DATE : 8 / 10 / 22
PAGE NO. :

Monday

Intelligence :

→ AI : not copying exact human-level intelligence
: not a replica

→ AI : creates env. to ease human functions

→ Birth of AI : 1956 (workshop at Dartmouth Clg).

→ Aim for general principles : Every aspect of learning ... (slide

→ Early success : Checker Board (1952)
: Problem Solving (1955) (Arithmetic & Trigon

→ Overwhelming Expectation Vs Underwhelming Results
: m/c translation
Eg: Translate ' The spirit is willing but the flesh is weak '

ERROR !!

↓ RUSSIAN (via AI)

' The vodka is good but the meat is rotten '

Reasons for failure :
→ No generalised rule for all languages.
→ Need to train the machine with every word & rule of languages involved.

Problems :
→ Limited computation
→ Limited information
• In 1956 - computation was expensive (expensive hard discs etc.)

Contributions
: LISP (list processing) (logic-based prog. like 'if', 'else')
: Garbage collection

DATE : / /
PAGE NO. :

- Knowledge Based Systems (KBS) (70-80's) - in Unit 2.
- Artificial Neural N/W. (ANN) (1943)
  - Thought: study human neurons to form artificial neural n/w.
- Front / Followed Propagation :

  State A $\xrightarrow[\text{propagate forward}]{\text{steps}}$ Objective

- Back Propagation : We know the src & dstn. & steps
  - Job is to go back & check if steps lead to source specified.

  Eg: X is father of Y, Z is father of X.
  Is Z grandfather of Y? ⟶ Back Prop.
  Is Y grandchild of Z? ⟶ forward Prop.

- convolution (LeCun) : to recognize handwritten digits

Deep Learning

- AlexNet (2012) Object Recognition, computer vision common
- Alphago (2016) Deep reinforcement learning, defeat world champion (Lee Sedol) in

Two views of AI

Agents : How to create intelligence
Tools : How to benefit society?

AR: Augmented Reality
VR: Virtual Reality

DATE: / /
PAGE NO.:

AR Vs VR

AR: In real world, using sensors, objects working IRL.

VR: no connection with real world,

Date: 10/10/22
Wed

AGENTS:

Intelligent Agent (PEAS str.)

- It will enact response to senses automatically.

P → Perceptors / performance measure

E → Environment

A → ~~Actions~~ Actuators (them which are take action)

S → sensor

Ideal rational agents

- Single Particular objective (Eg: cost minimization)
- ~~Most of the time, we need a trade-off~~ time minimization
- Mostly, we need a trade-off among all variables.
- optimization → single objective
  → multiple objective
  (in collaboration, need to appreciate all needs & constraints)

single obj → single constraint-satisfaction problem.

→ If more ppl are concerned on 1 factor, we bias towards that single-constraint-satisfaction prob.

HW: Medical Diagnosis System (Write PEAS)

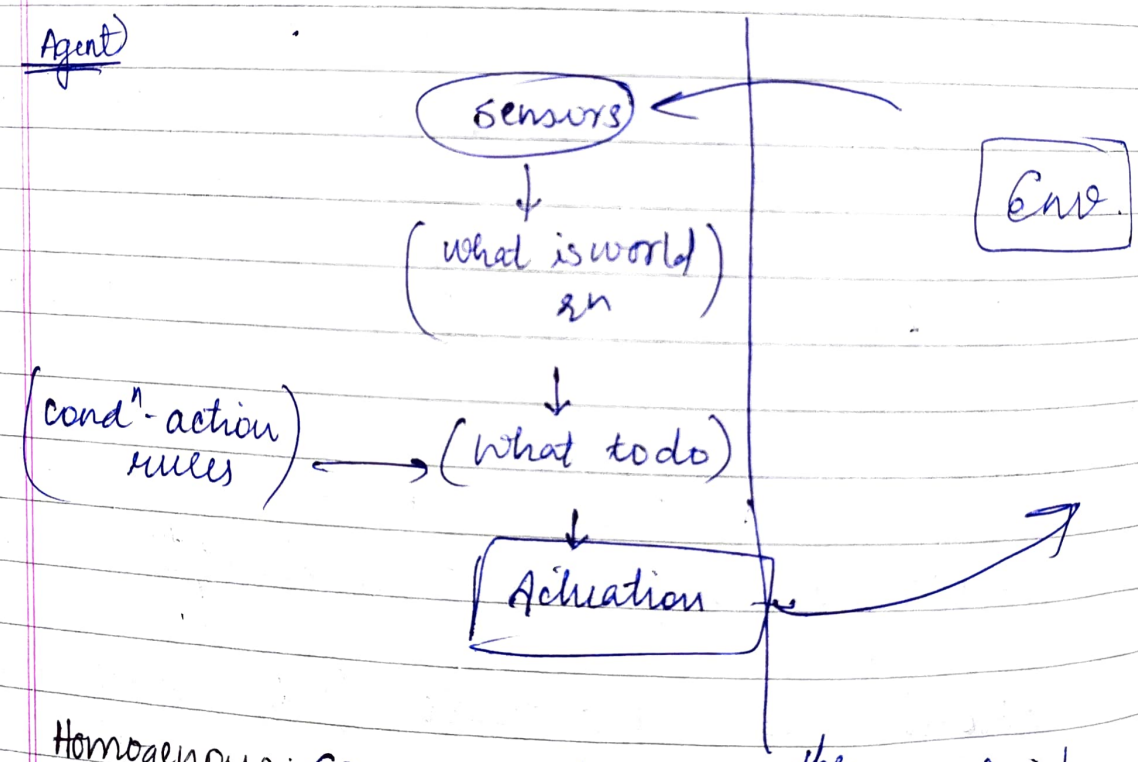
DATE: / /
PAGE NO.:

- Reflex: Catch the state of env. quickly & take action.

→ Only dependent on current state of env, ~~e~~ perceptible to you.

→ On that state, action is taken.

A simple Reflex Agent

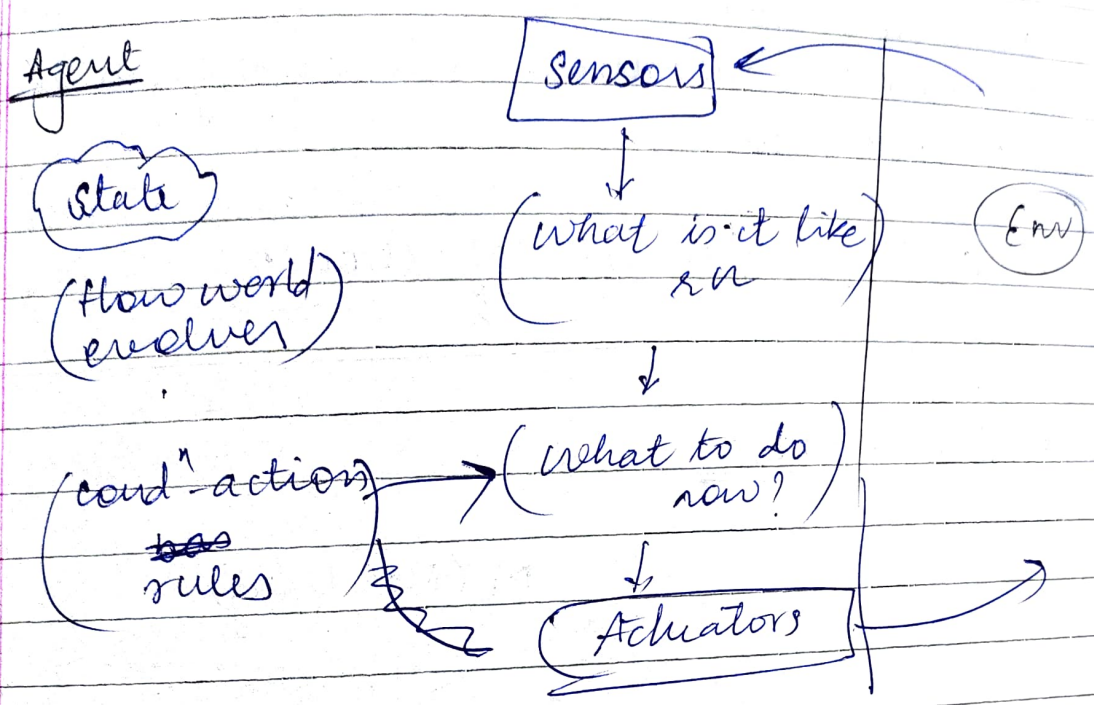


Homogenous: Same rules for all of the same kind.
Heterogenous: Not same attributes, rules.

** →
→ Based on current state only
→ Simple but limited
→ work only if env. is fully observable
Eg: Vacuum Agent.
(Vacuum cleaner)
→ works if dust in env.

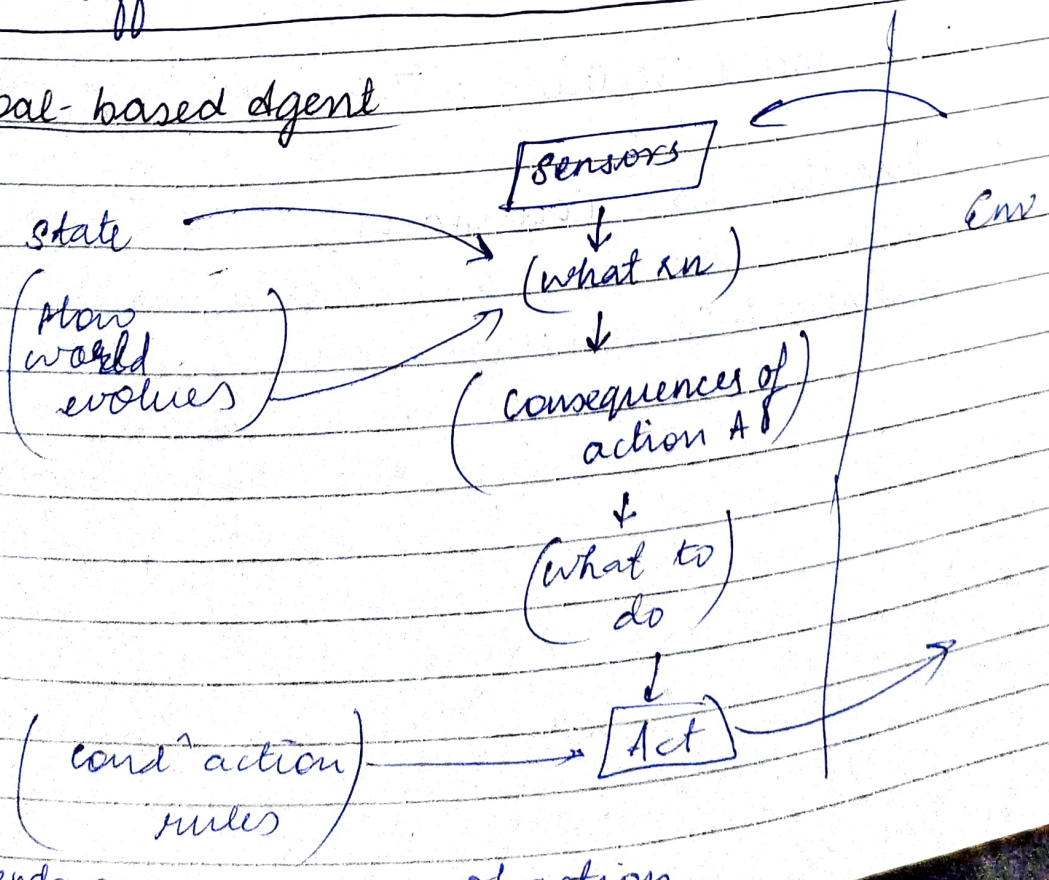
DATE :
PAGE NO. :

- **Model-Based Reflex Agent**



→ ~~It~~ considers history of environment.
→ Eg: Rain detection (Weather forecast)
→ Somehow consider partially observable env.
→ Eg: Traffic Detection.

- **Goal-based agent**



→ Depends on consequences of action

DATE: / /
PAGE NO.:

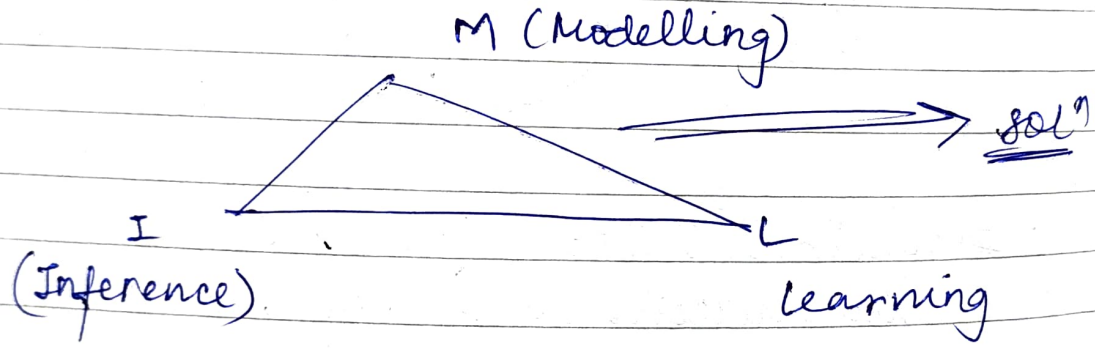
Utility Based Agent

→ Similar to goal-based
→ A performance-measure which also gives a token of appreciation.

Utility → Benefit of action A.

PEAS

- Environment



Modelling : What is our env & constraints / req.

Environment Defining : 7-key Dimensions (slides)

- Discrete Vs Continous

Discrete: Playing chess: 1st step, wait, 2nd step
Continous:- Car driving - acceleration
- All events are continous, even if dependent on env. factors.

DATE : / /
PAGE NO. :

- Complete Vs Incomplete
  (C): chess: ~~Initially~~ We know all rules all the time, response for all state.

  (In): Poker: Don't know what could happen after 2 chances.

- Fully observable: face recognition
  Vs. Partially obs: self-driven car.

- competitive Vs collaborative
  Eg: board games - win or lose
  Eg: Alexa connected to music sys, listening music then ask alexa to play a song.
  Eg: cricket is - collab for a team - compet for 2 team

Static Vs Dynamic
(S): Not changing on its own (voice/speech recog.) (changes, if action is taken (only) (speech is recorded, no change)
(D): Self-driven car
- Even if the car is stopped, there are changes in env.

DATE : / /
PAGE NO. :

- Deterministic
  → We know the result of our action
  Eg: speech recog.
  (like finite automata machine)

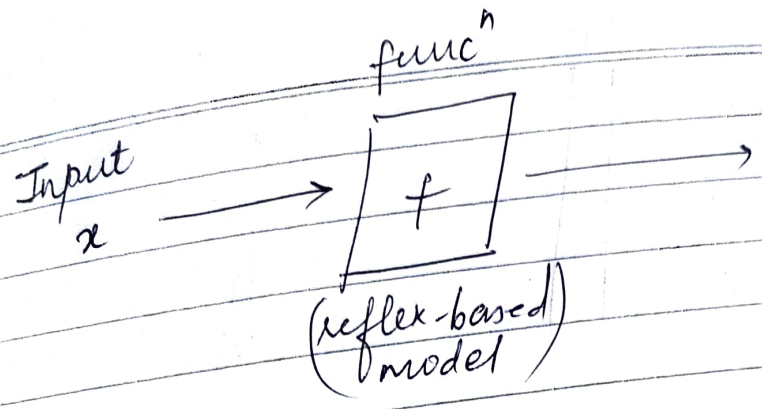
Vs

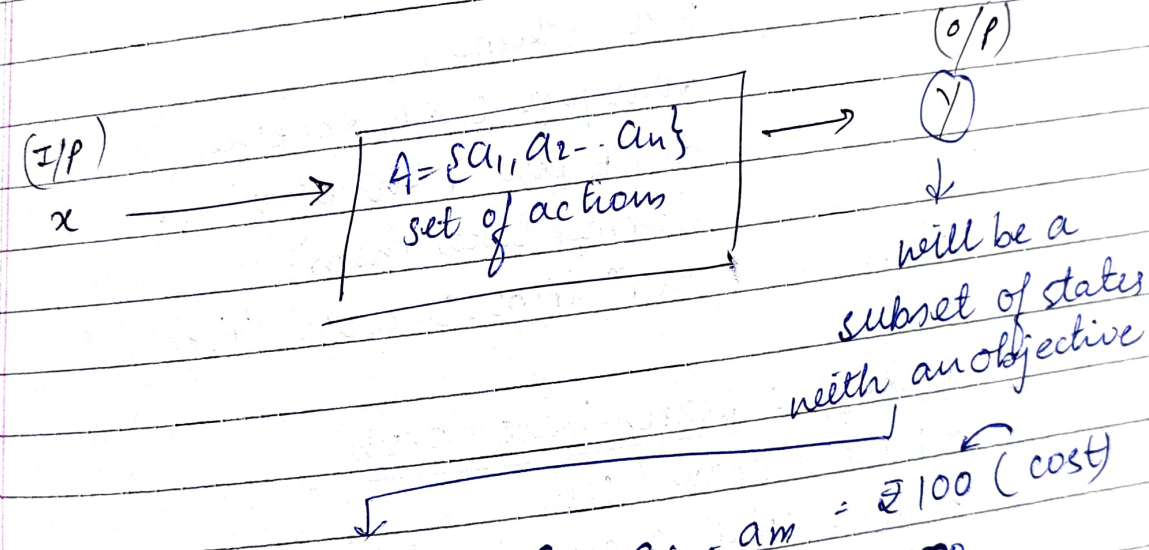
Stochastic
→ self-driven car
- Applying brakes may or may not cause accident
→ uncertain result for a specific action

- Single agent
→ Only 1 in the class attending
→ Not competitive or collab
→ Maybe utility-based

Multi agent
→ Either competitive or collaborati

DATE : 17/8/22
PAGE NO. :





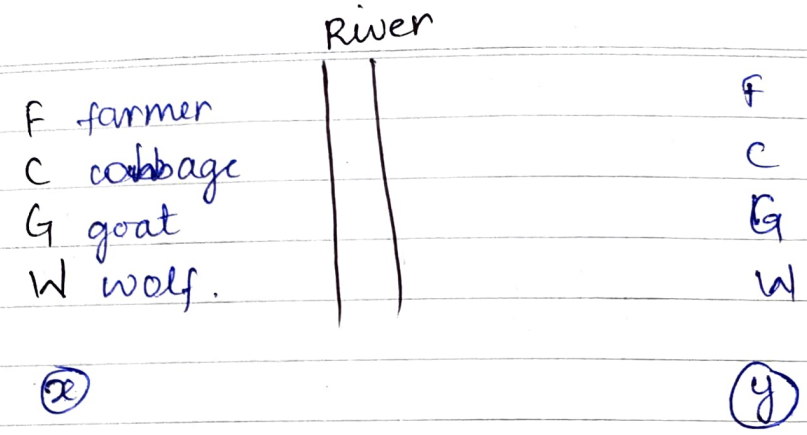
Search

↓ search problem

- Modelling real-world probs into a ~~set~~ subset of sol^n's.
- all possible set of actions
- Then comes constraints, and we go for search algos to achieve the objective

↓ search algorithm

Eg: A real world prob.



constraint : 2 at a time,
don't put C & G together,
don't put G & W together,
farmer takes them.

objective : Min^m trips to cross river

① Make it a search-based prob

- Make a graph/ tree
- Search space : set of Actions

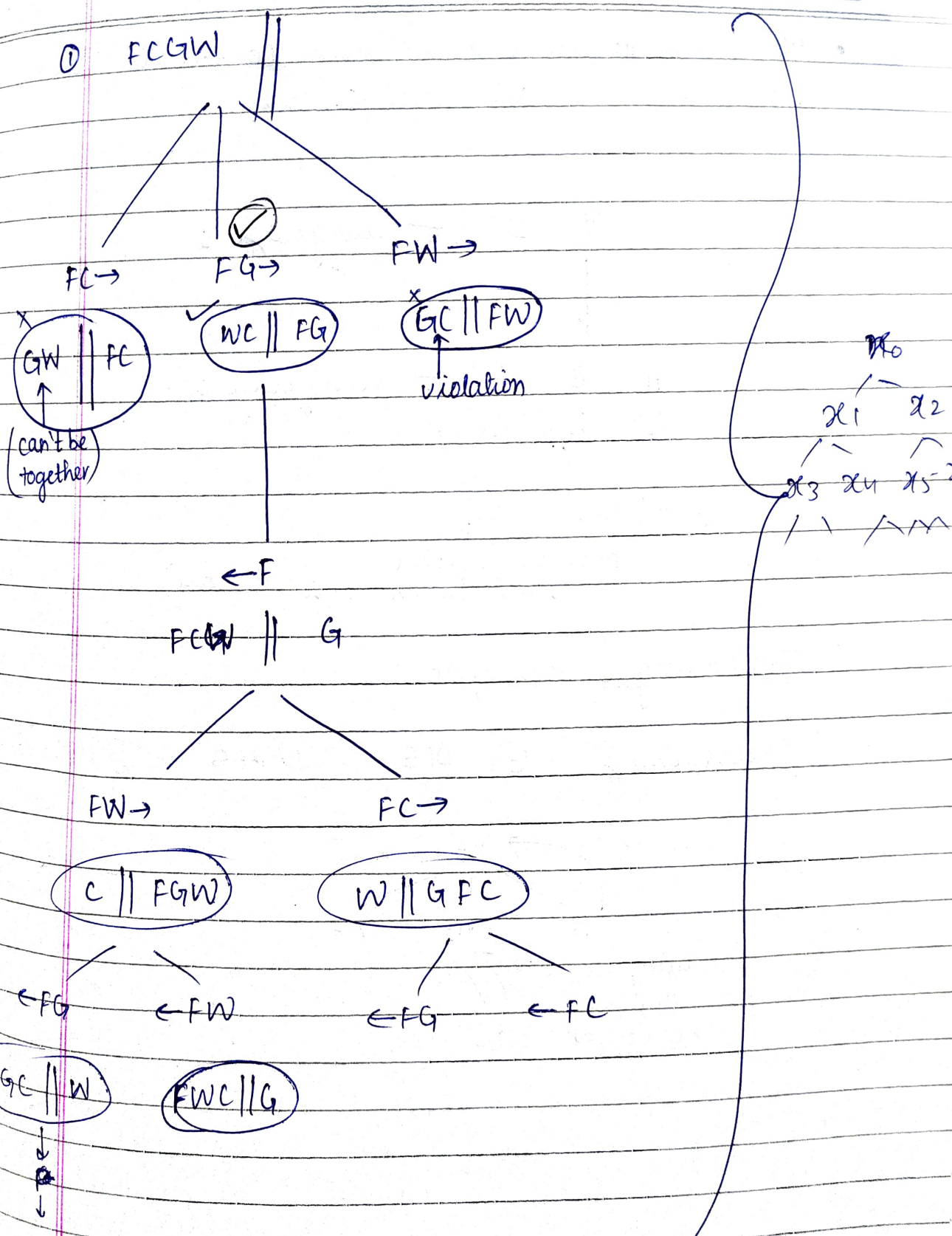
a_1 F →
a_2 FC →
a_3 FG →
a_4 FW →
a_5 ← F
a_6 ← FC
a_7 ← FG
a_8 ← FW

$A = \{a_1 -- a_8\}$

Search Tree

→ set of actions & possible states

DATE : / /
PAGE NO. :


① FCGW ||
FC→
FG→
FW→
GW || FC
can't be together
WC || FG
GC || FW
violation
←F
FCGW || G
FW→
FC→
C || FGW
W || GFC
←FG
←FW
←FG
←FC
FGC || W
FWC || G
x0
x1
x2
x3
x4
x5

DATE: / /
PAGE NO.:

- We need to train our machine to create these states & actions and then compute.

Eg:

X Y Z ← language 1

↓

A B C ← language 2

(I) (have) (class)

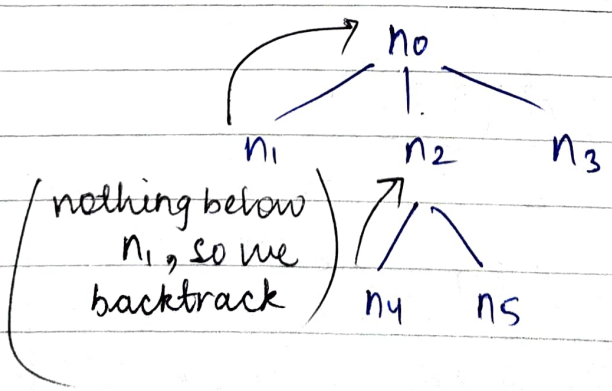
↓

A { a_1 - - - a_n }

pick the correct word (has/have/had/is/are)

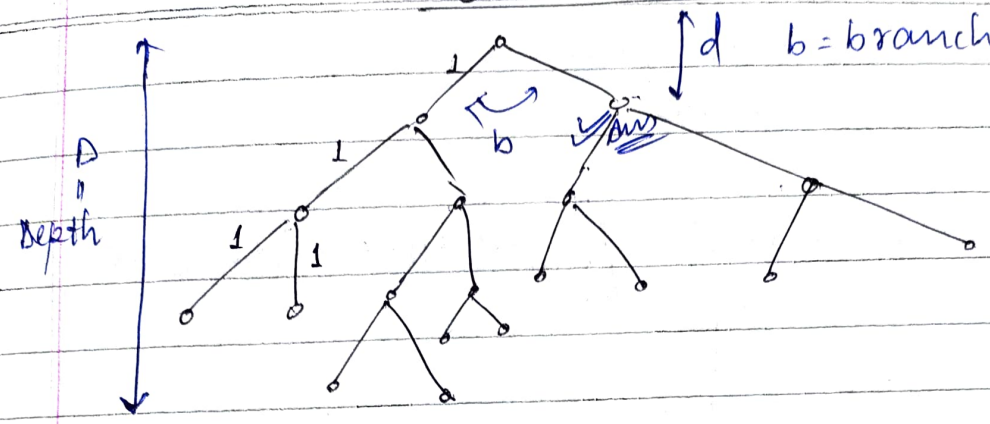
Traversing search space

① Backtracking ② DFS ③ BFS ④ DFS-ID



DATE : / /
PAGE NO. :

Search Tree



Search Algo	cost	Time	Space
① Backtrack	Homo/ Hetero (Any)	$O(b^D)$	$O(D)$

Prob: traverse the whole tree in any case

② DFS : Backtracking but if we reach the soln, we don't traverse the path further.
∴ Only in worst case it traverses whole tree

DFS	$c \geq 0$ (concerned by set of actions, not cost)	$O(b^D)$ (improved in avg/simple case)	$O(D)$

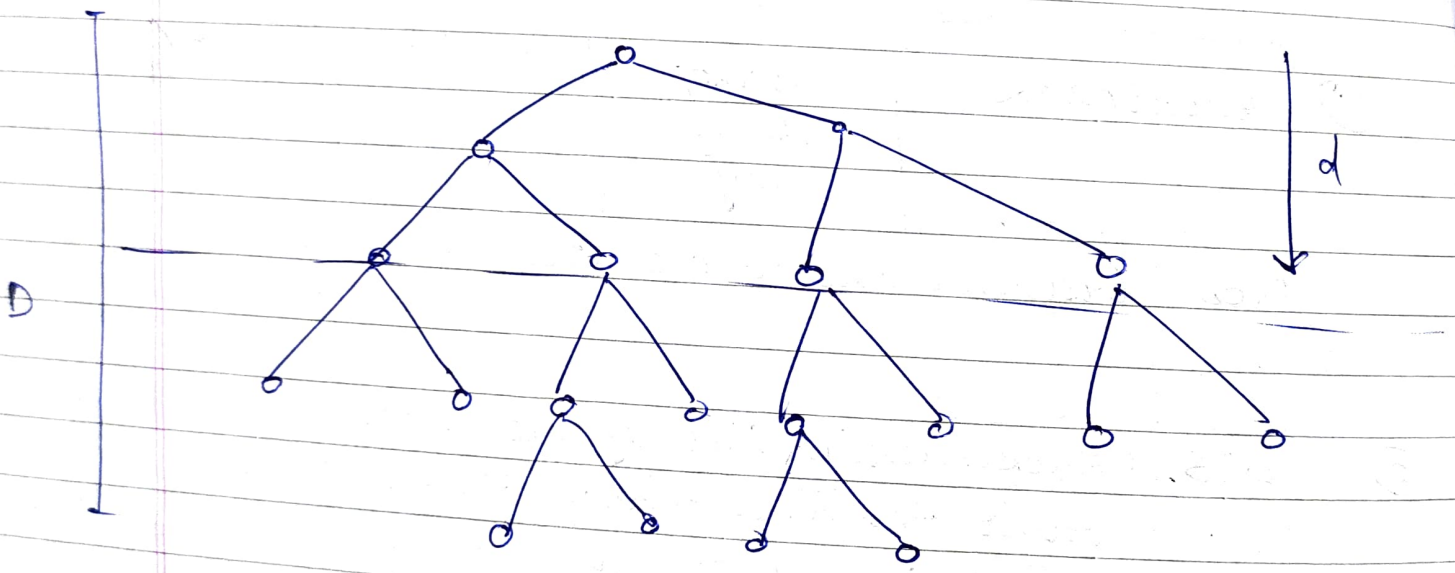
③ BFS : → level-by-level traversal.
→ stop if answer-node reached

BFS	$c \geq 0$	$O(b^d)$	$O(b^d)$ (keep track of every node)

DFS → less ~~time~~ space
BFS → less time

∴ We need a trade-off b/w time & space complexity

DFS - ID
↓
Iterative Deepening



→ First do DFS for d and hoping to get answer in this subtree
→ If not found, we increment d by 1 and do BFS in the new layer and keep ↑ing d and do BFS.

DFS-ID		O(d)	O(d)

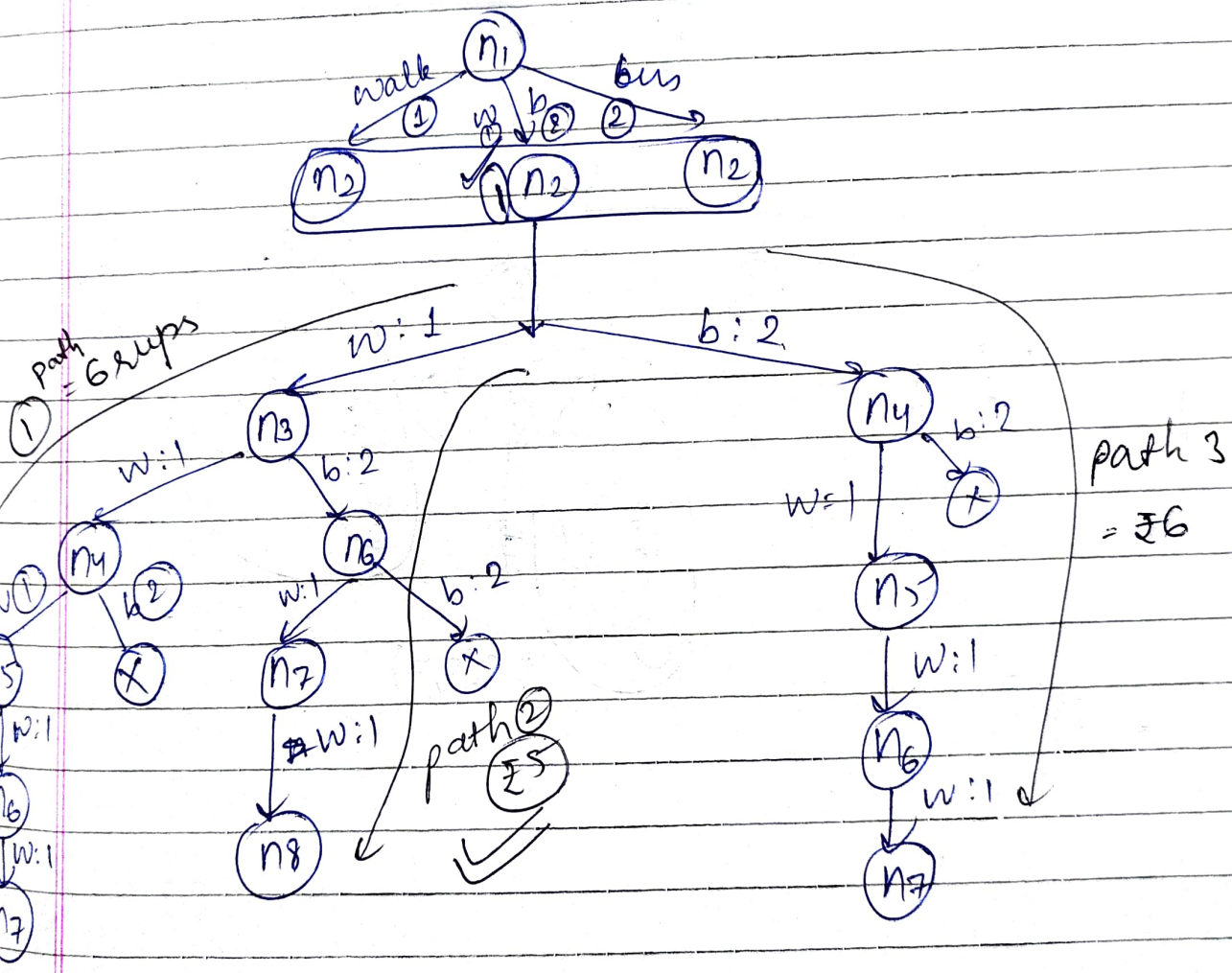
DATE : 22/8/22
PAGE NO. :

now

walk : cost = 1₹ $(n \rightarrow n+1)$

bus : skips stoppages, cost = 2₹ $(n \rightarrow 2n)$

You can only move forward.



Qu: choose path with minimum cost.

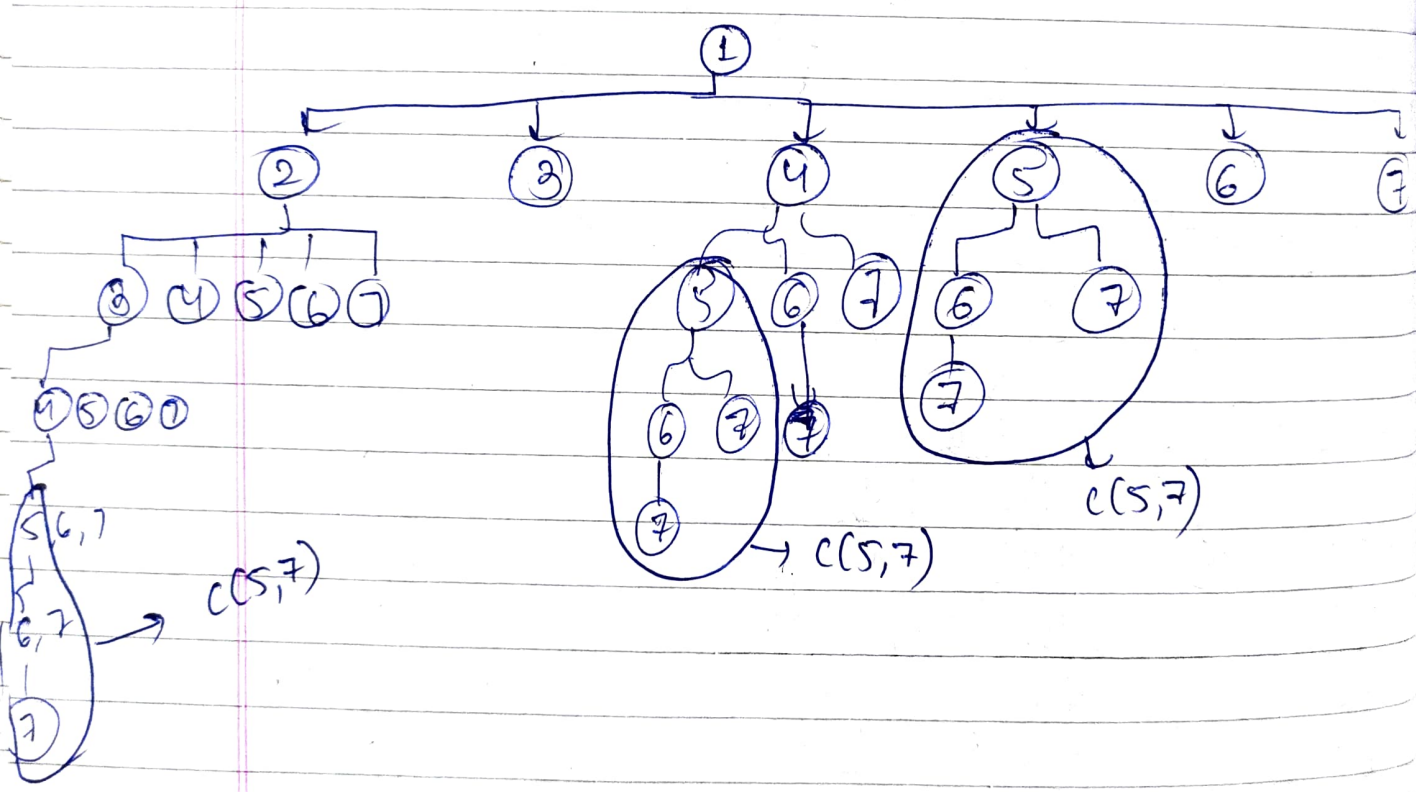
path 2 is optimal

For exponential tree → No improvement in complexity coz we need to see all paths.

Soln: We can save cost from n_4 to n_7 = ₹3 as it is used repetedly.

We can save costs which are constant & reuse them
- This is Dynamic Programming

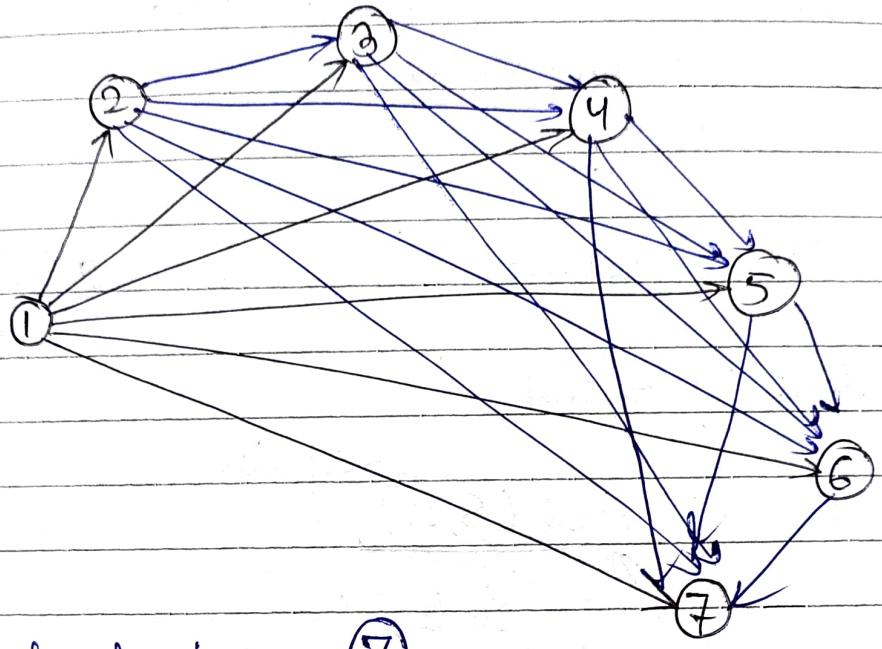
Eg: cost is $c(i,j) = $ ~~c constant~~
$i \to j$



- Exponential search tree (2^7 nodes)
  → Reusing c(5,7), and many more subtrees

DATE : / /
PAGE NO. :

Graph looks like:



No. of nodes here = (7)

Time complexity = $O(n^2)$ (in worst case) $\frac{n(n+1)}{2}$

∴ complexity has gone down from exponential to polynomial

→ DP is not always n^2, but definitely less than expo

Eg: city 1 ⟶ city 7
constraint → can't go to 3 consecutive odd no. city.

DATE : / /
PAGE NO. :

Concept of states

In search tree : states = all possibilities
graph : states

#

Start state : S_{start}

End State : S_{end}

cost : $\underbrace{cost(s, a)}_{\text{cost}}$, $\overbrace{}^{\text{state}}$ ← action

$a \in A$ ← all possible set of actions

$Succ(s, a) = s'$

↑ successor

{all states are subset of $\in S$.}

Binary Func^n : IsEnd → 0 / 1

= True (1) or False (0)

↑ ending cond^n.

DATE : / /
PAGE NO. :

Back to Example (odd consec.)

(Prev city, current city) ← keep track.
↓ 3
1 or 2

if → (1,3) → Now (1,3), (3,7) → Not possible

(2,3) → (2,3), (3,7) → possible

∴ current city = n options
prev city = n-1 options

∴ complexity = $O(n^2)$ (for search space)
$= n(n-1)$

- But we don't really need (n-1) options record.
- We only need to know if prev city is odd or even.

∴ prev city can have a flag for odd & even

Now complexity = $O(n)$ (search space = $2n$)

DATE: / /
PAGE NO.:

Que: Reach 1 → 7,
constraint: visit atleast 3 odd cities before 7

Soln: States:

[min (odd cities visited , 3) , current city]

{ O(n²) complexity if keep track of all prev. cities }

But now O(n) (coz min() returns a constant)

Que: Cost Minimization

$$\text{Future cost} = \begin{cases} \min(\text{cost}(s,a), \text{Future cost}(s')) \\ 0 \quad \text{IsEnd.} \end{cases}$$

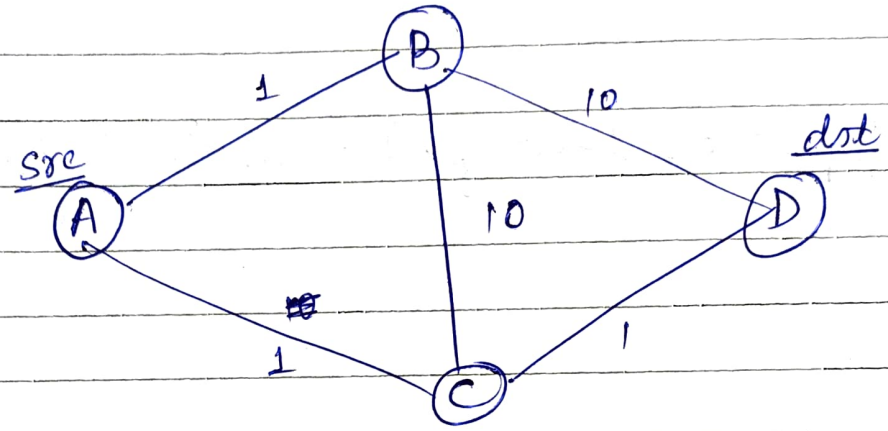
- If graph is cyclic, DP can't handle it
- For DP, graph should be acyclic, unidirectional
- For DP implementation, machine should have space for memorization. (In case space grows!)

DATE : / /
PAGE NO. :

Q: What if the graph is cyclic or bidirectional (can go back)

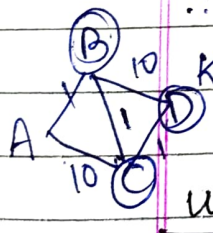
∴ We use UCS. **Uniform cost search**

DP



From A, AB = AC = 1. ← pick any, Pick B
From B, BD = 10, BC = 10 ← pick any, pick C
From C, CD = 1 = take
cost Σ = 1 + 10 + 1 = (12)
But optimal path = ACD (cost = 2)

∴ We need UCS



Keeps track of Explored, unexplored and frontiers (priority list)

unexplored	Frontiers (priority)	Explored
A	(A,0), (B,1), (C,10)	A,0 ①
B	(D,10)	B,1 ①
C	(D,1) ②	C,1 ②
D		D,1 ③

C is already in explored set, ∴ see B-D, if new value, update (C,1)

S_{start} = A

→ Update in frontier set,
→ Dont change what is already in explored set.

DATE: / /
PAGE NO.:

Complexity

$\frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + \cdots - - - 1 \Rightarrow \log(n)$ for frontier

from A, (1 out of n), then (1 out of n-1) <- - -

(not check explored nodes)

for explored → space = O(n)

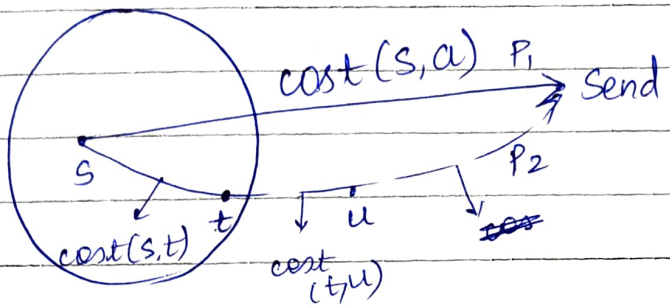
∴ Complexity = O(n log n)

(n: explore, log n: frontier)

DATE : 24/8/22
PAGE NO. :
Wednes

UCS

Unexplored → Frontier → Explored



Minimization Prob

P_1 → optimal (Proof by contradiction)

Let P_2 be optimal (min. cost)

$= cost(S-t-u-Send) \geq cost(s,t) + cost(t,u)$

$\geq cost(s,t)$

~~$cost(s,t) + cost(t,u)$~~ $\geq cost(s,a)$

(coz Send has more priority than t)

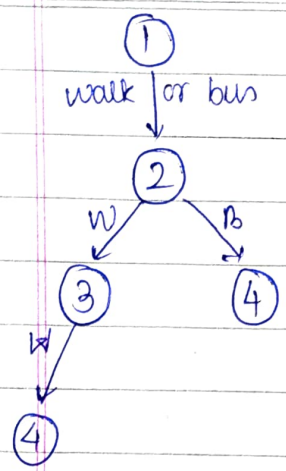
* -ve cost is not allowed in UCS

(6)

M (model)

I (inference) —— L (learning)

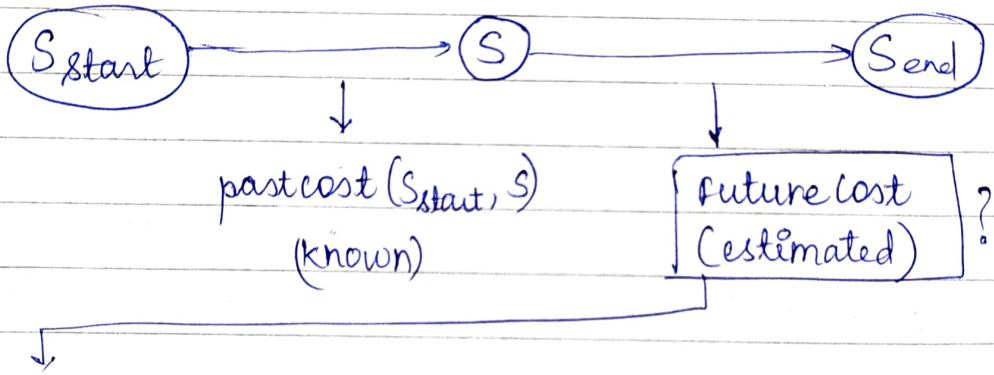
→ learning from previous data



Que: Predict the cost

W, B = 3+2 = 5

W, W, W = 3+3+3 = 9



Estimation func^n

→ Heuristic (h)

Prediction : cost (walk ~~s~~) = 3 (each step)

cost (bus) = 2 (each step)

Info provided / intuition / claim (from training data):

walk, walk, walk → optimal

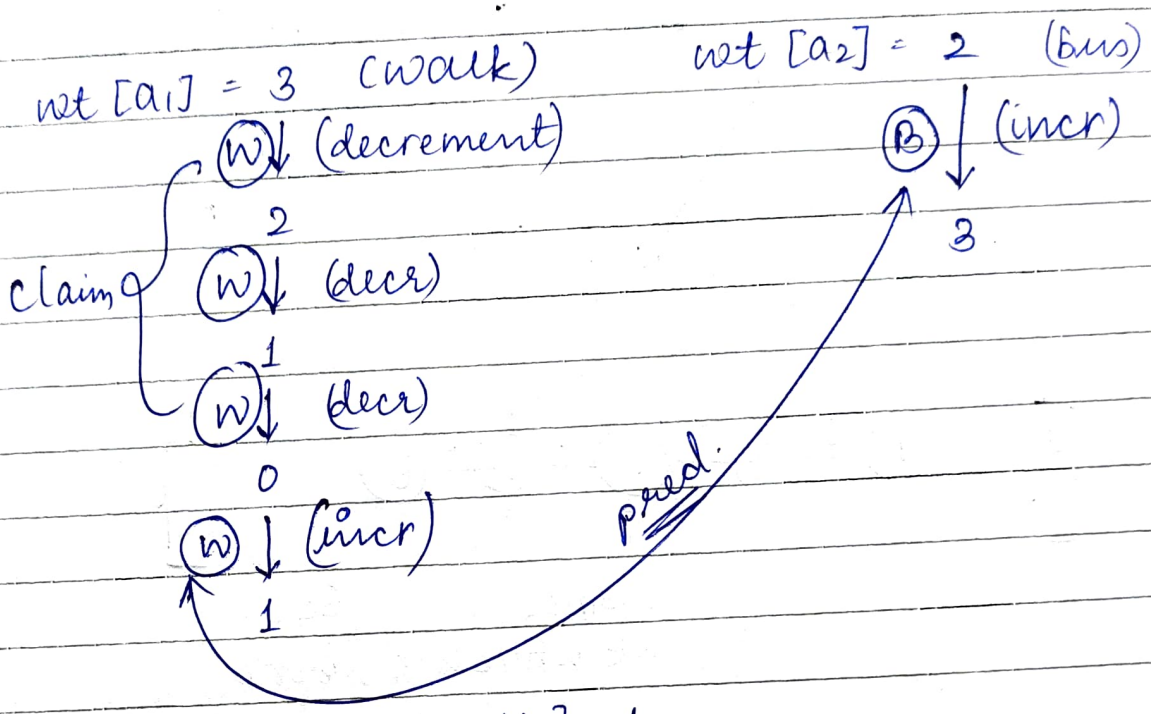
My heuristic (Prediction) : walk, bus → optimal

∴ with my prediction & prev. claim, if they match, we get optimal.

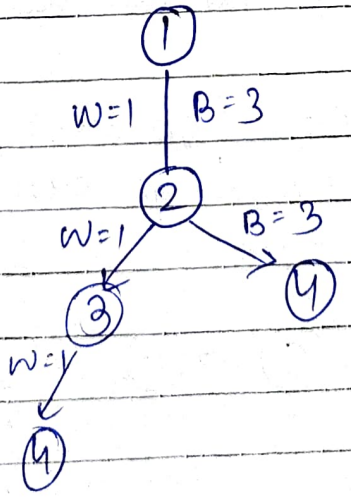
DATE : / /
PAGE NO. :

weights → wt [a1], wt [a2] action ↓

- We take a func" which ↑ or ↓ if we go by prediction or claim.



∴ Final values: wt [walk] = 1
wt [bus] = 3



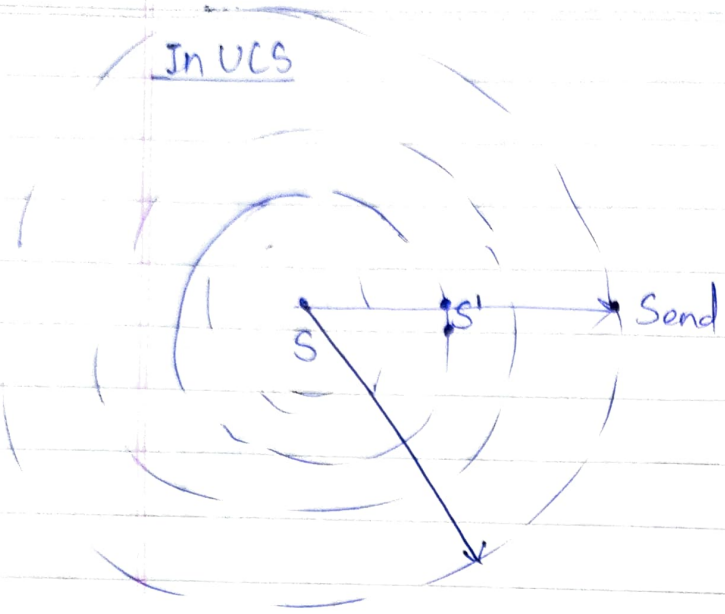
∴ Now optimal path = WWW ≠ prediction

∴ our prediction was wrong & we go ~~for~~ t by training set.

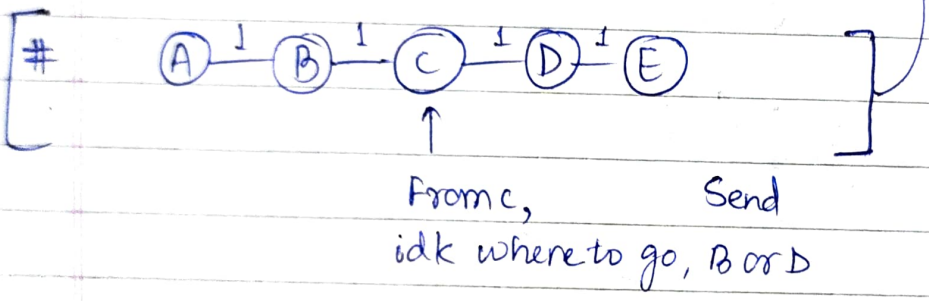
∴ heuristic told us that we predicted wrong weights

DATE: / /
PAGE NO.:

In UCS



similarly, here we may be going towards or away from Send

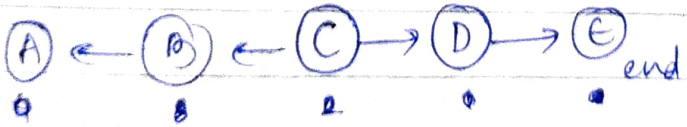


From c, idk where to go, B or D

Send

★ [But what if we can predict that C→B is taking me away from Send and we need to go from C to D.]

Q: How to ensure we don't go back?
A: We define a heuristic (hop count)



h(S) = 4 3 2 1 0

cost(s, a) ← not used

$$cost'(s,a) = \underbrace{past\ cost(c)}_{\text{(known)}} + heur.(s') - heur(s)$$

$s' = succ(s,a)$ ← successor of s

$past\ cost(c) \rightarrow 1$ (given)

∴ here,

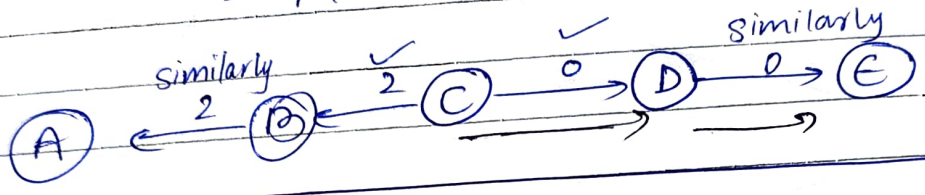
cost'(C, B) = past cost(c) + h(B) − h(C)

= 1 + 3 − 2

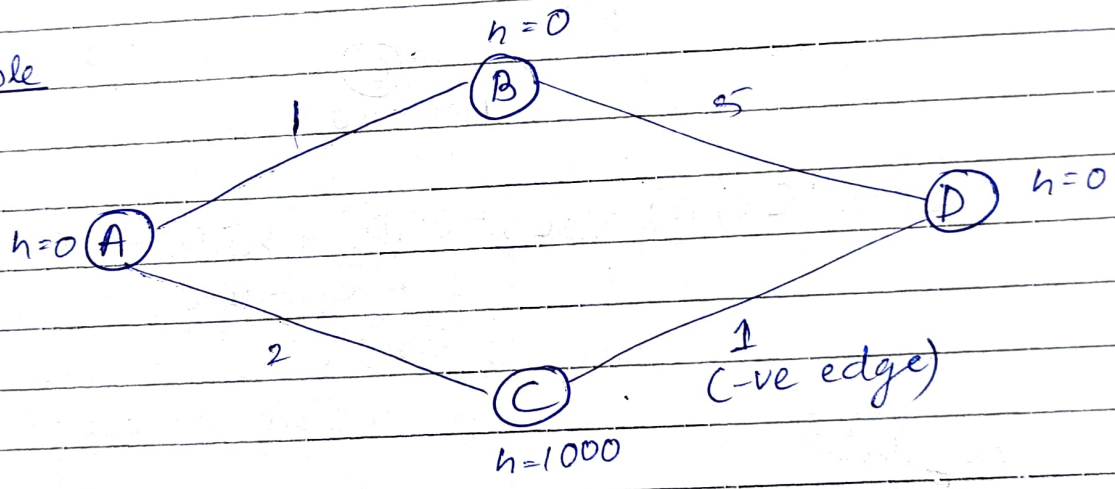
= (2)

cost'(C, D) = past cost (C) + h(D) − h(C)

= 1 + 1 − 2 = (0)



Example

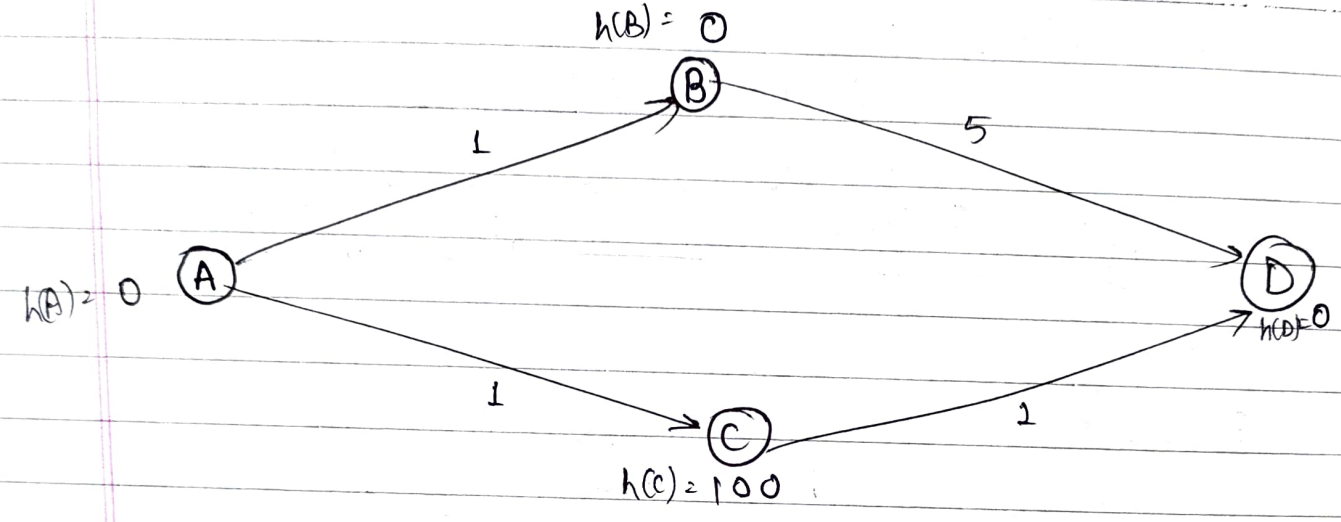


cost'(C, D) = 1 + (0 − 1000) = −999 ← not allowed in UCS

∴ can't apply 1 heuristic for all situations/datasets, even if we are doing the same optimization in them.

DATE : 26/8/22
PAGE NO. :
Friday

• consistent h
• corrections
• Greedy BFS
• CSP - COP



$$cost'(C,D) = cost(C,D) + h(D) - h(C)$$
$$= 1 + 0 - 100$$
$$= -99$$

-ve UCS not allowed (inconsistent h)

h (heuristic) is consistent iff it is always +ve.

∴ For it to be consistent,

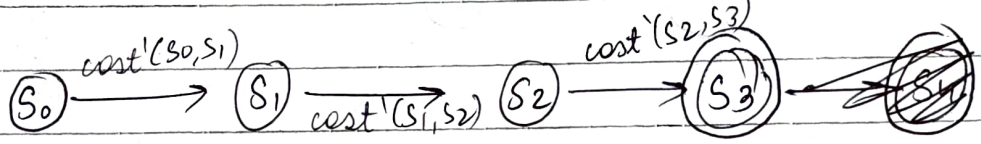
$$cost'(S_i, S_{i+1}) \geq 0$$
$$\therefore cost(C,D) + h(D) - h(C) \geq 0$$
$$\therefore cost(C,D) + h(D) \geq h(C)$$
$$\therefore \boxed{past\ cost(S) + h(S') \geq h(S)}$$

DATE : / /
PAGE NO. :

⊙ heuristic value of end state = 0, always.
- when you reach end state / goal, no point in calculating h for further paths.

Q: How will you know path found is correct?
A:

Eg: Goal → S_0 to S_3



Path obtained: $S_0 - S_1 - S_2 - S_3$

Estimated cost (S_0, S_1)
or

add {
$cost'(S_0, S_1) = cost(S_0, S_1) + h(S_1) - h(S_0)$
$cost'(S_1, S_2) = cost(S_1, S_2) + h(S_2) - h(S_1)$
$cost'(S_2, S_3) = cost(S_2, S_3) + h(S_3) - h(S_2)$

- $cost'(S_0, S_3)$ = Total cost. ⇓ optimal or not?

$$\sum_{i=1}^{n} cost'(S_{i-1}, S_i) = \sum_{i=1}^{n} cost(S_{i-1}, S_i) + h(S_n) - h(S_0)$$

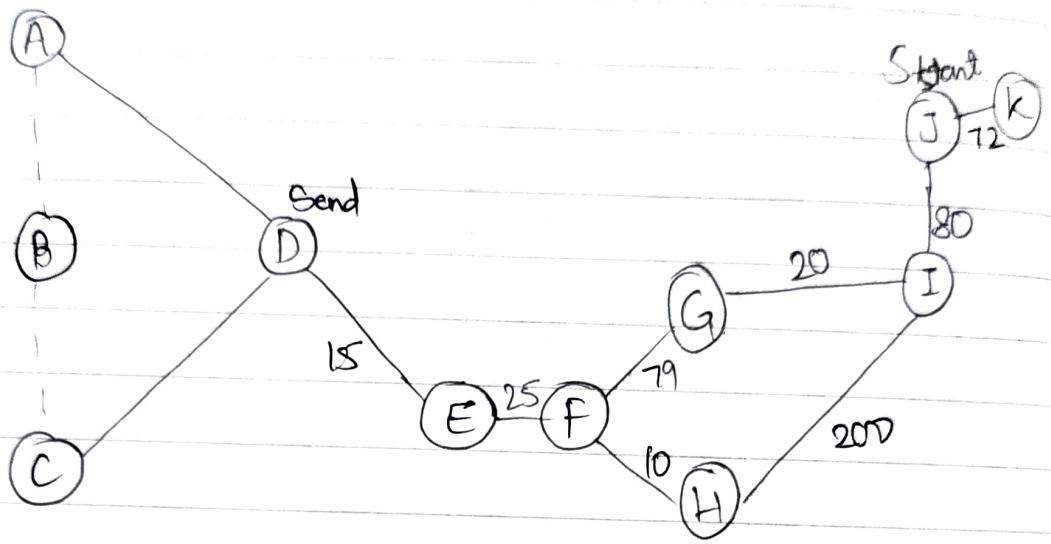
(end = n)

- $\sum_{i=1}^{n} cost(S_{i-1}, S_i)$ ⇓ - prev. path cost - algorithmic cost from UCS
- $h(S_n)$ ⇓ 0 (consistency)
- $h(S_0)$ ⇓ constant ≥ 0

- Since we already know that UCS gives optimal cost and we are decrementing some (the) constant from it, $\therefore \sum_{i=1}^{n} cost'(S_{i-1}, S_i)$ is the optimal one.

→ We can ~~only~~ do this : only estimate 'h' for next state.

Eg



→ h(S) = Euclidean dist^n of (S, S') (minimal)
(Not using past cost)

{ heuristic defined as }

Let,

⇒ Euc (J-K) = d_1 & Euc (J-I) = d_2
& $d_1 < d_2$ (known)

∴ We go from J to (K).

⇒ From K, ~~Euc~~ only 1 option,
∴ We go from K to (J)

Now again heuristic takes us to K.

∴ We are stuck in J ⇄ K loop.

∴ Can't depend on heuristic alone

Reason: greedy of ~~best~~ taking simple heuristic ~~made~~

This is the **Greedy BFS** approach

DATE : / /
PAGE NO. :

Greedy Best First Search

→ Best 1st search : coz we are prone do selecting the best ~~pat~~ edge in current scenario.

∴ We modify it with { pastcost + h(n) ↓ Eg: ~~h(S') - h(S)~~ euc($S_i \to S_{i+1}$) }

① J → K → cost'(J,K) = 72 + h

② K → ~~K~~ J → cost'(K,J) = (72+h) + 72 = 144 + h

③ J → K → cost'( ) = 144 + h ✗

↘ I → cost'( ) = 80 + h ✓ (picked)

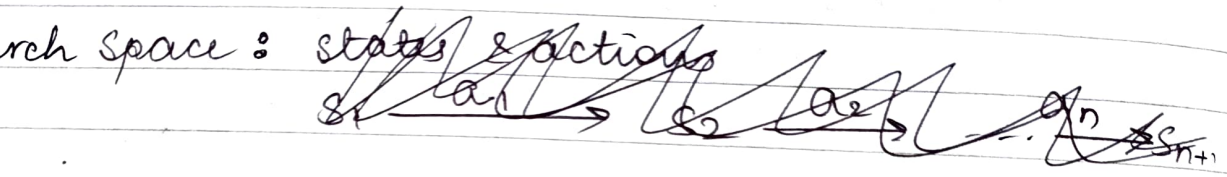
→ Greedy BFS → not optimal
→ in tree, no option of loop
∴ Not complete for tree } Greedy BFS
but complete for graph

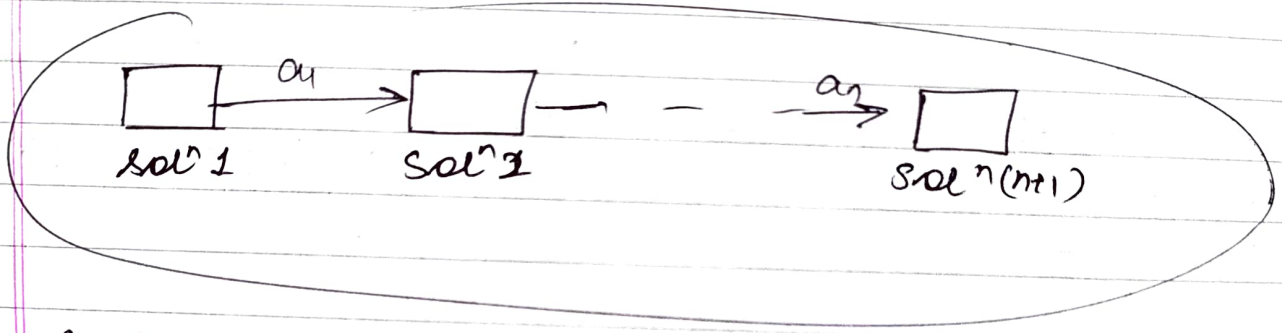
# Uninformed Search	VS	Informed search
→ ~~All~~ Brute force / blind search for goal / Send		→ Greedy BFS on heuristic → extra info provided

Goal : 'a path' or 'a set of action' } so far
Search space : states & actions

Hill-climbing search

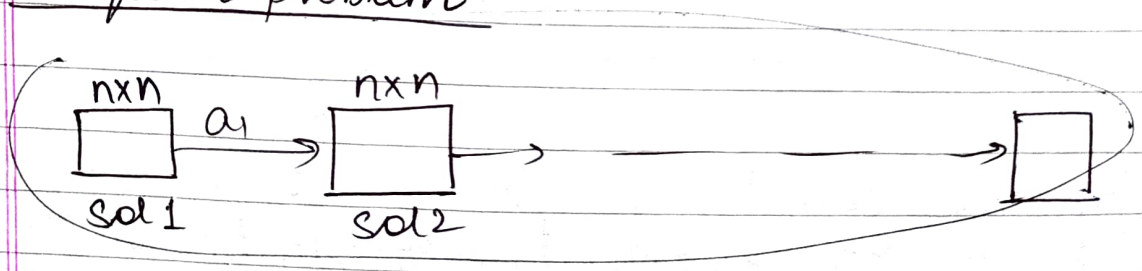
search space : ~~states & actions~~



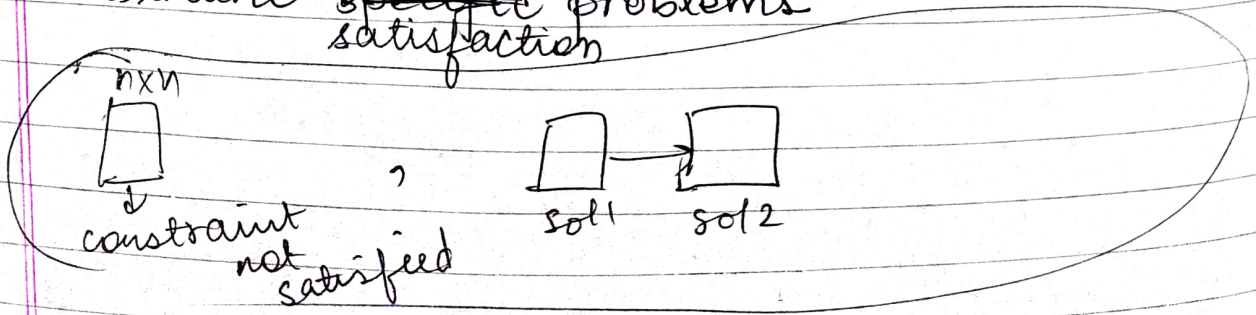


→ Each sol^n is a state
→ Goal states could be different but same sol space
→ Each state is a sol space

N-queen problem



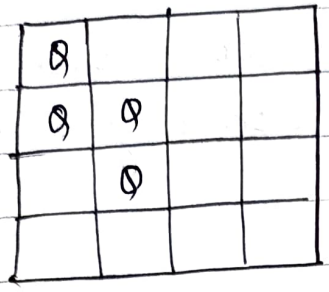
→ different solution ⇒ different placements for N-queens
→ No '1' goal state
→ Constraint - ~~specific~~ problems satisfaction



DATE : / /
PAGE NO. :

How?

(CSP) constraint-satisfaction for (N-queens) → to → (COP) constraint-optimization for (N-queens)



← constraint not satisfied

- We have to find minimum moves to change this to satisfactory placement
- That would be the optimal solution.

- ① Let after 'm' no. of moves → got ~~satisf~~ sol^n
② ,, ,, 'm' no. of moves → got sol^n

If $m < m'$ → (ans 1 → optimal)

Eg: CSP : ~~op~~ objective func^n, subject to constraints $C_1, C_2 - - - C_n$.

Que: If it satisfies 3 constraints, we reach our goal
$\therefore {}^nC_3$ options

COP - objective fun^c & n constraints

Que > Goal: the sol^n which satisfies all n constra-ints that will be the optimal sol^n.

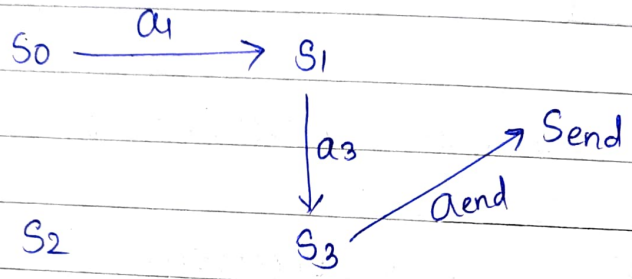
DATE : / /
PAGE NO. :

Q: COP is given, make it CSP.

A: ~~≥3 ∈ all sols~~

If optimality cond^n = ≥3 constraints
= all solutions will satisfy CSP.

29/8/22



solution: $a_1 a_3 a_{end}$.

(BUT)

Eg: In n-queen's prob : only 1 state is sol^n, not a set of actions

In a local search,

Eg: 8 queen prob.

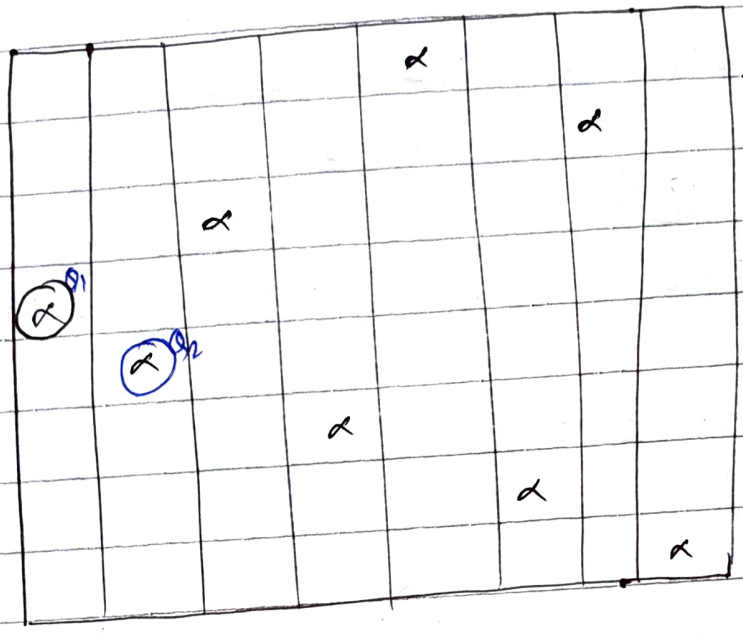
→ check for neighbourhood nodes with some function.

→

Catch : How you define neighbourhood func^n.
How many extra spaces you need to search?
determines if our neighbourhood func^n is good or bad.

DATE : / /
PAGE NO. :

8-queens



Possible actions for all queens : a_1, a_2, a_3, a_4.
Neighbourhood func" ? → how can you move taking every possible neighbour node.

case 1: <u>can move anywhere in column</u>, each column can have 1 queen atmost.

→ Q_1 = 7 moves (in its own column)
Q_2 = 7 moves
⋮
Q_8 = 7 moves

∴ 8 × 7 = 56 possible moves

→ If n is increased → $O(n^2)$ moves.

case 2: <u>move only 1 step in adjacent node</u>

Q_1 :
After every step, check if >1 queen in a column - Tiresome!!

DATE : / /
PAGE NO. :

Case 3: Move each queen anywhere on board.

∴ for q_1 : ~~$(8\times8)-1 = 8^8$ = huge no.~~

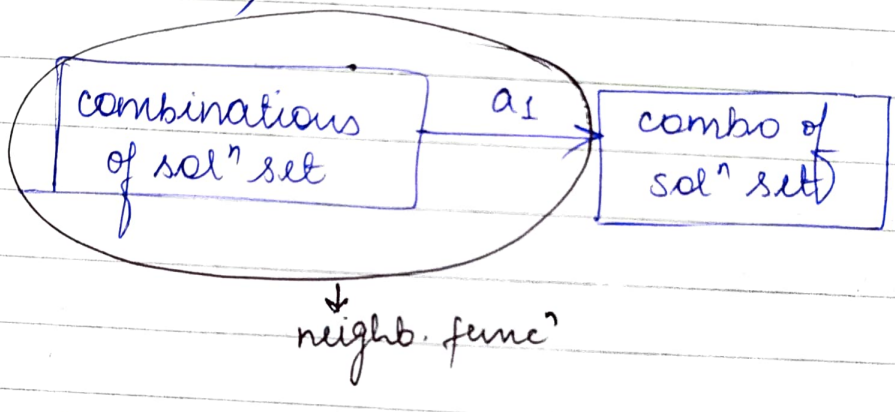
$(8\times8)-1$.

~~8 wrong place~~

$= (8\times8) - 1.$

∴ ~~8~~ $[8\times8-1]^8$ ← for all queen

Neighbourhood func^n :
(NO successors)



→ Local search : 1 action, possible states
{not a set of actions} (reachability)

How to choose neighbour ?

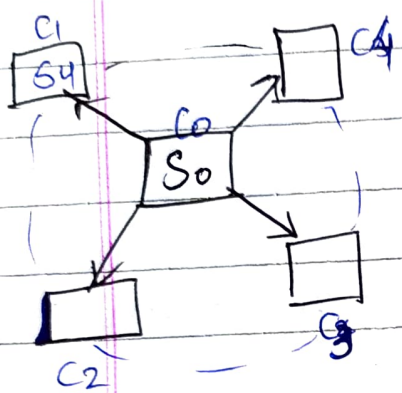
→ Random sapling method : pick any 1 outta 56 Case 1

→ Random walk : max^m similarity b/w neighbouring nodes.
: move towards most similar neighbour.

DATE : / /
PAGE NO. :

Hill Climbing Problem (HCP)

- 8-queens is also a HCP
- Greedy
- Local, looking for best neighbour



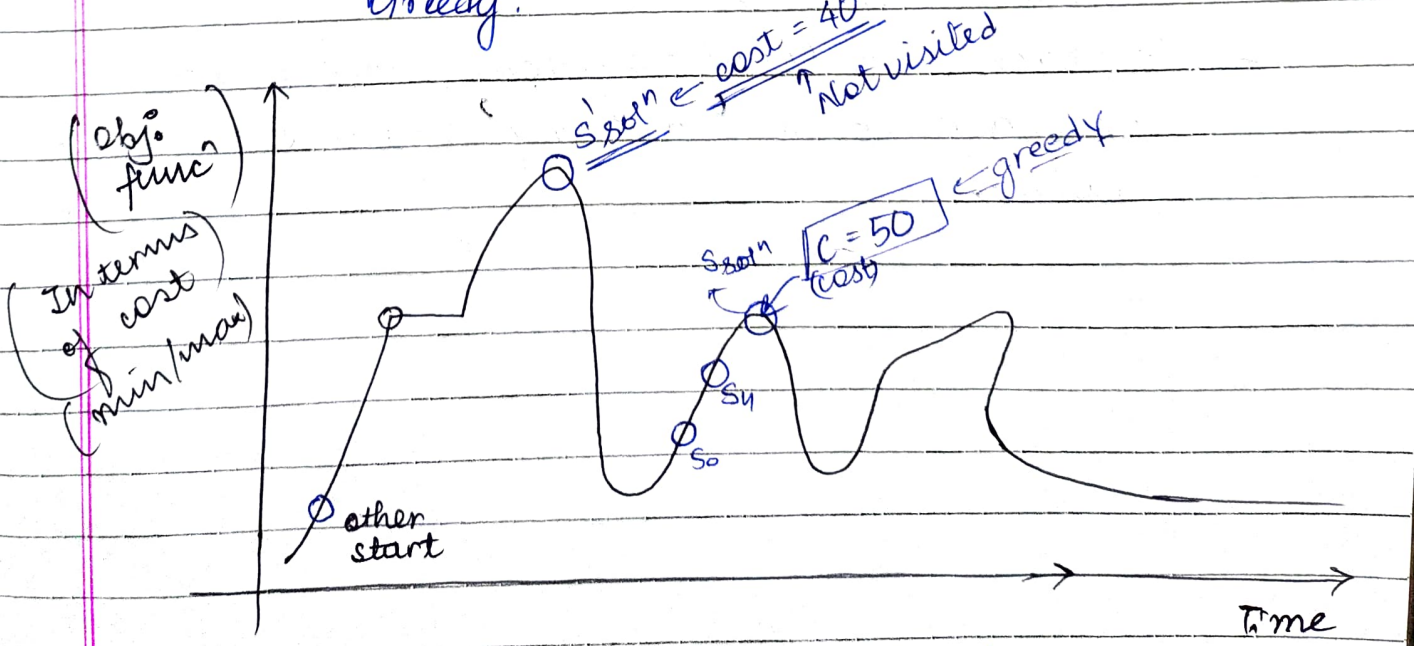
- Max/Min → optimisation
- only 1 state will have Max/Min (>) (<) next move (assumed)
- ∴ others will be nullified.

let it go from S0 → S4

Is it the desired answer / optimal answer?

- C_{sol^n} = 50, say, but I haven't travelled a path with cost = 40.

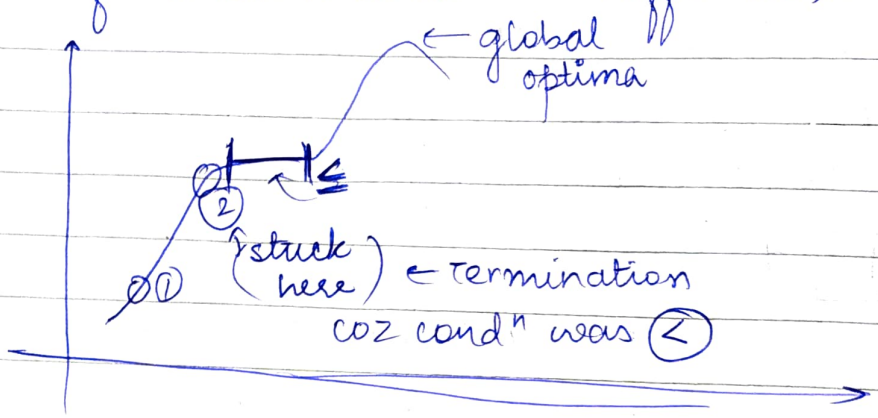
Drawback of HCP: Never reach global optimum. Greedy.



We didn't visit cost 40 sol^n coz neighbours of S0 were not optimal in the opposite dir^n.

DATE : / /
PAGE NO. :

But if the start state was different,



- But if cond" is ≤ ← now we can move on constant path.
- Now we are encouraging our machine to climb the hill.

If we put a stopping cond", it drastically improves the success rate.

* Read stats abt HCP.

To reach from local to global, you need to come back to the 'base camp' or the start point.