AI

DEEPAK KHEMANI

8-Puzzle :

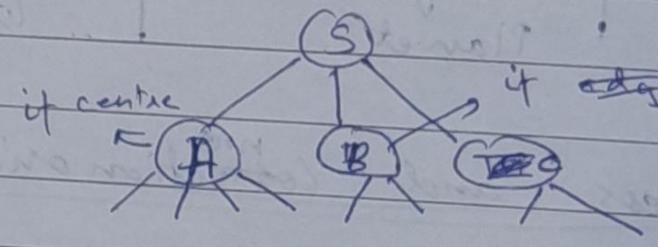
3	1	2
5	4	
6	7	8

(S)

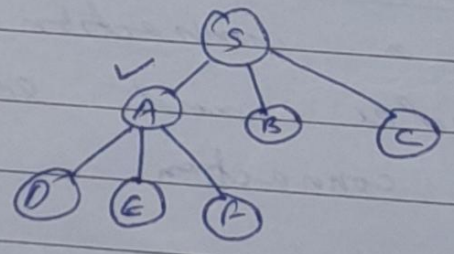
→

1	2	3
8		4
7	6	5

↳ Goal



OPEN SET : = {S} → {A, B, C} → {B, C, D, E, F}



↳ state space

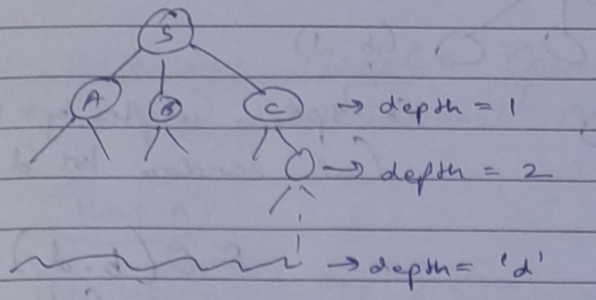
1) OPEN ← {S}
2) WHILE OPEN NOT EMPTY :
3) { Pick any node (n) from OPEN : ~~OPEN = OPEN~~
   if (n == GOAL) return TRUE
   else
   OPEN ← OPEN ∪ {neighbours (n) - Closed}
   }
4) RETURN FAILURE

CLOSED = CLOSED ∪ {n}
OPEN = OPEN - {n}

Page No.
Date

2) DFS
3) BFS

time complexity ~ O(|closed set|)



b = branching factor (avg. branches)

DFS time = $\frac{1}{2}$ [Goal at first node at depth 'd' + Goal at last node at depth 'd']

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

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

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

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

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

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

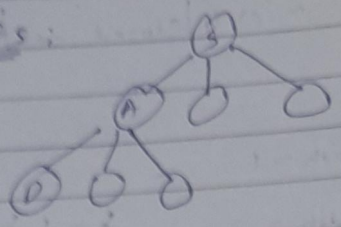
⇒ DFS is better than BFS

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

Page No.
Date

Space Complexity → examine by open set.

DFS:

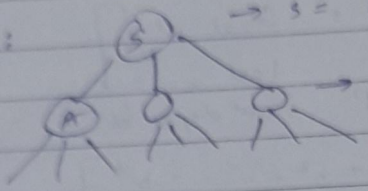


$(b-1)$

$(b-1)$ space complexity $= (d-1)(b-1)$
(at random lvl d)

$\approx O(b \cdot d)$
(linear)

BFS:



at depth $d \rightarrow b^d$
(Exponential)

∴ DFS is better than BFS in space complexity.

* Quality of sol^n : BFS > DFS
BFS gives optimal sol^n.

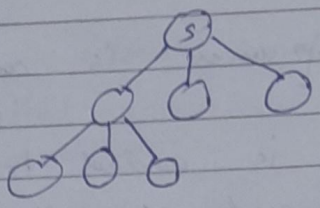
* Completeness:
⇒ both will be able us to reach goal state. ∴ both are complete.

× ×

AI

* **DB-DFS**: Depth bound DFS

⇒ Set up a bound on depth (like d = 3, 4, etc.) DFS will only go till depth.



⇒ good quality of soln
⇒ completeness ✗.

* **DFID**: Depth First iterative Deepening.

⇒ here depth bound ↑es ~~good~~ as we do not get the soln
⇒ good quality of soln
⇒ completeness ✓
⇒ Time and space complexity as DFS.

⇒ The tree gets deleted as we increse d.

d = 1
5 nodes
d = 2 (25 nodes)
d = 3 (125 nodes)

⇒ To ~~rec~~ check at d = 3 we need to check all nodes at d = 2

∴ ratio = $\frac{1+5+25=31}{125}$ $\underset{125}{}$ (not that significant work)

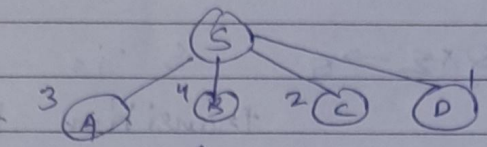
∴ we can delete and regenerate the whole tree again.

BFS, DFS, DB-DFS, DFID are blind algos as we don't know how close we are to the goal.

* Heuristic Algorithms : (we use heuristic f^n's to determine the node)

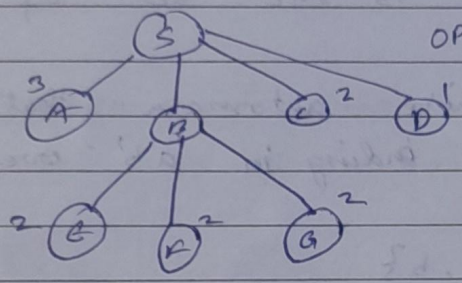
* Best First Search :

$h_1(n)$ = no. of correct nodes



OPEN = {A, B, C, D}
CLOSED = {S}

⇒ select B (B has max $h_1(n)$ in OPEN set)



OPEN = {A, C, D, E, F, G}
CLOSED = {S, B}

and so on....

⇒ Best way to maintain open set is in max heap. (node with highest $h_1(n)$ will be at root)

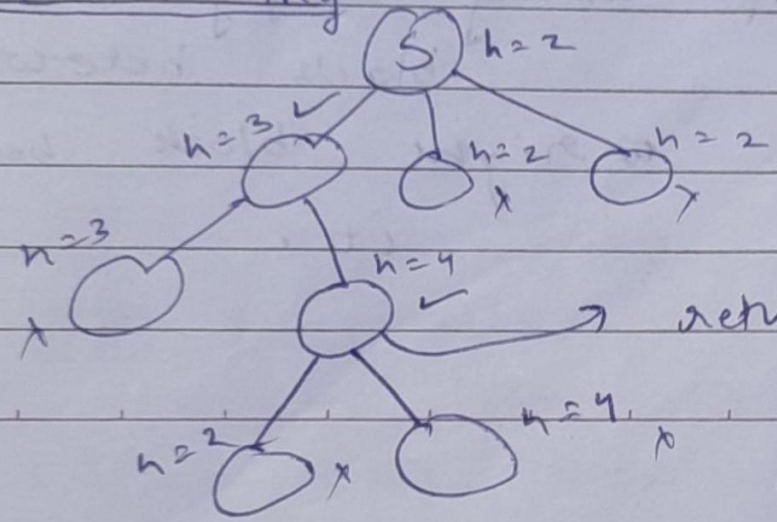
⇒ time and space complexity depends on heuristic f^n.

⇒ the heuristic algo we used is complete.

X X

AI

* Hill Climbing: → for all puzzles.



⇒ move strictly in the direction of inc/dec. $h(n)$.

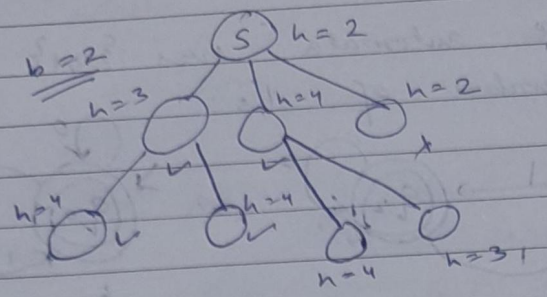
return this ⇒ Not complete

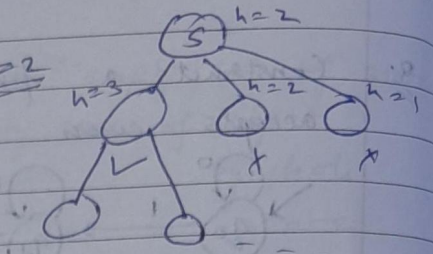
⇒ Space complexity = const.

time comp = $O(n)$ ↳ best feature.

* **BEAM SEARCH :**
  ↳ select 'b' best nodes in each lvl. (depth)
  it becomes Hill climbing.

⇒ All the nodes in 'b' should be better than the parent

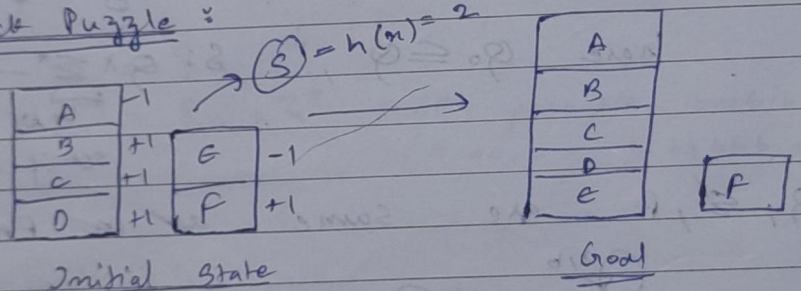




time complex = linear, space com = const.

⇒ still not complete.

* **Block Puzzle :**



Can put blocks on table or on top of another blocks

heuristic fn : -1 if wrong config (correct block below it)
+1 if right block below it.

not applicable all puzz

Page No.
Date

* SAT :

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

⇒ 3 - clauses ⇒ 2 literals inside each clause

∴ 2 - SAT

⇒ total 5 literals.

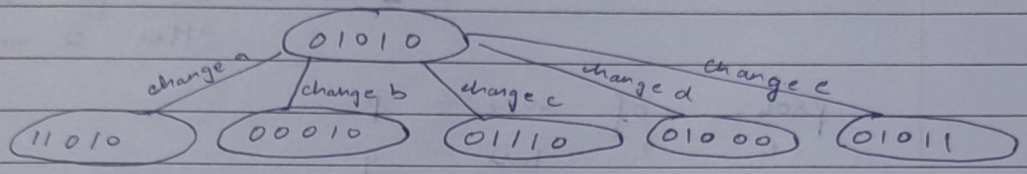
for $K \geq 3$: NP (K-SAT)

else : Polynomial time (2-SAT)

3SAT : eg. $(a \vee b \vee c') \wedge (d \vee a' \vee b') \wedge (e \vee d \vee c')$

take random :

$a = 0, \; b = 1, \; c = 0, \; d = 1, \; e = 0$



take heuristic value = no. of clauses having 1.

* VARIABLE NEIGHBOURHOOD APPROACH (VNA)

↳ HILL CLIMB

In 3 SAT :

if you change 1 bit : 5 nodes = ${}^nC_1$

" " 2 bits : 10 nodes = ${}^nC_2$

n = no. of literals.

not applicable for all puzzles. eg. 8 puzzle

⇒ Increasing the no. of nodes ↑es our prob. of getting the right answer.

⇒ VNA says apply at start apply such that neighbourhood is minimum. After a few lvls increase the density (no. of bits) (start sparse and go dense) (by changing hch f^n).

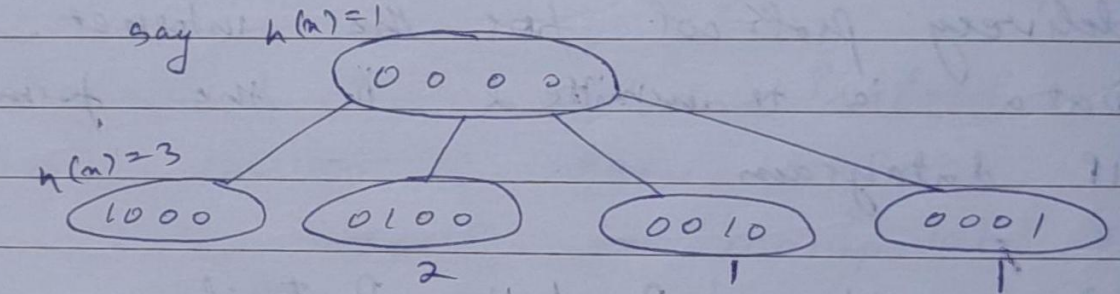
AI

* ITERATED HC : Using multiple start positions in HC
(Also not complete)

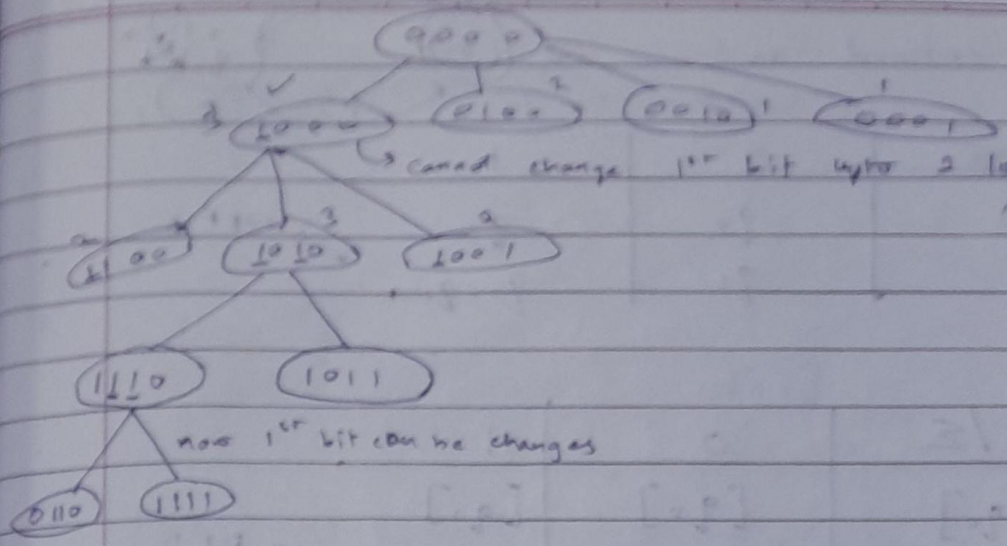
* TABU SEARCH : (concept of HC) not best first)

4 literals : a, b, c, d

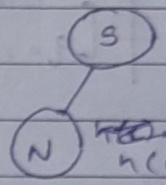
$(a \vee b) \wedge (c \vee d') \wedge (a \vee b') \wedge (a \vee d')$



tabu tenure = 2 (can change the bits of change bit again for 2 hrs



* STOCHASTIC HILL CLIMBING : (only goes one state)



prob. threshhold = 20%

$\Delta E = h(N) - h(c) = 120 - 107 = 13$

use sigmoid : $\frac{1}{1+e^{-\Delta E/T}}$ to calc. prob.

fix T at start (say T = 10)

⇒ change T iteration by iteration (start with high T or low T)
simulated Annealing

* Study Branch n Bound
TOC

* Conversion of NDFA ⟶ DFA

Q. Construct a DFA equivalent to
$M = (\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_0\})$ where

AI

* Genetic Algorithm :

a. selection   b. cross-over   c. Mutation → cus random soln.

	x		eval f^n (x^2)	Prob (4 outcomes)
eg.	13	01101	169	$4 \times 169/1170 \approx 0.8$ — ①
	24	11000	576	$4 \times 576/1170 \approx 1.67$ — ②
	8	01000	64	$4 \times 64/1170 \approx 0$ — ✗
	19	10011	361	$4 \times 361/1170 \approx 1.2$ — ①
			1170	↑ selection

13 → 01101
24 → 11000

→ 01100 (12), 11001 (25) → ②

24 → 11000
19 → 10011

→ cross over (single point)

→ 11011 (27) → ②, 10000 (16)

25 → 11001
27 → 11011

27 → 11011
25 → 11001

→ 3rd bit in both candidate is zero, so we will never reach the answer. ∴ we use mutation

mutation : change one bit in order to get to the answer.

11111
11101

* TSP :

using Gen. Algo.

$P_1 = 2\ 4\ 7\ 5\ 6\ 1\ 8\ 9\ 3$ } path representation.
$P_2 = 6\ 2\ 8\ 3\ 9\ 1\ 4\ 5\ 7$

1) Partially Mapped Crossover:

$P_1 = 2\ 4\ 7 \mid 5\ 6\ 1\ 8 \mid 9\ 3$
$P_2 = 6\ 2\ 8 \mid 3\ 9\ 1\ 4 \mid 5\ 7$

$P_2 \quad P_1 \quad P_2$

$C_1 = 5\ 6\ 1\ 8$ → mapping (if value already there then use the mapped value)
$\quad\quad\ \downarrow\ \downarrow\ \downarrow\ \downarrow$
$\quad\quad = 3\ 9\ 1\ 4$

$P_1 \quad P_2 \quad P_1$

$P_2 \qquad\qquad P_2$

$C_1 = 9\ 2\ 4\ 5\ 6\ 1\ 8$
$\quad\ \hookrightarrow 6 \to 9 \therefore 9$ from 628 → 4 ~~5~~ 8 → 4 (628)

$\therefore C_1 = \underbrace{9\ 2\ 4}_{P_2}\ \underbrace{5\ 6\ 1\ 8}_{P_1}\ \underbrace{3\ 7}_{P_2}$

$C_2 = \underbrace{2\ 8\ 7}_{P_1}\ \underbrace{3\ 9\ 1\ 4}_{P_2}\ \underbrace{6\ 5}_{P_1}$

2) Ordered Crossover: $P_1 = 2\ 4\ 7 \mid 5\ 6\ 1\ 8 \mid 9\ 3$
$P_2 = 6\ 2\ 8 \mid 3\ 9\ 1\ 4 \mid 5\ 7$

$C_1 = \underbrace{5\ 6\ 1\ 8}_{P_1}\ \underbrace{2\ 3\ 9\ 4\ 7}_{P_2}$

$C_2 = \underbrace{3\ 9\ 1\ 8}_{P_2}\ \underbrace{2\ 7\ 5\ 6\ 8}_{P_1}$

* Ordinal Representation
↳ study on your own.

INDEX = 1 2 3 4 5 6 7 8 9
$P_1 = 2\ 3\ 5\ 3 \mid 3\ 1\ 2\ 2\ 1$
$P_2 = 6\ 2\ 6\ 2 \mid 5\ 1\ 1\ 1\ 1$

Page No.
Date

$c_1 = \underbrace{2\ 3\ 5\ 3}_{P_1}\ \underbrace{5\ 1\ 1\ 1\ 1}_{P_2}$

$c_2 = \underbrace{6\ 2\ 6\ 2}_{P_2}\ \underbrace{3\ 1\ 2\ 2\ 1}_{P_1}$

} using single pt. ~~crossover~~

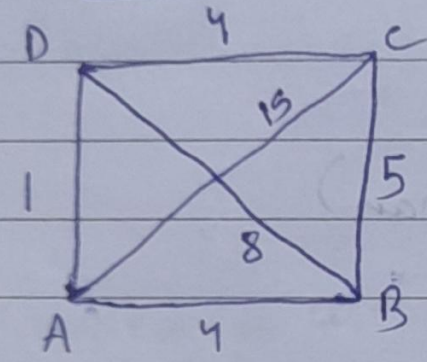
$c_1 = 2\ 4\ 7\ 5\ 9\ 1\ 3\ 6\ 8$ } path rep.

TOC

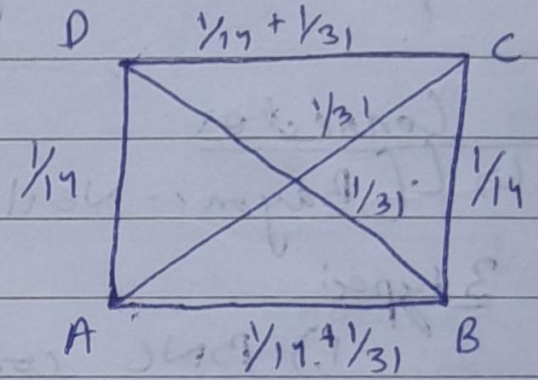
AI

* **Ant Colony Optimization :**

cost matrix



Pheromone Matrix



Path 1 → A B C D A = 14

Path 2 → A C D B A = 31

⇒ To update the lvl of pheromone in pheromone matrix, we use the path sum reciprocal

Page No.
Date

Now to find prob. of another ant going from A based on cost or pheromone lvl

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

$$A \to B = \frac{0.3 \times \frac{1}{15}}{\Sigma} \qquad A \to C = \frac{0.1 \times \frac{1}{4}}{\Sigma}$$

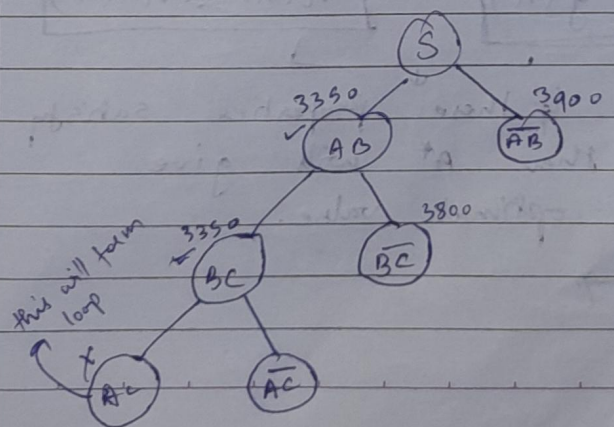
Vapourization Effect: let $\rho = 0.6$

$\therefore$ $(1-\rho) \times$ initial lvl of pheromone $= (1-0.6) \times 1 = 0.4$

* Branch n Bound vs (TSP) → BnB, Ant col, Genetic.

	A	B	C	D	E		AB (exclude AB edge)
A	0	300	600	900	1000	→ 900	600+900
B	300	0	500	800	1200	→ 800	500+800
C	600	500	0	1100	1400	→ 1100	1100+1400
D	900	800	1100	0	1500	→ 1700	1100+1500
E	1000	1200	1400	1500	0	→ 2200	1400+1500
						6700	7800 → 7800/2 = 3900

$\therefore$ Estimate cost / bound $= \frac{6700}{2} = 3350$

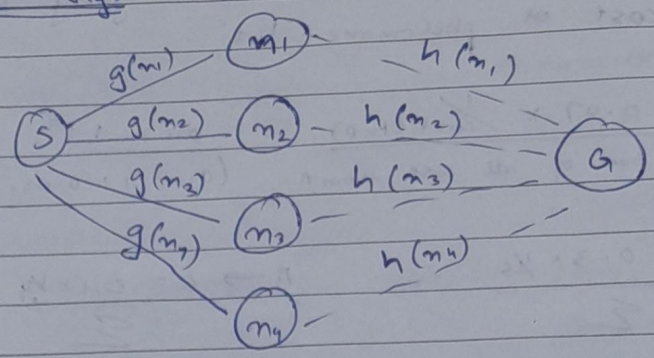


$\Rightarrow$ AB, BC are chosen as they are min. in matrix.

Page No.
Date

* **A* Algo :** (* → Always gives optimal result)

$h(n)$ → heuristic
$g(n)$ → cost till node n



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

8- puzzle :

1	7	4
2		5
3	8	6

→

1	2	3
8		4
7	6	5

$h(n)$ = no. of wrong
$g(n)$ = no. of levels travelled.

Use Branch n Bound.

$f(n) = g(n) + h(n)$ ⇒ Value we evaluate

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

$$g^*(n) \leq g(n) \qquad h(n) \leq h^*(n)$$

↓

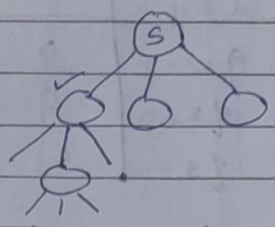
if these properties satisfy then A* will give optimal value.

✕ → ✕

Page No.
Date

AI

* **IDA*** : (Iterative Deepening A*)
  ↳ learn eg. from YT.

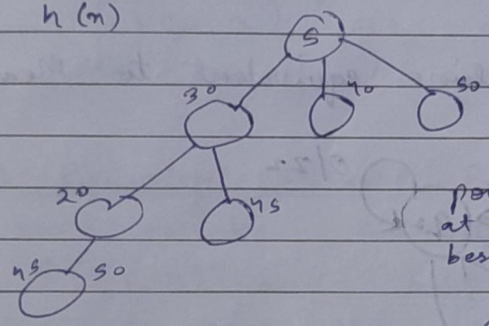


$f(n) = g(n) + h(n)$

⇒ Instead of lvls, ~~take~~ generate till $f(n)$ increases more than a certain bound, or you get the answer.
(Almost same as DFID)

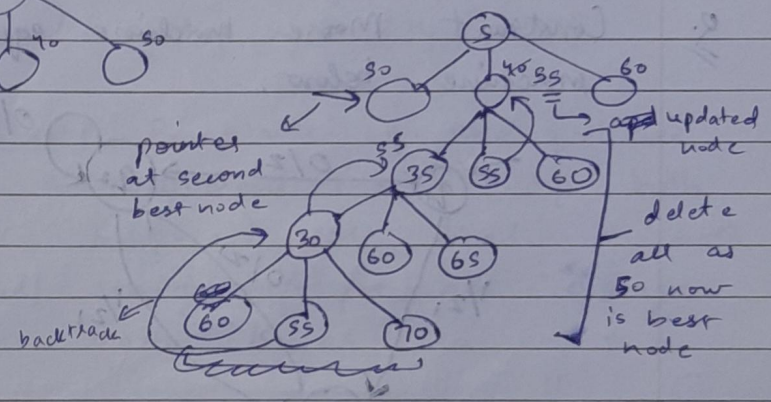
* **Best First Search** :

$h(n)$

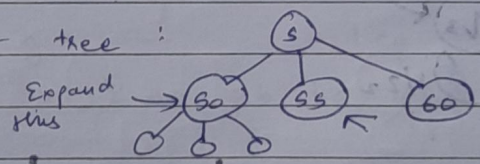


* **Recursive Best First Search**

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



∴ new tree :



⇒ Recursive BFS saves space.

⇒ Thrashing

⇒ Values can be updated in Rec. BFS but not in BFS.

* — *

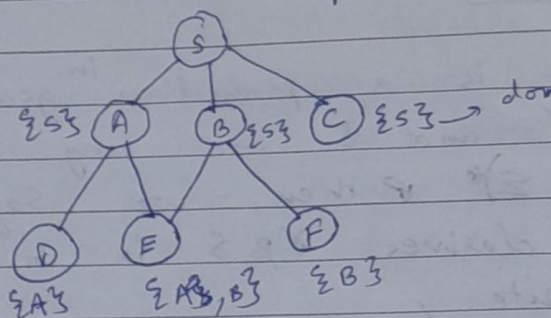
AI

* PRUNING THE CLOSED LIST:

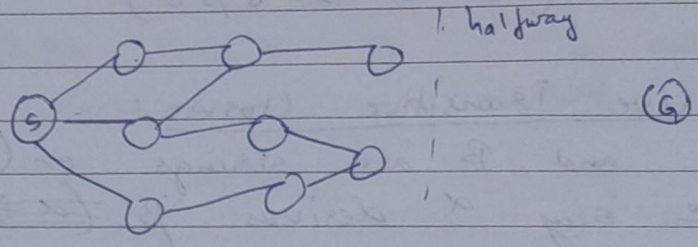
⇒ Stop leaking back (not exploring the already explored nodes)

⇒ Path construction: path from starting to goal node.

1) DCFS : (Divide n Conquer Frontier Search)



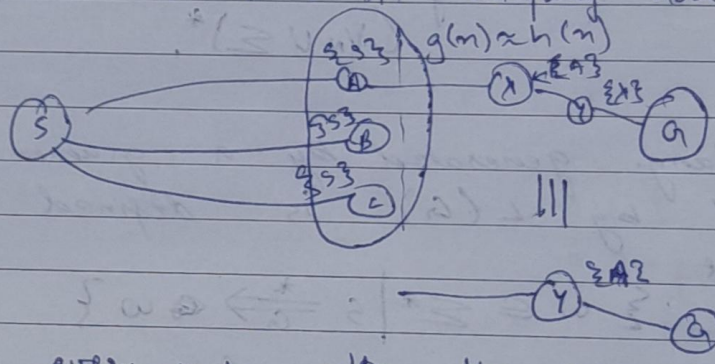
→ don't generate these



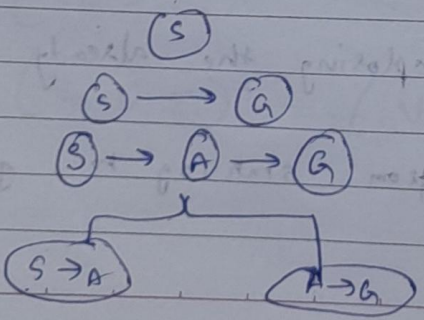
At halfway $f(n) = g(n) + h(n)$

$g(n) \approx h(n)$

Delete all nodes till halfway node



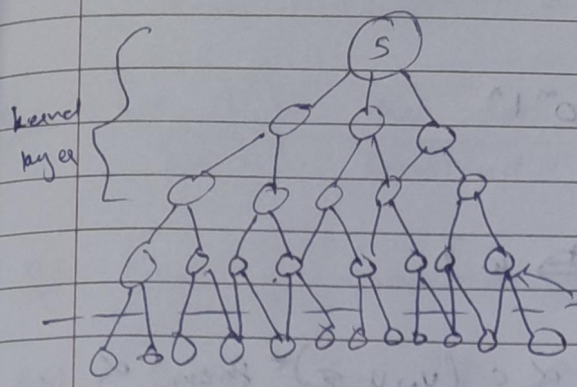
⇒ At every node, after the relay layer we will delete the previous explored node and the list will contain the parent only from relay layer.



and so on...

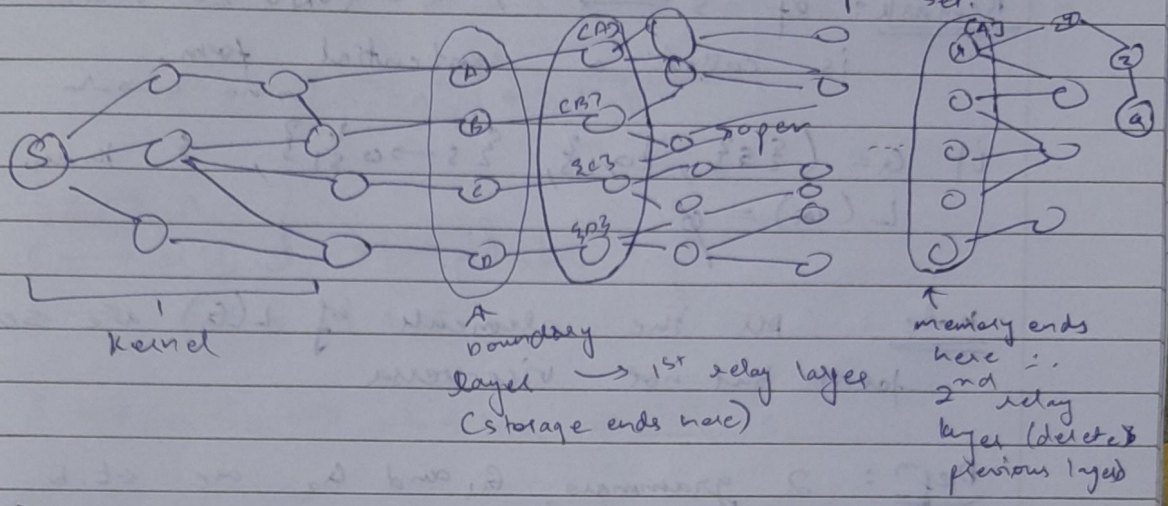
Page No.
Date

2) Sparse Memory Graph Search :



CLOSED SET = KERNEL LAYER + BOUNDARY LAYER

↳ whose neighbours are in open set.



⇒ Delete kernel layer − {S}

⇒ never delete boundary (eg keep 1st bound layer even during nth kernel layer being deleted)

Path reconstruction : (S→A) + (A→X) + X→Y→Z→G

TOC

Q. G = ({S}, {0,1}, {S → 0S1, S → ∧}, S). Find L(G)

↳ Dutomata S → 0S1/∧

G = (V_N, Σ, P, S) → ∈ V_N starting symbol

↳ Set of variables (capital Alphabets) ↳ terminals

α → β

∈ (V_N ∪ Σ)*

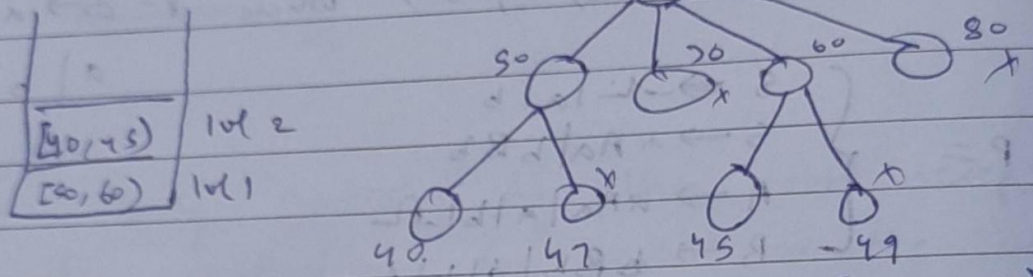
AI

* Pruning the OPEN list:

→ Hill climbing
→ Beam search
→ BFHS - (Breadth First heuristic search)
→ Beam Stack Search

In this we set an upper bound depending on a heuristic f^n and if the heuristic val. is higher than bound the we discard that node

* Beam Stack Search:



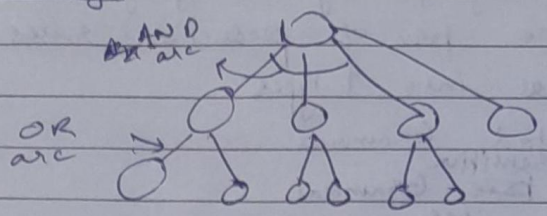
⇒ if we don't get ans from [40, 45) then we change stack lvl 2 from [40, 45) → [45, 47) and so on.

⇒ If we don't get a result in lvl 2 the backtrack to lvl 2-1

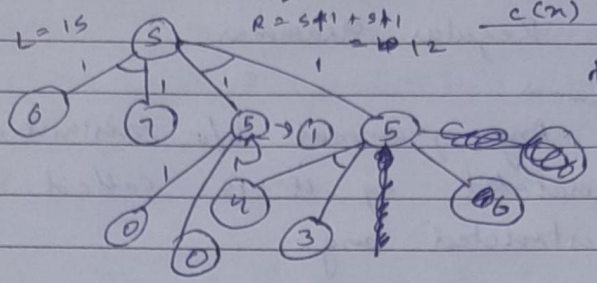
⇒ Complete Algo.

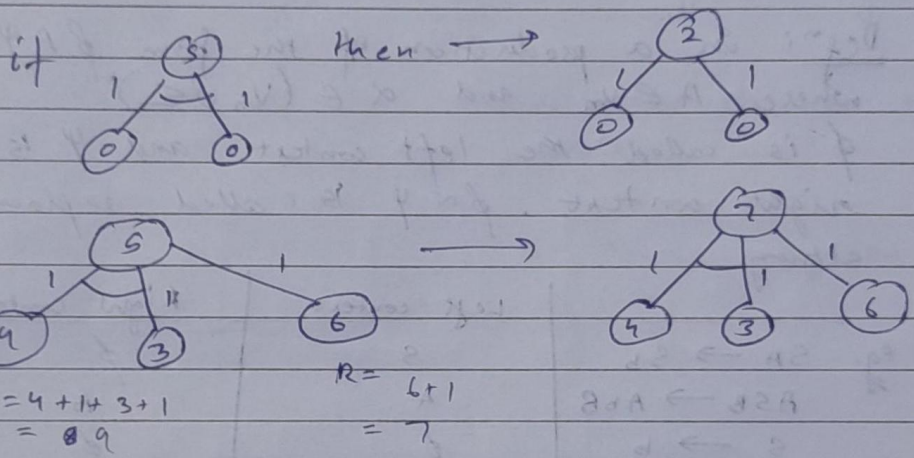
⇒ Study DCBSS and DCBFMS

* AO* Algo :

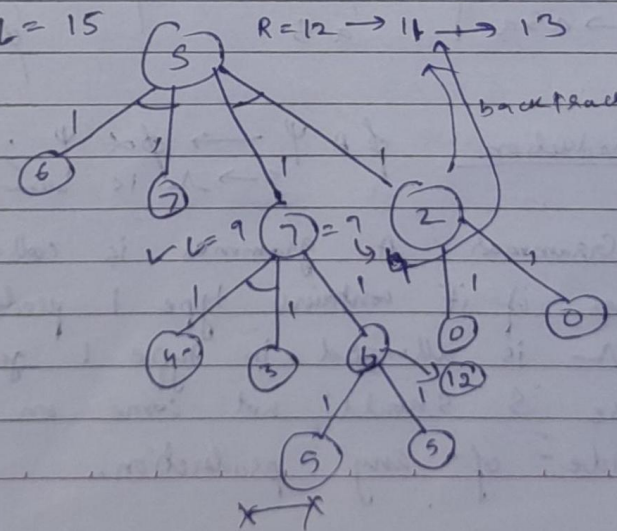


→ ~~not complete~~, complete
⇒ But not optimal



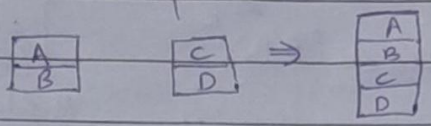


∴



AI

Planning : Set of states



Initial State	(Represent States)	Action (Y)
on table (B)	Domain Predicate (S)	Pickup(x) ⇒ x on table No one is above.
on table (D)	on (X,Y) → X is on Y	
on (A, B)	on table ⇒ on Table(T(x))	Put Down(X) ⇒ Putting block on table
on (C, D)	Clean (X) → no one is above X.	Unstack (X,Y) ⇒ Pick X from Y.
Arm Empty (AE)	Holding(X) → Robert holds X.	
Clean (A)	Arm Empty → Robert not holding anything	Stack (X,Y) ⇒ Putting X on Y
Clean (C)		

Pre Cond^n:

on T(x) and clean (x) and AE	← For pickup
Holding (x)	← For putdown (x)
on (x,y) and clean (x) and AE	← Unstack (x,y)
Clean(Y) and Holding(Y)	← Stack (x,y)

After Action Affects :

Holding (x)

AE, on table, clean (x)

Holding (x), clean (x)

AE clean X, on (X,Y)

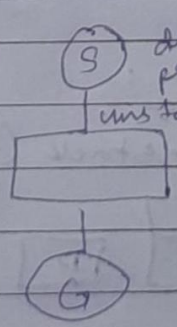
Action unstack (A,B)

Holding (A)

Clean (B)

on (C,D)

Clean (L)



FSSP (Forward State Space Planning)

⇒ All algos we have learned that come under this

(S) —a→ (S')

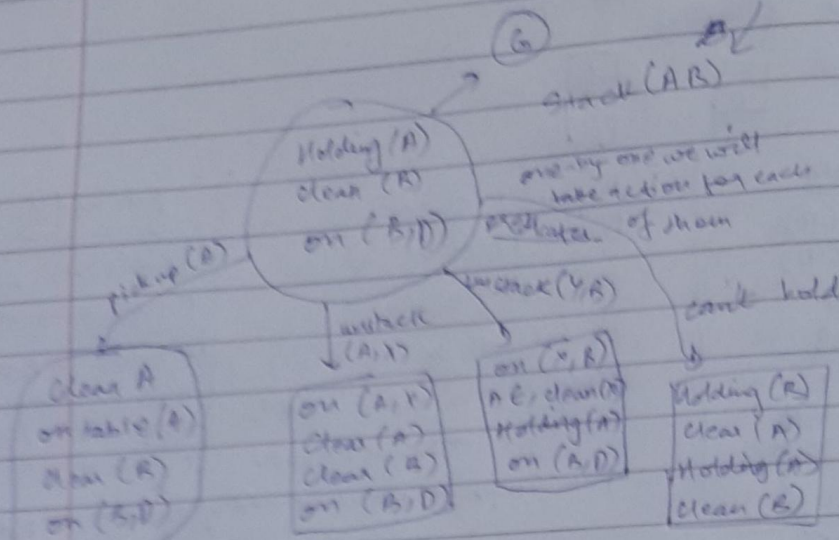
$S' = (S - effect^{-}(a)) \cup (effect^{+}(a))$

* BCSP (Backward State Space Planning)

$G \xrightarrow{a} G'$

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

[on (A,B) ∧ on (B,D)]



* Goal Stack Planning: