcus Alive Now?

5. Is the solution a state or a path?

A path-solution problem can be reformulated as a state-solution problem by describing a state as a partial path to a solution.

The question is whether that is natural or not

6. What is the role of knowledge

Playing Chess

Knowledge is important only to constrain the search for a solution.

Reading Newspaper

Knowledge is required even to be able to recognize a solution.

7. Does the task require human-interaction?

Solitary problem, in which there is no intermediate communication and no demand for an explanation of the reasoning process.

Conversational problem, in which intermediate communication is to provide either additional assistance to the computer or additional information to the user.

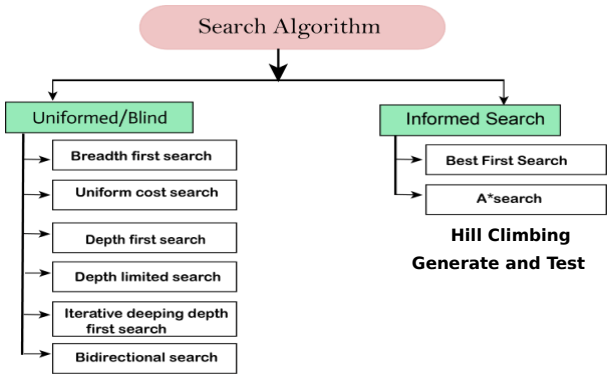
Classes of Production System

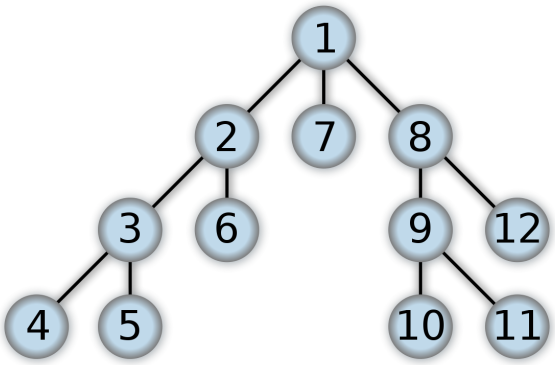
- **Monotonic production system:**
  the application of a rule never prevents the later application of another rule that could also have been applied at the time the first rule was selected.
- **Non-monotonic production system:**
  Is one in which this is not true.
- **Partially commutative production system:**
  the application of a particular sequence of rules transforms state x into state y, then any permutation of those rules that is allowable also transforms state x into state y.
- **Commutative production system:**
  system that is both monotonic and partially commutative.

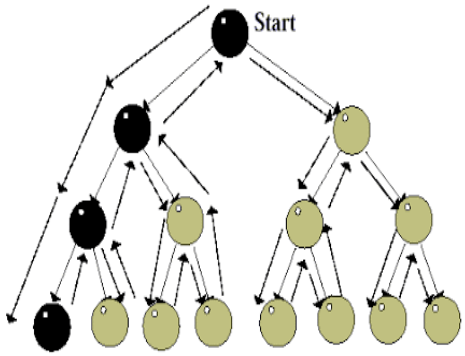
Characteristics of an algo

- How does it work?
- Complete?
- Optimal?
- Time complexity?
- Space complexity?

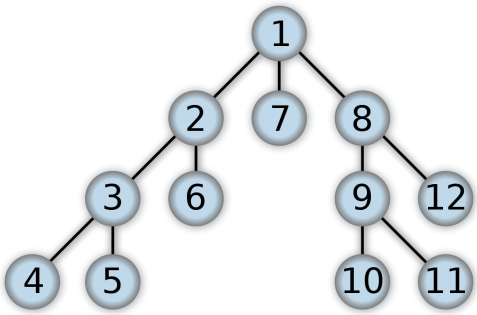
Classification of Search techniques



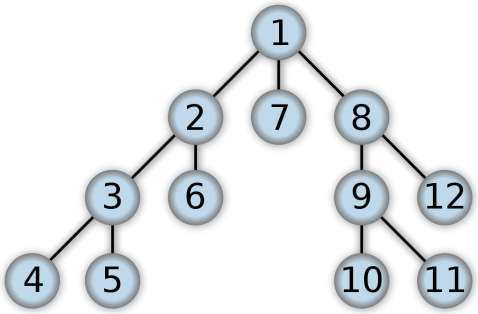




Complete?



Optimal?



DFS

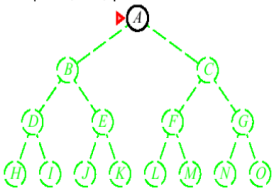
- The search begins by expanding the initial node, generate all successors of the initial node and test them.
- Depth-first search always expands the deepest node in the current frontier of the search tree.
- Depth-first search uses a **LIFO approach.**

Depth-first search

Expand deepest unexpanded node

Implementation:

fringe = LIFO queue, i.e., put successors at front

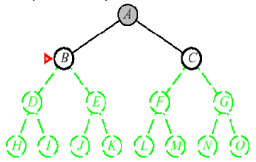


Depth-first search

Expand deepest unexpanded node

Implementation:

fringe = LIFO queue, i.e., put successors at front

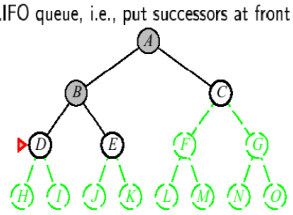


Depth-first search

Expand deepest unexpanded node

Implementation:

fringe = LIFO queue, i.e., put successors at front

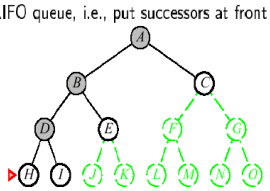


Depth-first search

Expand deepest unexpanded node

Implementation:

fringe = LIFO queue, i.e., put successors at front

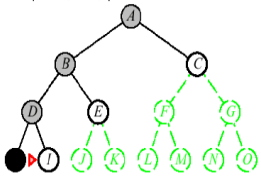


Depth-first search

Expand deepest unexpanded node

Implementation:

fringe = LIFO queue, i.e., put successors at front

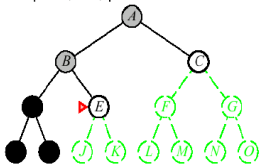


Depth-first search

Expand deepest unexpanded node

Implementation:

fringe = LIFO queue, i.e., put successors at front

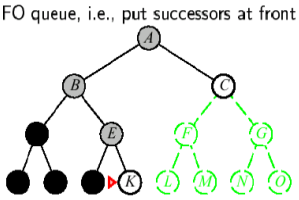


Depth-first search

Expand deepest unexpanded node

Implementation:

fringe = LIFO queue, i.e., put successors at front

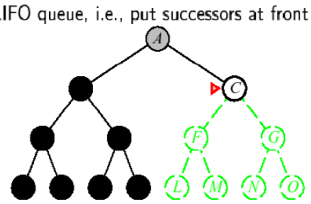


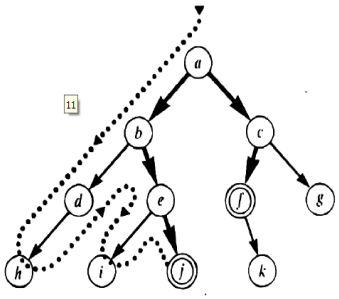
Depth-first search

Expand deepest unexpanded node

Implementation:

fringe = LIFO queue, i.e., put successors at front





Algorithm for Depth First Search

1. If the initial state is a goal state, quit and return success.
2. Otherwise, do the following until success or failure is signaled:
   a) Generate a successor, E, of the initial state. If there are no more successors, signal failure.
   b) Call Depth-First Search with E as the initial state.
   c) If success is returned, signal success. Otherwise continue in this loop.

Time and space complexity

Time Complexity :

$1 + b + b^2 + b^3 + \ldots + \ldots\ldots b^d$.

Hence Time complexity = O (b^d)

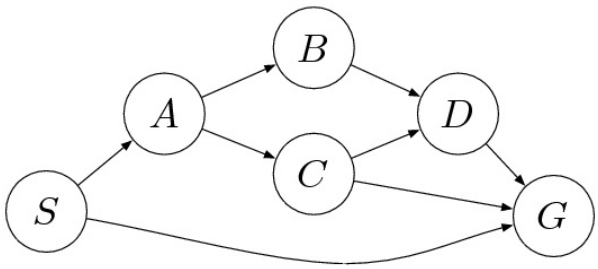
Where b-> branching factor

d-> Depth of a tree

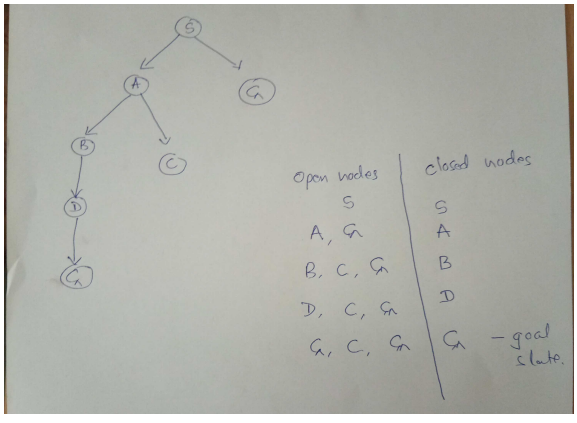
Space Complexity :

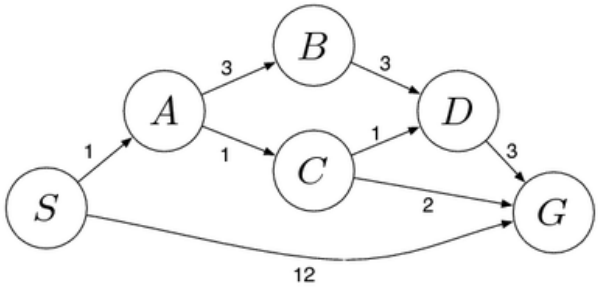
Hence Space complexity = O (d)

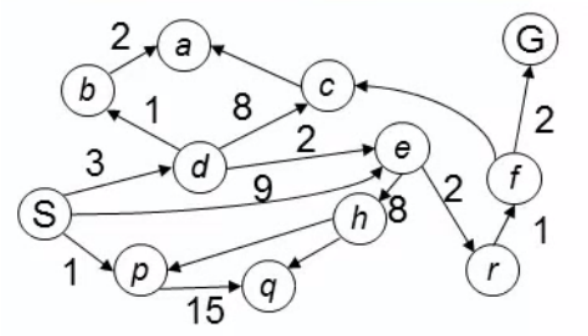
Example



Solution







Advantages of Depth-First Search

i. It requires less memory since only the nodes of the current path are stored.
ii. By chance, it may find a solution without examining much of the search space at all.

Disadvantages of Depth-First Search

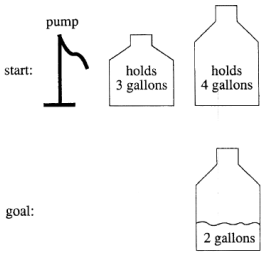
i. Determination of the depth until which the search has to proceed. This depth is called cut-off depth.
ii. If the cut-off depth is smaller, solution may not be found.
iii. If cut-off depth is large, time complexity will be more.
iv. And there is no guarantee to find a minimal solution, if more than one solution exists.

A Water Jug Problem

• You have a 4-gallon and a 3-gallon water jug

• You have a pump with an unlimited amount of water

• You need to get exactly 2 gallons in 4-gallon jug



Representation of the problem

- State representation: **(x, y)**
  - x: Contents of four gallon
  - y: Contents of three gallon
- Start state: **(0, 0)**
- Goal state **(2, n)**
- Operators
  - Fill 3-gallon from pump, fill 4-gallon from pump
  - Fill 3-gallon from 4-gallon , fill 4-gallon from 3-gallon
  - Empty 3-gallon into 4-gallon, empty 4-gallon into 3-gallon
  - Dump 3-gallon down drain, dump 4-gallon down drain

State Space Search: Water Jug Problem...

1. (x, y) if $x < 4$ → (4, y) if x is less than 4, fill the 4 gallon jug
2. (x, y) if $y < 3$ → (x, 3) fill the 3-gallon jug
3. (x, y) if $x > 0$ → (x – d, y) pour some water out of the 4 gallon jug
4. (x, y) if $y > 0$ → (x, y – d) pour some water out of the 3 gallon jug
5. (x, y) if $x > 0$ → (0, y) empty the 4-gallon jug on the ground
6. (x, y) if $y > 0$ → (x, 0) empty the 3-gallon jug on the ground
7. (x, y) if $x + y \geq 4, y > 0$ → $(4, y - (4 - x))$ pour water from the 3- gallon jug into the 4-gallon jug until the 4-gallon jug is full
8. (x, y) if $x + y \geq 3, x > 0$ → $(x - (3 - y), 3)$ pour water from the 4- gallon jug into the 3-gallon jug until the 3-gallon jug is full
9. (x, y) if $x + y \leq 4, y > 0$ → $(x + y, 0)$ pour all the water from the 3-gallon jug into the 4-gallon jug
10. (x, y) if $x + y \leq 3, x > 0$ → $(0, x + y)$ pour all the water from the 4-gallon jug into the 3-gallon jug
11. (0, 2) → (2, 0) pour the 2 gallons from the 3-gallon jug into the 4-gallon jug
12. (2, y) → (0, y) empty the 2 gallons in the 4 gallon jug on the ground

Example