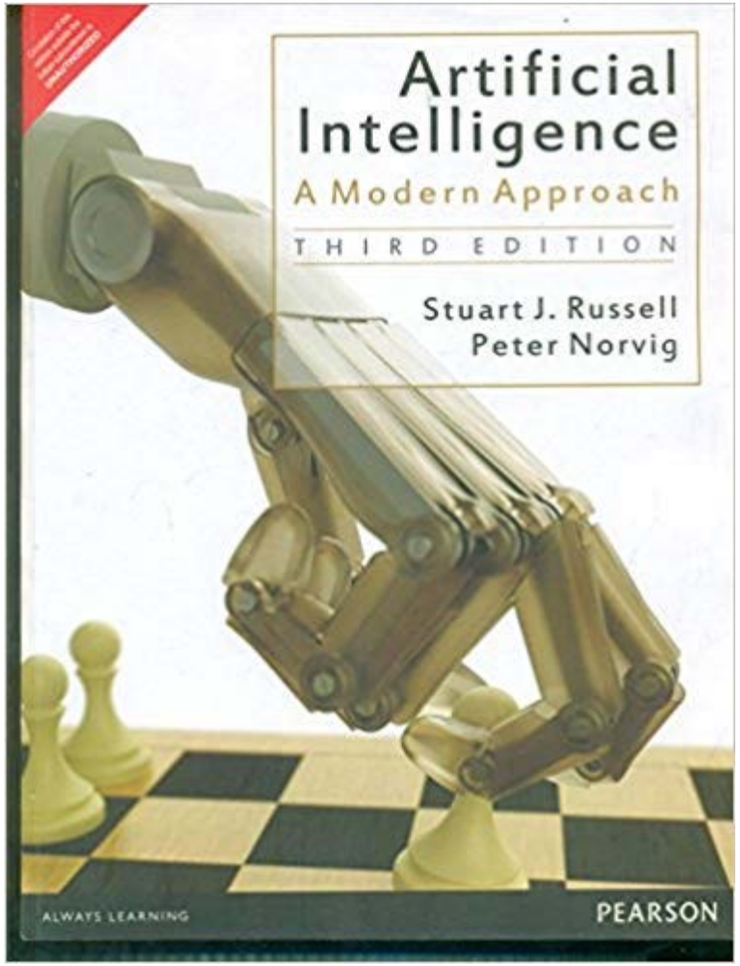
Automated Problem Solving by Search

COURSE: CS60045

Pallab Dasgupta
Professor,
Dept. of Computer Sc & Engg

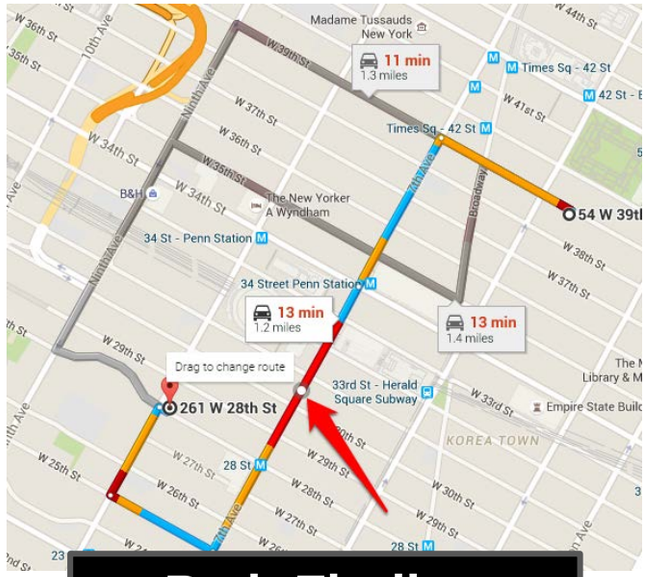


The book that we will follow mostly for this and many other topics:

Artificial Intelligence – A Modern Approach
Stuart J Russell, Peter Norvig

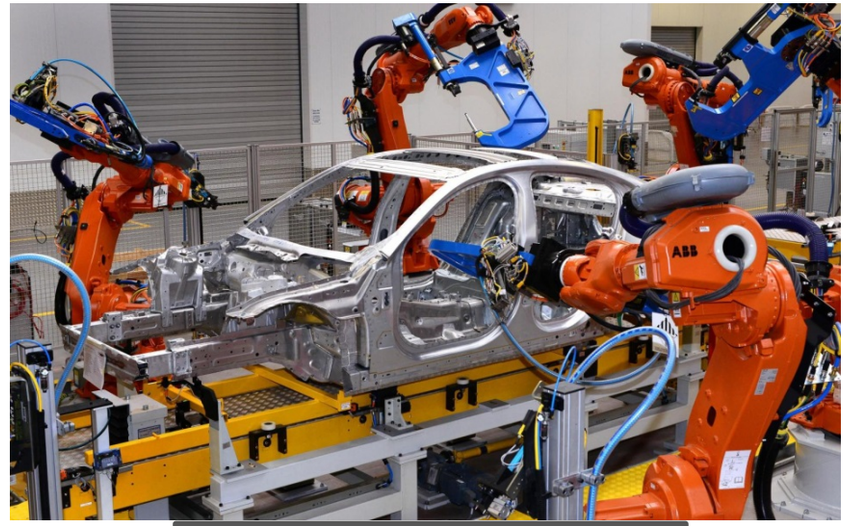
Pearson Education India

COMPLEX PROBLEMS AND ALGORITHMS

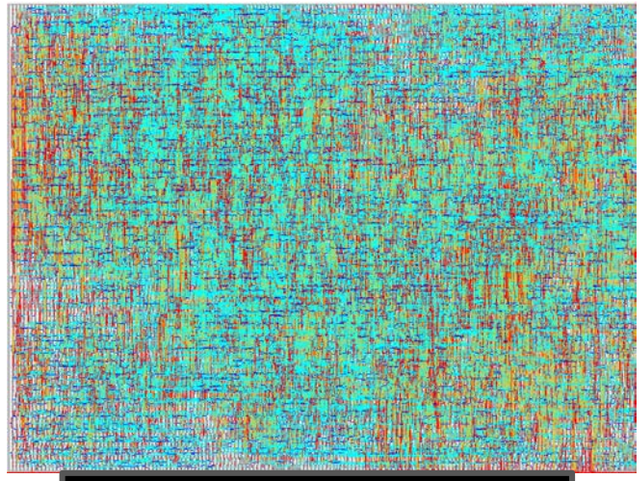


Path Finding

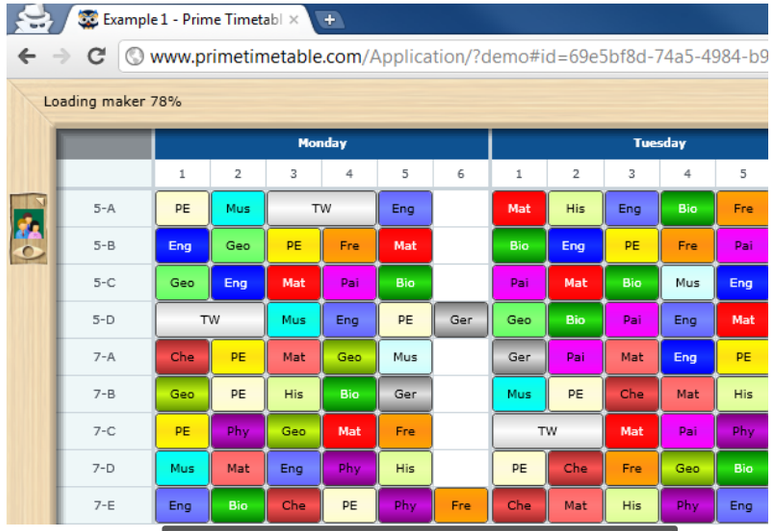
Chess Playing



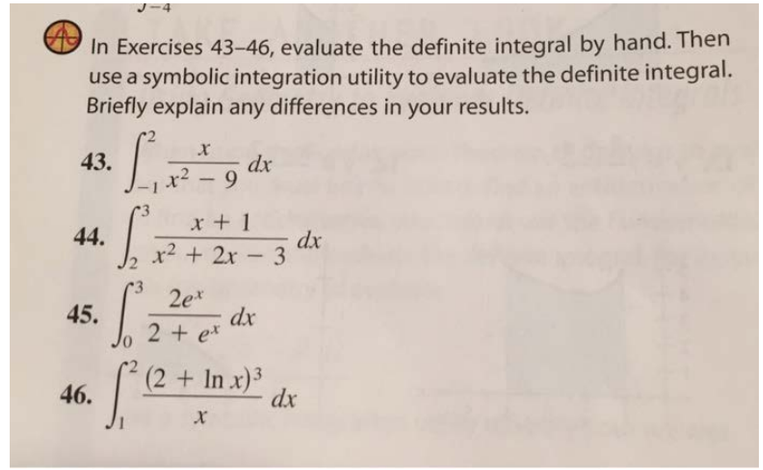
Robot Assembly



VLSI Chip Design



Time-Table Scheduling



In Exercises 43–46, evaluate the definite integral by hand. Then use a symbolic integration utility to evaluate the definite integral. Briefly explain any differences in your results.

43. $\int_{-1}^{2} \frac{x}{x^2 - 9}\,dx$

44. $\int_{2}^{3} \frac{x + 1}{x^2 + 2x - 3}\,dx$

45. $\int_{0}^{3} \frac{2e^x}{2 + e^x}\,dx$

46. $\int_{1}^{2} \frac{(2 + \ln x)^3}{x}\,dx$

Symbolic Integration

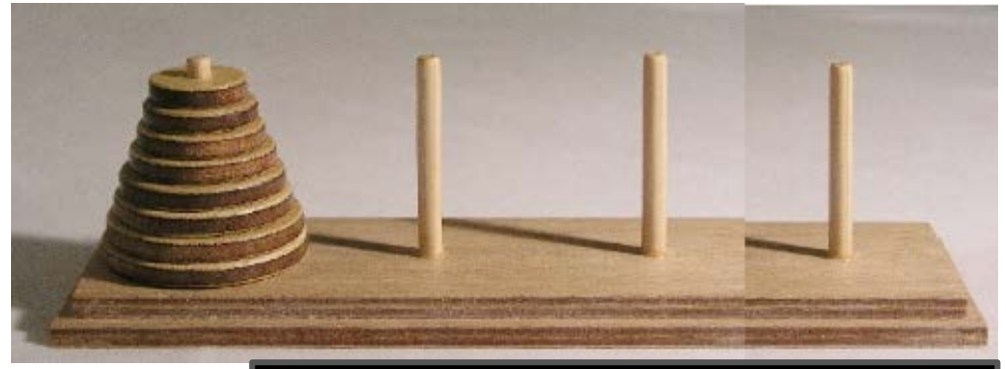
AUTOMATED PROBLEM SOLVING BY SEARCH

Generalized Techniques for Solving Large Classes of Complex Problems

Problem Statement is the Input and solution is the Output, sometimes even the problem specific algorithm or method could be the Output

Problem Formulation by AI Search Methods consists of the following key concepts

- Configuration or State
- Constraints or Definitions of Valid Configurations
- Rules for Change of State and their Outcomes
- Initial or Start Configurations
- Goal Satisfying Configurations
- An Implicit State or Configuration Space
- Valid Solutions from Start to Goal in the State Space
- General Algorithms which SEARCH for Solutions in this State Space



Multi-Peg Tower of Hanoi / Brahma

Issues

- Size of the Implicit Space, Capturing Domain Knowledge, Intelligent Algorithms that work in reasonable time and Memory, Handling Incompleteness and Uncertainty

TWO JUG PROBLEM

There is a large bucket B full of water and Two (02) jugs, J1 of volume 3 litre and J2 of volume 5 litre. You are allowed to fill up any empty jug from the bucket, pour all water back to the bucket from a jug or pour from one jug to another. The goal is to have jug J1 with exactly one (01) litre of water

State Definition: <J1, J2>

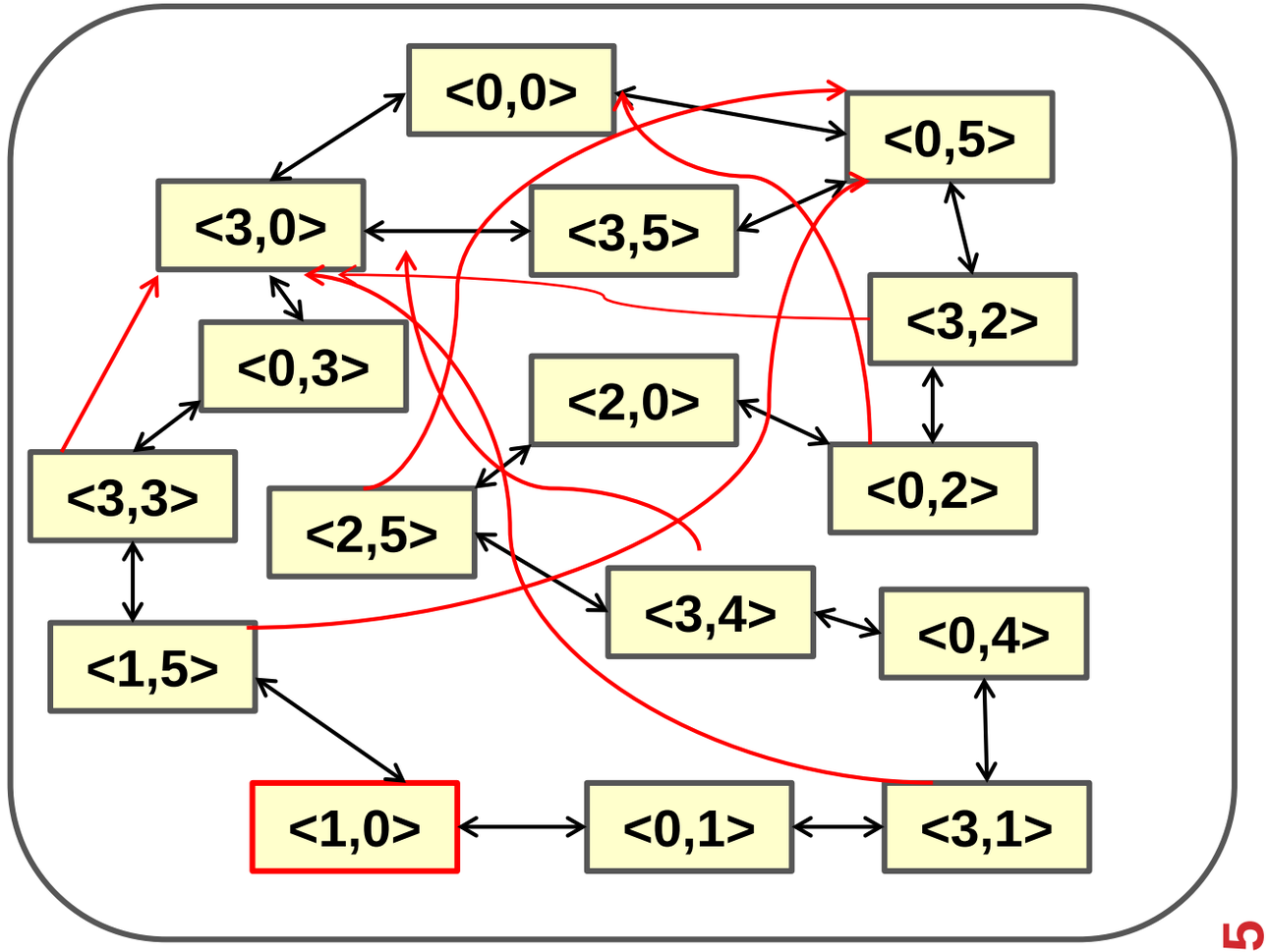
Rules:

- Fill (J1): <J1, J2> to <3,J2>
- Fill (J2): <J1, J2> to <J1, 5>
- Empty (J1), Empty (J2): Similarly defined
- Pour (J1, J2): <J1, J2> to <X,Y>, where
  - **X = 0 and Y = J1 + J2 if J1+J2 ≤ 5,**
  - Y = 5 and X = (J1+ J2) – 5, if J1+J2 > 5
- Pour (J2, J2): Similarly defined

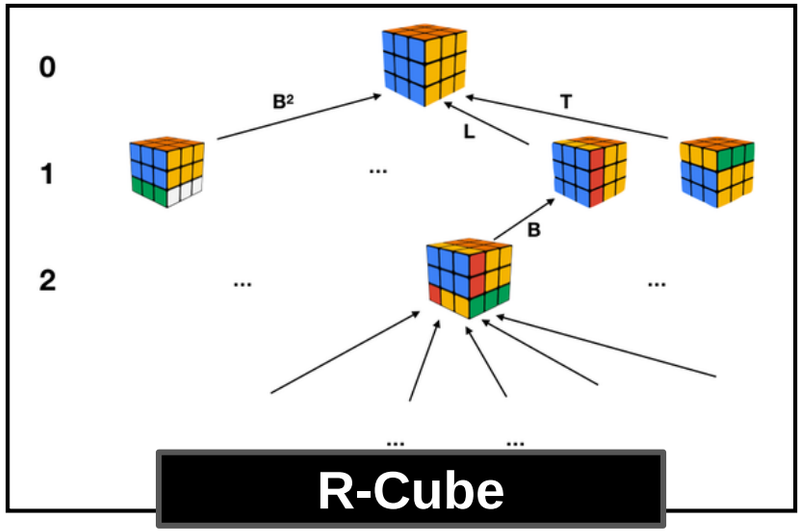
Start: <0,0>, Goal: <1,0>

Part of State Space Shown on the right

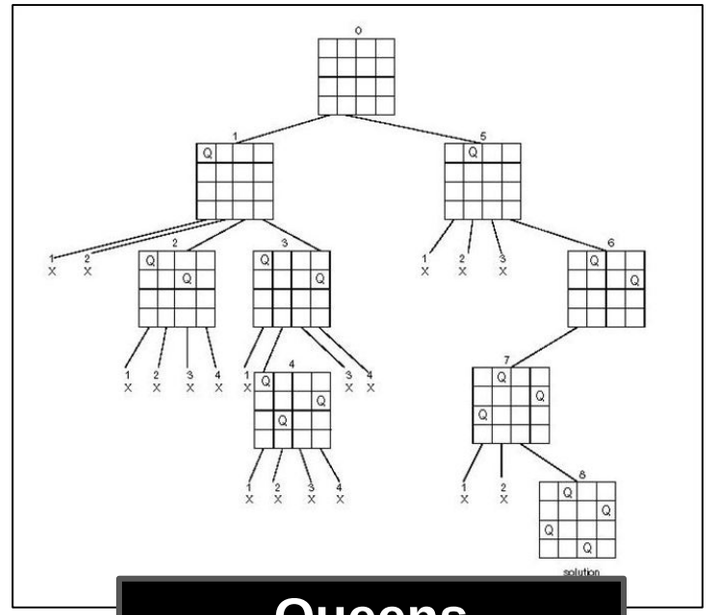
(Not all Links shown here)



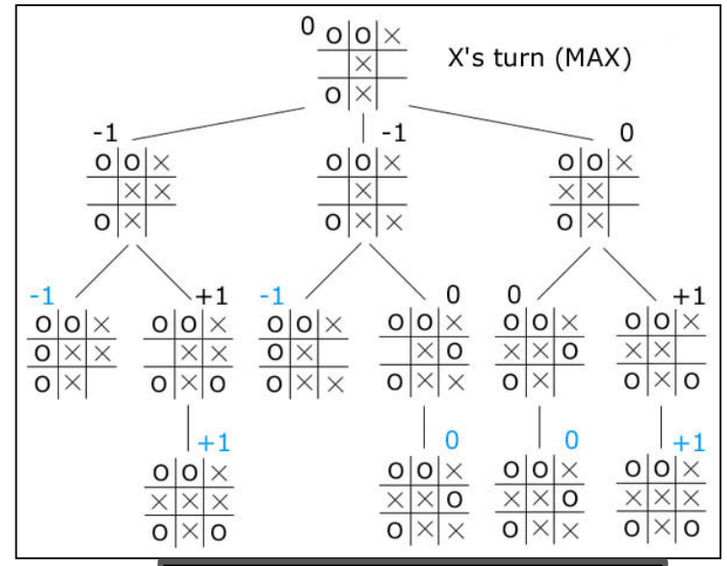
STATE SPACES



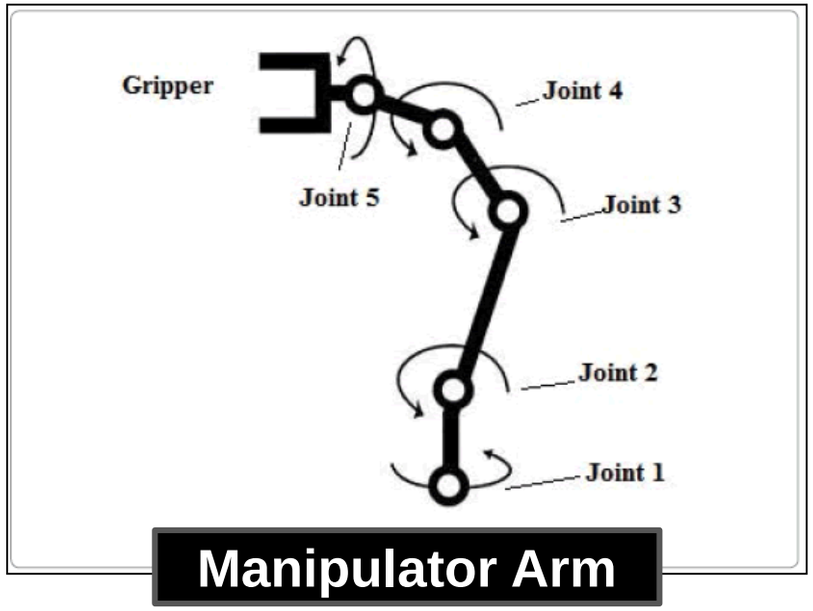
R-Cube



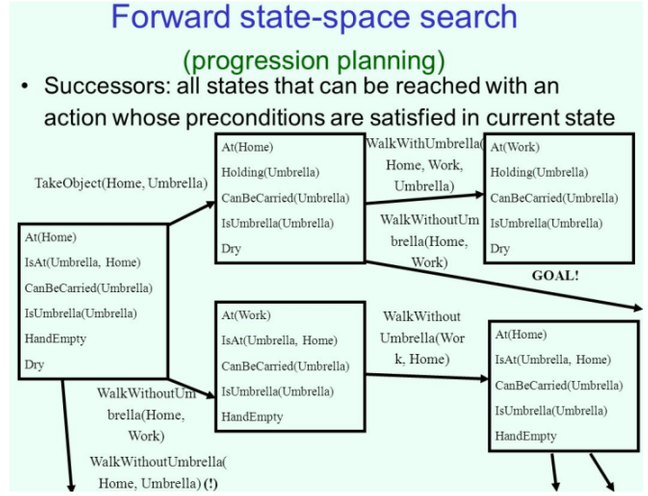
Queens



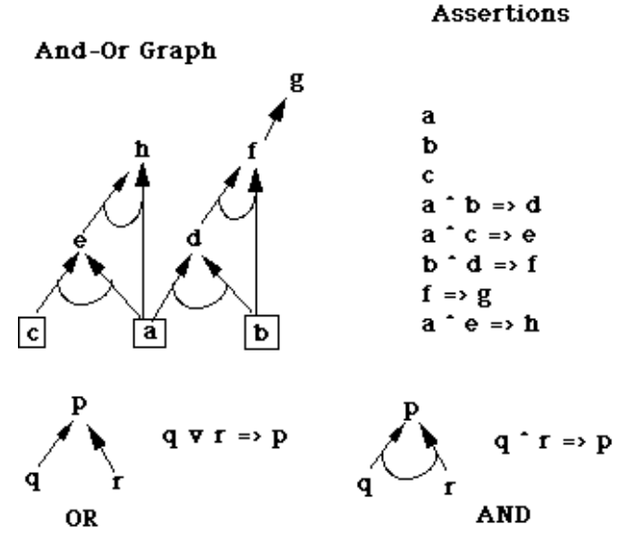
Game



Manipulator Arm



Planning



Reasoning

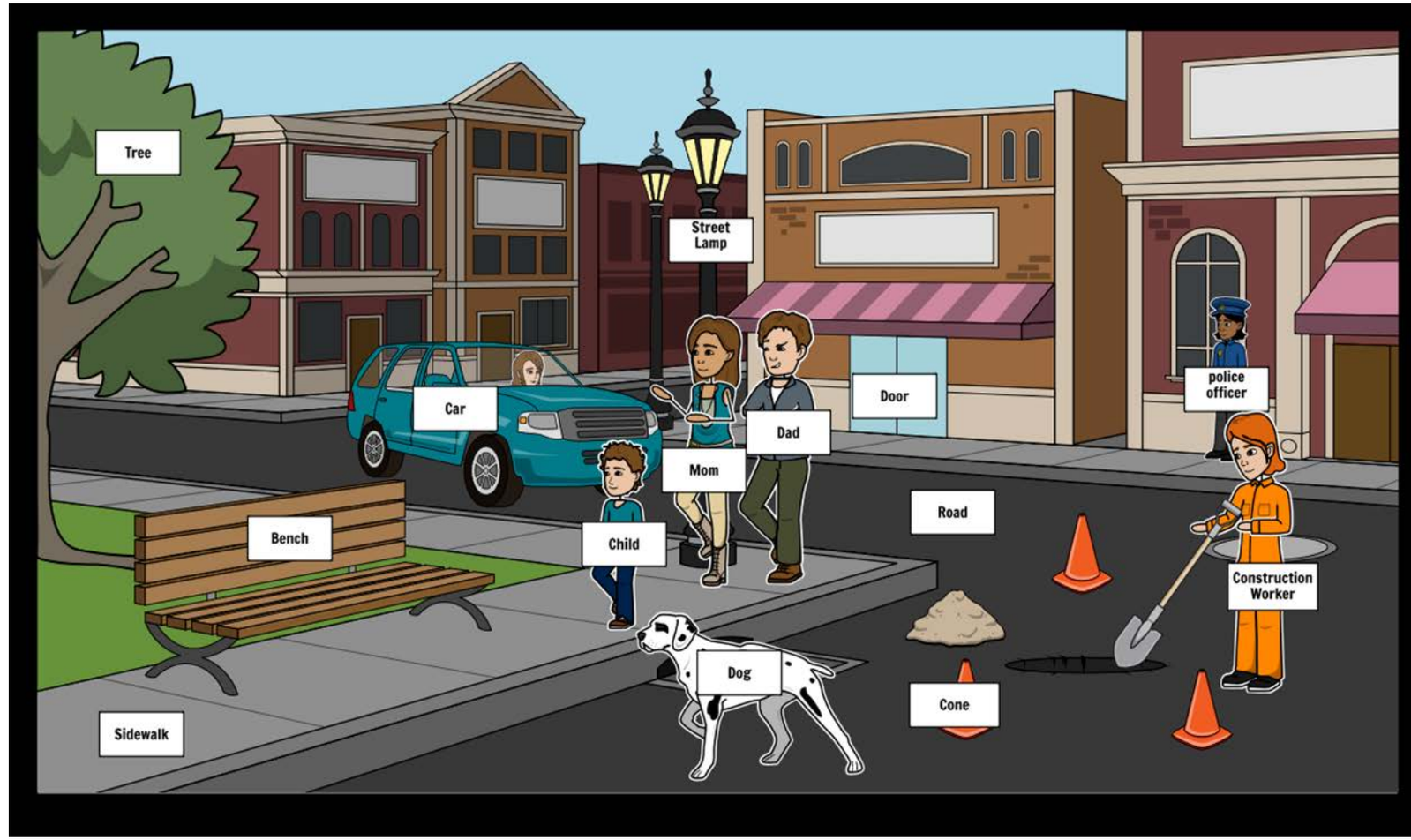
CONSISTENT LABELLING BY CONSTRAINT SATISFACTION

1 V	I	2 T	A	3 M	I	4 N	5 D			6 B	I	7 O
A		H		A		8 A	I	M		U		N
	9 B	R	U	X	I	S	M		10 M	R	I	S
		O		I		A				N		E
11 R		12 A	I	L		13 L	E	14 G		15 F	A	T
16 H	O	T		L		S		17 O	18 S	A		
I			19 S	A	L	T			20 S	T	O	21 P
22 N	E	23 T				E		24 L				A
I		O		25 C	I	R	C	A	D	26 I	A	N
27 T	A	R		L		O		C		T		G
I		28 P	R	O	B	I	O	T	I	C	S	
29 S	K	I		C		D		I		H		30 P
		31 D	I	K	E			C		32 Y	A	M

Crossword

```
    B O B
  x B O B
  -------
  M E O Y
M I L O
M E O Y
-------
M A R L E Y
```

Cryptarithmetic



Scene Analysis

STATES, SPACES, SOLUTIONS, SEARCH

States

- Full / Perfect Information and Partial Information States

State Transformation Rules

- Deterministic Outcomes
- Non-Deterministic / Probabilistic Outcomes

State Spaces As Generalized Games

- Single Player: OR Graphs
- Multi-Player: And / Or, Adversarial, Probabilistic Graphs

Solutions

- Paths
- Sub-graphs
- Expected Outcomes

Costs

Sizes

Domain Knowledge

Algorithms for Heuristic Search

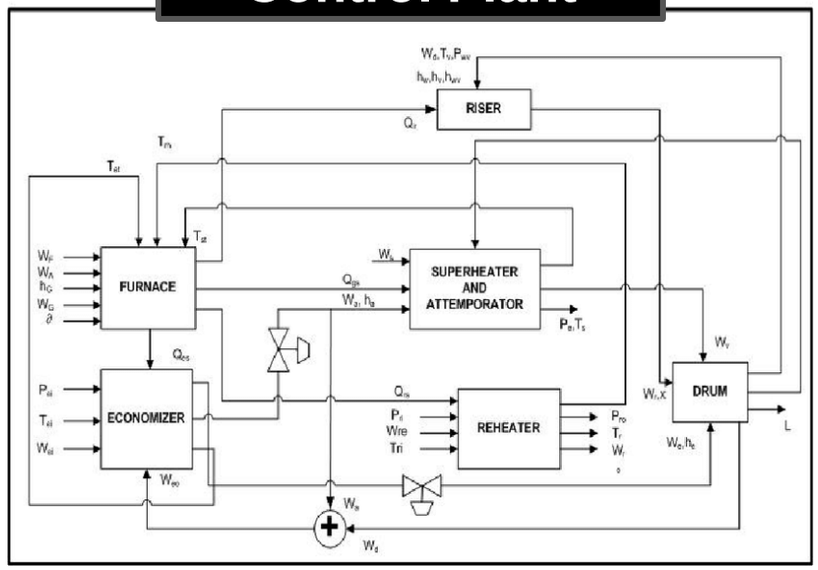
Card Game

South Deals
N-S Vul

	♠ K J 8 5 ♥ J 10 2 ♦ J 8 ♣ A J 10 4	
♠ 10 6 ♥ A 6 4 3 ♦ A K 10 5 2 ♣ Q 2	N W E S	♠ Q 9 2 ♥ 8 7 5 ♦ Q 9 6 3 ♣ K 8 6
	♠ A 7 4 3 ♥ K Q 9 ♦ 7 4 ♣ 9 7 5 3	

West	*North*	*East*	*South*
			Pass
1 ♦	Dbl	2 ♦	2 ♠
3 ♦	3 ♠	All pass	

Control Plant



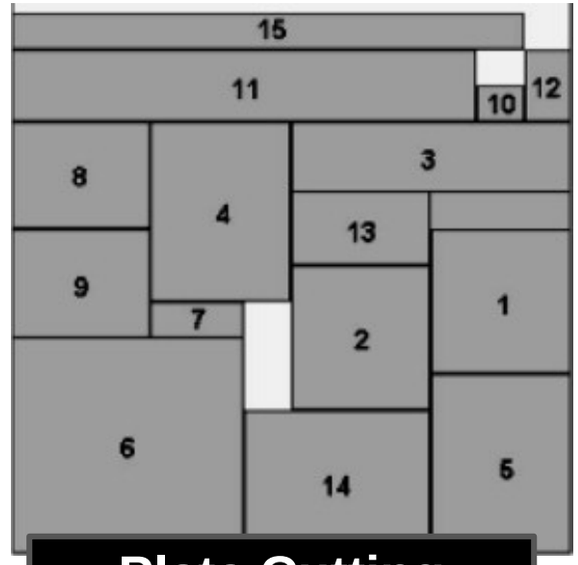


Plate Cutting

BASICS OF STATE SPACE MODELLING

STATE or CONFIGURATION:

- A set of variables which define a state or configuration
- Domains for every variable and constraints among variables to define a valid configuration

STATE TRANSFORMATION RULES or MOVES:

- A set of RULES which define which are the valid set of NEXT STATE of a given State
- It also indicates who can make these Moves (OR Nodes, AND nodes, etc)

STATE SPACE or IMPLICIT GRAPH

- The Complete Graph produced out of the State Transformation Rules.
- Typically too large to store. Could be Infinite.

INITIAL or START STATE(s), GOAL STATE(s)

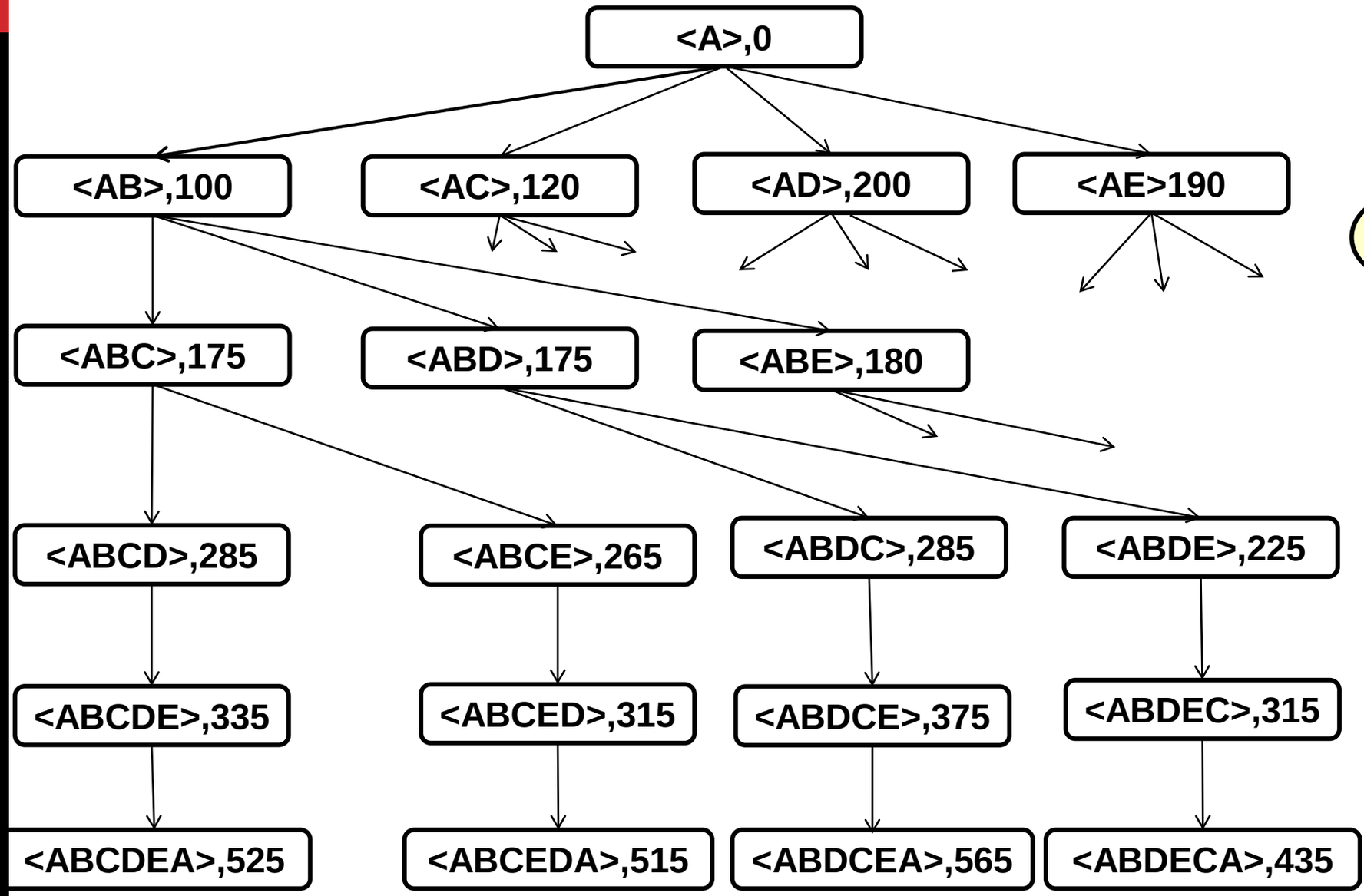
SOLUTION(s), COSTS

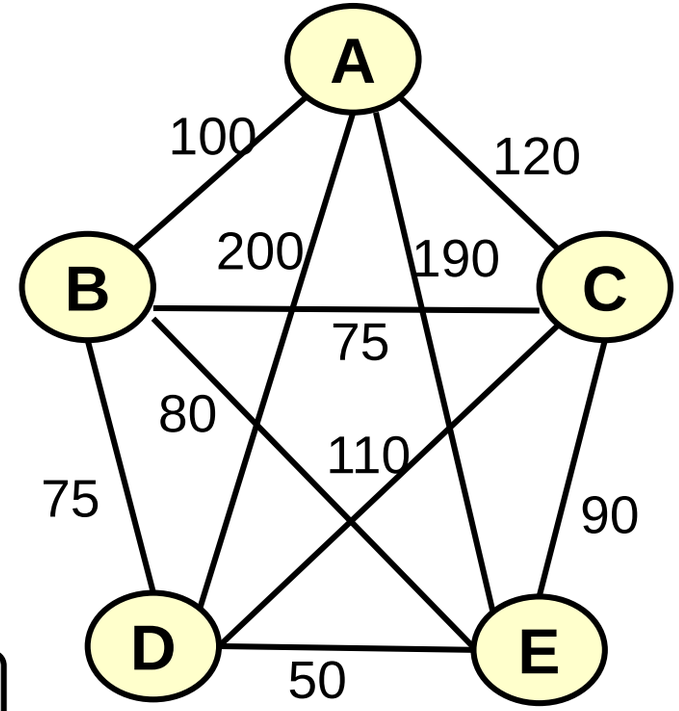
- Depending on the problem formulation, it can be a PATH from Start to Goal or a Sub-graph of And-ed Nodes

SEARCH ALGORITHMS

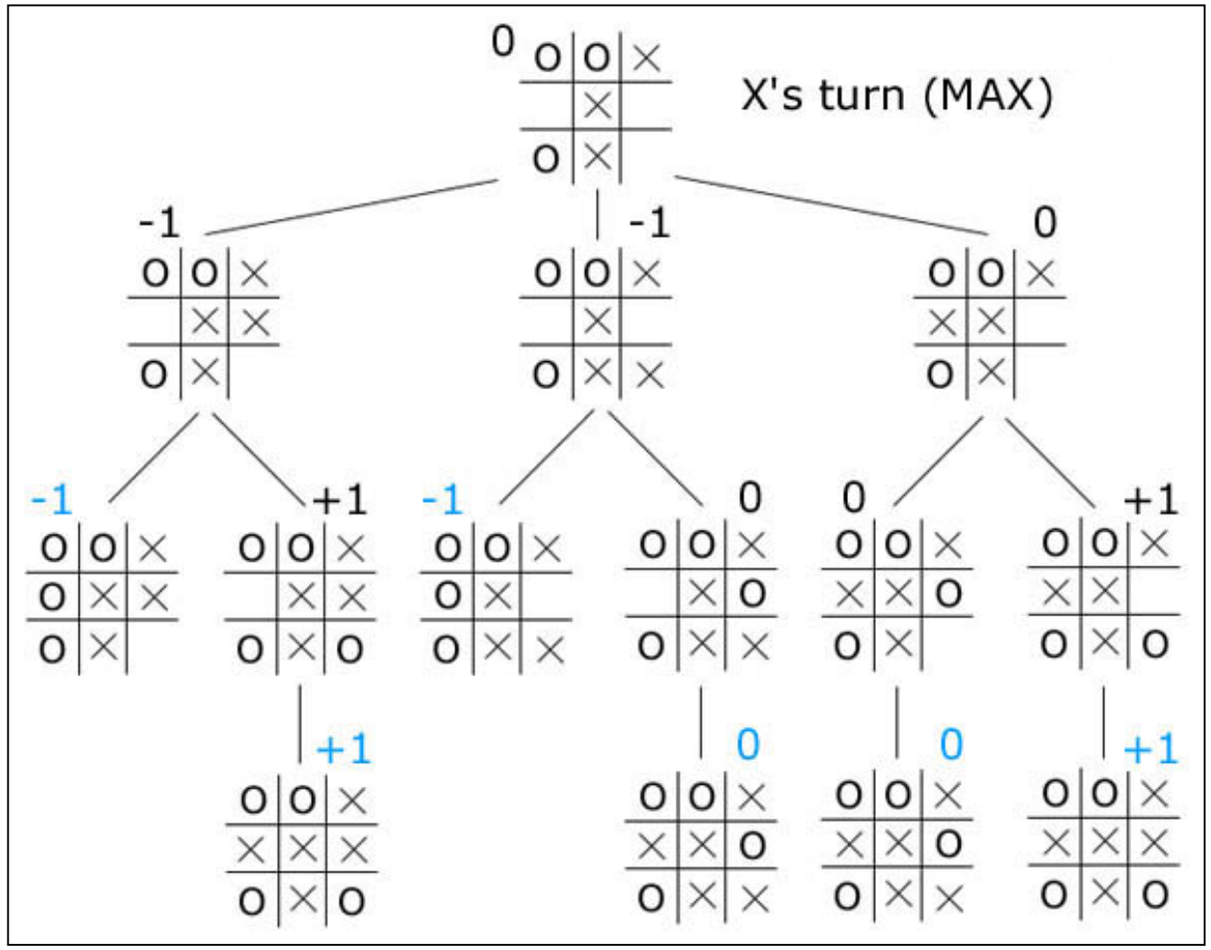
- Intelligently explore the Implicit Graph or State Space by examining only a small sub-set to find the solution
- To use Domain Knowledge or HEURISTICS to try and reach Goals faster

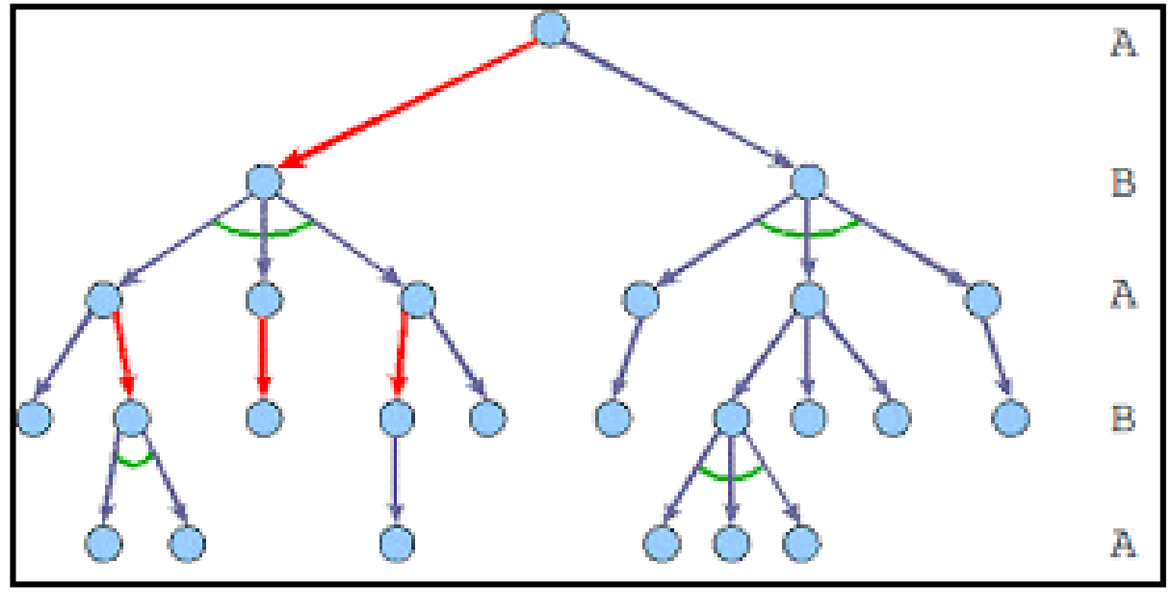
OR-Graph: TRAVELLING SALESPERSON PROBLEM





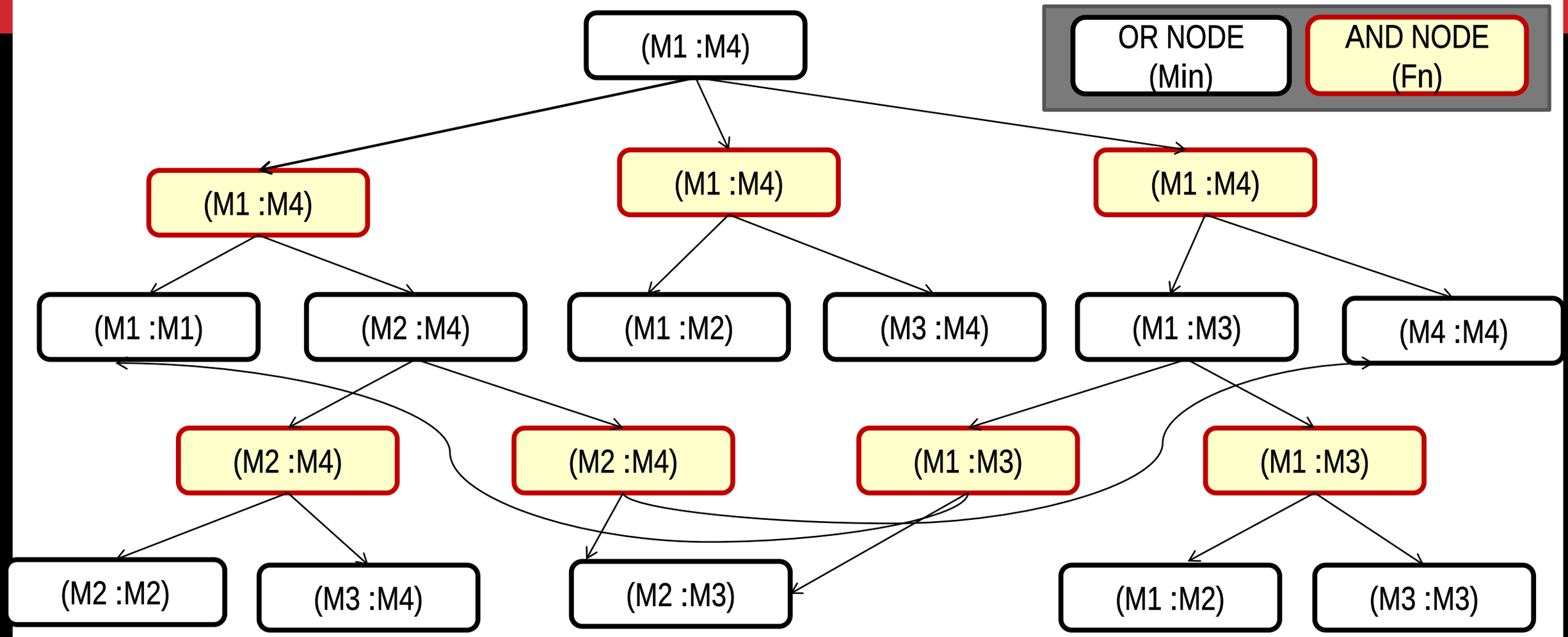
AND/OR GRAPHS: COMPOSITIONAL / ADVERSARIAL / PROBABILISTIC





OR Nodes are ones for which one has a choice. The AND nodes could be compositional (sum, product, min, max, etc, depending on the way the sub-problems are composed), Adversarial (game where the other parties have a choice) or Probabilistic (Environmental Actions)

COMPOSITIONAL AND/OR GRAPHS: MATRIX CHAIN MULTIPLICATION



(M1 X (M2 X (M3 X M4))) = ((M1 x M2) X (M3 X M4)) = (((M1 x M2) X M3) X M4) = (M1 X (M2 X M3)) X M4)

BUT THE NUMBER OF MULTIPLICATIONS TO GET THE ANSWER DIFFER !!

SEARCHING IMPLICIT GRAPHS

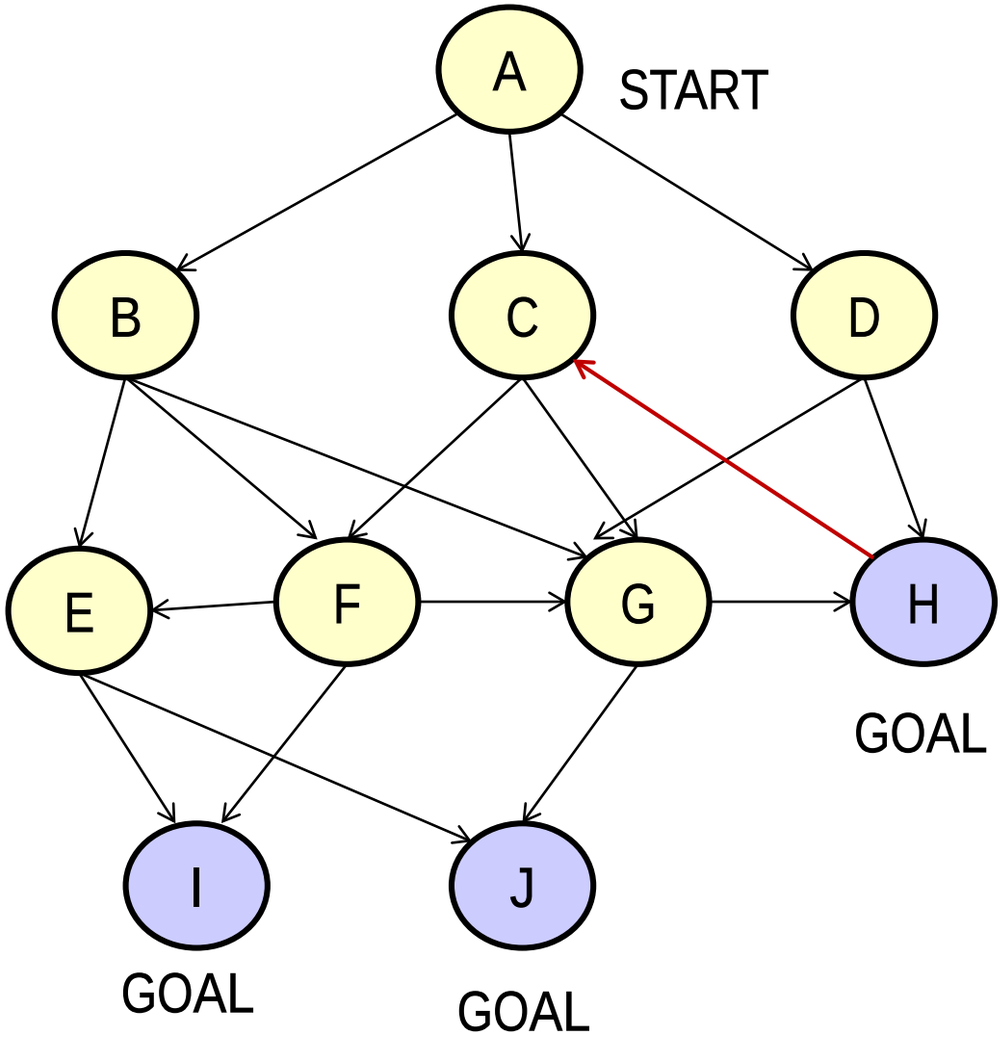
- Given the start state the SEARCH Algorithm will slowly create successors based on the State Transformation Rules and make part of the Graph EXPLICIT. It will slowly EXPAND the Explicit graph INTELLGENTLY to rapidly search for a solution without exploring the entire Implicit Graph or State Space
- For OR Graphs, the solution is a PATH from start to Goal. Cost is usually sum of the edge costs on the path, though it could be something based on the problem
- For And/OR Graphs, the Solution is an AND Subgraph rooted at the Start and each leaf is a Goal Node. The Cost of OR Node is usually a Min or Max. The Cost at the AND Node depends on the type of Node (Compositional, Adversarial, Probabilistic). For Adversarial two player games, Max / Min is used at AND Node (reverse of Or Node)
- The various Search Algorithms include
  - BASIC Algorithms: Depth-First (DFS), Breadth-first (BFS), Iterative Deepening (IDS)
  - COST-based Algorithms: Depth-First Branch-and-Bound, Best First Search, Best-First Iterative Deepening
  - Widely Used: A* (Or Graphs), AO* (And/Or Graphs), IDA*, Alpha-beta Pruning (Game-Trees)

BASIC ALGORITHMS in OR GRAPHS: DFS, BFS, IDS

1. [Initialize] Initially the OPEN List contains the Start Node s. CLOSED List is Empty.
2. [Select] Select the first Node n on the OPEN List. If OPEN is empty, Terminate
3. [Goal Test] If n is Goal, then decide on Termination or Continuation / Cost Updation
4. [Expand]
   a) Generate the successors n_1, n_2, ... n_k, of node n, based on the State Transformation Rules
   b) Put n in LIST CLOSED
   c) For each n_i, not already in OPEN or CLOSED List, put n_i in the FRONT (for DFS) / END (for BFS) of OPEN List
   d) For each n_i already in OPEN or CLOSED decide based on cost of the paths
5. [Continue] Go to Step 2

Algorithm IDS Performs DFS Level by Level Iteratively (DFS (1), DFS (2), ... and so on)

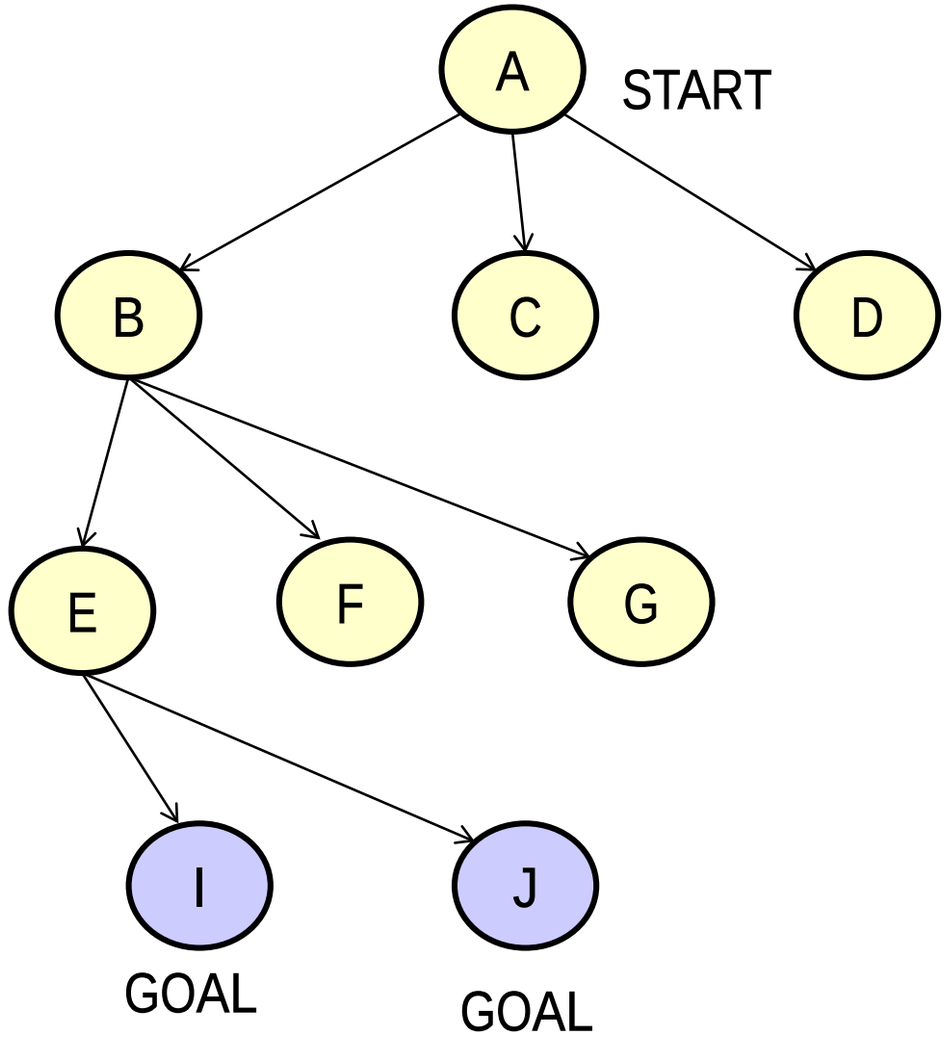
EXAMPLE: SEARCHING A STATE SPACE GRAPH



DETERMINE THE EXECUTION TRACES:

- DEPTH-FIRST SEARCH (DFS)
- BREADTH-FIRST SEARCH (BFS)
- ITERATIVE DEEPENDING SEARCH (IDS)
- PROPERTIES
  - SOLUTION GUARANTEES
  - MEMORY REQUIREMENTS

EXAMPLE: SEARCHING A STATE SPACE GRAPH



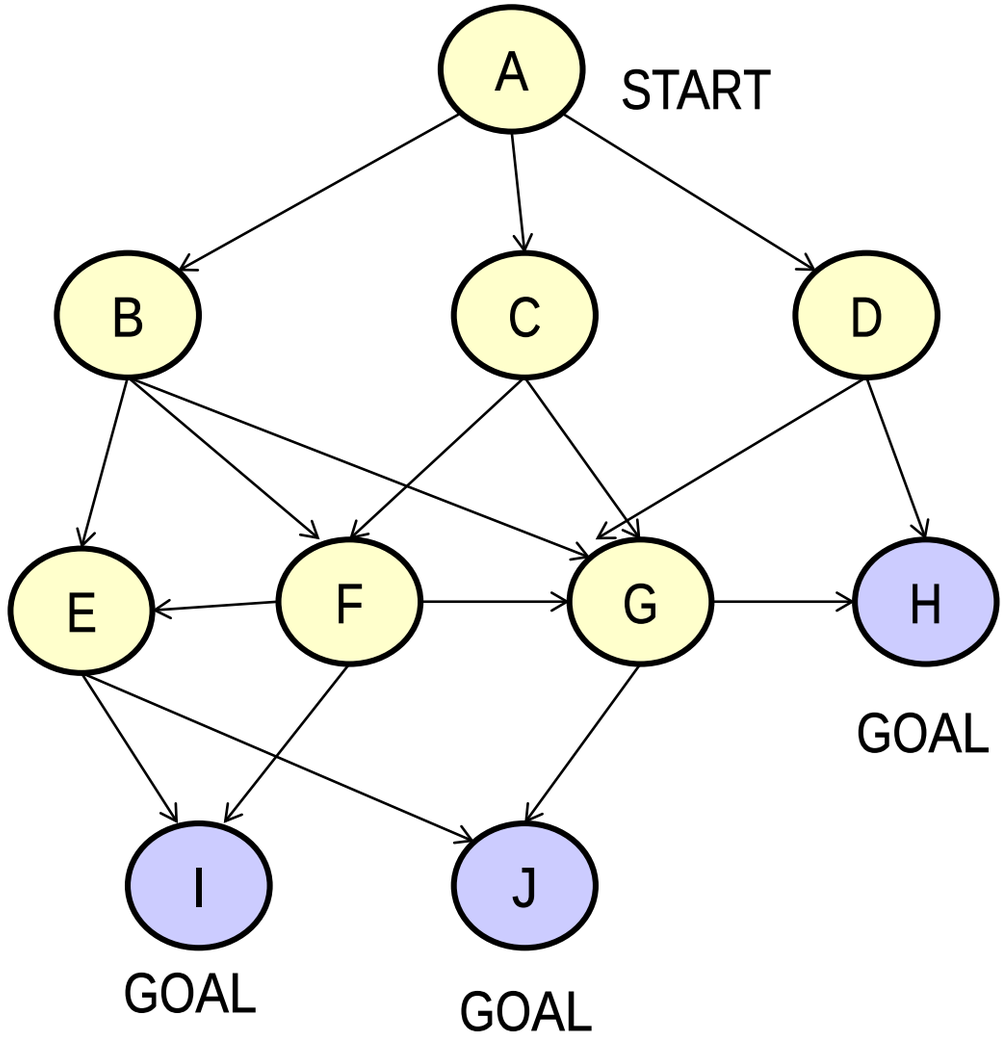
DEPTH-FIRST SEARCH:

1. OPEN ={A}, CLOSED = {}
2. OPEN = {B,C,D}, CLOSED = {A}
3. OPEN = {E,F,G,C,D}, CLOSED = (A,B}
4. OPEN = {I,J,F,G,C,D}, CLOSED = {A,B,E}
5. Goal Node I Found. Can Terminate with Path from A to I or may Continue for more Goal nodes if minimum length or cost is a criteria

DFS MAY NOT TERMINATE IF THERE IS AN INFINITE DEPTH PATH EVEN IF THERE IS A GOAL NODE AT FINITE DEPTH

DFS HAS LOW MEMORY REQUIREMENT

EXAMPLE: SEARCHING A STATE SPACE GRAPH

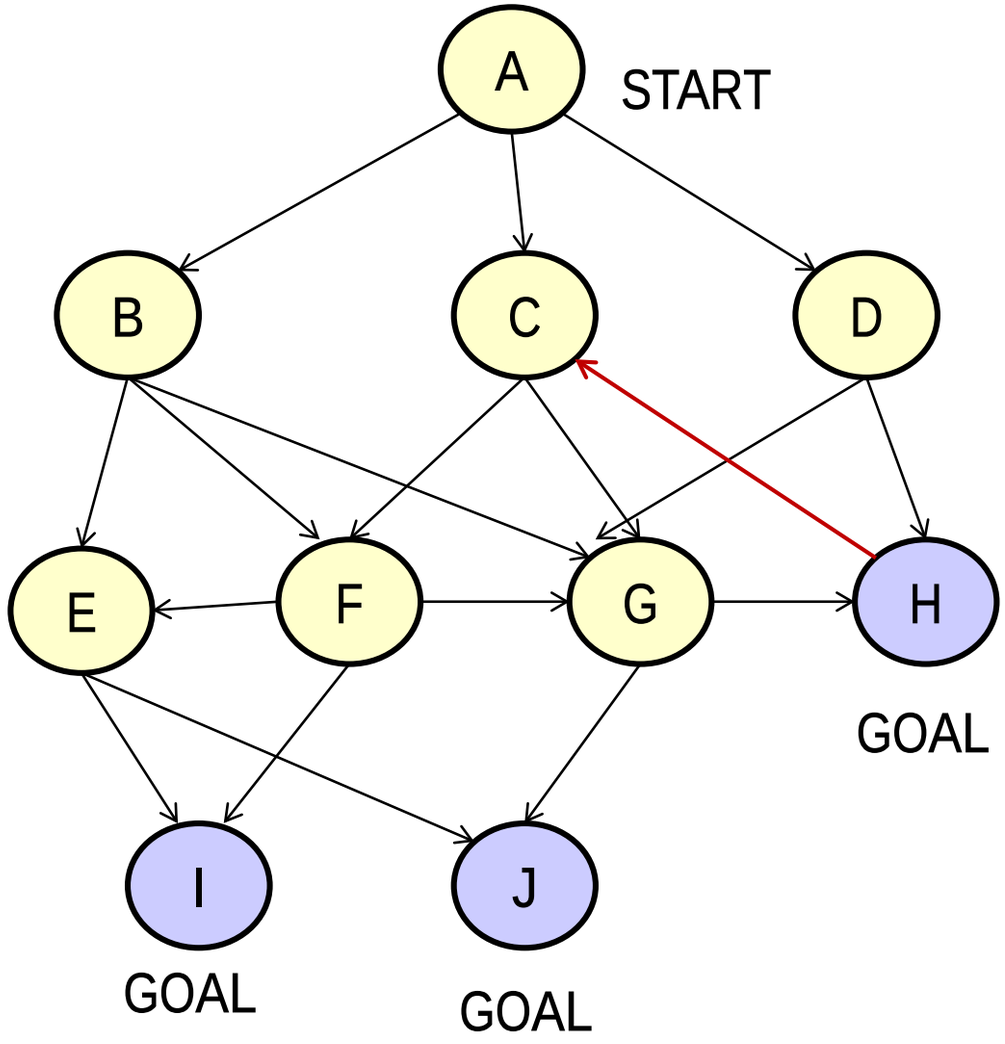


BREADTH-FIRST SEARCH:

1. OPEN ={A}, CLOSED = {}
2. OPEN = {B,C,D}, CLOSED = {A}
3. OPEN = {C,D,E,F,G}, CLOSED = (A,B}
4. OPEN = {D,E,F,G}, CLOSED = {A,B,C}
5. OPEN = {E,F,G,H}. CLOSED = {A,B,C,D}
6. OPEN = {F,G,H,I,J}, CLOSED = {A,B,C,D,E}
7. OPEN = {G,H,I,J}, CLOSED = {A,B,C,D,E,F}
8. OPEN = {H,I,J}, CLOSED = {A,B,C,D,E,F,G}
9. Goal Node H Found. Can Terminate with Path from A to H. This is guaranteed to be the minimum length path.

BFS GUARANTEES SHORTEST LENGTH PATH TO GOAL BUT HAS HIGHER MEMORY REQUIREMENT

EXAMPLE: SEARCHING A STATE SPACE GRAPH



ITERATIVE DEEPENING SEARCH:

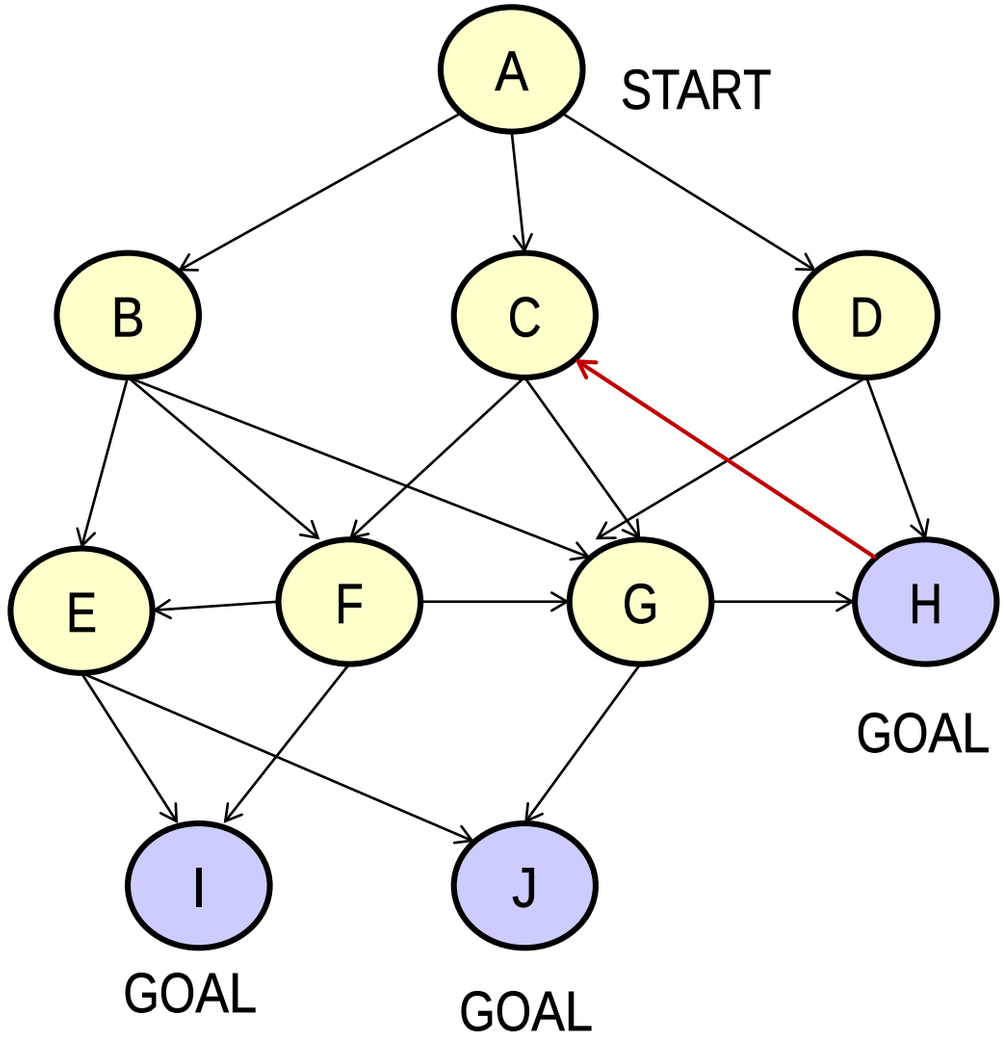
1. PERFORM DFS TILL LENGTH 1. NO SOLUTION FOUND
2. PERFORM DFS TILL LEVEL 2. GOAL NODE H REACHED.
3. Can Terminate with Path from A to H. This is guaranteed to be the minimum length path.

IDS GUARANTEES SHORTEST LENGTH PATH TO GOAL

IDS MAY RE-EXPAND NODES MANY TIMES

IDS HAS LOWER MEMORY REQUIREMENT THAN BFS

NEXT: SEARCHING STATE SPACE GRAPHS WITH EDGE COSTS



DETERMINE THE EXECUTION TRACES:

- COST ORDERED SEARCH:
  - DFBB
  - Best First Search,
  - Best First IDS
  - Use of HEURISTIC Estimates: Algorithm A* (Or Graphs), AO* (And/Or Graphs)
- PROPERTIES
  - SOLUTION GUARANTEES
  - MEMORY REQUIREMENTS

Uniform Cost Search

This algorithm assumes that all operators have a cost:

1. Initialize: Set OPEN = {s},
   CLOSED = { } Set C(s) = 0

2. Fail: If OPEN = { }, Terminate & fail

3. Select: Select the minimum cost state, n,
   from OPEN and save n in CLOSED

4. Terminate: If $n \in G$, terminate with success

Uniform Cost Search

5. Expand:

Generate the successors of n using O.

For each successor, m:

If m $\notin$ [OPEN $\cup$ CLOSED]

Set C(m) = C(n) + C(n,m)

and insert m in OPEN

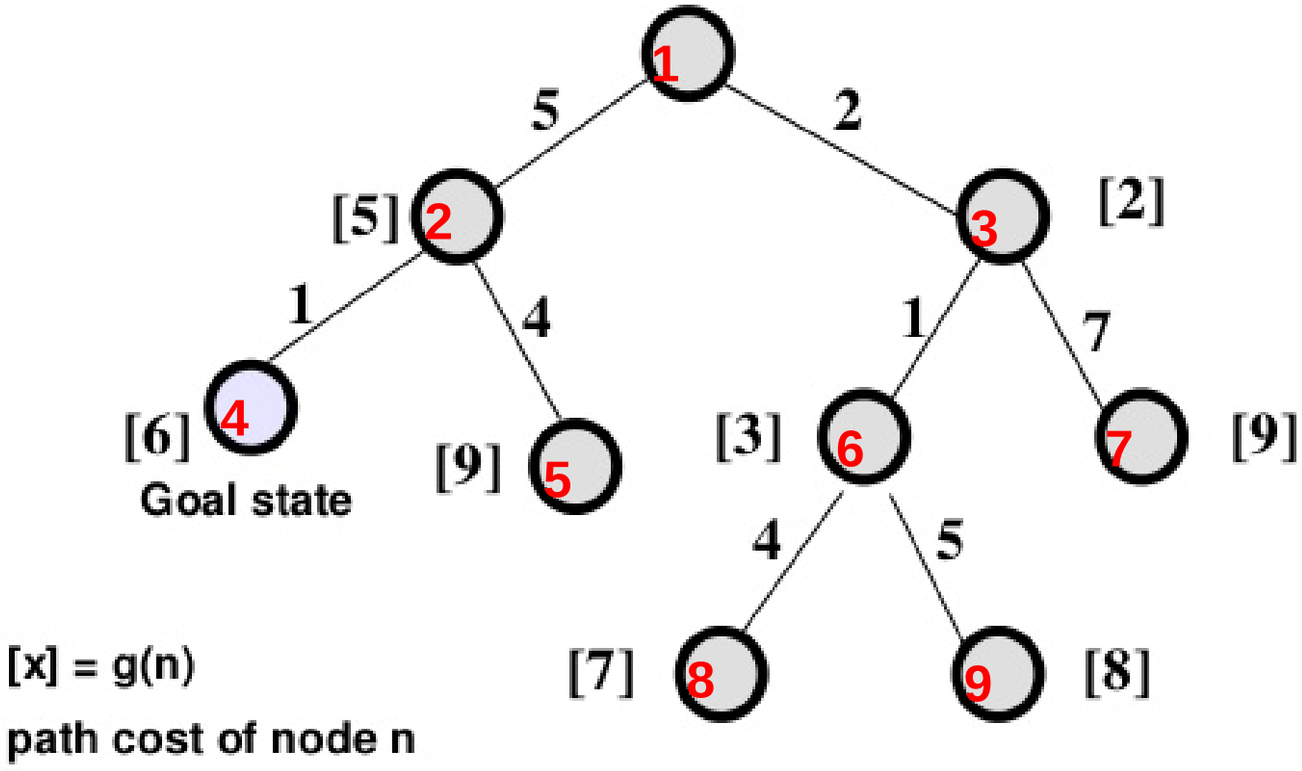
If m $\in$ [OPEN $\cup$ CLOSED]

Set C(m) = min {C(m), C(n) + C(n,m)}

If C(m) has decreased and

m $\in$ CLOSED, move it to OPEN

Sequence of selection of nodes from OPEN:
1 → 3 → 6 → 2 → 4



Searching with costs

- If all operator costs are positive, then the algorithm finds the minimum cost sequence of transitions to a goal.
  - No state comes back to OPEN from CLOSED
- If operators have unit cost, then this is same as BFS
- What happens if negative operator costs are allowed?

SUMMARY

- State Space Search consists of definitions of States, State Transformation Rules, Constraints, Initial and Goal States, Costs
- The Implicit State Space is created using OR Nodes (Choice) and AND Nodes (Compositional, Adversarial, Probabilistic / Environment)
- Solutions can be Paths or Sub-Graphs of the Implicit State Space
- The problem is solved by finding a solution path or sub-graph in the state space so that we reach from start to goal in some optimal way and the solution path must satisfy the constraints in the states as well as follow valid state transformation rules
- Costs are problem specific and goals include Minimization, Maximization, or can be Multi-Objective
- The complete state space is extremely large and usually cannot be completely searched or stored in any reasonable time and space
- Basic Algorithms Include DFS, BFS, IDS,
- Cost Ordered Search required more sophisticated Algorithms