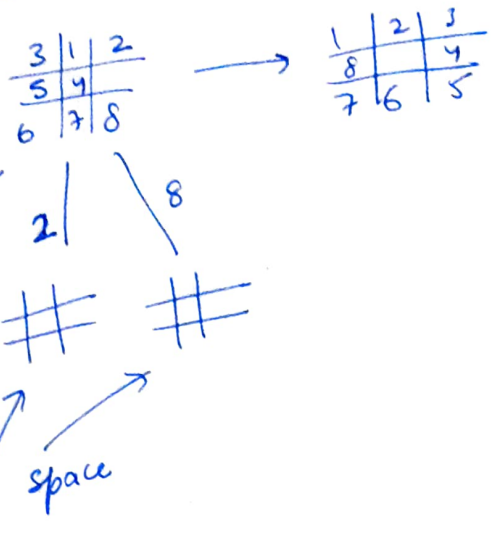
5/8/25

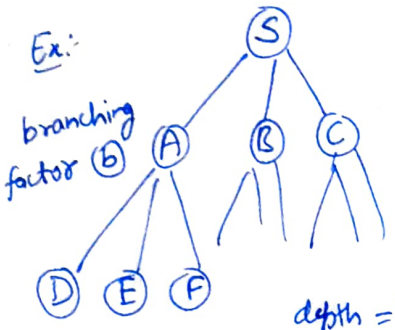
Artificial Intelligence

→ 8-puzzle.



Brute force ≡ Generate & Test

Ex:



OPEN Set = ($) { → The nodes that have to be examined in future }

d = 1 [A, B, C]

d = 2 (B, C, D, E F)

Closet set → { contains nodes that have been examined }

Pseudo Code (Generate & Test)

```
OPEN ← [S]
while OPEN is not Empty
{   Pick any node n from OPEN
    OPEN ← OPEN - (n)
    CLOSED = CLOSE U (n)
    if (n = GOAL) return True
    else
        OPEN ← OPEN U
            {neighbours(n) - CLOSED}
return failure
```

2.) DFS, BFS } comparison points

(a) Time Complexity

↳ depends on CLOSED SET

DFS → Time = $\frac{1}{2}$ [Goal at first node + GOAL at last node]

$= \frac{1}{2}\left[d + \frac{b^{d+1}-1}{b-1}\right]$ { b → branching factor ↳ no. of branches each node have → $1+b+b^2+\dots+b^d$

$\simeq \frac{b^d}{2}$

BFS → Time = $\frac{1}{2}$ [all nodes till $(d-1)$ depth + Goal at last node]

$1+b+b^2+\dots+b^{d-1}$ $\quad$ $1+b+b^2+\dots+b^d$

$= \frac{1}{2}\left[\frac{b^d-1}{b-1} + \frac{b^{d+1}-1}{b-1}\right]$

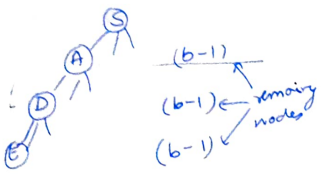
$= \frac{b^d}{2}\left(\frac{1}{b}+1\right) \simeq \frac{b^d}{2}\left[\frac{1+b}{b}\right]$

so, DFS is marginally better than BFS $\left[\frac{b^d}{2} < \frac{b^d}{2}\left(\frac{1+b}{b}\right)\right]$

$\frac{BFS}{DFS} = \frac{1+b}{b}$

(b) Space Comp. → OPEN set determines space comp.

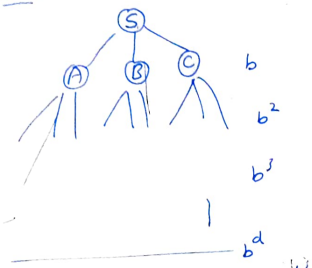
DFS



depth = d

so, space (of OPEN) $= (d-1)(b-1) \simeq O(b \cdot d)$

BFS



so, space $= 1+b+b^2+\dots+b^d$

$= \frac{b^{d+1}-1}{b-1} \simeq O(b^d)$

so, in space comp. again DFS is better than BFS

(c) Quality of sol^n → That is probability of finding the goal (starting cell se kitne pass answer h)

→ BFS ≥ DFS → BFS has better quality of sol^n than DFS

(d) Completeness → Both DFS & BFS are complete in nature (can algo. always help us to reach the goal)

6/8/25

→ (4) DB-DFS → Depth Bound DFS

↳ Find depth tak search karega algo then terminate.

↳ Not complete

(5) Depth First Iterative Deepning (DFID):-

↳ Variable Depth

↳ First search till a fixed depth using DFS, if goal state not found then we inc. the depth and the whole tree will be regenerated till that depth

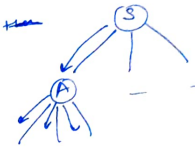
↳ complete

↳ Quality of sol^n best

↳ Time & Space is similar to DFS complexity

⇒ d=1

If sol^n not found delete whole tree

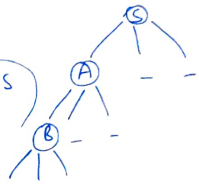


full tree will be regenerated.

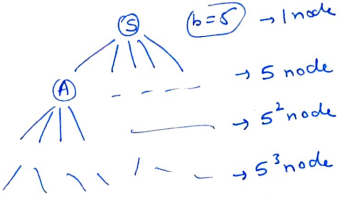
d=2

↓

d=3 and so-on.



→



so, to generate the whole it is not heavy on the memory because we have generate for eg: $125 = 5^3$ nodes at d=3 for that we have to also generate the previous tree nodes. $(1+5+5^2 = 31)$ since the ratio is not significant $\left(\frac{31}{125}\right)$ therefore it doesn't put stress on memory.

(Ch-3) in Book.

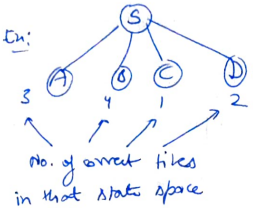
→ Heuristic Algorithms :-

↳ To check how ~~we are~~ close we are to goal state

let h_1 = no. of correct tiles

→ So, (B) is best state to continue with.

Eg:

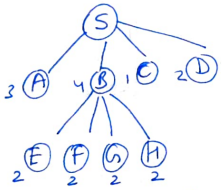


(1) Best First Search :-

Priority Queue (Min Heap) set →

OPEN = [A, C, D, E, F, G, H]

CLOSED = [S, B]



→ Best (A) after exploring B.

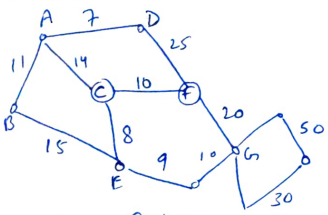
→ Time & Space depends on heuristic func" Comp.

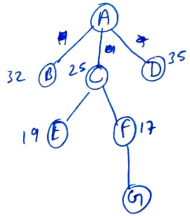
→ Complete Algo.

Not gives always optimal soln

Now, we will check all the heuristic value of nodes present in OPEN Set, so (A) will be explored next.

Eg:





St. Line Distance

A → G = 40

B → G = 32

C → G = 25

D → G = 35

E → G = 19

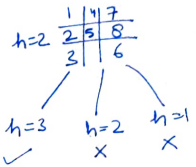
F → G = 17

A → G = 10

G → G = 0

7/08/25

#(i) Hill Climbing



↳ Constant Space Complexity → since we are only examining the best node.

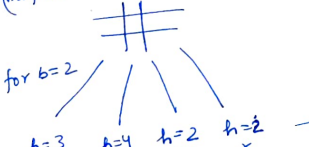
↳ Linear Time Comp. : cause we are only exploring one node at each depth.

↓ chosen Node, should be better than parent and all other nodes are deleted. If we don't find any better heuristic value, the algo. will terminate & return the previous parent node.

(ii) BEAM Search :- (modified Hill Climbing)

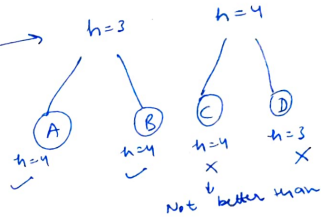
→ beam width → 'b' = no. of best nodes chosen at each depth

→ all best 'b' nodes should be better than parent

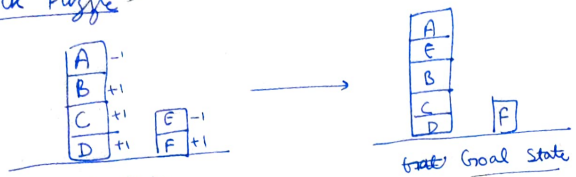


→ 2 'b' nodes are chosen at each depth.

→ Time & Space comp. same as HILL climbing



Block Puzzle :-



$h = 4 - 2 = +2$

→ Apply HILL or BEAM Search.

2-SAT

$(a \vee b) \wedge (c \vee d) \wedge (d \vee e')$ → CNF form.

3- clauses

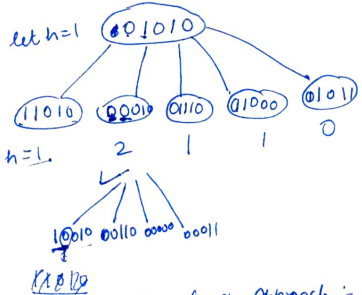
and each clause has maximum of two literals hence 2-SAT

a, b, c, d, e'

3-SAT,

↳ $(a \vee b \vee c') \wedge (d \vee a' \vee b') \wedge (e \vee d \vee c')$

let $a=0$, $b=1$, $c=0$, $d=1$, $e=0$ initial state



(3.) Variable Neighbourhood Approach :- (Not Abbliable

(variation of hill climbing)

Changing one bit at a time = 5 nodes

changing two bits at a time = 10 nodes

↳ increased probability of finding better heuristic value.

for n- literals,

change one bit at a time = nC_1

in two - = nC_2

→ In initial depth, start with low dense heuristic f^n, like nC_1 and then after few depths increase the density of f^n like nC_2 or nC_3.

12/08/25

Iterative H.C :-

→ we can assure multiple initial cond^n.

→ Like H.C.

for eg: to solve SAT Problem

we can assure 0010, 1100, - ——

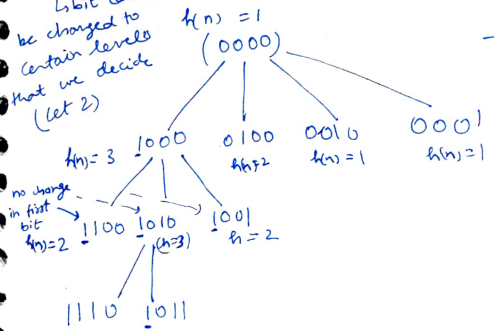
TABU Search SAT Problem :-

4 literals a, b, c, d

$(a \vee b) \wedge (c \vee d) \wedge (a \vee b') \wedge (a \vee d')$

Tabu Tenure

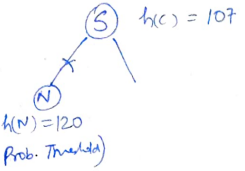
↳ bit cannot be changed to certain levels that we decide (let 2)



Programmy

→ we can maintain an array like [2|0|0|0] acc. to it only those bits are changed whose values are 0.

Stochastic Hill Climbing



(∵ < Prob. Threshold)

→ Probability threshold → A threshold value acc. to which we decide if the child node is better than parent or not.

→ If it is not better, we will generate another node.

and To find probability threshold,

$$\Delta E = \underset{\text{(child)}}{h(N)} - \underset{\text{(Parent)}}{h(C)}$$

we will sigmoid f" → $\frac{1}{1+e^{-x}}$ → since it gives value → b/w 0 & 1.

so, $\frac{1}{1+e^{-\Delta E/T}}$ → for higher values of heuristic f".

$\frac{1}{1+e^{\Delta E/T}}$ → for lower value in heuristic f".

where, T is usually small like, T=10 ← final value

h(c) = 107	h(N) = 120	h(N) = 150	h(N) = 80
T = 1	~ 0.8	~ 0.9	~ 0.000—
T = 10	•		
T = 100000	~ 0.5	~ 0.5	~ 0.5

so, we will use diff. values of T at diff. levels.
↳ we will start with higher value of T, and decrease as we go down.
↳ This algo. is called Simulated Annealing.

Genetic Algorithm

(a) Selection → Start with random sol" and select better candidates

(b) Crossover :- Producing new sol" from selected candidate

(c) Mutation :- (Rarely used)

		evaluation $f"(x^2)$	Probability	Expectation
y:	13 01101	169	169/1340	≈ 0.2 × 4 ≈ 0.8 ≈ 1
	24 11000	576	576/1340	≈ 0.4 × 4 ≈ 1.6 ≈ 2
	8 01000	64	64/1340	≈ 0.0 × 4 ≈ 0.0 ≈ 0
	19 10110	361	361/1340	≈ 0.3 × 4 ≈ 1.2 ≈ 1
		1340		

To find max. possible value of sixth (i.e 11111)
$= 2^4+2^3+2^2+2^1+2^0 = 31$

so, now selection

crossover single point

13 → 01101 → 01100 (12)
24 → 11000 → 11001 (25)
19 → 10011 → 11011 (27)
10000 (16)

→ 1st Method
If we have 'n' initial candidates, & and then we generate 'n' new candidates, then we can replace the previous weaker candidates from new better candidates

→ 2nd Method → or we can replace entire old candidates.

→ 11001 (25) → 11001, 11011
11011 (27) → 11011, 11001

we cannot change the third bit to one by any crossover.

so, now we use mutation → and we change third bit to 1.

TSP :-

Path representation: $P_1 = 24756|1893$, $P_2 = 6283|91457$ ⇒ 24756 91457 / 6283 61893 (repetation)

To solve repetation :-

(i) Partially Mapped Crossover :

$P_1 = 247|5618|93$

$P_2 = 628|3914|57$

$C_1 = \underbrace{924}_{P_2}\ \overbrace{56\ 18}^{P_1}\ \underbrace{37}_{P_2}$ / $\underbrace{287}_{P_1}\ \underbrace{39\ 14}_{P_2}\ \underbrace{65}_{P_1}$

→ we will take some random bits from both P_1 & P_2 and then we will map them together $\begin{pmatrix}5&6&1&8\\ \updownarrow&\updownarrow&\updownarrow&\updownarrow\\ 3&9&1&4\end{pmatrix}$

Then, if we encounter a repetation we will use these mapping.

$P_1 = 2\overset{*}{4}7|5\ 6\ \overset{*}{1}8|\overset{*}{9}\overset{*}{3}$

$P_2 = \overset{*}{6}\ 2\overset{*}{8}|3\ 9\ \overset{*}{1}\overset{*}{4}|\overset{*}{5}7$

(ii) Ordered Crossover :-

$C_1 = \overbrace{5\ 6\ 1\ 8}^{P_1}\ 23947$ → Just put some random bits first, and discard all the repetation.

$C_2 = \underbrace{3\ 9\ 1\ 4}_{P_2}\ 27568$

⇒ ordinal representation : Let index $\overset{*}{1}\ \overset{*}{2}\ \overset{*}{3}\ \overset{*}{4}\ \overset{*}{5}\ \overset{*}{6}\ \overset{*}{7}\ \overset{*}{8}\ 9$

$P_1 = 24756\ 1893$

$P_1 = 2\ 3\ 5\ 3\ 3\ 1\ 2\ 2\ 1$

$P_2 = 628\ 3914\ 57$

$P_2 = 6\ 2\ 6\ 2\ 5\ 1\ 1\ 1\ 1$

→ write index no., of the no. in path representation

→ also cut the index you have used, then for next index, count from start by 1

New, $P_1 = 2353|3122|$

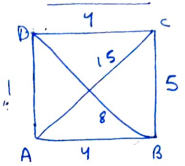
$P_2 = 62625|51111$

⇒

14/08/25

Ant Colony Optimization :-

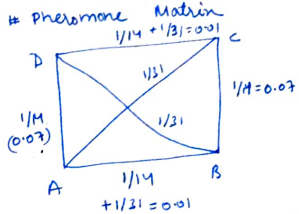
→ cost Matrix :-



Path 1 → ABCDA = 14

Path 2 → ACDBA = 31

Pheromone Matrix



→ To update the level of pheromone in the pheromone matrix, we use the path sum reciprocal.

→ Now to find probability of another ant going from A, based on cost & pheromone level,

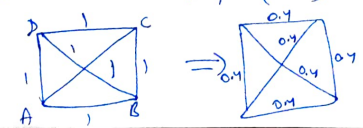
$A \to D = \dfrac{(0.07)\times\frac{1}{1}}{\text{sum of all cost from A}} = \dfrac{0.07\times\frac{1}{1}}{0.07\times\frac{1}{1}+0.03\times\frac{1}{15}+0.1\times\frac{1}{4}}$

$A \to B = \dfrac{0.03\times\frac{1}{15}}{0.07\times\frac{1}{1}+0.03\times\frac{1}{15}+0.1\times\frac{1}{4}} =$

$A \to C = \dfrac{0.1\times 1/4}{\left(0.07\times\frac{1}{1}\right)+0.03\times\frac{1}{15}+0.1\times\frac{1}{4}} =$

→ Vaporization Effect : $\rho = 0.6$ (Let)

(Optional)

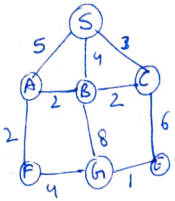


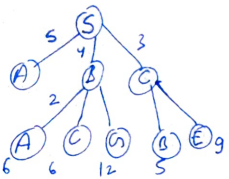
so, $(1-\rho)\times$ initial level of pheromone $= (1-0.6)\times 1 = 0.4$

19/08/25

Branch & Bound :-

(Solving TSP)





→ after getting bound, removed node whose cost is less than bound

→ complete

→ ~~Est~~ No. of iteration we don't know.

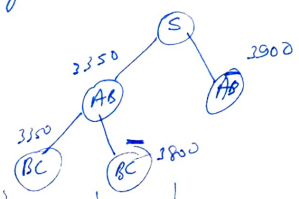
AB ✗	AB ✓		A	B	C	D	E
1500	900 ←	A	0	300	600	900	1000
1300	800 ←	B	300	0	500	800	~~800~~ 1200
1100	1100 ←	C	600	500	0	1100	1400
1700	1700 ←	D	900	800	1100	0	1500
2200	2200 ←	E	1000	1200	1400	1500	0

= 7800 ÷ = 6700

Estimated cost = $\frac{6700}{2}$ = 3350 , for $\overline{AB}$ = 3500

/bound for AB

→ Selection of min. of two and adding it the get estimated cost for every non selected edge or selected edge

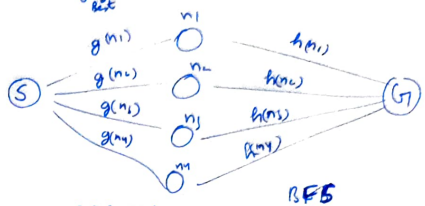


A* algo :

↳ combination of BFS & Branch & Bound

Time = $O(b^d)$

Space = $O(b^d)$



~~B~~ BKB

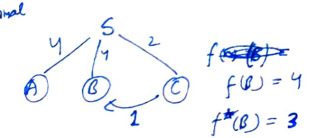
$(g(n))$

BFS

$(h(n))$

value we evaluate $f(n) = g(n) + h(n)$

Actual value /optimal $f^*(n) = g^*(n) + h^*(n)$

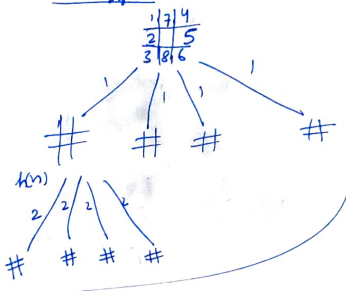


~~f(B) = 4~~

$f^*(B) = 3$

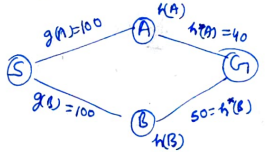
so, $g^*(n) \leq g(n)$

8-Puzzle :-



Now for $h(n)$ & $h^*(n)$

eg:-



(underestimation) ✓

Case II :- $h(A)=30$, $h(B)=20$

$S \to B \to G = 120$

but actual = 150 $(h^*(B))$

so, we will evaluate

$S \to A \to G = 130$ ✓

then 140 so optimal answer is find

→ $\therefore h(n) \leq h^*n$

Case I (overestimation)

$h(A) = 80$, $h(B) = 70$

so, $S \to B \to G = 150$

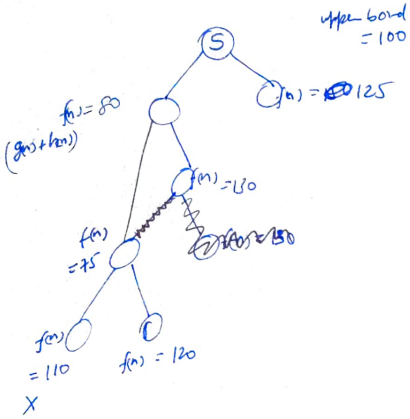
but we ignore (A),

but if we $S \to A \to G \to (140)$

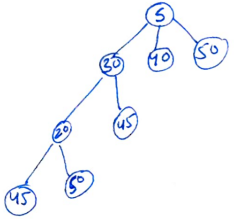
so, optimal answer gets is not

Iterative deepening A* (IDA) :-

like DFID but we compute $f(n)$



Best First Search ($h(n)$) :-



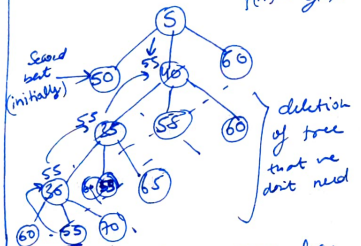
(i) works on $h(n)$

(ii) only picks the best possible value based on $h(n)$.

go down, when we don't have initial second best (i.e 50) we update the best value of existing node (i.e 55) to it's top most parent, then it becomes the second best value and 50 is the best value & we will explore it

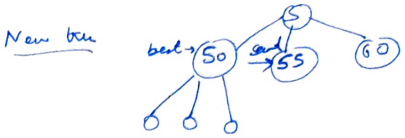
Recursive Best first Search :-

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



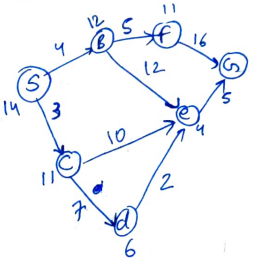
(i) works on $f(n) = g(n) + h(n)$

(ii) we always maintain a pointer to the second best value (i.e 50) then we exhaust the node that is selected first (i.e 40) as we any best values other than



Thrashing → when we generate & delete existing various paths acc. to our need.

A* Example :- $f(n) = g(n) + h(n)$



$f(S) = 14$

$S \to B \to 4+12 = 16$

$S \to C \to 3+11 = 14$

$SC \to E \to 3+10+4 = 17$

$SC \to d \to 3+7+6 = 16$

~~$SCD \to e \to 3+7+2+4 = 16$~~

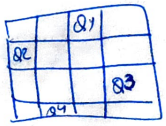
$SCD \to e = 16$

~~$SCDE \to G = 17$~~

$SCDE \to G = 17$

$SB \to F = 4+5+12 = 20$

$SB \to e = 20$



26/08/25

Pruning the Closed list

↳ Eliminating the unnecessary nodes present in the closed list

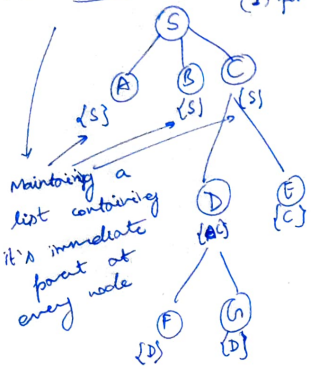
Cond" → (i) Stop looking back (Not exploring the nodes we have already explored)

(Purpose of closed list) (ii) Path construction

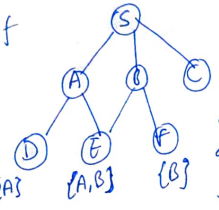
↳ Path from starting to Goal node.

(i) DCFS (Divide & Conquer Frontier Search) :-

(1) for Stop looking back

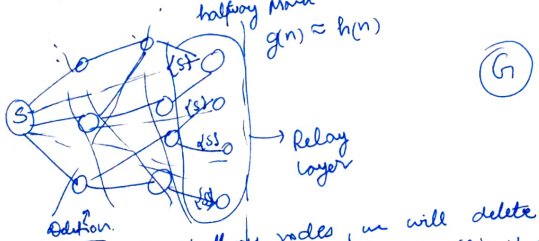


if

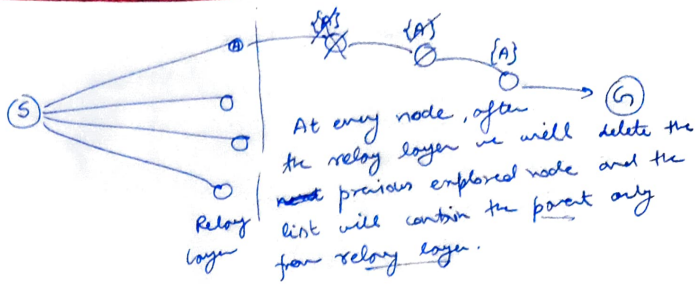


we have common child of any two parent, then all parents are added in the list.

(2) for Path construction



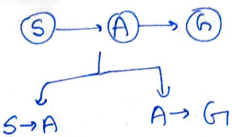
→ After reaching the halfway nodes, we will delete all the previous nodes, and everone's parent will be (S) stored in the list as {S}. And the layer of all nodes at halfway mark are called relay layer.

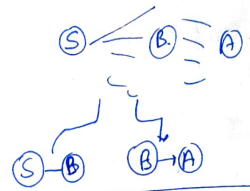


At every node, after the relay layer we will delete the previous explored node and the list will contain the parent only from relay layer.

Path reconstruction

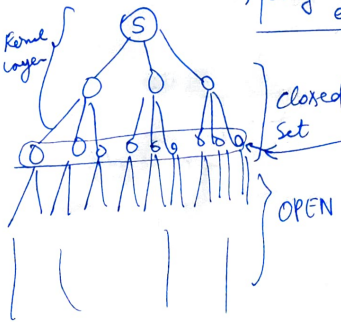
↳ we have





(ii) SMGS (Sparse Memory Graph Search) :-

↳ Only delete/prune when you don't have enough space

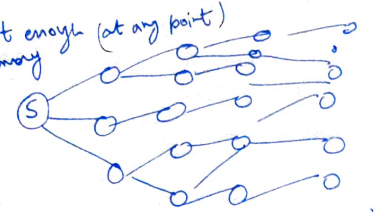


Kernel layer

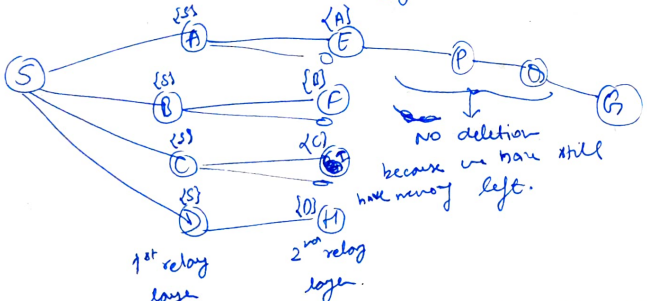
Boundary Layer
↳ whose neighbouring level nodes are in the OPEN Set.

→ Acc. to SMGS :- If you have enough memory then avoid pruning.

→ if not enough (at any point) memory



↓ Delete all nodes in kernel layer except and their parent (S) will be from relay layer.



Path reconstruction → S → A → E → P → Q → G

→ S→A + A→E + E→P→Q→G

28/8/25

Pruning the OPEN list :-

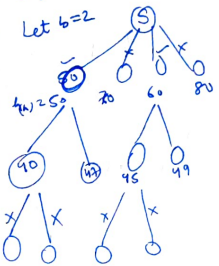
(i) → Hill Climbing

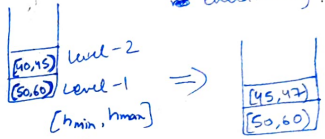
(ii) → BEAM search

(iii) → BFHS (Breadth First Heuristic Search) → Not complete

↳ we set an upper bound depending on a heuristic f". and if the heuristic value is higher than the bound then we discard that node.

(iv) → BEAM Stack Search :- [complete because we also search discarded nodes if we didn't find any better child nodes].





like in [40,45]

If we didn't find any answer better than parent, then we will take the next best value from hmax and make it to new hmax and the hmax will become new hmin

[40, 45) ⇒ [45, 47) ⇒ [47, 49)
hmin hmax — if no answer — hmin hmax — if no answer — hmin hmax

---

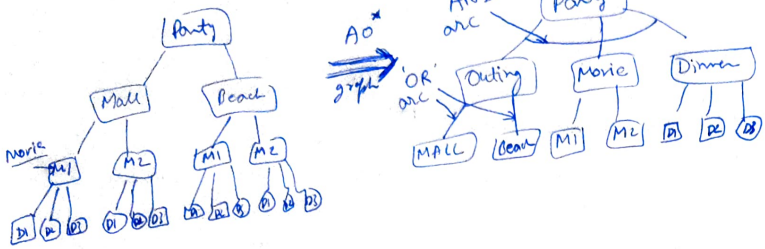
Merging pruning of OPEN & CLOSED list Algo :-

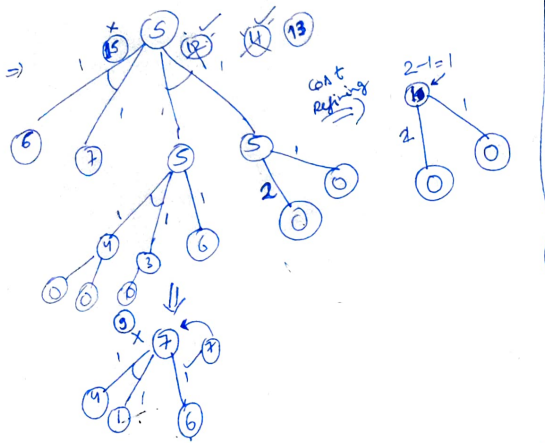
(No running)

(i) DCBSS → Divide & Conquer Beam Stack Search. (DCFS + BSS) (Close) (OPEN)

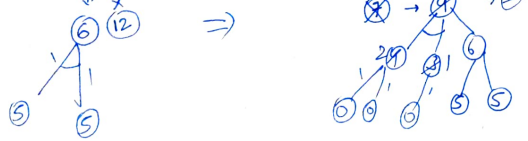
(ii) DCBFHS → Divide & conquer Beam first heuristic Search. (DCFS + BFHS) (Close) (OPEN)

AO* Algo. (AND OR Algo.)

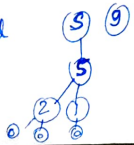




→ Only generate the level of each node, then compare and update the cost after refining at each node.



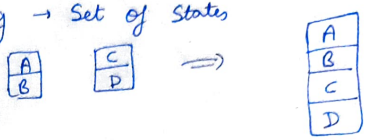
So, final answer:



Note :- It may or may not give the optimal sol.

02/09/25

Planning → Set of States



Initial State

On Table (B)
on Table (D)
on (A, B)
on (C, D)
Arm Empty
Clear (A)
Clear (C)

(Represent States)
Domain Predicate (S)

On (X, Y) → X is on Y
on Table → on Table (T(X))
Clear (X) → No one is above 'X'.
Holding (X) → Robert holds X
Arm Empty → Robert not holding any thing

Action (Y)

Pickup (X) → X on Table & None is above
Put Down (X) → Putting block on table
Unstack (X, Y) → Pick X from Y
Stack (X, Y) → Putting X on Y

Pre Cond":-

On T(X) & Clear(X) & A.E ← For Pickup
Holding (X) ← For putdown (X)
On (X, Y) & Clear(X) & A.E ← Un Stack (X, Y)
Clear(Y), & Holding (Y) ← Stack (X, Y)

After Action Effect

Holding (X)
A.E, onTable, Clear(X)
Holding (X), Clear(X)
A.E Clear X, on(X, Y)

Action Unstack(A, B)
Holding (A)
Clear (B)
On(C, D)
Clear C

S domain predicate
unstack (C, D)

FSSP
↓
Forward State Space Planning
(all the Algo's we have learnt that come under this)

In a state space action A = progress (S, a) = S' = {S - effect⁻(a)} ∪ {effect⁺(a)}

BSSP (Backward State Space Planning) :-

$G \xrightarrow{a} G'$

$G' = [G - effect^{+}(a)] \cup precond^{-}(a)$

$[on(A,B) \wedge on(B,D)]$

(G)

Stack(A,B)

Holding (A)
Clear (B)
On (B, D)

Stack (B, D)
Holding (B)
Clear (D)
On (A, B)

One by one we will take predicates action for each of them.

Pickup(A)

Unstack (A, X)

Unstack (Y, B)

Can't hold more than

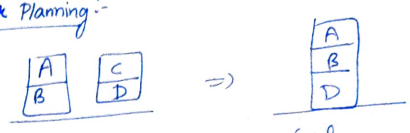
Clear A	On (A, X)	On (X, B)	Holding (B)
on Table (A)	Clear (A)	A.E, Clear (X)	Clear (A)
Clear (B)	Clear (B)	Holding (A)	Holding (A)
on (B, D)	on (B, D)	on (B, D)	Clear B

Drawback FSSS

↳ Large branching factor

04/09/25

Goal Stack Planning :-



Initial ⇒ Goal

Order doesn't matter

Goal state predicate

→ Stack (Reciprocal for representation)

Top

Not True → Action → 6th Action → Stack (B, D)

Pre cond" → Holding (B) & Clear (D)

↳ Holding (B) Pop

Clear (D) Pop

Unstack (C, D) Pop

1st action →

A.E, on(C,D) & Clear(C)

↳ A.E Pop

On (C, D) Pop

Clear (C) Pop

5th Action → Pickup (B)

A.E & OnTable(B) & Clear

A.E

Clear (B) Pop

On Table (B)

Unstack (A, B)

3rd Action →

A.E

Clear A Pop

On (A, B) Pop

Putdown (C)

Holding (C)

2nd Action →

Putdown A

4th Action →

Holding (A)

Stack (A, B)

8th Action →

Holding (A) & Clear (B)

Clear (B) Pop

Compare it from start state if not true then pop and write the action what needs to be done

and along with action also push it's precond"

when an action gets popped that means that will be the first action of our Plan

⇓

Also now, the initial state will be changed, acc. to action

C ← Robot

Trace

New stack after 3rd action →

PLAN

1) Unstack (C, D)
2) Putdown (C)
3) Unstack (A, B)
4) Putdown (A)
5) Pickup (B)
6) Stack (B, D)
7) Pickup (A)
8) Stack (A, B)

7th Action → Pickup(A)

~~A-E~~

~~Clear(A)~~

→ Goals are serializable → that means a specific order of goals will be better than other

Better → | On(A,B) ∧ On(B,D) | / | On(B,D) |

or | On(A,B) ∧ On(B,D) | / | On(B,D) | / | On(A,B) |

Both will give correct answer

→ But we have to do less work in first one.

→ Sussman Anamoly → we have to rework

| On(A,B) ∧ On(B,C) / On(A,B) / On(B,C) |

Δ

| On(A,B) ∧ On(B,C) / On(B,C) / On(A,B) |

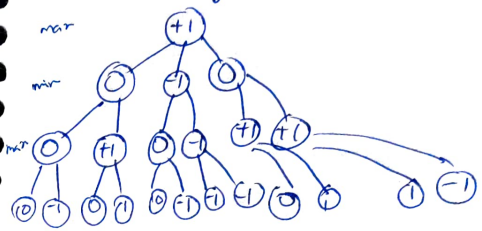
In Both cases we have to do one action twice so there is no better serial, ∴ this is an Sussman Anamoly

10/09/25

Game Planning

↳ Back-up rule

Tick-Tac-Toe software → more all game



b → brandy factor

d^{th} level

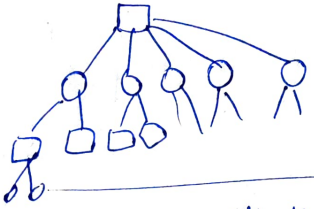
time comp. $\Rightarrow b^d$

space $\Rightarrow bd$

Chess avg. branching factor ≈ 35

no. of moves to end game = 50 (both players)

35^{50} → Back up rule failed.



Chess → we go upto K-level

↓ material value ↓ positional value

(intermediate board pos" & generate again & again

DFS algo.

min-max algo

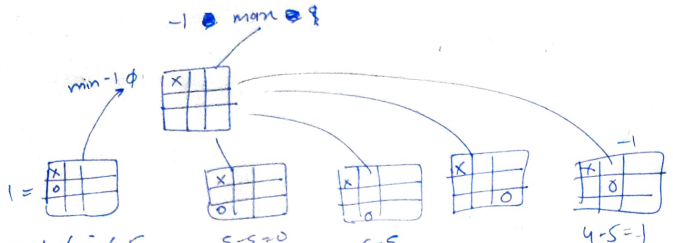
based on heuristic value

10/09/25

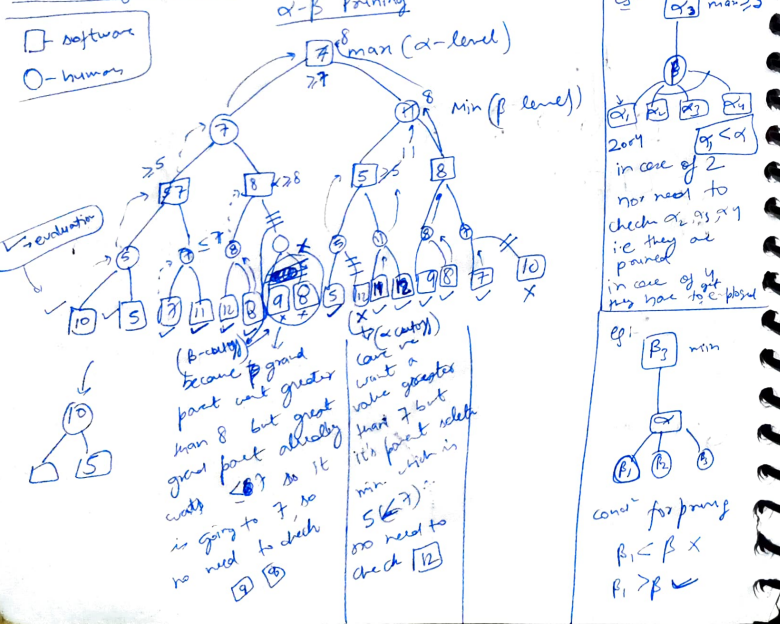
Game Playing (Min-Max Algo) Time → $O(b^d)$

- Time $O(b^d)$
- Space $O(b \cdot d)$

} like DFS



(no. of possible rows + colm of (1) + diagonal − no. of possible rows + colm of (0) + diagonal)



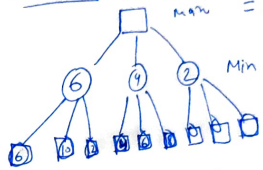
if the edge we cutoff / prune is going to a min node (○) then it is called β-cutoff.

if edge going to max node (□) → α-cutoff.

→ Space complexity = $O(b \cdot d)$

→ Time comp.

Best case $T(d) = 1 \cdot T(d-1) + (b-1)\,T(d-2)$

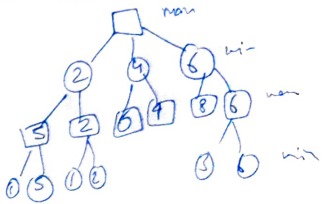


↓

For best case, in min level the values should be in decreasing order, like (6, 4, 2)

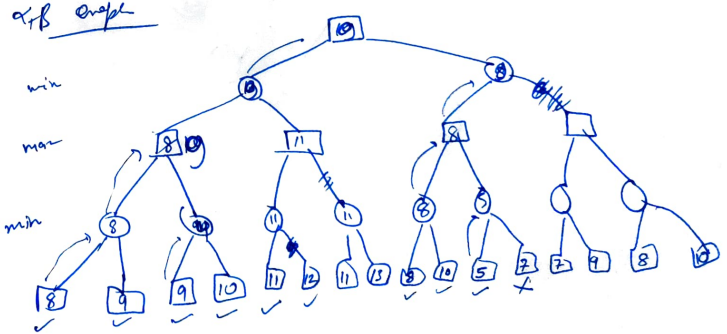
and for max level → in inc. order like (6, 10, 12)

Worst case $T(d) = b \cdot T(d-1)$



→ For worst case, vice-versa of best case.

α-β example



30/09/25

Naive - Bayes's (Ex: Email spam or not spam) classifier → generative supervised learning algo.

→ categorical data

The prob of new example x_{new} belonging to class $y \in Y$ can be calculated using Bayes's Thm,

$$P(y/x_{new}) = \frac{P(x_{new}/y) \cdot P(y)}{\sum P(x_{new}/y) \cdot P(y)}$$

↓ class label

Example

words	Label (Class)
Algo, Tree, Graph	CS
Tree, life, Gene, Algo	Bio
Graphs, NP, Algo, Tree	CS
Protein, Assay, Cell	Bio

Now, $P(y = CS) = 2/4 = 0.5$

$P(y = Bio) = 2/4 = 0.5$

→ class-cond' prob,

$$P(Algo \mid y = CS) = \frac{P(Algo, y = CS)}{\sum_w P(w, y = 'CS')}$$

$$= \frac{\#\text{ Algo. in 'CS' docs}}{\#\text{ (Total no. of words in CS doc)}} = \frac{2}{7} \approx 0.3$$

Similarly, $P('Tree' \mid y = CS) = 2/7$

$P(Algo, \mid y = Bio) = 1/7$

$P(y = CS \mid x_{new}) \propto P(y = CS)\ \pi_w (P(w/y = CS)$

Prob. of doc x_{new} containing the words (Algo, Tree)

$\propto P(y = CS) * P(Algo \mid y = 'CS') * P(Tree \mid y = C$

$\propto \frac{1}{2} \times \frac{2}{7} \times \frac{2}{7} = \frac{4}{98}$

$P(y = CS \mid x_{new}) \propto \frac{4}{98} = 2/49$

Similarly,

$P(y = Bio \mid x_{new}) = P(y = Bio) * P(x_{new} \mid Bio)$

$= (1/2) \times (2/7 \times 1/7)$

$= 1/98$

Now

$$P(y = 'CS' \mid x_{new}) = \frac{P(y = CS) \times \pi P(w/y = CS)}{P(x_{new})}$$

$$= \frac{4/98}{2/49 + 1/98} = 4/5 = 0.2$$

$$P(y = Bio \mid x_{new}) = \frac{P(y = Bio) \times \pi_w (P(w/y = Bio)}{P(x_{new})}$$

$$= \frac{1/98}{2/49 + 1/98} = 1/5 = 0.2$$

So, $P(y = 'CS' \mid x_{new}) > P(y = 'Bio' \mid x_{new})$, so we assign the 'CS' to x_{new} (ie new doc) containing the words (Algo, Tree)

→ Numerical data

Study	Sleep	Result
2	10	Fail
4	12	Fail
3	11	Fail
8	8	Pass
6	9	Pass
7	7	Pass

→ In numerical data if Naive Bayes' is applied then it is called Gaussian Naive Bayes

$$P(y \mid x_{new}) = \frac{P(x_{new} \mid y) \cdot P(\theta)}{\sum_y P(x_{new} \mid y) \cdot P(y)}$$

y ↓ Study = 4, Sleep = 6

Now, $P(\text{Fail}) = P(\text{Pass}) = 0.5 \quad (= 3/6)$

Mean = ? Variance = ?

For Fail Class

Study (mean) = 3

Sleep (mean) = 11

Study variance $= \frac{(2-3)^2 + (4-3)^2 + (3-3)^2}{3} = 2/3$

Sleep variance $= \frac{(10-11)^2 + (12-11)^2 + (11-11)^2}{3}$

$= 2/3$

For Pass Class

Study mean = 7

Sleep mean = 8

Study variance $= \frac{(8-7)^2 + (6-7)^2 + (7-7)^2}{3}$

$= 2/3$

Sleep variance $= \frac{(8-8)^2 + (9-8)^2 + (7-8)^2}{3}$

$= 2/3$

Gaussian N.B

$P(x_{new} \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \text{Exp}\left[\frac{-(x-\mu)^2}{2\sigma^2}\right]$

x_{new}: for Study = 4, Sleep = 6; μ: mean; σ^2: variance

For Fail Class

$P(\text{Study} = 4 \mid \mu = 3, \sigma^2 = 2/3)$

$= \frac{1}{\sqrt{2\pi(2/3)}} \times e^{\left[\frac{-(4-3)^2}{2\times 2/3}\right]}$

$= 0.33 \times e^{-0.375}$

$= 0.09$ (≈ Let 0.03)

$P(\text{Sleep} = 6 \mid \mu = 3, \sigma^2 = 2/3)$

$= \frac{1}{\sqrt{2\pi \cdot 2/3}} \times e^{\left(\frac{-(6-3)^2}{2\times 2/3}\right)}$

$= 0.33 \times 0.001$

Let ≈ 0.04

For Pass Class

$P(\text{Study} = 4 \mid \mu = 7, \sigma^2 = 2/3)$

$= \frac{1}{\sqrt{2\pi(2/3)}} \times e^{\left(\frac{(4-7)^2}{2\times 2/3}\right)}$

(≈ Let 0.04)

$P(\text{Sleep} = 6 \mid \mu = 7, \sigma^2 = 2/3)$

$= \frac{1}{\sqrt{2\pi \cdot 2/3}} \times e^{\left(\frac{-(6-7)^2}{2\times 2/3}\right)}$

Let ≈ 0.01

Now,

$0.03 \times 0.04 \times P(\text{Fail})$

$0.03 \times 0.04 \times 0.5$

⇒ Let (0.7)

$\rightarrow (0.04)(0.01) \times P(\text{Pass})$

$\rightarrow 0.04 \times 0.01 \propto 0.5$

⇒ Let = (0.8)

That new for new (Study = 4, Sleep = 6) ⇒

$P(\text{Fail} \mid x_{new}) < P(\text{Pass} \mid x_{new})$

$0.7 < 0.8$

⇒ Then x_{new} will pass.

Decision Tree (Discrimination Trees)

Experience	Education	Hands-on	Salary (Class label)
Low	BTech	No	Low
Low	PHd	No	Medium
Medium	PHd	No	Medium
High	MTech	No	High
High	MTech	Yes	Very high

Possible Output :-

Exp → Low, medium, high

Education → BTech, MTech, Phd

Hands → Yes, No.

Salary → Low, Medium, high, very high

⇒ Let Total rows = 48

Salary class label

Low = 10, Medium = 15, high = 15, V.High = 8

Entropy → Uncertainty in the dataset

Eni-

x	w	Gender
6	70	M
5	60	M
5.5	40	M

$$\text{Entropy} = \sum \left[-P_i \log_2(P_i)\right]$$

For salary class label $= \left(\frac{-10}{48} \times \log\left(\frac{10}{48}\right)\right) + \left(\frac{-15}{48} \log\left(\frac{15}{48}\right)\right) + \left(\frac{-15}{48} \times \log\frac{15}{48}\right) + \left(\frac{-8}{48} \times \log\frac{8}{48}\right) \approx$ (set of 8)

→ Education

BTech	MTech	Phd	Salary
7	4	5	Low
	3	5	Medium
		7	High
8	8	1	Very high
Σ = 15	15	18	

→ Entropy

BTech

$$\left(\frac{-7}{15} \times \log\frac{7}{15}\right) + \left(\frac{-8}{15} \log\frac{8}{15}\right) \approx 0.02$$

MTech

$$-\frac{4}{15} \log\frac{4}{15} + \left(\frac{-3}{15} \log\frac{3}{15}\right) + \left(\frac{-8}{15} \log\frac{8}{15}\right) \approx 0.01$$

Phd

$$\left(\frac{-5}{18} \times \log\frac{5}{18}\right) + \left(\frac{-5}{18} \log\frac{5}{18}\right) + \left(\frac{-7}{18} \times \log\frac{7}{18}\right) + \left(\frac{-1}{18} \log\frac{1}{18}\right) \approx 0.01$$

Info. Gain [Education]

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v)$$

$$= 1.8 - \left[\frac{15}{48} \times 0.02 \times \frac{15}{48} \times 0.01 \times \frac{18}{48} \times 0.01\right]$$

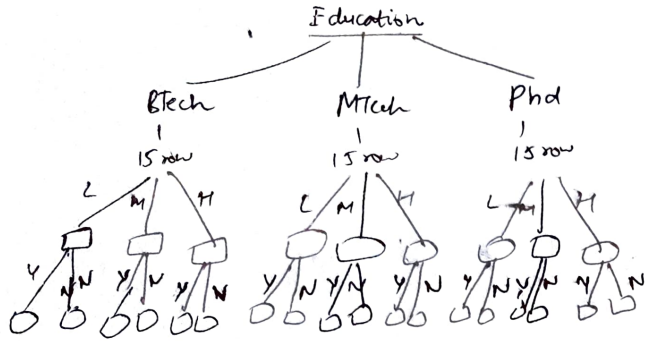
$$= 1.8 - \left(\frac{0.3}{48} + \frac{0.6}{48} + \frac{0.18}{48}\right) \approx 0.051$$

$$\approx 0.5$$

Similarly, I.Gain (Exp) ≈ 0.3

I. Gain (Hands on) ≈ 0.2

→ Better I.Gain → Better distinguishable factor.



Bayesian Network :-

prob. graphical model

DAG (Directed Acyclic Graph)

Given, $P(Rain) = 0.6$ (P(R)), so $P(No\ Rain) = 0.4$ [P(~R)]

→ Prob. of sprinkler depending on Rain,

Rain	Sprinkler	No Sprinkler
Y (R)	0.6	0.4
N (~R)	0.3	0.7

→ Prob. of wet grass depending upon Sprinkler & Rain,

Rain	Sprinkler	Wet Grass	No Wet Grass
Y	Y (S)	0.99	0.01
Y	N (~S)	0.7	0.3
N	Y	0.6	0.4
N	N	0.2	0.8

Q) $P(Wet\ Grass, Sprinkler, No\ Rain)$

$= P(W, S, \sim R) = P(W \cap S \cap \sim R)$

$= P(W \mid S, \sim R) \times P(S, \sim R) \times P(\sim R)$

$= 0.6 \times 0.3 \times 0.4$

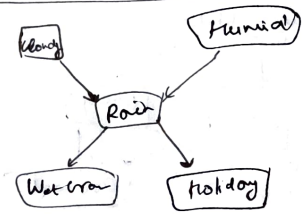
$= 0.072$

Q) $P[\sim W \cap S \cap \sim R) = P(\sim W \mid S, \sim R) \times P(S \mid \sim R) \times P(\sim R)$

$= 0.4 \times 0.3 \times \times 0.4$

$= 0.048$

Eg.



$P(holiday, \sim R, \sim H, \sim C)$

$= P(H \mid \sim R, \sim H, \sim C) \times$

$P(\sim R \mid \sim H, \sim C) \times$

$P(\sim H \mid \sim C) \times$

$P(\sim C)$

Markov Model :- (only depends on immediate past)

(Seq^l data pred^n)

Sunny	Cloudy	Rainy
0.3	0.5	0.2

Given

→ Next Day Table (Transition table)

	Sunny	Cloudy	Rainy
Sunny	0.6	0.3	0.1
Cloudy	0.2	0.5	0.3
Rainy	0.1	0.4	0.5

Today(1st) Next day(2nd) 3rd Day 4th Day

Ex:- (i) Raining → Sunny → Cloudy → Raining

P(R) P(S|R) P(C|S) P(C|R)

$0.2 \times 0.1 \times 0.3 \times 0.3$

$= 0.0012$

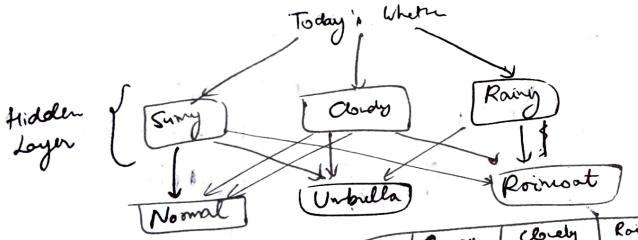
(ii) S → C → R → S

$\Rightarrow P(S) \times P(C|S) \times P(R|C) \times P(S|R)$

$\Rightarrow 0.3 \times 0.3 \times 0.3 \times 0.1$

$\Rightarrow 0.0027$

Hidden Markov ~~Rain~~ Model :-



Sunny	Cloudy	Raining
0.3	0.5	0.2

	Sunny	Cloudy	Raining
Sunny	0.6	0.3	0.1
Cloudy	0.2	0.5	0.3
Raining	0.1	0.4	0.5

Observation / Emission Prob. Table

	Normal	Umbrella	Raincoat
Sunny	0.6	0.3	0.1
Cloudy	0.4	0.4	0.2
Raining	0.1	0.4	0.5

Ex:- Raincoat → Normal → Umbrella → Raincoat

C|R|S C|R|S C|R|S C|S|R

$3 \times 3 \times 3 \times 3 = 81$ cases

Now out of 81 cases, we have given the each day's

Day → Raining Sunny Cloudy Raining

↓ ↓ ↓ ↓

Raincoat Normal Umbrella Raincoat

$P(R_c|R) \times P(N|S) \times P(U|C) \times P(R_c|R)$

$\times 0.6 \times 0.4 \times 0.5$

$\Rightarrow 0.5$

× Today Rain × Next Sunny × Next Cloudy × Next Raining

~~2~~ Day Day Day

$0.2 \times 0.1 \times 0.3 \times 0.3$

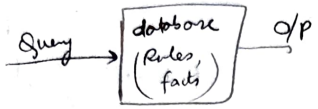
$\Rightarrow 0.5 \times 0.6 \times 0.4 \times 0.5 \times 0.2 \times 0.1 \times 0.3 \times 0.3$

$\Rightarrow 0.0000432$

Prolog

→ logical programing

eg:- cakes are delicious
chocolates are delicious
Pickles are delicious



→ Priya likes food if food is delicious

Prolog :- delicious (cakes).
" (chocolates).
" (Pickles).
} facts (end with full stop)

likes (Priya, food) :- delicious (food)
then statement — if statement

Variable - with capital letter (starts with)
constants → small/lowercase

In above example,
→ like, priya, cakes, chocolates, pickles } constants - small letter
→ food } variable, can be cake, pickle
↓ capital letter

ex:- cakes are sweet
chocolates are sweet
} adjective ( ) noun → Sweet (cakes)
Sweet (chocolate)

ex:- Prakash likes food if food is delicious and tastes sweet.
likes (Prakash, food) :- delicious (food), sweet (food).
if and used separates by comma (,)

ex:- if (OR used in if condn)
↳ separate by semicolon (;)
likes (Prakash, Food) :- delicious (food); sweet (food).

For Queries :-

ex:- Priya likes which food ?
Query => ? likes (priya, food) } gives only first matched result
o/p => X = cakes
for all output : Press ;

Ex:- give all person with like food?
=> ? likes (Y, X).
Y = priya
X = cakes
→ ∀ Press;

ex:- car (maruti, white).
car (maruti, black).
car (hyundai, grey).
car (" , yellow).
} if we want only the colour irrespective of the company then query,
? { car (—, color).
o/p :- colour = white
∀ o/p Press;

9/10/25 uses backtracking

, → and
; → or

Text: ML & Prologic

Prologue (extension .pl)

then it is in the output

likes (Prakash, X) :- sweet (X), delicious(X).] → rule
sweet (chocolates).] → fact
sweet (cakes).
delicious (cakes).

↳ If this is true

Ex:
query (i) ?- likes (Prakash, Food)
Food = cakes; { ∵ it satisfies both cond sweet & delicious }
No

Now, to check if there are more available answers to the query then we add ';' after the first output (like Food = cakes;), and if there is no answer possible then it says no.

(ii) ?- likes(prakash, cakes) ← If this is true then answer will be yes otherwise no.
Yes

Prologic :- likes (Prakash, X) :- sweet (X), delicious (X), write(X), nl, fail(X).
sweet (chocolates)
sweet (cakes)
delicious (cakes)
delicious (chocolates)
sweet (pastry)
delicious (pastry)

→ To get output without using semicolon again & again.
→ fail(X) means that the work you have done till now is null/failed, so it produces it again.

query :- ?- likes (Prakash, food)
output → Food = chocolates
Food = cakes
Food = pastry
no.

Program to find max/min of two no.

max (X, Y, X) :- X >= Y

query : ?- max (10, 8, X)
X = 10 ← check if (10 > 8)

?- max (8, 10, X)
no ← check if (8 > 10)

So, for this case we will add another rule

max (X, Y) :- Y > X

therefore final rules,

max (X, Y, X) :- X >= Y
max (X, Y, Y) :- Y > X

Program to find factorial :-

fact (0, 1).
fact (N, result) :- N > 0, N1 = N - 1,
fact (N1, R1)
Result = N * R1

working

query → fact (3, X)
N = 3
N1 = 2
fact (2, R)
Result 3 * R1

fact (2, R1)
N = 2
N1 = 1
fact (1, R2)
R1 = 2 * R2(1) = 2

fact (1, R2)
N = 1
N1 = 0
fact (0, R3 = 1)
R2 = 1 * R3(1)

List = [1,2,3,4] = [1 | [2,3,4]] = [H|T]

(H|T)

Head (First element)

Tail (remaining elements)

Program to find sum of elements of the list.

```
sum ([],0)
sum ([H|T], Result) :- Sum([Tail], R1),
                       Result = H+R1.
```

query

sum([1,2,3], X)

X = 6

Given any list if any given no. belongs to the list or not

member([1,2,3], 3)

Yes

```
member([X|_], X)
member([_|T], X) :- member(T, X)
```

_ → represents anonymous variable

Determine the length of list

Append / Concatination of two list

Reversing a list

Finding last element of a list

14/8/25

CUTS (!) symbol → It prunts backtracking

likes (prakash, food) :- sweet (food), delicious (food), !, healthy (food).

! → no backtrack for other variable

sweet (cakes)
sweet (chocolate)
delicious (cakes)
delicious (chocolate)
healthy (chocolate)

? likes (Prakash, X).
no { because there is no healthy food (cake) }

It will not do backtrack but if one variable satisfies all it is condⁿ then it will search for new variable after prevy ';'
If (then we can press semi colon)

Predicate Logic :-

ex:-
① marcus was a man → man (marcus)
② " is christian → christian (marcus)
③ all christian are roman → ∀ x: christian (X) → roman(X)
④ all romans were either loyal to caesar or hated him → ∀ x: roman (X) → loyal (x, caesar) ∪ hate (x, caesar)
⑤ everyone is loyal to someone → ∀ x, ∃ y: loyal (x, y)
people only try to assassinate rulers they are not loyal to → ⑥ ∀ x, ∀ y: person(x) ∩ ruler(y) ∩ try to assassinate (x, y) → ¬ loyal (x, y)

Marcus try to assassinate caesar
⑧ → try to assassinate (marcus, caesar)

⑦ ∀ x man(x) → person (x)

Prove that
Marcus is not loyal to caesar
using 7th
person (m) ∩ ruler (c) ∩ try to assassinate (M, C)
↓ ⑧
person (m) ∩ ruler (c)
↓ ④
person (m)
↓ ①
man (m)
↓ ⑩
true

Using resolution

(i) P ∨ Q → P ∨ Q
(ii) P → R → ¬P ∨ R
(iii) Q → R → ¬Q ∨ R

⑧ CNF :- (P ∨ Q) ∩ (¬ ∨ P) ∩ (E ∨ P) ∩ Q, Prove is R is True

How to solve :- Start with negation & assume it to be true & reach the contradiction point

Sol :- → Assume (¬) → R is True
using ① & ②
(v) Q ∨ R ← (because at least one of Q or R is True)
using ④ & (v).
(vi) → Q
using ③ & ⑤.
(vii) R → ∴ our assumption is wrong, hence R is True

To contradicts the fact that ¬R is True,

8/10/25

Solve using resolution. Prove R.

1) $\sim(P \to Q) \to Q$

2) $(P \to P) \to R$

3) $(R \to S) \to \sim(S \to Q)$

ex: 1) $\sim(P \to Q) \vee Q$

$\to (\sim P \vee Q) \vee Q$

$= (P \wedge Q) \vee Q$

$= (P \vee Q) \wedge (Q \vee Q)$

$= P \vee Q$ —(i)

(2) $\sim(\sim P \vee P) \vee R$

$(P \wedge \sim P) \vee R = \underbrace{(P \vee R)}_{(ii)} \wedge \underbrace{(\sim P \vee R)}_{(iii)}$

(3) $(R \to S) \to \sim(S \to Q)$

$\Rightarrow \sim(\sim R \vee S) \vee \sim(\sim S \vee Q)$

$\Rightarrow (R \wedge \sim S) \vee (S \wedge \sim Q)$

$= \underbrace{(R \vee S)}_{(iv)} \wedge \underbrace{(R \vee \sim Q)}_{(v)} \wedge \underbrace{(S \vee S)}_{} \wedge \underbrace{(\sim S \vee \sim Q)}_{(vi)}$

let $\sim R$ (7)

assume

using $S \& (7) \to \sim Q$

using $1 \to P$

using $3 \to R$ is True (contradiction)

$\therefore \sim R$ is wrong assumption

$\therefore R$ is true

gt → greater than

lt → less than

gt(10,1) → 10 is greater than 1

lt(1,10) → 1 is smaller than 10.

born(marcus, 40)

all number mortal → man(x) → mortal

all christan died due to volcanic eruption,

$\forall x \Rightarrow$ christian(x) $\wedge$ velocity (79)

→ died (x, 79)

No, mortal lives

$\forall x$ mortal(x) $\wedge$ born(x, t_1) $\wedge$ gt ($t_2 - t_1$, 150)

→ died (x, t_2)

alive mean not dead

alive(x) → → dead(x)

If someone dies then, he is dead all future times.

To, prove, died(x, t_1) $\wedge$ gt(t_2, t_1)

→ died (x, t_2)

To prove,

Marcus is now dead,

x now (1991)

→ died (man, 1991)

→ alive true.

29/10/25

CSP

Constraint Satisfaction Prob.

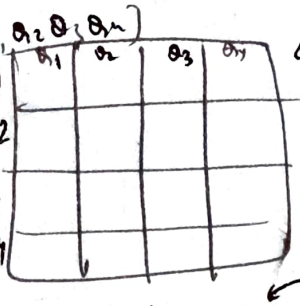
def. by

[X] = set of variables

(D] = Domain, [C] = constraint

[X] = [Q_1, Q_2, Q_3, Q_4]

(D] = {1, 2, 3, 4}



← n-queen

[Constraint on Q_1 & Q_2] → C_{12} = [(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)]

← pairs on which Q_1 & Q_2 are allowed

C_{13} = [ ]

C_{14} = [ ]

→ Binary constraints → constraints on two variables

How to solve

Let a = { $Q_1 = 1$, $Q_2 = 2$ } ← values from domain

assignment

now check in C_{12} if (1,2) is allowed

∴ it's not allowed,

∴ a = { $Q_1 = 1$, $Q_2 = 3$ }

Let → $Q_3 = 2$

Now check in C_{13} & C_{23} if (1,2) & (3,2) are allowed

∴ it's not allowed,

∴ a = { $Q_1 = 1$, $Q_2 = 3$, $Q_3 = 3, 1, 4$ → all are not allowed }

so, we backtrack to Q_1 & Q_2 & change their values.

→ a = { $Q_1 = 1$, $Q_2 = 4$, .

repeat steps to get the answer.

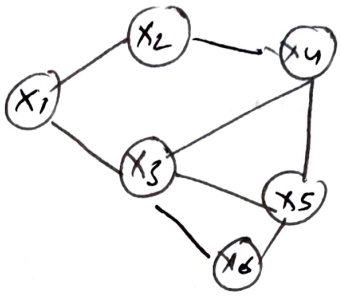
Graph colouring problem from CSP :-

Variables = Nodes = [X_1, X_2 ⟶ X_6]

Domain = [R, G, B]

Constraints = ↓ X_1, X_2 = [(R,G), (R,B), (G,R), (G,B), (B,R), (B,G)]

that are allowed



Criptarithmetic Problem

Message → EAT + THAT = APPLE

$X_4 X_3 X_2 X_1$

A = 1, P = 0, T = 9, E = 8, L = 3

$X_1 = 1$, $X_2 = 0$, $X_3 = 1$, $X_4 = 1$, H = 2

[X] = {A, E, H, L, P, T, X_1, X_2, X_3, X_4}

[D] = {0, 1, 2, ⋯ 9}

[C] = [T + T = E + 10 × X_1,

A + A = L + 10 · X_2, + X_1

E + H = P + 10 · X_3, + X_2

T + X_3 = P + 10 · X_4.

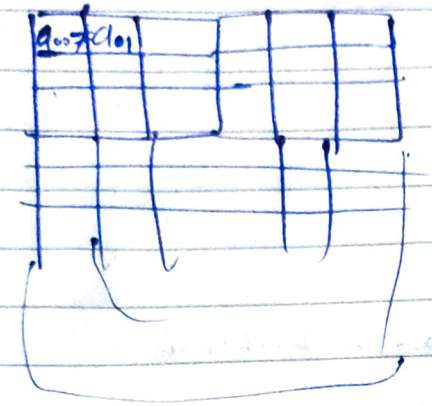
A = X_4.

Ex: SEND + MORE = MONEY

4th Nov, 2025

AI

Sudoku as CSP

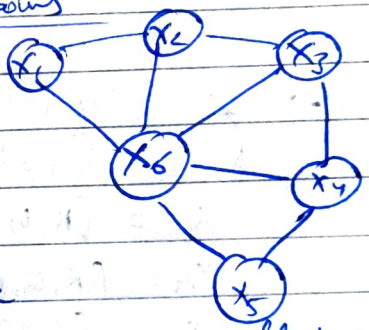


Domain = 1 to 9
Variable = cells.
Constraints = that in a column, row or a box no two cells can have same value.

Three approaches to improve backtracking :-

1. Forward checking / Constraint propagation.
   ↳ checks for constraints in its neighbours

eg: Graph coloring



(i) X_1 = [Red] ← Let one colour

(ii) Now, check the constraints where X_1 is present, and remove the colour you selected for X_1 in it's neighbour.

eg: (X_1, X_2) = {RG, RB, GR, GB, BR, BG}
↓ only possible set left (since X_1 = Red)
$\Rightarrow X_2 \rightarrow$ G or B

(iii) Now, X_2 = [G, B]

(iv) Now, check for X_3,
$[X_1, X_3]$ = {RR, B,B, GG, ____}
→ all allowed since X_1 & X_3 are not connected.

Limitation :- If $X_3 = [Green]$

↓

It delays in identifying the failure.

$X_1 = [Red]$ $\quad X_4 = [R, B]$

$X_2 = [B]$ $\quad \xleftrightarrow{\times} \quad X_6 = [B]$

$X_3 = [G]$ $\quad X_5 = [R, G, B]$

Date : / /

Page No.

→ If fails when any variable is left with no domain, so we backtrack.

eg:- $X_1 = [Red]$

$X_2 = [Blue]$

$X_3 = [Green]$

$X_4 = [B]$

$X_6 = [\,]$ ← Now we backtrack.

$X_5 = [R, G, B]$

(2.) Arc Consistency :-

[in presence of forward checking]

→ $X_1 = [Red]$

$X_2 = [G, B]$

$X_3 = [R, G, B]$

$X_4 = [R, G, B]$

$X_6 = [G, B]$

$X_5 = [R, G, B]$

Assume arcs,

$X_2 \rightarrow X_1$

$X_1 \rightarrow X_2$

Def :→

If $A \rightarrow B$

$\forall x$ in $A \Rightarrow \exists y$ in B which doesn't violates the constraint.

In arc $X_1 \rightarrow X_2$

Red $\quad$ B, G

since there is atleast one value in X_2 that satisfies the constraint for X_1, arc is consistent

→ Do that for every arc

↓

arc is consistent otherwise backtrack.

Date : / /
Page No.

Now, Let $X_3 = G$

$X_1 = [R]$

$X_2 = [B]$

$X_3 = [G]$

$X_4 = [R, B]$

$X_6 = [B]$

$X_5 = [R, G, B]$

Now when we check the arc, $X_2 \rightarrow X_6$, we see that the arc is not consistent

So, we backtrack, and change the output of X_3

Similarly, $X_4 \rightarrow X_6$ is also not consistent.

2nd approach in Arc consistency,
we can try to make the edge consistent,
for eg:- $X_4 = [R, B]$
$X_6 = (B)$

So we remove, B for X_4 to make arc $X_4 \rightarrow X_6$ consistent.

(3.) Ordering:-

Order in variable (forward check)

3.(a) Variable with least domain values
↳ Assign the variable first which has the least domain values.

3(b) when there is two or more variable with same values then take its degree into consideration.

Date : / /
Page No.

Ordering in domain values

↳ $X_3 = [R, G, B]$ current

↳ Pick that value from domain such that it removes fewest no. of values from other domains.

If we pick $X_3 = G$ or B, the domains of X_2, X_4, X_6 will be changed.

but if we pick $X_3 = R$, only the domain of X_4 will be changed

so, $\boxed{X_3 = R}$ → is better.