PROBLEM SOLVING USING SEARCH:

* Search Strategy
  → Blind / Uninformed
  → Heuristic
* Constraint Satisfaction Problems (No final State, only Final State test)
  ↳ Finite Search Tree
  ↳ All solution same depth. So path doesn't matter. (All equal)
* 2- Player Games

SEARCH FORMULATION:

* Initial State
* Final State or Final State Test (CSP)
* Actions (Successor function - checks if action / state are legal).

OPTIMIZE:

1. Cost of a solution
   eg: Number of steps from IS to FS.
2. Cost of finding a solution.
   eg: 2 Search Strategy. Both find same solution. One does by expanding full tree, other doesn't.

MEASURE OF DIFFICULTY OF A SEARCH PROBLEM:

1. Number of possible states (State space).
2. Branching factor
3. Depth of solution

SEARCH PROCESS:

Repeat:

1. Find a fringe / frontier node based on the strategy.
2. Expand the node
   - → Check if FS
   - → Generate Children

SEARCH STRATEGIES:

EVALUATION:

* Completeness : Find solution if exists
* Time Complexity : Number of nodes expanded
* Space Complexity : Maximum number of nodes in memory,
* Optimality : Always find least-cost solution.

Complexity expressed in terms of

* b : branching factor
* d : depth of least-cost solution
* m : maximum depth of search tree.

$$N(b,d) = 1 + b + b^2 + \dots + b^d$$

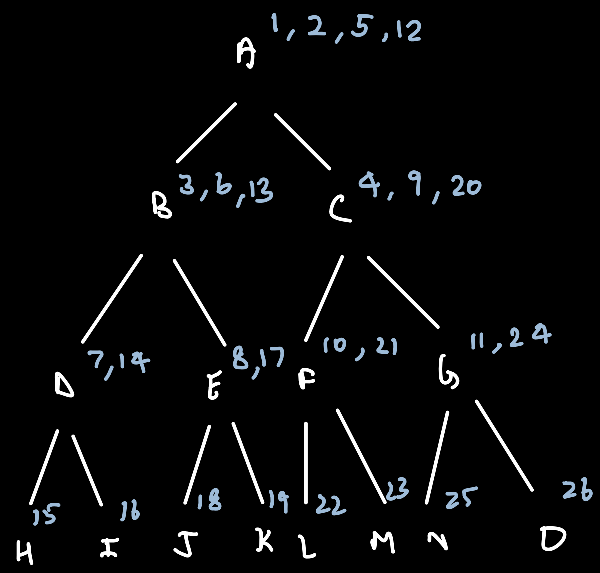
$$bN(b,d) = b + b^2 + \dots + b^{d+1}$$

$$N(b,d) = \frac{b^{d+1} - 1}{b - 1} \qquad O(b^d)$$

BLIND SEARCH STRATEGIES:

	BFS	DFS	LIMITED DFS (ℓ)	ITERATIVE DEEPENING
Complete	✓	✗	$d \leq \ell$	✓
Optimal	✓	✗	✗	✓
TC	b^d	b^m	b^ℓ	b^d
SC	b^d	bm	$b\ell$	bd

ID:

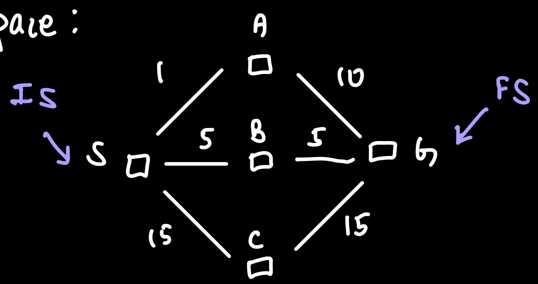


HEURISTIC SEARCH STRATEGIES :

$h(n)$: Heuristic

	Uniform Cost Search	Greedy Best First Search	A* Search
Complete	✓	×	✓
Optimal	✓	×	✓ (if $h(n)$ is admissible)
TC	$b^{\lceil C^*/\varepsilon \rceil}$		
SC	$b^{\lceil C^*/\varepsilon \rceil}$		
Cost	$g(n)$ Distance travelled till 'n'	$h(n)$ Straight line distance from n to goal	Evaluation function : $f(n) = g(n) + h(n)$

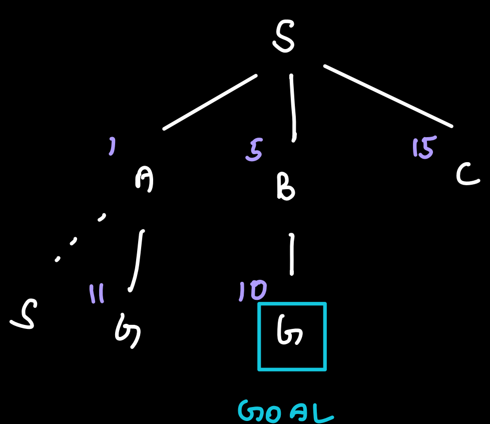
State Space:



Choose least costly fringe node to expand

UCS:

Choose least costly fringe node to expand (generalize BFS)



$g(B) = 5$

$g(G) = 10$

S - B - G

(cost is 10)

ε : smallest cost of any action

Optimal cost: C^*

Optimal Depth $\leq \lceil C^* / \varepsilon \rceil$

BFS $\subseteq$ UCS

So b^d and $d = \lceil C^* / \varepsilon \rceil$

HEURISTIC: Admissible

$h(n) \leq h^*(n)$ [least cost to go from n to goal]

$h(n) \geq 0$

↳ $h(G) = 0$ for a goal state G.

$h(n)$ = admissible => A^* is optimal

CONSTRAINT SATISFACTION PROBLEMS (CSPs):

PRIMAL / CONSTRAINT GRAPH:

connect nodes that are in a constraint.

TYPES:

Unary vs Binary vs Higher-constraints

Hard vs Soft.

Variable
Domain
Constraints

STRATEGY: DFS

IMPROVEMENTS:

1. BACKTRACK SEARCH:

Avoid children that violate constraints.

Iterate through constraints and see if assignment violates it.

2. VALUE ORDERING:

Least constraining value ; Choose value that rules out the fewest values of remaining variables.

3. VARIABLE ORDERING:

Most constrained variable ; Choose variable with fewest legal values.

4. FORWARD CHECKING (FC):

* Maintain set of possible values for each variable.
* Update as you assign.
* Declare "bad state" if variable loses all values.

5. ARC CONSISTENCY (AC) :

* Draw arcs between constrained variables with their allowed values
* Arc $x \to y$ is consistent if

  $\forall$ value of x, there is a compatible value for y.

  $O(n^2 d^3)$

Two Player Games:

See it as one player move followed by another.

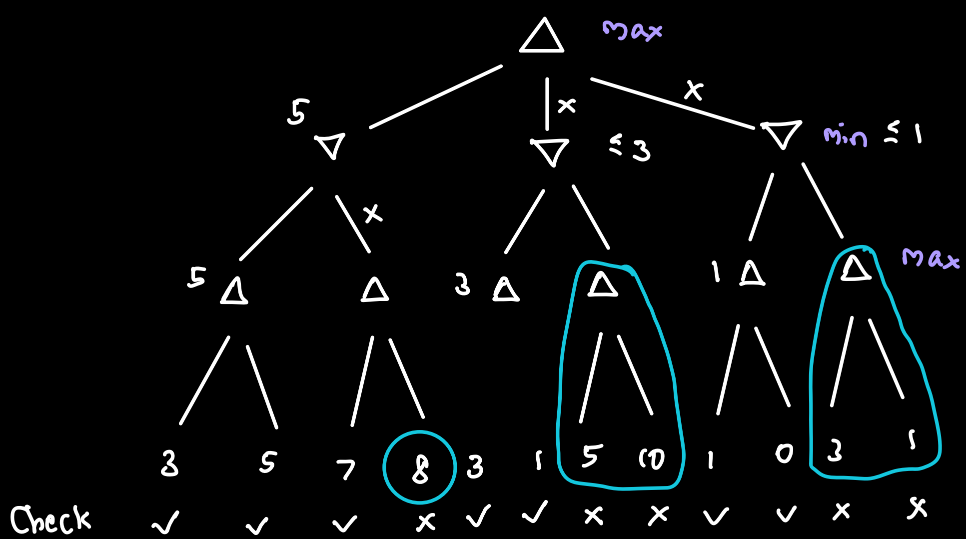
Terminal States (Not Goal / Final)

Minimax Algorithm:

Player 1 maximizes, 2 minimizes.

DFS. $O(b^d)$.

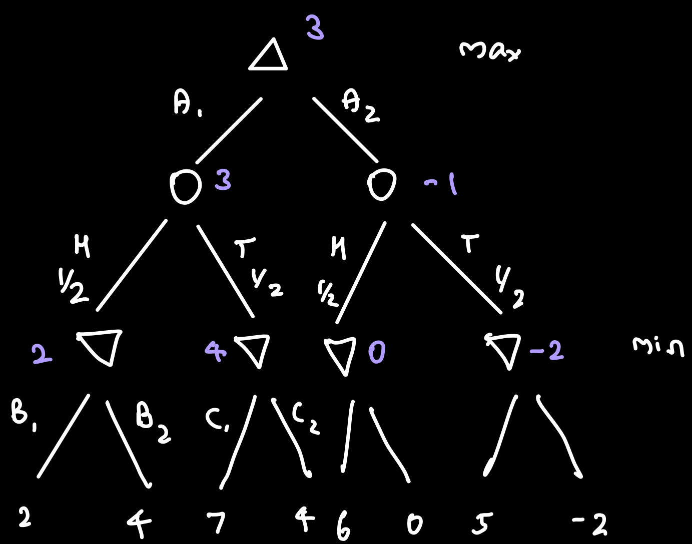
α-β Pruning:



Best case: $O(b^{d/2})$

GAMES WITH CHANCE:

Expected Minimax:



PROPOSITIONAL LOGIC:

SYNTAX:

1. $\alpha \Rightarrow \beta$

   $\neg\beta \Rightarrow \neg\alpha$ CONTRAPOSITIVE

2. $\alpha \Rightarrow \beta$

   $\neg\alpha \vee \beta$

3. $\neg(\alpha \wedge \beta)$

   $\neg\alpha \vee \neg\beta$

4. $\neg(\alpha \vee \beta)$

   $\neg\alpha \wedge \neg\beta$

DE MORGAN'S LAW

a. Boolean Variables

b. $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$

c. Variable is sentence

S_1, S_2 are sentences, then

$\neg S_1, S_1 \wedge S_2 \ldots$

sentences.

NORMAL FORMS:

	UNIVERSAL	TRACTABLE (SAT is easy)
CNF	✓	
DNF	✓	✓
HORN		✓
NNF	✓	
DNNF CIRCUITS	✓	✓

CNF : Conjunction of clauses
($\wedge$)

DNF : Disjunction of terms

HORN : ≤ 1 Positive literal per clause

Conjunction of such clauses

NNF : Negation only to literals

DNNF : AND Gate inputs are separate.

SEMANTICS :

$M(\alpha)$: set of worlds where α holds at

$\downarrow$

Meaning of α
Models of α $\{ \omega : \omega \models \alpha \}$

Opposite : $\omega \not\models \alpha$

* α equivalent to β

$$M(\alpha) = M(\beta)$$

* α is contradictory / inconsistent

$$M(\alpha) = \phi$$

* α is valid / tautology

$$M(\alpha) = \text{all worlds}$$

* α and β are mutually exclusive

$$M(\alpha) \cap M(\beta) = \phi$$

* α implies β

$$M(\alpha) \subseteq M(\beta)$$

INFERENCE:

KB: Δ

Query: α

$\Delta \models \alpha$ reads : Δ implies α, Δ entails α

α follows from Δ.

INFERENCE METHODS:

1. TRUTH TABLE (MODEL ENUMERATION):

Δ imply α : $M(\Delta) \subseteq M(\alpha)$

(Brute Force)

REFUTATION THEOREM:

$\Delta \models \alpha$: Δ implies α

$\boxed{\Delta \wedge \neg\alpha \quad \text{contractory}}$ $M(\Delta \wedge \neg\alpha) = \phi$

2. INFERENCE RULES:

INFERENCE RULES R

$\Delta \vdash_R \alpha$: α derived from Δ using R.

Complete:

If $\Delta \models \alpha$ then $\Delta \vdash_R \alpha$

Sound:

If $\Delta \vdash_R \alpha$ then $\Delta \models \alpha$

INFERENCE RULES:

1. MODUS PONENS:

$$\frac{\alpha, \ \alpha \Rightarrow \beta}{\beta}$$

2. OR - INTRODUCTION:

$$\frac{\alpha, \beta}{\alpha \vee \beta}$$

3. AND - INTRODUCTION:

$$\frac{\alpha, \beta}{\alpha \wedge \beta}$$

RESOLUTION:

$$\frac{\alpha \vee \beta, \ \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

Unit Resolution → One of the 2 clauses is unit.

Linear time! But it is not refutation complete.

Is $\Delta \models \alpha$?

1. $\Delta \wedge \neg\alpha$ convert to CNF

2. $\Delta \wedge \neg\alpha$ prove unsat

   - Contradiction $\Rightarrow$ $\Delta \models \alpha$

   - no contradiction $\Rightarrow$ $\Delta \not\models \alpha$

CONVERTING SENTENCES INTO CNF:

Any sentence can be converted into CNF.

1. Get rid of all connections but for $\wedge, \vee, \neg$

   $\alpha \Rightarrow \beta \rightarrow \neg\alpha \vee \beta$ - 1

   $\alpha \Leftrightarrow \beta \rightarrow (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
   use 1

2. Use de Morgan's law to push negations inside

   $\neg(\alpha \wedge \beta) \rightarrow \neg\alpha \vee \neg\beta$

   $\neg(\alpha \vee \beta) \rightarrow \neg\alpha \wedge \neg\beta$

3. Distribute $\vee$ over $\wedge$

   $(\alpha \wedge \beta) \vee \gamma \rightarrow (\alpha \vee \gamma) \wedge (\beta \vee \gamma)$

3. By Reduction SAT (Satisfiability) - Search

a. Complete Method: Systematic Search (dfs)

Always completes

DPLL Algorithm:

1. Origin CNF Δ

$$\left.\begin{array}{l} \to \Delta \wedge A \\ \to \Delta \wedge \neg A \end{array}\right\} \to \text{Branching on variable } A.$$

2. Unit Resolution

b. Incomplete Method: Local Search

Completes if sat.

If unsat, might loop infinitely.

* Min-Conflicts / Hill Climbing:

for i = 1 ... N : (Number of restarts)

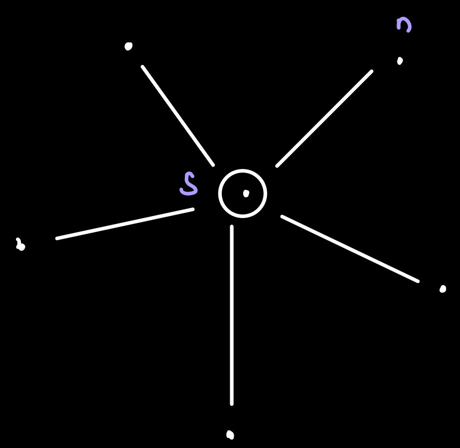
Choose random starting point.

While the violated constraints decrease:

Find neighbor with least violated constraints

Move to that neighbor.

* SIMULATED ANNEALING:



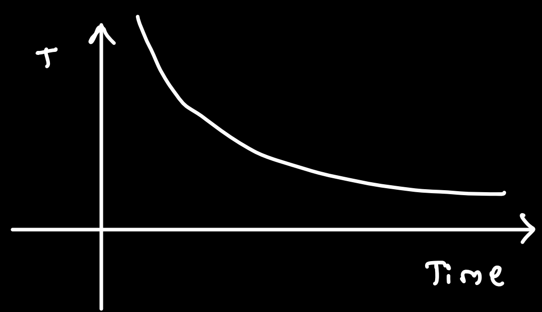
Pick random neighbor (n)

ΔE : violations (n) - violations (s)

↳ $\Delta E < 0$: go to n

↳ otherwise

go to n with $P \propto \frac{1}{e^{\Delta E / T}}$



T : Temperature

As time proceeds, chances to go to n decreases.

Randomization:

→ Avoids local minima

→ Reaches almost all nodes.

4. CONVERTING KB (Knowledge Base) into "tractable forms"

(Knowledge Compilation)

"Compile it" into a "tractable circuit".

NNF :

DNNF

→ Decomposability :

AND Gates : Children should not share variables

d-DNNF

→ DETERMINISM :

OR Gates. 2 Children → α, β then α, β are mutually exclusive. $\alpha \wedge \beta$ unsat.

→ SMOOTHNESS :

OR Gates → Children same variables.

SOLVES:

SAT / MODEL COUNTING:

→ Counts # of assignments that solve CNF.

HOW TO SOLVE #SAT:

1. BASE CASE:

True gets 1, False gets 0

Every literal get 1.

(+ve / -ve)

2. AND Gates : Multiply

OR Gates : Add