LECTURE 1

SATURDAY DATE: 20/01/24

INTRODUCTION

AI winter 1 → AI use stop হয়ে গিয়েছিল

* Domain specific system — just একটা domain এর মধ্যে limited
  — একটা single problem নিয়ে কাজ করে

* Logical agent — logic assign করে
  — ever growing কাজ গুলোর জন্য scalable না
  — problem এর সাথে সাথে logic auto change হয় না, manually change করা লাগবে।
  — Domain change হলে, logic agent কাজ করে না

AI winter 2 → hardware introduce করার পরেও AI model কে train করাতে পারত
→ chess solve করতে পারে

* Computer দিয়ে image কে set of points দিয়ে represent করা tough. To resolve this, the values of the face is matched with the coordinates of geometric matrix. Geometric shape handle করে।

* Main change came between 2013 — 2014 when DL was introduced.

* DL can extract features automatically.

* We are taking data and model is trained.

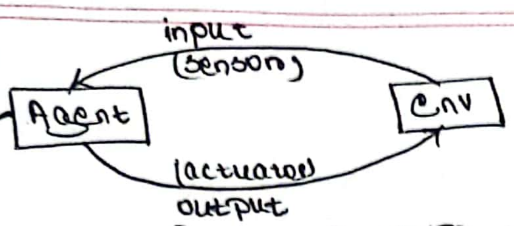
CNN at a time entire program কে process
করা start করে, 180, can be expensive → weakness of CNN

To solve the problem faced by CNN, we use transformer has attention module which can know which part of the data can be changed using the trained model.

Transformers can learn from available data which can be a security threat to humans.

Models যত বেশি advanced হচ্ছে, আমাদের তত security breach create হয়।

All the systems are referred as agents.
Agent environment থেকে info নেয়, input নিয়ে process করে output generate করে।



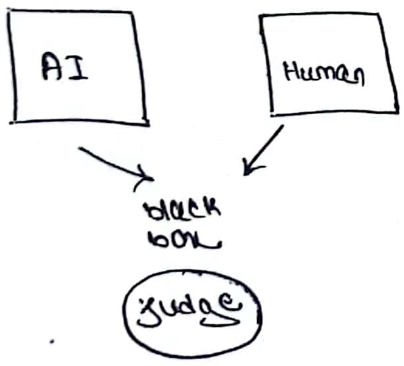
Agent একটা env এ থাকে, Agent → solves a problem.

ChatGPT এর sensor → input from keyboard
actuator → computer monitor

→ Rational Behaviour → efficient কাজ check করা → human emotion থাকে না
→ একটা task as efficiently as possible কাজ করায় থাকি

→ Humanlike Behaviour → Human feelings কে machine এ simulate করার try করি

The Dead button → AI কোনো rules violate করলে, instantly এ button use করে turn off করা possible.

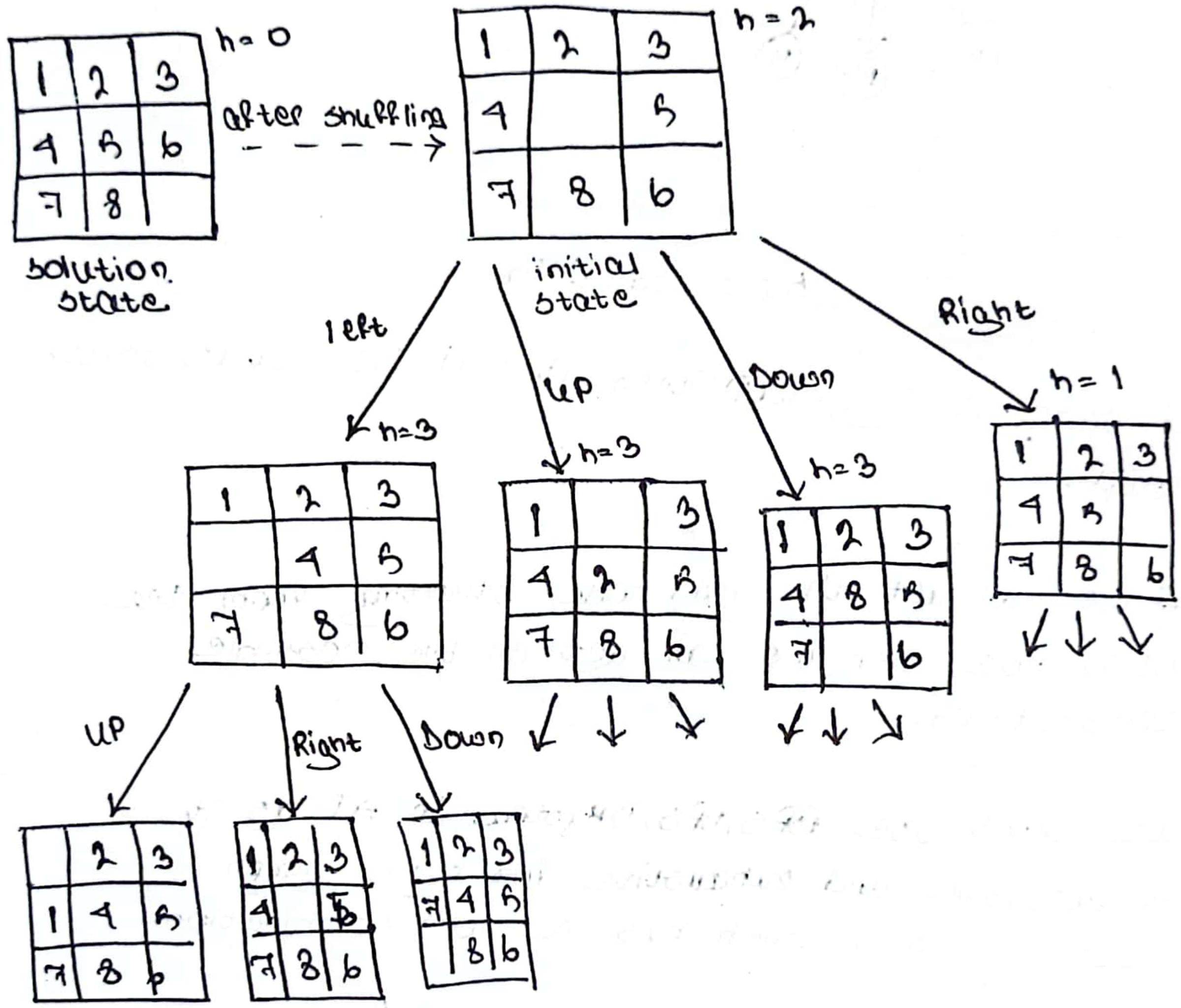


* Judge interacts with AI and Human.
If the judge is successful in differentiating betn AI and Human, AI is not intelligent.
If the judge is unsuccessful in differentiation, AI is behaving like a human & is intelligent.

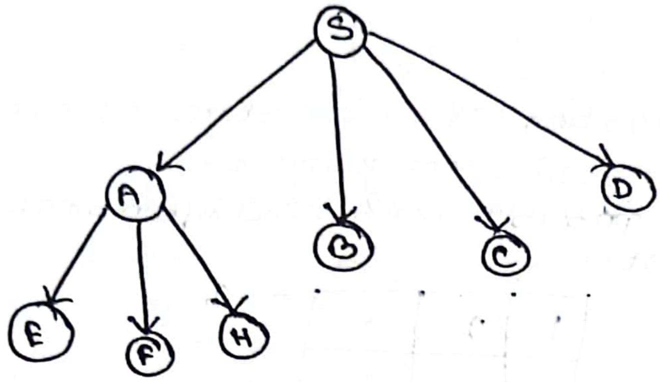
LECTURE 2

THURSDAY DATE: 25/01/24

State space → transition of one state to another in a specific environment.
→ all possible configuration/combination of a state



State space of a 8 puzzle game



State Space Tree

* If there is any loop, then it will be state space graph.

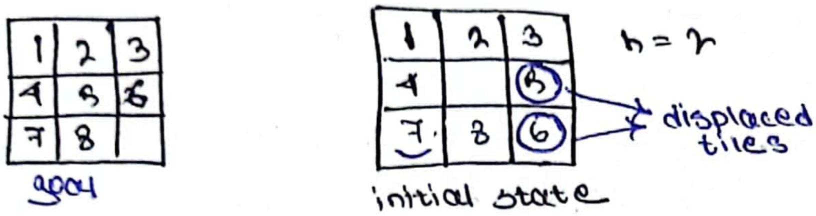
→ If we can get the goal node starting from the initial node, then we can get all the possible combinations.

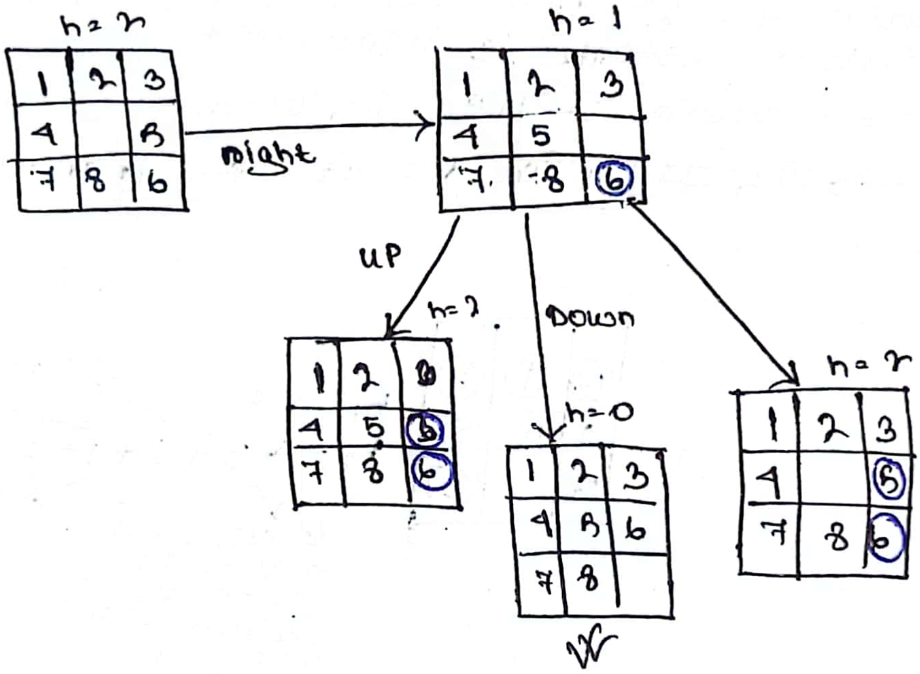
→ We cannot use BFS, DFS, Dijkstra in AI as it is expensive and exhaustive for huge graph such as chess which has a 10^{120} combination.

→ BFS, DFS are blind search as they search the entire tree to find the shortest path to reach the goal node.

→ Informed search → has general idea of where the goal node can be present.
→ need to reduce heuristic to get to the goal
→ The node with reduced/smallest "h" that is taken into consideration and expanded further.

Heauristic → define the estimated distance from a node to the goal node.
→ is a function





* need to choose the node with the lowest value of h and then expand it to get to the goal.

1	2	3
4	5	6
7	8	

1	2	3
4	5	
7	8	6

X

$h_1 = 1$
$h_2 = 1$

	2	3
4	5	6
7	8	1

Y

$h_1 = 1$
$h_2 = 4$

Here, h=1 because only one tile is displaced. However, the distance of displaced tile is greater for Y than the X one. To solve this, manhattan distance is introduced.

$h_1 \rightarrow$ ~~dis~~ number of tile displaced.

$h_2 \rightarrow$ ~~dis~~ কয় ঘর displaced count হবে (manhattan distance)

6	1	2
4	5	8
7		3

$h_1 \rightarrow 5$

$h_2 \rightarrow 3 + 1 + 1 + 2 + 2 = 9$

'1' displaced 1 tile

'8' displaced 2 tiles

'3' displaced 2 tiles

'2' displaced 1 tile

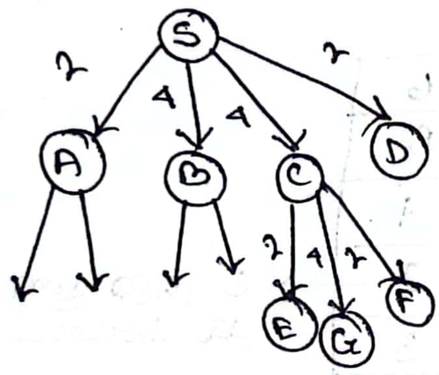
'6' displaced 3 tiles to get to the shuffled position from goal

* The heauristic which returns the larger value contains more values while returning the displaced tile — need to be taken, at first.

* Heauristic needs to be defined according to the scenario. So, when to consider h_1 and when to consider h2 depends on the given scenario.

weighted state space tree → If the edges have different value, then the state space tree is weighted.

The one which has the highest priority cost action needs to have low value on its edge and vice versa.



[Left and right are priotised over up and down]

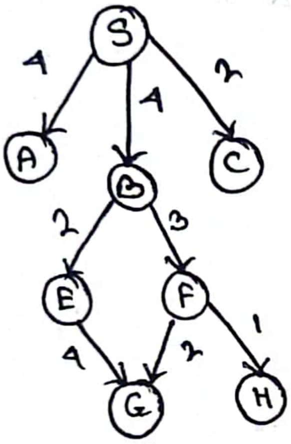
h-table → arbitary heauristic function is used

S	2
A	3
B	3
C	3
D	1
E	2
G	0
F	2

Informed search

- → Greedy Best first search
- → A* search

Greedy Best first search



S	6
A	5
B	4
C	3
E	3
F	2
G	7
H	0

→ C থেকে goal node এর distance 3

- Here, Evaluation function is the heuristic itself.

Data structure
↳ Fringe

Expand → when children of a node is visited

push → any node is inserted

pop → any node is taken out

| S^6 |

* start state কে push করি

S^6 | A^5 B^4 C^3 |

* S এর children কে push করব

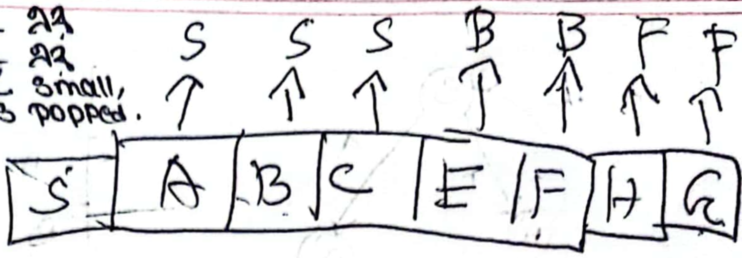
C^3 | A^5 B^4

*A, B, C এর মধ্যে C এর h value small, so it is popped.

B^4 | A^5 E^3 F^2

F^2 | A^5 E^3 G^0 H^7

G^0 | A^5 E^3 H^7



S→B → F → G

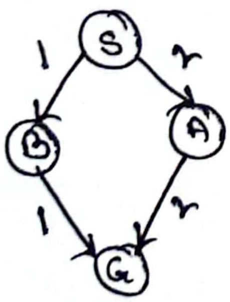
This is the ~~so~~ shortest path found

S ⟶ B ⟶ F ⟶ G

G এর immediate parent is F

*priority linked list is used to store the pop out node. [not in 422 syllabus]

Greedy Best First search works based on heuristic value only ignoring the priority.



S	2
A	2
B	3
G	0

Disadvantage

(1) not optimal

→ cost of optimal is less compared to the path found here.

S^2

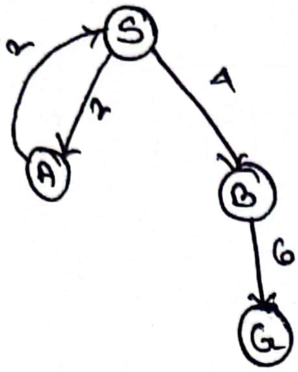
S^2 | A^2 B^3

A^2 | B^3 G^0

G^0 | B^3

S ⟶ A ⟶ G [cost = 4] - - - না

S ⟶ B ⟶ G [cost = 1] - - - → optimal



S	3
A	2
B	4
G	0

$\boxed{S^3}$

S^3 $\boxed{A^2 \; B^4}$

A^2 $\boxed{B^4 \; S^3}$

S^3 $\boxed{B^4 \; A^3}$

A^3 $\boxed{B^4 \; S^3}$

S^3 $\boxed{B^4 \; A^2}$

⋮

Disadvantage

2 is incomplete

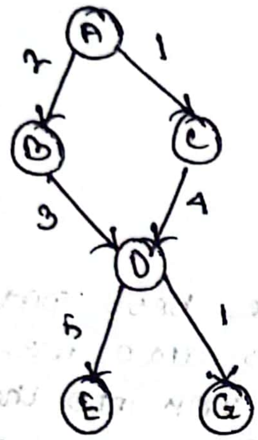
- stuck in a loop and cannot return the goal node.

So, it is incomplete.

LECTURE 3

SATURDAY DATE: 27/01/24

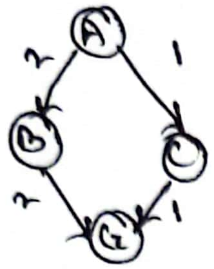
Informed search



A	7
B	2
C	3
D	4
E	2
G	0

	A^7
A^7	B^2 C^3
B^2	C^3 D^4
C^3	D^4
D^4	E^2 G^0
G^0	E^2

A → B → D → G



A	4
B	2
C	3
G	0

cost of LHS = 4

cost of RHS = 2 (optimal)

To get the best path using this algo, we have to look into the heuristic function, which is not possible in GBFS.

A^4

A^4 | B^2 C^3

B^1 | G^0 ~~B^2~~ C^3

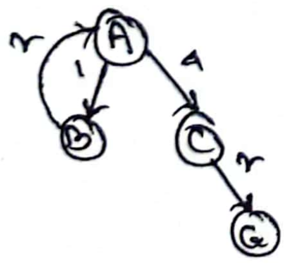
G^0 | ~~B^2~~ C^3

$A \rightarrow B \rightarrow G$

cost = 4

⟶ is not optimal

∴ Best path is not returned.



A	4
B	2
C	3
G	0

A^4

A^4 | B^2 C^3

B^2 | C^3 A^4

A^4 | C^3 B^2

B^2 | C^3 A^4

A^4 | C^3 B^2

we cannot reach the goal node as it is in a loop.

To get the complete process and node in a loop, we can keep track of already visited node.

A^4

A^4 | B^2 C^3

B^2 | C^3

C^3 | G^0

G^0 |

✗

* প্রত্যেক node এর জন্য আলাদা memory allocate করি, so is expensive.

However, doing so doubles the memory. So, we try to avoid this

Tree search → Don't keep track of visited node

Graph search → keep track of visited node.

GBFS

↳ $f(n) = h(n)$

$h(n)$ → heuristic function

$f(n)$ → evaluation function

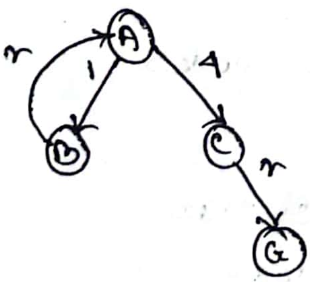
A* search

A*

↳ $f(n) = h(n) + g(n)$

$h(n)$ → heuristic function of node "n"

$g(n)$ → Path cost from root node to node "n"



A	4
B	2
C	5
G	0

$A^{4+0=4}$

A^4: $B^{2+1=3}$ $C^{5+4=9}$

B^3: C^9 $A^{4+3=7}$

A^7: C^9 $B^{2+4=6}$ $C^{5+7=12}$

B^6: C^9 $A^{4+6=10}$

C^9: G^{0+6} A^{10}

G^6: A^{10}

- C^9 is considered over C^{12}, so C^{12} is excluded.
- For B^6, $g(n)$ can be found $6 - 2 = 4 + 2 = 6$

  6 → evaluation cost of B^6; 2 → $h(n)$ of B; 4 → cost of A from B

A → C → G

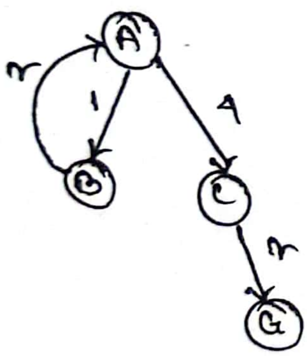
↳ C^9 first time A^4

To make the A* search optimal, we have to make changes to heuristic function.

Tree A* ⟶ Admissible ⟶ $h(n) \leq g(n)$ . . .

↓

h-value has to be lower or equal than the actual cost of the goal
(Not overestimated)



A	4
B	2
C	$\cancel{8}$ 2
G	0

$h(A) \leq g(A)$
$\Rightarrow 4 < 6$ ✓

$h(B) \leq g(B)$
$\Rightarrow 2 < 8$ ✓

$h(C) > g(C)$
$\Rightarrow 8 > 2$ ⟶ need to change h(C) so that it is less than or equal to 2.

For a tree A* to be optimal, all the nodes needs to be admissible.

If any node has $h(n) > g(n)$, then h(n) of that node is forcefully reduced to g(n) or less than g(n).

Graph A^* → consistency → every node is visited once
(↳ this is maintained)

↳ h-cost of node N has to be lower than the h-cost of child (N)

For C,

$$h(C) \leq c(C,G) + h(G)$$

$h(C)$ → heuristic of parent node

$c(C,G)$ → node থেকে immediate child যাওয়ার path cost

$h(G)$ → heuristic function of immediate child node

$$h(C) \leq c(C,G) + h(G)$$
$$5 \not\leq 2 + 0 \quad ✗$$

* কোন node inconsistent হলে, সে node বার বার visited হয় so optimal path না ও পেতে পারি

For A,

$h(A) \leq c(A,B) + h(B)$ - - - → using B as child of A

$4 \leq (1+2) = 3$

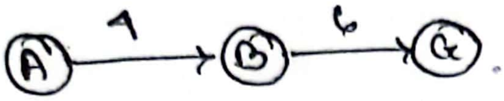
$h(A) \leq c(A,C) + h(C)$ - - - → using C as child of A

$4 \leq (4+1) = 5$

$c(A,B) + h(B) < c(A,C) + h(B)$

So, we have to check $h(A)$ needs to be smaller than $c(A,B) + h(B)$ to be consistent.

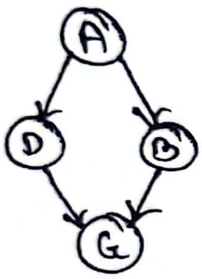
If consistency is ensured, then admissible can be ensured. The reverse is not true.



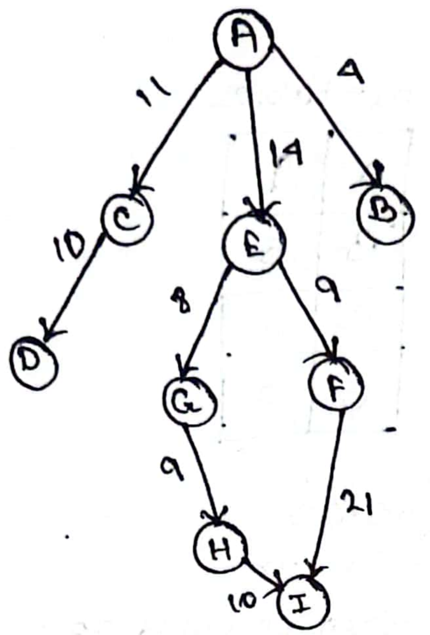
A	9
B	4
G	0

Admissible ✓

consistent ✗



G can be reached using multiple paths. even though it's a tree.
So, we use ~~A~~graph A^* here.



A	36
B	37
C	32
D	24
E	28
F	17
G	19
H	9
I	0

$\boxed{A^{36+0=36}}$

A^{36} $\boxed{B^{37+7=44} \quad C^{32+11=43} \quad E^{28+14=39}}$

E^{39} $\boxed{B^{44} \quad C^{43} \quad G^{19+22=41} \quad F^{17+33=40}}$

^{40} $\boxed{B^{44} \quad C^{43} \quad G^{41} \quad I^{0+44=44}}$

^{41} $\boxed{B^{44} \quad C^{43} \quad I^{44} \quad H^{9+31=40}}$

^{40} $\boxed{B^{44} \quad C^{43} \quad \cancel{I^{44}} \quad I^{0+41=41}}$

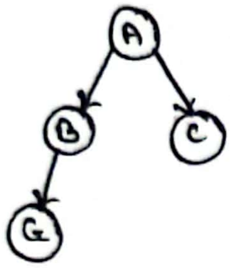
evaluation cost of $I^{41} < I^{44}$, so I^{44} is excluded.

^{41} $\boxed{B^{44} \quad C^{43}}$

$\rightarrow E \rightarrow G \rightarrow H \rightarrow I$

LECTURE 4 THURS ~~SATURDAY~~ DATE: 01/02/24

Heauristic Dominance



h_1 - table

A	7
B	4
C	5
G	0

h_2 - table

A	9
B	5
C	6
G	0

h_2 ~~table~~ ~~is~~ dominates h_1 as every node in h_2 table is greater than or equal to h_1 - table.

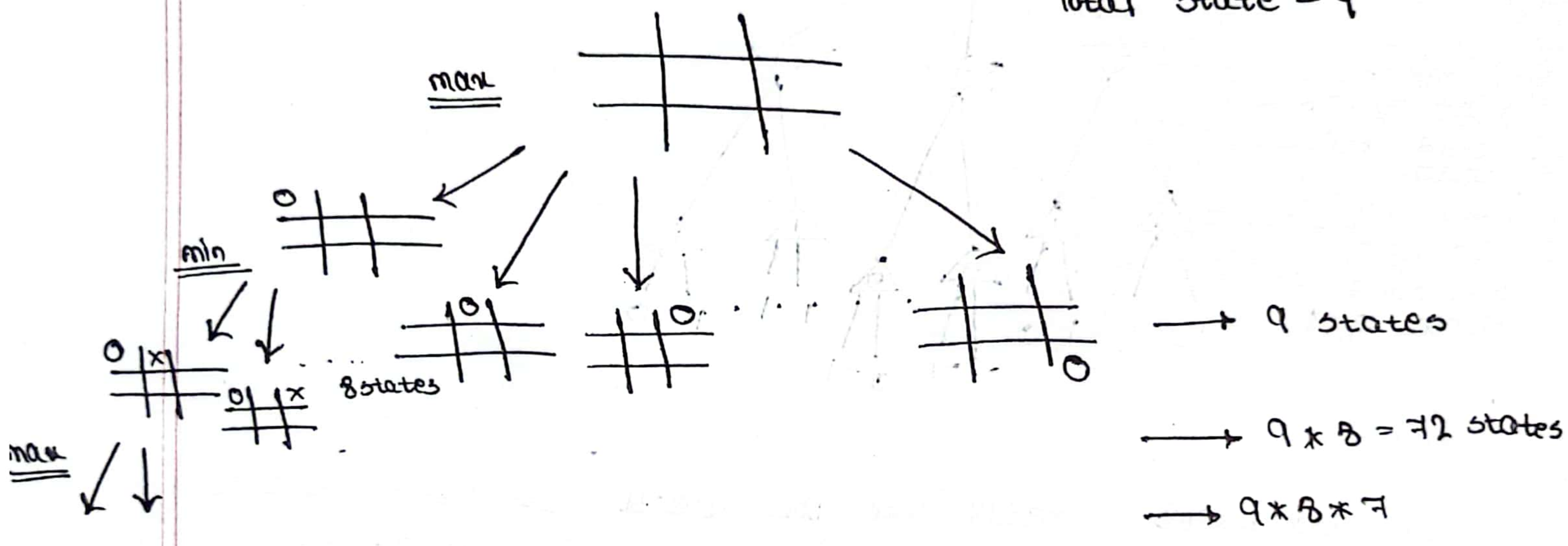
$h_2 \geq h_1$

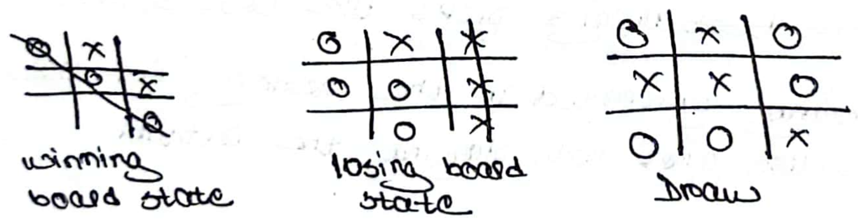
~~he~~

GAMES

→ chess is not a pre-determined game.
→ series of action cannot be determined in ~~this~~ multi-player game like chess.
→ we cannot use Greedy or A^* search here.

Total state $= 9^3$





→ we have to lookahead the move of ~~what~~ the opponent to make the move.

Zero sum game → If you gain sth, you are gaining in your perspective ~~and~~ ~~the~~ by that amount and the opponent is losing by that amount in his opponent.

$(+1) + (-1) = 0$

max → △ [chance of win]

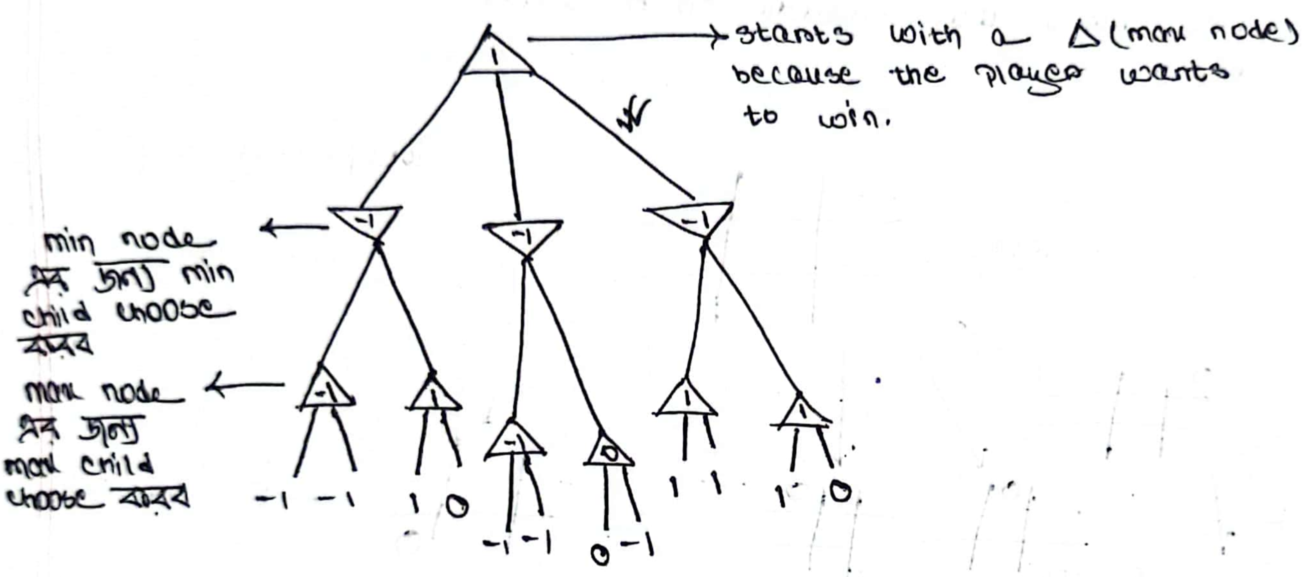
min → ▽ [chance of lose]

w = 1

l = -1

d = 0

p1 - perspective



p1 - prepares himself for the worst as he considers his opponent ~~to the best~~ always picks the best. ~~So, the~~

perspective is being generated in the memory for the lookahead purpose. We are not playing the actual game.

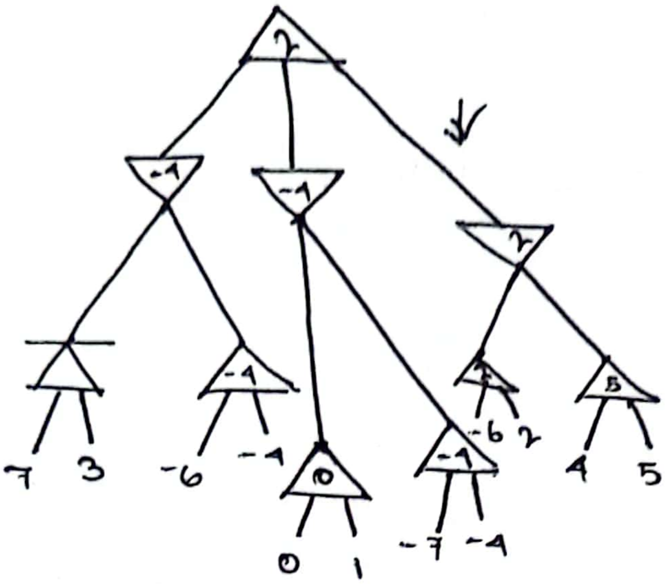
The entire process is called min-max algorithm.

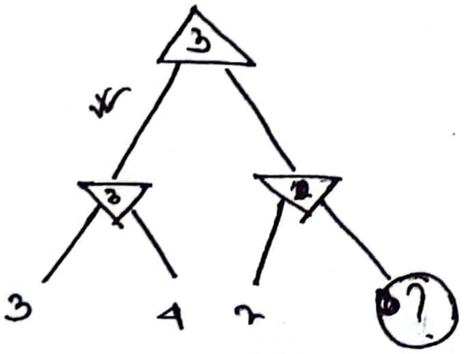
Total state of chess = 10^{120}

→ is difficult to store in memory

→ difficult to generate the final state.

A goodness function is used. It uses the previous history. It returns a +ve value if player in perspection has the chance to win. and returns a -ve value if he loses.

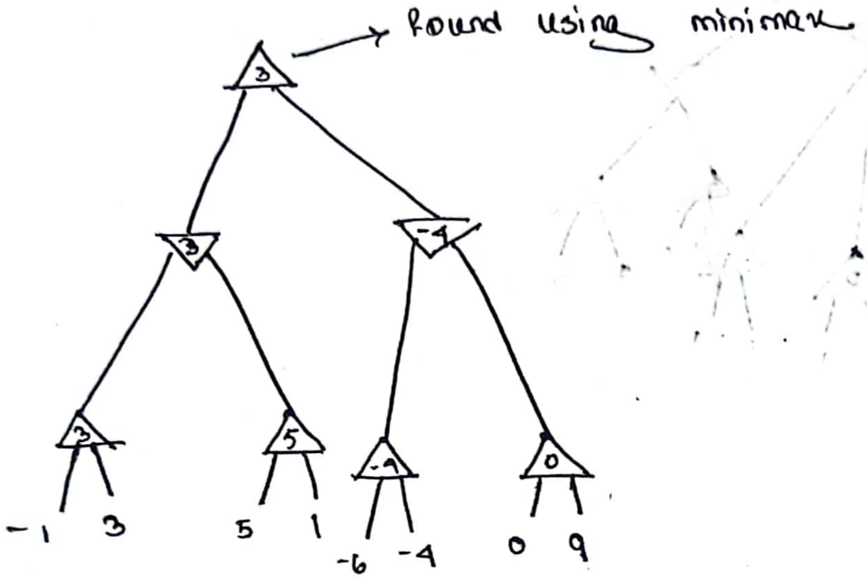




→ এখানে যে value এ থাকবে, start node always win path choose করবে (in this scenario)

* এখানে RHS part টা না দেখতে পারি, exclude করে ফেললেও best winning path ই পাবো।

Alpha–beta planning



~~[found u'~~

Using Alpha – Beta Pruning,

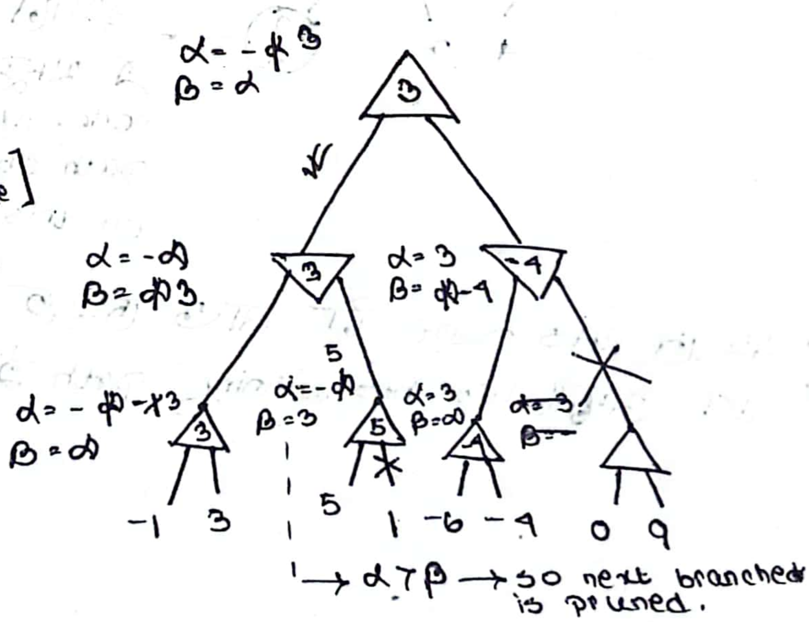
$\alpha \to -\infty$

$\beta \to \infty$

max → update α [with new max value]

min → update β [with new min value]

$\alpha \geq \beta$ → prune



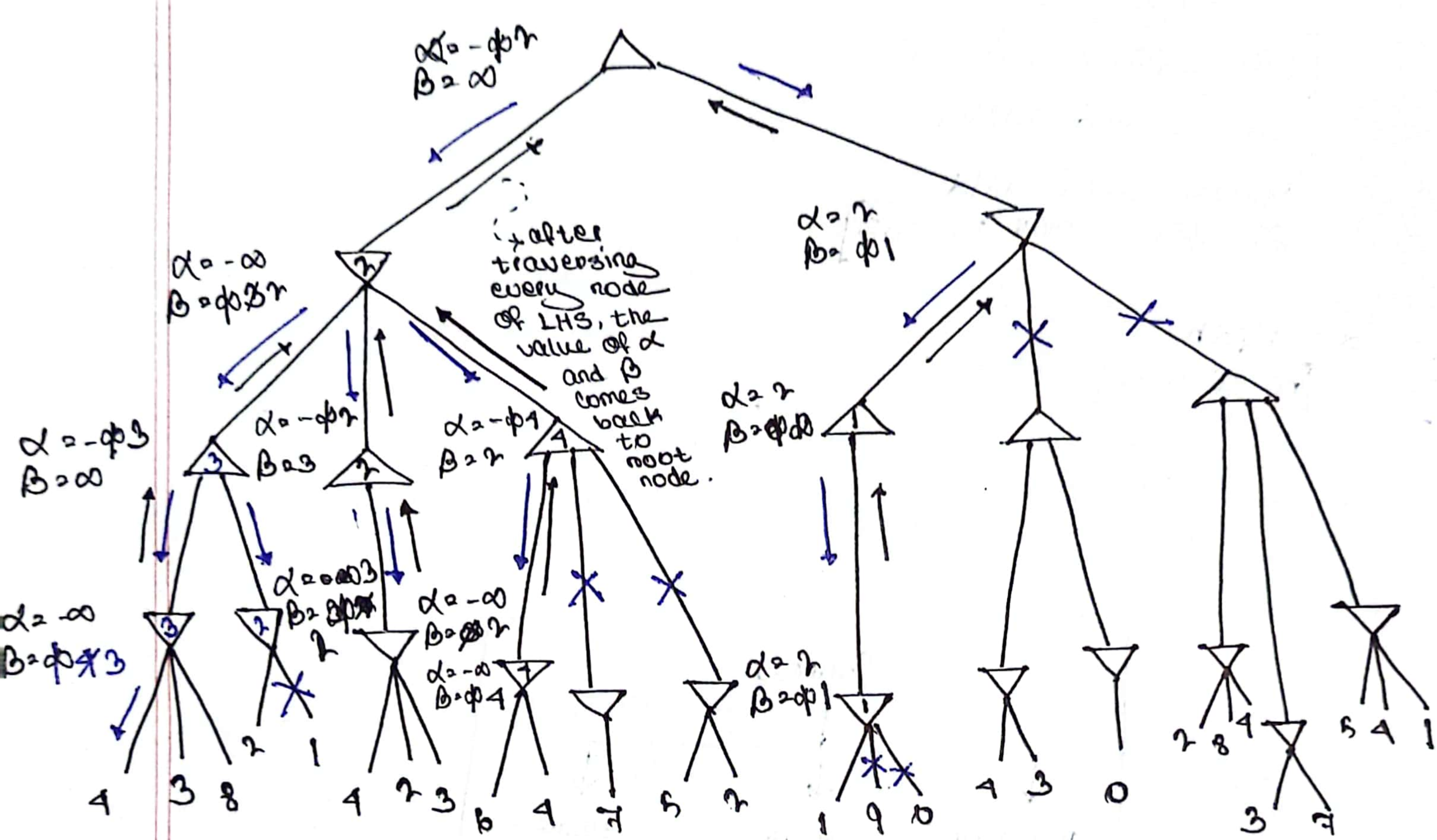
* α and β values are carried from top to bottom only.

* α and β update and check করবার পর node এর value বসায় দিবে।

LECTURE 6

SATURDAY DATE: 03/02/24

EXAMPLE OF α-β PRUNING



$\alpha = -\infty$

$\beta = \infty$

max $\rightarrow$ update α [with max value]

min $\rightarrow$ update β [with min value]

$\alpha \geq \beta$

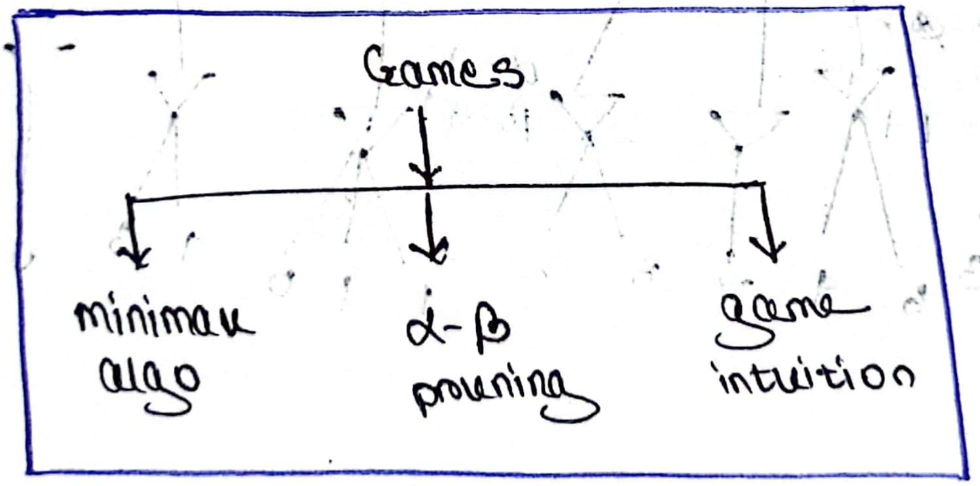
Here, $\alpha = 4$

* node value এর সাথে α-β এর কোন relation নাই

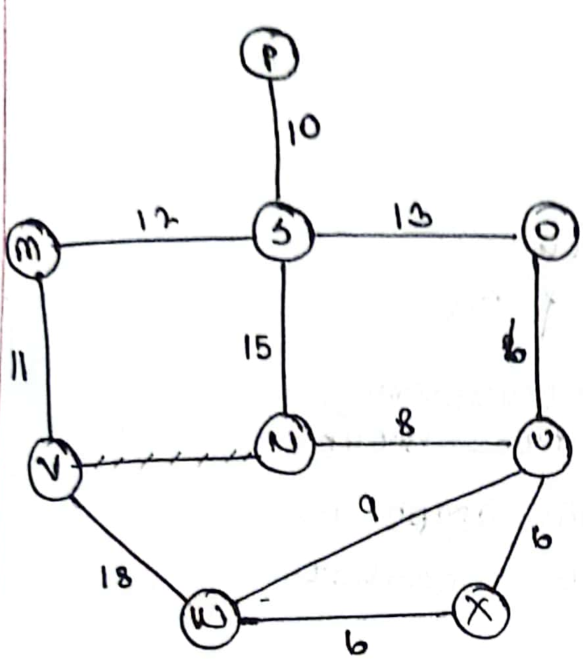
$b^{d/2}$

b = branching factor
[কোন একটা node এর
কয়টা child থাকে,
varied no. of child
থাকলে → max টা নিই]

d = depth
[no. of level]



Informed search



h-table

S	32	→ start
M	19	
N	16	
O	20	
P	22	
U	18	
V	12	
W	0	→ goal
X	3	

Admissible :-

- → goal node এর কাছাকাছি node আগে check করব
- → h-value ≤ path cost (min)

Here, $h(U) \nleq path(U)$

$18 \nleq 9$

∴ This graph is not admissible.

* goal node না চেনা থাকলে, যে node এর h-value = 0 ~~থাকে~~ থাকে goal node তাকে consider করব।

* Better heauristic function হলে more relevant node going to the goal will be visited first.

Consistency

$h(n) \leq c(n, n') + h(n')$

n → parent, n' → child, c(n, n') → path cost from parent & child

$h(X) \leq c(X, W) + h(W)$

$3 \leq 6 + 0$ → check this

$h(X) \leq c(X, U) + h(U)$

$3 \leq 6 + 18$

$h(S) \leq c(S,P) + h(P)$

$32 \leq 10 + 22 \leq 32$

$h(S) \leq c(S,m) + h(m)$

$32 \leq 12 + 19 \leq 31 \longrightarrow$ check

$32 \nleq 31$

$\longrightarrow$ consistency rule breaks.

$\therefore$ The graph is not consistent

$h(S) \leq c(S,O) + h(O)$

$32 \leq 13 + 20 \leq 33$

$h(S) \leq c(S,N) + h(N)$

$32 \leq 15 + 16 \leq 31$

* সব node এর জন্য consistency check করব, যেখানে consistency rule break করবে, full graph টা inconsistence হয়ে যাবে।

* Consistent হলে admissible হবে।

* Admissible হলে consistent হতেও পারে, না ও পারে।

→ যে node visit করা হয়েছে, further visit করবে না

Graph A* (যদি node টা queue তে enqueue হলে dequeue হতে পারে)

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

$g(n)$ → source node থেকে "n" node এ আসার path cost.

Queue তে reverse alphabetically node insert করব।

$S^{37+0=37}$

S^{37}: $P^{27+10=37}$ $O^{30+3=33}$ N^{31} M^{31}

N^{31}: P^{37} O^{33} M^{31} $U^{18+23=41}$

M^{31}: P^{37} O^{33} U^{41} $V^{12+23=35}$

P^{37}: O^{33} U^{41} V^{35}

O^{33}: $\cancel{U^{41}}$ V^{35} $U^{18+19=37}$

→ U fringe এ আছে, so U^{41} এর চেয়ে কম lower cost path follow করে f(n) update হয়, so U^{41} বাদ দিয়ে ওই updated U টা বসাবে দিব

V^{35}: U^{37} $W^{0+41=41}$

U^{37}: $\cancel{W^{41}}$ $X^{3+25=28}$ W^{28}

X^{28}: W^{28} $\cancel{W^{31}}$ → W^{28}

A^{8} B^{8} C^{8} D^{11}

→ A আগে insert হয়েছে, so A আগে pop out হবে

— x — x —

B^{8} C^{8} D^{11}

A^{8} B^{8} C^{8} D^{11}

→ queue তে B আগে insert হয়েছে, so B আগে pop out হবে

→ এই data structure কে fringe বলে

Path:

S → O → U → W

* shortest path না ও বের হতে পারে as graph is not admissible.

Pure Graph search

└→ একটা node visit করলে তার insert করবেও না

S^{32}

S^{32} | P^{32} O^{33} N^{31} M^{32} |

N^{31} | P^{32} O^{33} M^{31} U^{41} |

M^{31} | P^{33} O^{33} U^{41} $V^{12+23=35}$ |

P^{32} | O^{33} U^{41} V^{35} |

O^{33} | U^{41} V^{35} |

V^{35} | U^{41} $W^{0+41=41}$ |

U^{41} | W^{41} $X^{3+29=32}$ |

X^{32} | W^{41} |

W^{41} | |

Path :- $S \rightarrow M \rightarrow V \rightarrow W$

LECTURE 6

THURSDAY DATE: 08/02/24

GENEPTIC ALGORITHM

- → part of local search
- → survival of the fittest

State randomly generate করা হয়। যে state fit সেই state গুলো থেকে আরো state generate করা হয় which goes close to the goal.

* monkeys are selected randomly.

covid is pushed to check how they survive

→ | M1 | M2 | M3 | M4 | → Generating initial population

M1	M2	M3	M4
↓	↓	↓	↓
excellent	good	poor	good
		↓	
		excluded	

| M1 | M2 | M4 | → selection

* Using M1, M2 and M4, new generations are created by mating. [Gender is not taken into account]

M1 M2 ⟶ M5 M6
M1 M4 ⟶ M7 M8 } cross over + mutation

M5 M6 M7 M8.

* The entire process is repeated until we receive a completely immuned generation.

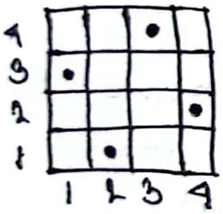
Steps of Genetic

1. Generating intial population
2. Selection
3. Crossover
4. mutation
5. Fitness Calculation

runs in a loop

n-queen

King & queen and ~~(or) they~~ are kept away from each other

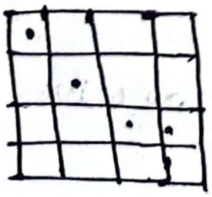


Here, n = 4
$\hookrightarrow$ no. of queens

This board is represented using number in the computer

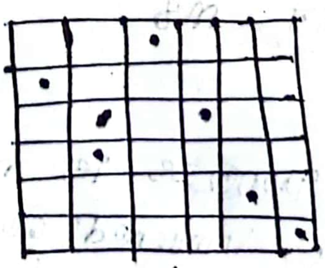
1st ~~row~~ col → 3
2nd ~~row~~ col → 1
3rd ~~row~~ col → 4
4th col → 2

⇒ 3 1 4 2



⇒ 4 2 2 2

n = 4



⇒ 5 3 6 4 2 1

n = 6

n = 8

8- queen

g1 g3 → gene

c1 → 24748552

c2 → 32752411

c3 → 24415124

c4 → 32543213

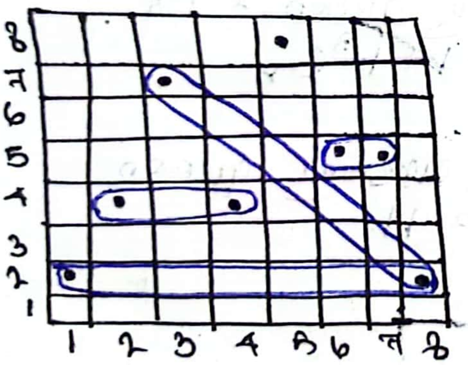
population

→ chromosome

gene use করে chromosome করে, chromosome দিয়ে population create করে।

Fitness → ~~qu~~ max queens don't clash

c1

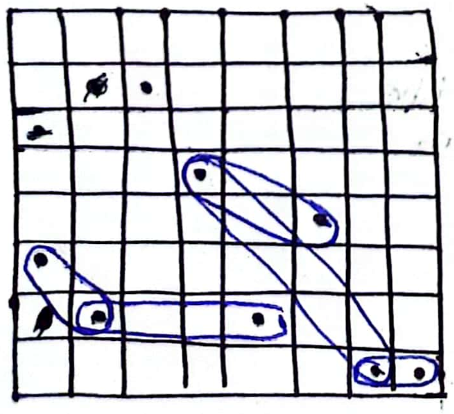


$\underset{n}{8}C_{\underset{pair}{2}}$ combination of non-~~clash~~ queen = 28

Combination of clash queen = 4.

no. of non-clashing queen = 28-4 = 24

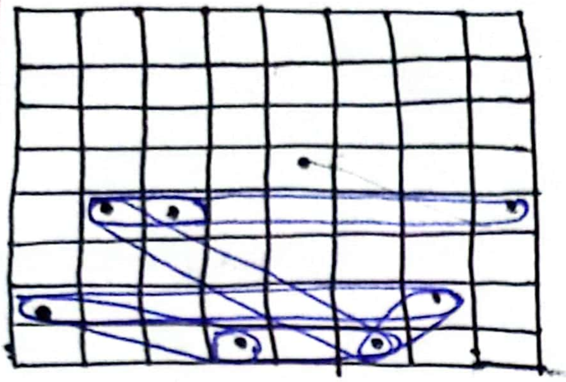
c2



no. of non-clashing queen = 28-5 = 23

no. of clashing queen = 5

C3 ^3C_r

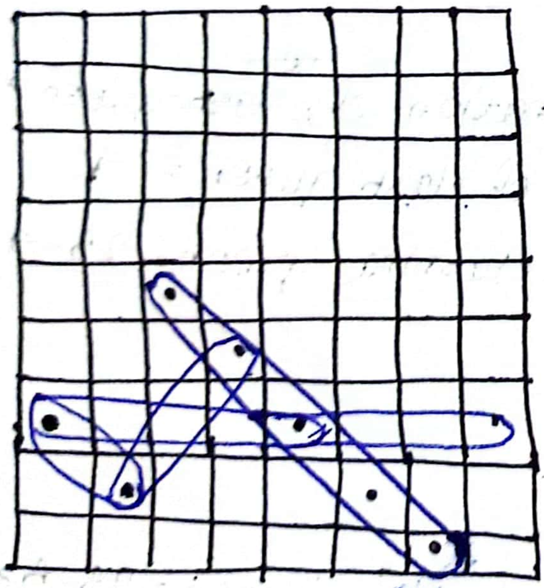


no. of clashing queen = 8

no. of non-clashing queen = 28 − 8
= 20

* একটা row তে / diagonal এ multiple queen থাকলে, combination use করে clashing queen বের করব for that part

C4



no. of clashing queen = 17.
$= {}^3C_2 + 1 + 1 + 1 + 1 + {}^3C_r$

no. of non-clashing queen
= 28 − 17 = 11

C4 → is excluded as it has the lowest non-clashing queen.

C1 C2 $\quad$ 24748 | $\boxed{552}$

C2 C3 $\quad$ $\boxed{32752}$ | 411

→ randomly choosen index = 5

mutated

C1 C2 $\quad$ 247 | $\boxed{48552}$ → 32748552 → 32744552

C2 C3 $\quad$ $\boxed{327}$ | 52411 → 24752411 → 24752411

→ index = 3

2474 | ~~4~~8552 → 32758552 → 32758552

3275 | 2411 → 24742411 → 24742411

* At least one children is picked randomly and one of its value is randomly changed. → for mutation.

Solution of 4-queen

		•	
•			
			•
	•		

3142 → all 4 genes present

initial population

3313

2123

3112

2221

Here, we have same numbers in all state.
But, we don't have gene-4, so we cannot land into the solution.
To solve this, we can use mutation to introduce 4.

LECTURE 7

THURSDAY DATE: 15/02/24

0-1 KNAPSACK USING GENETIC ALGORITHM

Object	Reward	Weight
A	20	1
B	5	2
C	10	3
D	40	8
E	15	7
F	25	4
G	~~15~~ 4	5
H	7	2

1 → Object is picked
0 → Object is not picked

* Capacity needs to be utilised and ~~maximu~~ maximum reward can be taken

Initial Population formation → at least 4 chromosome needs to be selected

		A	B	C	D	E	F	G	H
ACD	→	1	0	1	0	0	0	0	0
ABDE	→	1	1	0	1	1	0	0	0
DF	→	0	0	0	1	0	1	0	0
BCDEGH	→	0	1	1	1	1	0	1	1

Cross over will be effected if all the chromosomes present have different number of genes / different length.

To solve this, the above process of ~~0-1 knapsack~~ problem encoding is taken.

Fitness Check

* Fitness value will increase if ~~more~~ objects with more values are taken.

$\xrightarrow{}$ object

$O_1, O_2, \dots\dots\dots\dots, O_8$

$\xrightarrow{}$ reward

$R(O_1), R(O_2), \dots\dots\dots\dots, R(O_8)$

$\xrightarrow{}$ weight

$W(O_1), W(O_2), \dots\dots\dots\dots, W(O_8)$

} global

$\xrightarrow{}$ binary

$B(O_1), B(O_2), \dots\dots\dots\dots, B(O_8) \rightarrow$ specific

$$Re = \sum_{i=1}^{8} R(O_i) * B(O_i)$$ → যে গুলো object choose করি তার reward কে জুড়ে add হয়।

For chromosome 1,

$$Re = R(O_A) * B(O_A) + R(O_B) * B(O_B) + \dots\dots\dots + \cancel{= (20*1)}\, R(O_H) * B(O_H)$$

$$= (20*1) + (5*0) + (10*1) + (40*1) + (15*0) + (25*0) + (4*0) + (7*0)$$

$$Re = 70$$

$$We = \sum_{i=1}^{8} W(O_i) * B(O_i)$$ → যে object choose হয়, তার weight ~~sum~~ add করছি

If $We >$ ~~W~~ max W → fitness = 0

If $We <= W$ → fitness = Re

$$\boxed{\text{Fitness} = \begin{cases} 0, & We > \text{max-weight} \\ Re, & We \leq \text{max-weight} \end{cases}}$$

একটা fitness function এ যত বেশি values include থাকবে, তত better.

DG → 8+5 = 13 ----→ exceed max-weight by 1

DEG → 8+7+5 = 15 ----→ exceed max-weight by 8

Here, fitness function will be 0. ~~using our fitn~~
However, DEG should have been penalised as it exceeds max-weight by a greater value than that of DG.

To solve this, we use:-

$$\text{fitness} = \begin{cases} 0 - (w_e - \text{max-weight}), & w_e > \text{max-weight} \\ R_e, & w_e \leq \text{max-weight} \end{cases}$$

DG → 13 | $0 - (13 - 12) = -1$

DEG → 20 | $0 - (20 - 12) = -8$

Fitness value:-

C1: ACD, $R_e \to 70$, $w_e \to 12$, $f \to 70$ → best

C2: ABDE, $R_e \to 80$, $w_e \to 18$, $f \to 0 - (18 - 12) = -6$

C3: DF, $R_e \to 64$, $w_e \to 12$, $f \to 64$

C4: BCDEGH, $R_e \to 81$, $w_e \to 27$, $f \to 0 - (27 - 12) = -15$

lowest, so excluded

$C1 * C2 \rightarrow$ 1011 | 0000 $\rightarrow$ 11010000

11011 | 000 $\rightarrow$ ~~00001101~~ 10111000

$C2 * C3 \rightarrow$ 10 | 110000 $\rightarrow$ 00110000

00 | 010000 $\rightarrow$ 10010000

LECTURE 8

SATURDAY DATE: 17/02/24

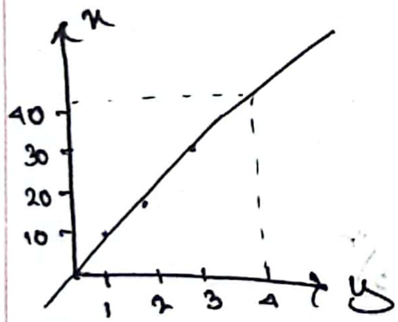
LOCAL SEARCH

→ We can solve infinite domain problems where search space is almost infinite.

amount (x)	price (y)
1	10
2	19
3	30
4	?

→ predict the amount of 4 kg

We can try fit the data points using a function.



$y = mx + b$

m → slope

b → y-intercept

We have derived a linear function using the given data points.

Here, $y = mx$ is the function.

→ need to figure out m.

↳ correctness depends on the value of m

Finding value of m

① Randomly place a value of m to the equation, error is checked.
The smaller the error, the better the value of m is. → "Hill climbing search"

Initially, $m = 9$

$y = mx \Rightarrow y_p = 9x$

For, $x_1 = 1$, $y_{p1} = 9 * 1 = 9$

$x_2 = 2$, $y_{p2} = 9 * 2 = 18$

$x_3 = 3$, $y_{p3} = 9 * 3 = 27$

~~$x_4 = 4$, $y_4 = 9 * 4 =$~~

error $= (y_{p1} - y_{p1}) + (y_{p2} - y_{p2}) + (y_{p3} - y_3)$

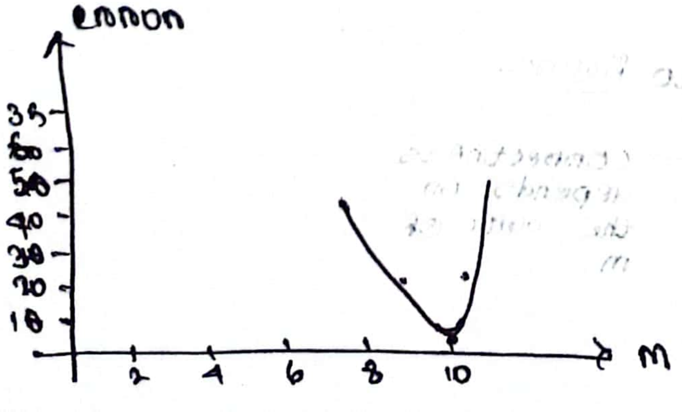
→ error is a distance where it can be -ve and needs to be converted into +ve

modified:—

error $= (y_{p1} - y_1)^2 + (y_{p2} - y_2)^2 + (y_{p3} - y_3)^2$

$= (9 - 10)^2 + (18 - 19)^2 + (27 - 30)^2$

$= 11$



$m = 9$, error $= 11$

$m = 8$, error $= 49$

$m = 10$, error $= 1$

$m = 8$

$y = 8x$

for $x=1,\ y_{p1} = 8*1 = 8$

$x=2,\ y_{p2} = 8*2 = 16$

$x=3,\ y_{p3} = 8*3 = 24$

$x=4,\ y_{p4} = 8*4 = 32$

$$\text{error} = (8-10)^2 + (16-19)^2 + (24-30)^2$$

$$= 4 + 9 + 36$$

$$= 49$$

Here, error এর value ~~কমানোর~~ বাড়তে যাবে if m ↓

$m = 10$

for $x=1,\ y_{p1} = 1*10 = 10$

$x=2,\ y_{p2} = 2*10 = 20$

$x=3,\ y_{p3} = 3*10 = 30$

$x=4,\ y_{p4} = 4*10 = 40$

$$\text{error} = (10-10)^2 + (20-19)^2 + (30-30)^2 = 1$$

Here, error এর value কমতে যাবে if m ↑

$m = 11$

for $x=1,\ y_{p1} = 11*1 = 11$

$x=2,\ y_{p2} = 11*2 = 22$

$x=3,\ y_{p3} = 11*3 = 33$

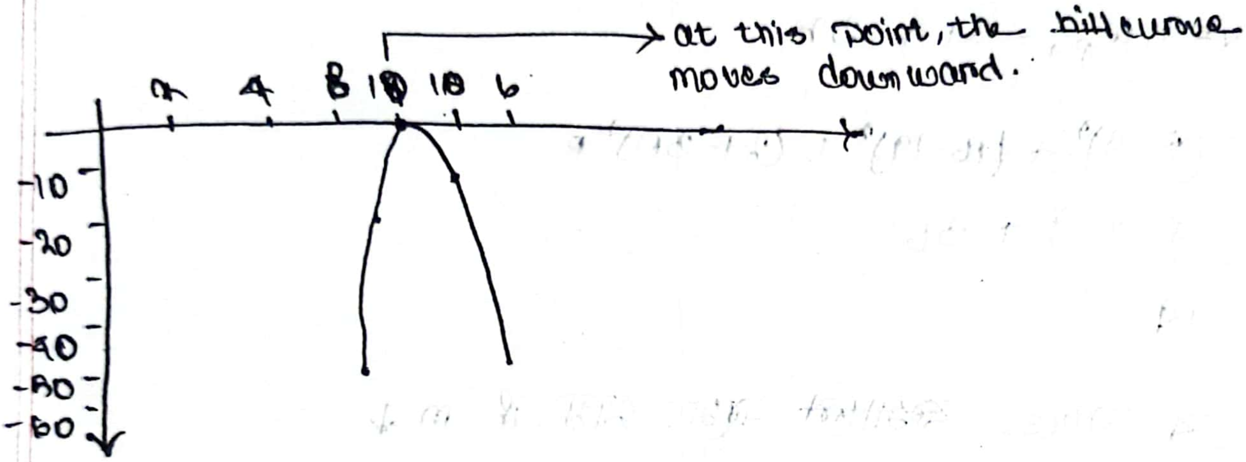
$x=4,\ y_{p4} = 11*4 = 44$

$$\text{error} = (11-10)^2 + (22-19)^2 + (33-30)^2$$

$$= 19$$

error $= -\left[(y_{P3} - y_3)^2 + (y_{P1} - y_1)^2 + (y_{P8} - y_8)^2\right]$

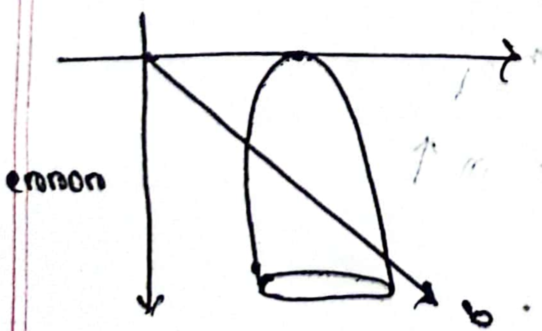
to change direction of graph



* when all the error values fall, that point of m is the most appropriate parameter.

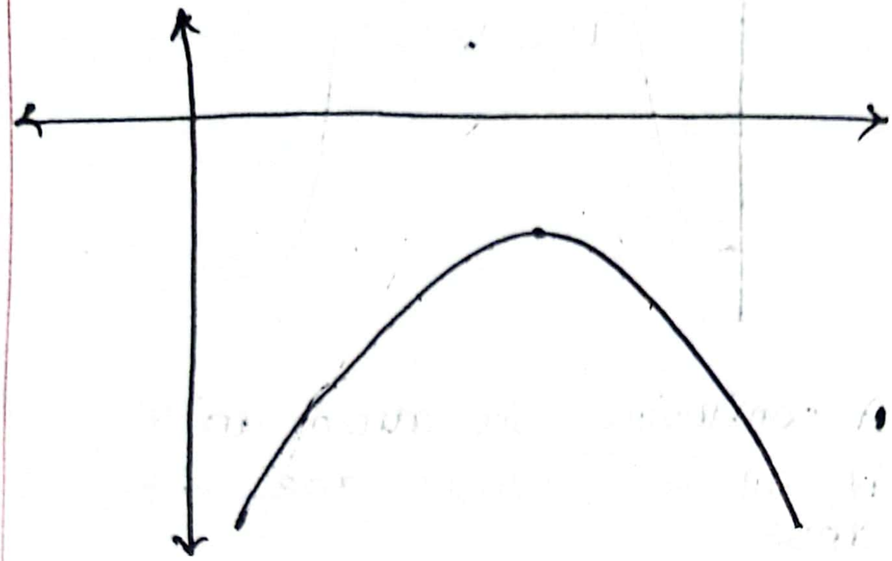
Hill Climbing search.

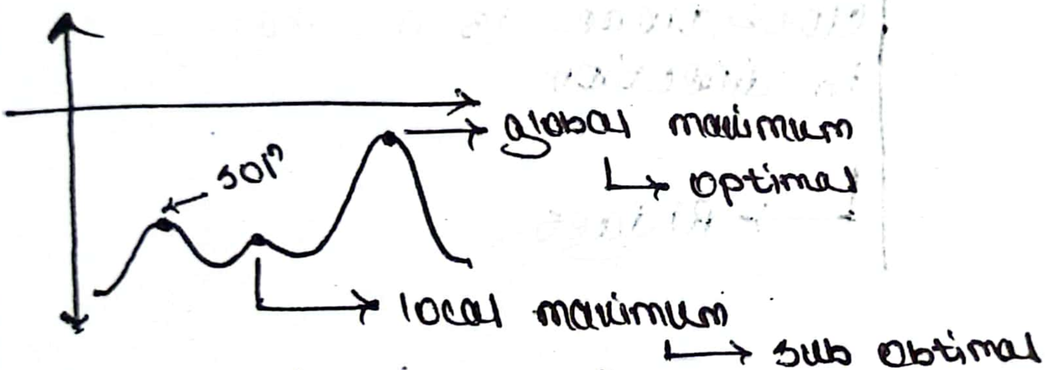
$y = mx + b$



A graph becomes 3D when 2 variables are unknown.

Increasing the no. of unknown variables, increases the error.



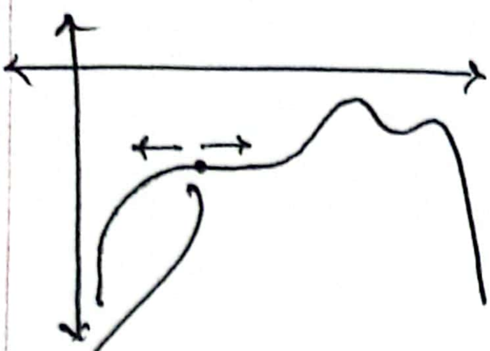


* Here, moving towards left decreasing error.

starting point is selected and goal is selected randomly.

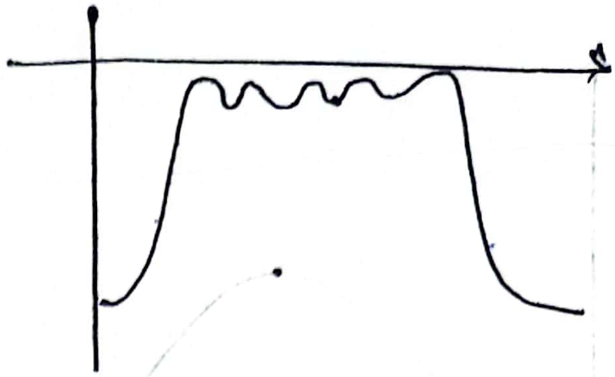
It may return global maximum

Gives sub-optimal solution.



no elevation change so does not have any idea of moving

Plateaus



A confusing situation arises if all the curves are very close.

~~A to~~ When maxima are close, there is a change in direction

Ridges.

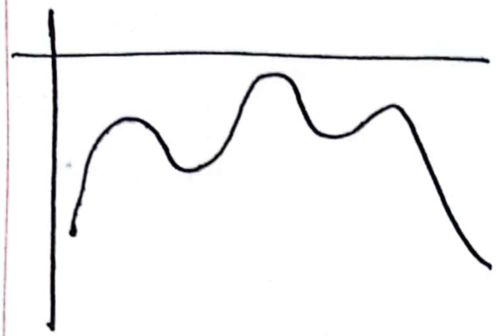
Drawbacks :— ① Local maximum

② Plateaus

③ Ridges

Remedy :— ① Random restart

Random Restart



~~Randomly select a value for~~

we select a value for restarting point randomly in such a way that the chance of getting global maximum increases.

Problem Reformation

লোকাল maxima, ridges, plateaus এগুলো দূর করতে use করি

If we define the error function or hypothesis differently, that will effect the hill.

We have to generate new hill which will be more favourable than the previous function.

Simulated Algorithm

→ global maximum ~~det~~ hit হবে কিনা তা detect করে

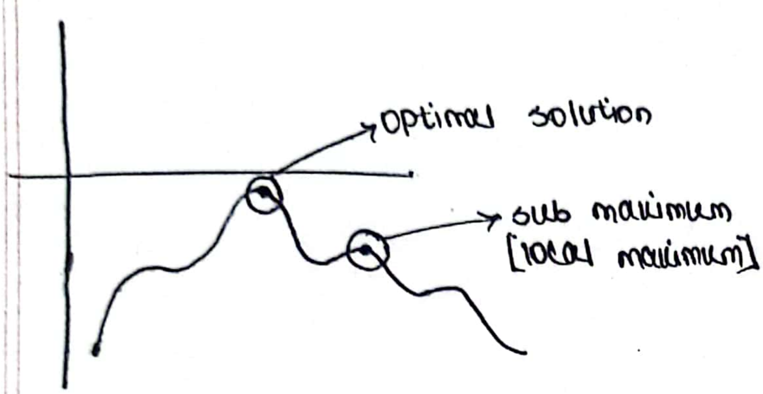
LECTURE
9

THURSDAY DATE: 22/02/24

Local search

Drawbacks

① Local maximum



② Plateaus

* Plane region থেকে start করলে, algo confused হয়ে যায় কোন way তে move করবে।

③ Ridges

* অনেক গুলো ছোট ছোট maxima থাকলে, loop এ পরে

Remedy:

① Random Restart

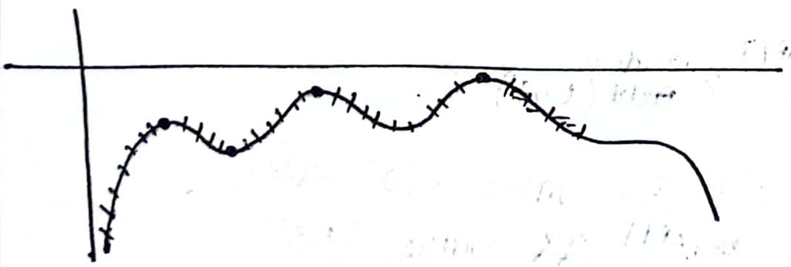
Restart "n" number of times. Using this, there's a chance of avoiding plateaus and ridges.

② Problem Reformulation

Underlying high shape and structure shape can be changed if hypothesis or function are changed.

③ Simulated

* Downward যাওয়ার tendency থাকলেও সে যাবে, starting এ অনেক বার downward যাওয়ার chance থাকে even if peak is reached.



* একটা peak এ থাকে না, it moves forward until a maximum is obtained.

* Temp value থাকে, Initially, flexibility of the simulated annealing is the highest so temp variable is the highest, so at the algo accepts downward movement.

Temp variable এর initial value বড় থাকবে and each peak approach করার পরে বড় Temp variable থেকে value কম-কম হবে তা problem domain decide করবে।

$C = C_{init}$ → current state

For $T = T_{max}$ to T_{min}

$E_c = E(C)$

$N = Next(C)$ → C এর child

$E_N = E(N)$

$\Delta E = E(N) - E(C)$ → if $E(N) > E(C)$, $\Delta E = +ve$

∴ next step is better than current step.

∴ ~~next~~ current step is now considered to be the next step

$C = N$

else if $e^{\Delta E/T} > random(0,1)$

→ T এর value বড় হবে, $e^{\Delta E/T}$ এর value ও বড় হবে

$C = N$

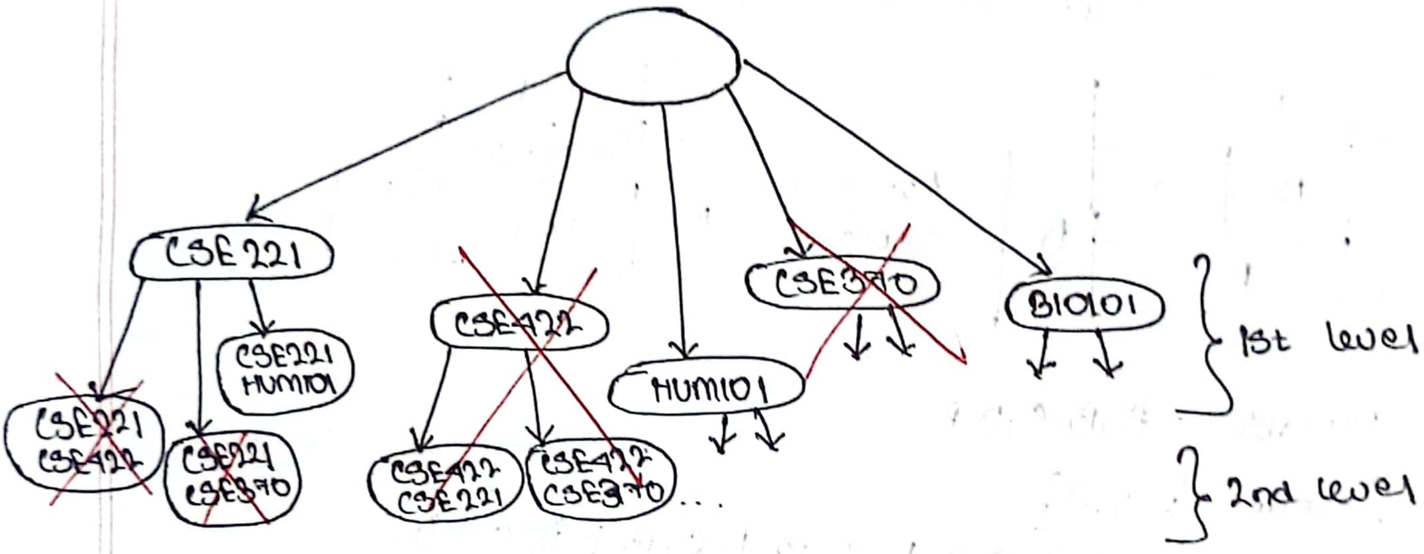
* Temperature এর value বড় হলে, cliff এর ভিতর ও enter করার chance বেশি থাকে।
* Temperature এর value low হলে, cliff এর ভিতর enter করা না

LECTURE 10

SATURDAY DATE: 24/02/24

CONSTRAINT SATISFACTION PROBLEM (CSP)

* এখানে problem এ constraint add করলে space state এর ছোট করে আনা যায়।
* constraint add করলে available options কমে যায়



constraint :- CSE221 is the prereq of CSE422 & CSE370. So, branch containing CSE422 & CSE370 are excluded.

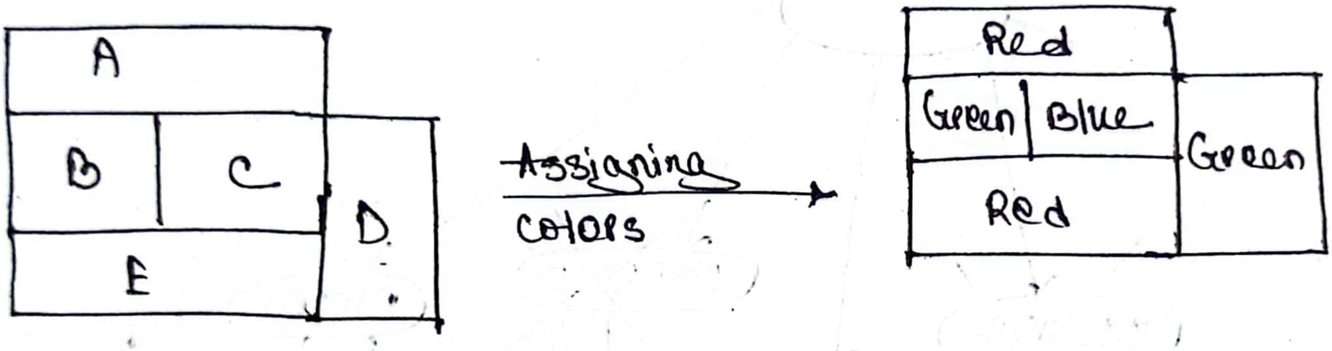
Introducing a constraint removes some part of the space state making it smaller and easier to find goal.

Variable — Time slots
Domain — Course section - - -→ set of values (যেখানে values assign করি)
constraints — No clashes, prereq completion
Goal — Completed routine.

MAP COLORING PROBLEM

→ map দেওয়া থাকবে, zone দেওয়া থাকবে, প্রত্যেক zone কে color করা যাবে using constraint

constraint: ~~color~~ neighbouring zone same color হবে না



Variable : A, B, C, D, E

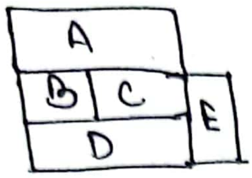
Domain: $\{R, G, B\}$

Constraint: $A \neq B$, $A \neq C$, $B \neq C$, $B \neq D$, $C \neq D$, $E \neq C$, $E \neq D$

Goal: One or multiple

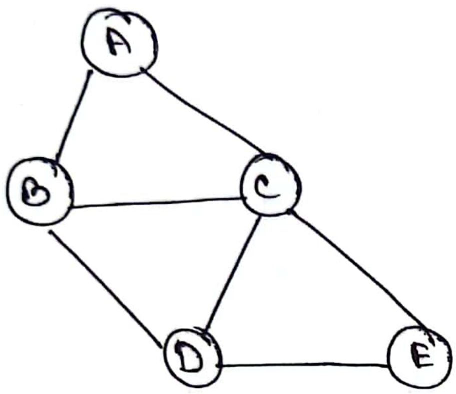
Back track করবে and values assign করবে। ~~the~~ যা

Constraint Graph



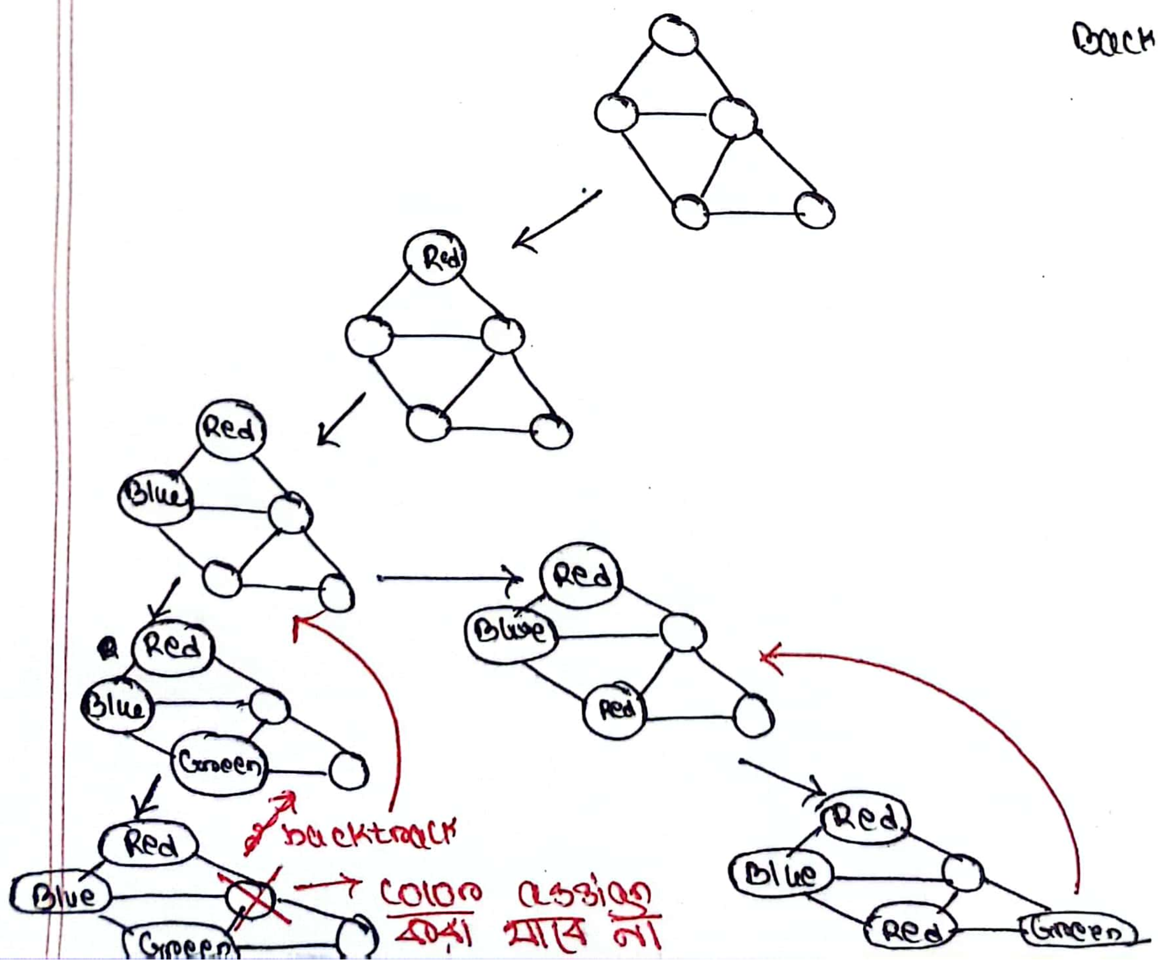
variable → node

constraint → edge



given constraint graph

Backtracking search



*General search গুলো বেশি efficient না

~~* কোনো~~

VARIABLE ORDERING → used to reduce backtracking

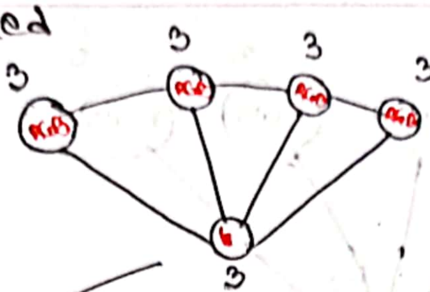
- Most constraint variable

- most constraining variable

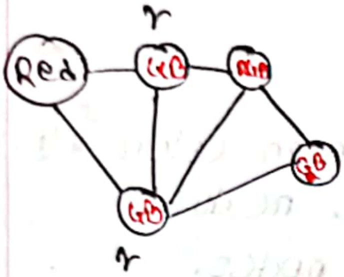
While picking up value, the least constraining value needs to be picked up.

Value = Least constraining value

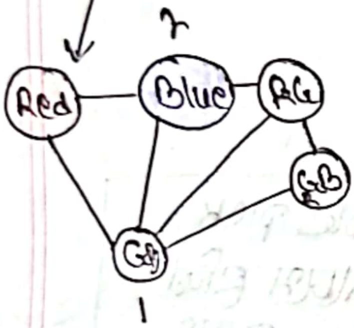
Most Constrained variable



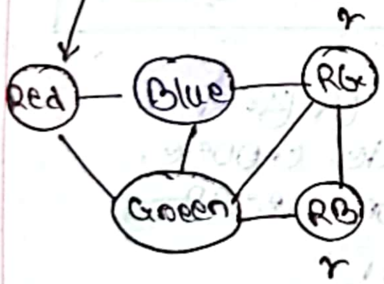
→ Domain size = 3
Tie between all variables, so selected randomly



Domain size of neighbouring nodes = 2



যে variable এর domain এ সবচেয়ে কম color আছে, তাকে আগে hit করা লাগবে (color দেওয়া লাগবে)



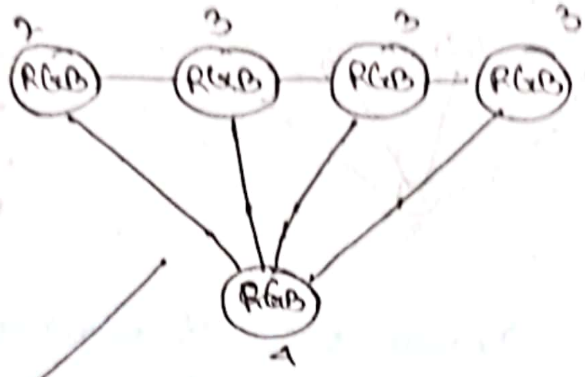
— X — X — X — X — X —

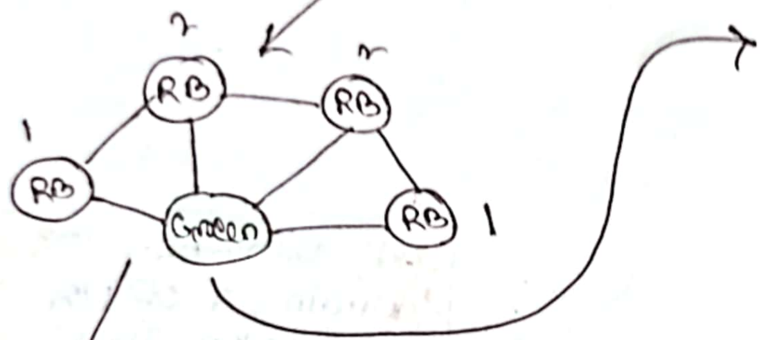
যে variable টা সবচেয়ে বেশি constraint reduce করে using constraint is the most constraining variable.

~~[illegible]~~

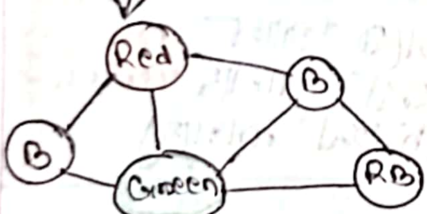
Domain এ tie থাকলে, most constraining variable choose করব।

Most Constraining Variable

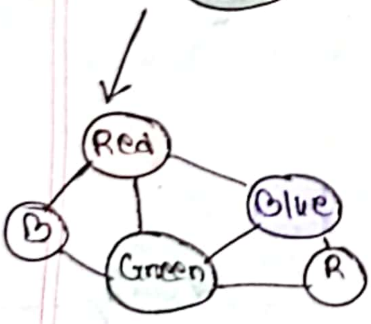




এখানে Green color দিলে, এই node reduce 4 other nodes.



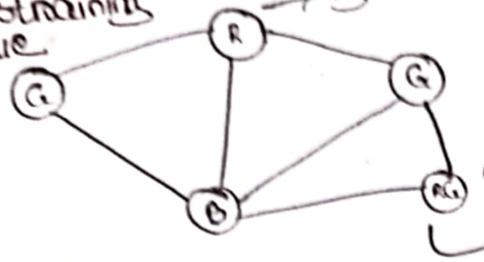
এমন variable pick করব, যে সবচেয়ে বেশি domain reduce করবে



Tie হলে যে কোন একটা node choose করে color assign করে দিব।

* Using most constraining variable, we can reduce backtracking

Least constraining value



→ given

↗ if we place Green here, it reduces 1 variable.

↪ if we place Red here, it reduces 0 variable.

→ least constraint reduce করে

↓

we place R here

① Check the domain.

ⓐ If there is a ~~hire~~ large value of domain, then use most constrained variable.

ⓑ If there is Tie, then most constraining variable is used.

---

most constrained variable {অনেক domain constrained variable আছে}
(1st priority)

↓ multiple tie হলে

most constraining variable / Degree Heuristic
(2nd variable)

↓ then least value assign করা হয়

least constraining value

---

* when actually we implement this, we use a combination of all these approaches.

(FINAL)
LECTURE
11

SATURDAY DATE: 02/03/24

CONSTRAINT SATISFACTION PROBLEM

Variable Ordering:

(i) most constrained variable
→ prioritised at first

(ii) most constraining variable
→ tie breaker এর time এ use করি

Value ordering:

Least constraining value
→ multiple value থেকে select করা লাগবে randomly

*Backtracking কে efficient করে

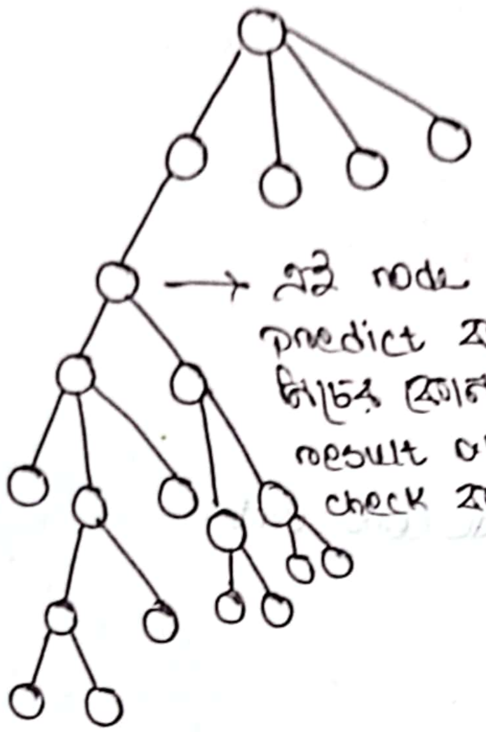
Node consistency checking
→ Forward checker

Arc consistency checking
→ stronger version of node consistency checking

next depth এ determine করতে পারি।

Constraint Propagation
- Node — 1 step
- Arc — multi steps

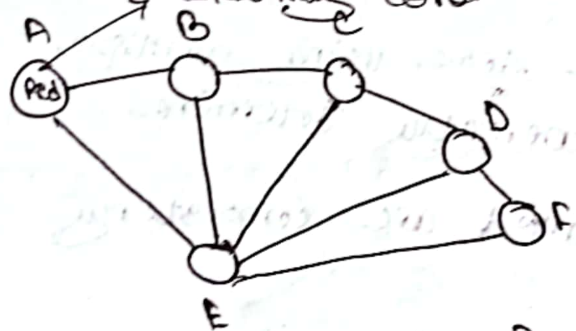
CamScanner



এই node থেকে predict করতে পারি নিচের কোন node এ desired result আছে কি না check করে।

Node consistency checking

randomly color বসায় দিলাম



A	B	C	D	E	F
RGB	RGB	RGB	RGB	RGB	RGB

→ কোনো node এ color assign করিনি

A তে Red দিলাম, B and E থেকে Red এর option চলে যাবে as B and C are neighbours of A. A বাদে আর কারো color assign করিনি

Red	GB	GB	RGB	RGB	RGB

C তে Green color দিলাম

Red	GB	GREEN	RB	B	RGB

Red	GB	Green	RB	Blue	RGB

A	B	C	D	E	F
Red	?	Green	Red	Blue	RG

→ here, graph is **NOT** node consistent.

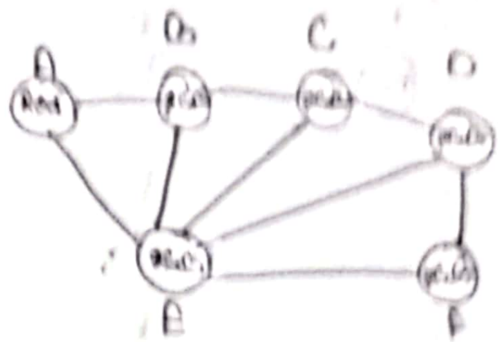
→ domain is empty

* B এর color change হওয়া পর্যন্ত wait করি না,

This saves time.

forward checking ~~an~~ can be done using multiple steps which is better to correctly determine the empty domain. This is called arc consistency checking.

Arc consistency checking



A	B	C	D	E	F
Red	BG	RG / RGB	RGB	GB	RGB

A	B	C	D	E	F
Red	GB	RGB	RB	B	Green

According to forward checking, the tree is not consistent.

→ domain check করছি

~~⇒ remaining value গুলোর জন্য~~

→ remaining value দিয়ে use করে domain reduce করা যাবে।

A	B	C	D	E	F
Red	G~~B~~	RG~~B~~	R~~B~~	Blue	Green

↳ কোন সময় color assign করলেই green cross করা লাগবে

A	B	C	D	E	F
Red	Green	~~Red~~	Red	Blue	Green

↳ empty node.

* Empty node হলে সেখানে stop করব।

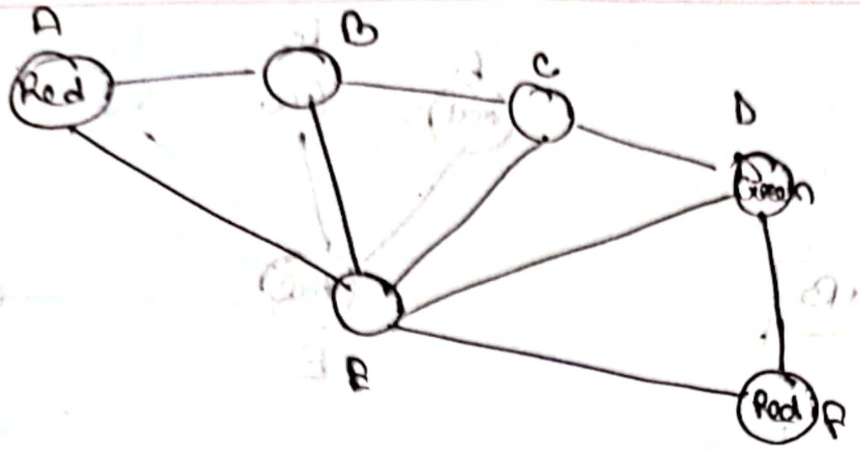
∴ It's not arc consistent.

Step 1: যেসব variable এ one & only color আছে

Step 2: ওইটার basis এ সেই color এর adjacent গুলো color টা বাদ দিব

Step 3: adjacent যে গুলো বাদ দিলাম, তাদের আবার করবো one color available কি না

Repeat upto we reach an empty node
↳ no domain



Arc consistent or not?

Red	G/B	R/B	Green	B	Red

In this particular stage, no domain can be reduced against any variable. So, we stop right here.

∴ We can conclude that it is arc consistent.

* If any node gets reduced to an empty domain, then it is not ~~an~~ arc consistent. otherwise, it is arc consistent.

Arc Consistency Simulation

$A = \{1,2,3\}$
$B = \{1,2,3\}$
$C = \{1,2\}$

$\left.\begin{array}{l} A = \{1,2,3\} \\ B = \{1,2,3\} \\ C = \{1,2\} \end{array}\right\}$ variable

$\left.\begin{array}{l} A < B \\ B > C \\ C = B \\ C = A \end{array}\right\}$ constraint

Queue → queue তে bi-directional constraint রাখিনি, so একটা রাখি।

Queue	Constraint
~~$A < B$~~	$A < B$
~~$B > C$~~	$B > C$
~~$C = B$~~	$C = B$
~~$C = A$~~	$C = A$
$A < B$ → re inserted	
$B > C$ → re inserted	

$A = \{1, 2, \cancel{3}\}$ → $A \nless B$

$B = \{1, \cancel{2}, \cancel{3}\}$

$C = \{1, \cancel{2}\}$

* value এর against এ variable reduce হচ্ছে কি না। যদি reduce হয়, এই ~~value~~ variable যদি constraint এ থাকে, and queue তে নাই, সেই constraint queue তে যাবে।

$A = \{1, 2, 3\}$

$B = \{1, 2, 3\}$

$C = \{1, 2\}$

Queue	Constraint → bi-directional
~~A < B~~	A < B
~~B > A~~	B > C
~~B > C~~	C = B
~~C < B~~	
~~C = B~~	
B = C	
~~A < B~~ → reinserted	
~~B > C~~	
C = B	

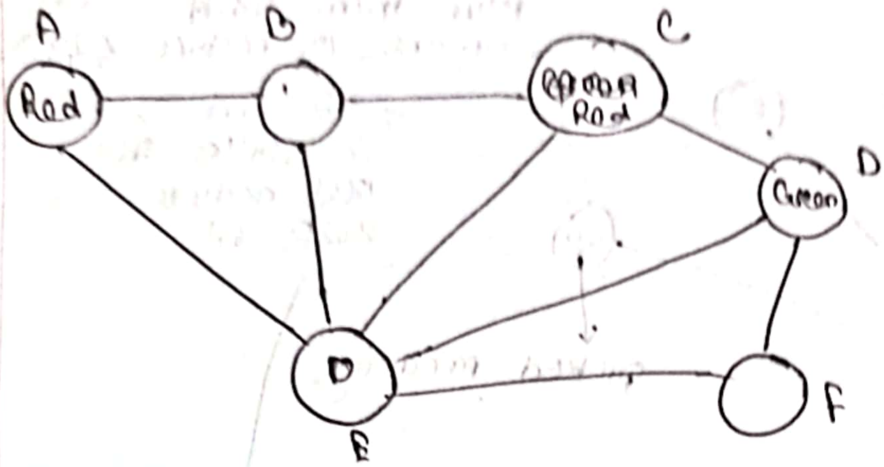
$A = \{1, \cancel{2}, \cancel{3}\}$

$B = \{\cancel{1}, \cancel{2}, \cancel{3}\}$ ⟶ empty domain

$C = \{\cancel{1}, 2\}$

∴ Not arc consistent.

যে variable reduce হচ্ছে, তাকে constraint এর RHS খুঁজব, খুঁজে পেলে দেখব ওই constraint Queue তে আছে কি না, Queue তে না থাকলে, constraint re-insert করি।



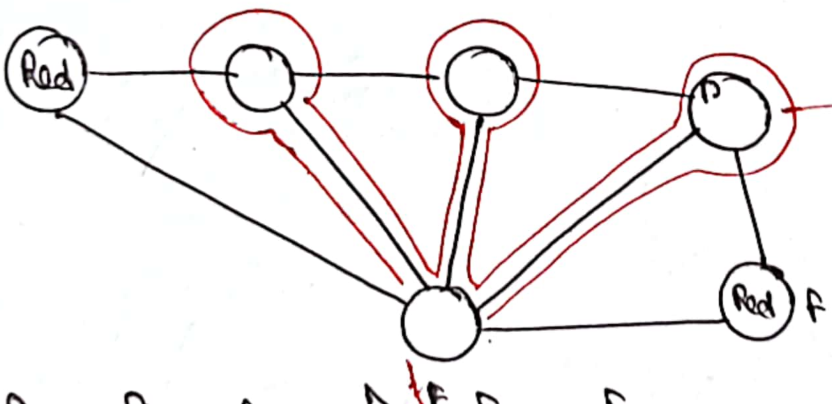
A	B	C	D	E	F
Red		Red	Green		

unassigned

most constrained variable
- least domain আছে

	(2)			(1)	(2)
Red	BG	Red	Green	Blue	RB

unassigned



these 3 can be reduced by E.

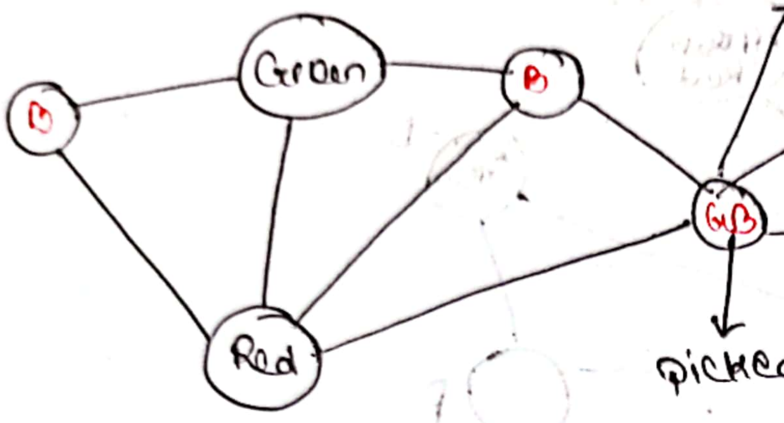
A	B	C	D	E	F	
	(2)	~~(3)~~	(3)	(3)		
Red	BG	RGB	BG	BG	Red	~~Red~~

unassigned

no. of reduction is the highest, so most constraining variable.

Blue দিলে একটা variable কে reduce করবে

এই node এ G বসালে একটা কেউ reduce করবে না



picked randomly

$G \rightarrow 0$

$B \rightarrow 1$

least constraining value

— least no. of reduction, so Green can be placed here

FINAL

LECTURE 12

THURSDAY DATE: 21/03/24

PROBABILITY THEORY

A → event

P(A) → 0 ~ 1

Lower value → lower likelihood
Higher value → higher likelihood

Discrete random variable

weather = {sunny, rainy, foggy, cloudy} → finite amount of values.
.6 .15 .05 .02

→ total হওয়া লাগবে 1 এমনি,
otherwise কোন value missing আছে

umbrella = {Yes, No} | {Yes, Yes'}
0.2 0.8

→ marginal probability of sunny

P(sunny) = 0.6

→ marginal probability of Yes

P(~~Yes~~ Umbrella = Yes) = 0.2

Joint → 2 টা independent event এক সাথে occur হওয়ার probability

→ joint probability

P(rainy and Yes) = ?
or
$\wedge$
or
$\cap$

P(sunny $\wedge$ No) = ?

Conditional probability — given scenario তে multi event হওয়ার probability বের।

→ Y conditional probability

P(Umbrella = Yes | Rain)

→ given event

→ যেটার probability calculate করতে চাই

$$\boxed{P(a|b) = \frac{P(a \wedge b)}{P(b)}}$$

'OR' probability → 2 টা independent event এর যে কোনো একটা হবে

P(rainy OR Yes)

$\cup$

$\vee$

Joint distribution Table — সব গুলো joint combination এর tabular representation প্রকাশ করে

— joint combination এর associate value থাকবে।

	sunny	rainy	foggy	cloudy
Yes	P(sunny ∧ Yes)	P(rainy ∧ Yes)	P(foggy ∧ Yes)	P(cloudy ∧ Yes)
No	P(sunny ∧ No)	P(rainy ∧ No)	P(foggy ∧ No)	P(cloudy ∧ No)

Burglary = {Burg, ¬Burg}

Alarm = {Alarm, ¬Alarm}

	Burglary	¬Burglary
Alarm	P(Burglary ∧ Alarm) 0.05	P(¬Burglary ∧ Alarm) 0.2
¬Alarm	P(Burglary ∧ ¬Alarm) 0.15	P(¬Burglary ∧ ¬Alarm) 0.6

$P(Alarm \wedge \neg Burglary) = 0.2$

$P(\neg Alarm \wedge \neg Burglary) = 0.6$

$P(\neg Burglary) = 0.2 + 0.6 = 0.8$

$P(\neg Alarm) = 0.15 + 0.60 = \cancel{0.\ldots}\ 0.75$

$P(Alarm) = 0.05 + 0.20 = 0.25$

P(Alarm ∧ ¬Burglary)

P(Alarm ∧ Burglary)

কোন একটা event এর marginal probability বের করতে চাইলে, তার সাথে অন্য event থাকলে, ~~and~~ joint probability এর যোগ লাগবে।

~~P(A)~~

Calculate the probability of burglary taking place <u>while</u> an alarm is ringing → (conditional)

$$P(Burglary \mid Alarm) = \frac{P(Burglary \wedge Alarm)}{P(Alarm)} = \frac{0.05}{0.05 + 0.2} = \frac{0.05}{0.25}$$

$t = \{t, \neg t\}$

$c = \{c, \neg c\}$

$x = \{x, \neg x\}$

	t		$\neg t$	
	x	$\neg x$	x	$\neg x$
c	$P(c \wedge x \wedge t)$ 0.102	$P($ 0.24		
$\neg c$	$\boxed{0.086}$	$\boxed{A}$	$P(\neg t \wedge x \wedge \neg c)$	

$\neg t$ ~~এর~~ এই column ~~row~~ তে আছে,

x এই column ~~row~~ তে আছে

$\neg c$ এই ~~row~~ তে আছে

Given, $P(\neg t) = 0.5$,

calculate ~~$P(t)$~~ $P(\neg c \wedge t)$

$P(A) = 1 - (0.102 + 0.24 + 0.086 + 0.5)$

$P(\neg c \wedge t) = 0.086 + \cancel{P(A)}\, A$

	t		$\neg t$	
	x	$\neg x$	x	$\neg x$
c	0.108	0.012	0.072	0.001
$\neg c$	0.016	0.064	0.144	0.576

$P(\neg t \wedge \neg c \wedge \neg x) = 0.144$

$P(t \wedge \neg c \wedge \neg x) = 0.064$

$P(c \wedge \neg x) = 0.012 + 0.001 = 0.013$

$P(\underset{a}{t} \mid \underset{b}{\neg c \wedge \neg x}) = \dfrac{P(t \wedge \neg c \wedge \neg x)}{P(\neg c \wedge \neg x)}$

$= \dfrac{0.064}{0.064 + 0.576}$

LECTURE 13

SATURDAY DATE: 23/03/24

P(A or B)

	A	¬A
B	0.6	0.15
¬B	0.2	0.05

P(A or B) = 0.6 + 0.2 + 0.15 = 0.95

P(¬A or B) = 0.6 + 0.15 + 0.05 = 0.8

formula for "Or Probability"

P(A or B) = P(A) + P(B) − P(A ∧ B)

= (0.6 + 0.2) + (0.6 + 0.15) − 0.6

= 0.95

Independence

$\hookrightarrow$ multiple events can occur simultaneously, but their outcome does not affect each other.

$$A \xrightarrow{ind} B$$

$$P(A \wedge B) = P(A) * P(B)$$

$\left.\begin{array}{l} P(H) = 0.5 \\ P(T) = 0.5 \end{array}\right\}$ probability of H and T are independent of each other

2 coins are tossed :-

$P(H \wedge H) = 0.25$ ~~→ P(H~~

$P(H \wedge T) = 0.25 \longrightarrow P(H \wedge T) = P(H) * P(T)$

$= 0.5 * 0.5$

$= 0.25$

$P(T \wedge H) = 0.25$

$P(T \wedge T) = 0.25$

$+ \overline{}$

1

$\hookrightarrow$ coin is fair

	smart		¬smart	
	study	¬study	study	¬study
prepared	0.432	0.16	0.084	0.008
¬prepared	0.48	0.16	0.36	0.072

*Is smart independent of not study?

$$\boxed{P(A \wedge B) = P(A) * P(B)}$$

LHS and RHS needs to be equal to proof A and B to be independent of each other.

$P(\text{smart} \wedge \neg \text{study}) = P(\text{smart}) * P(\neg \text{study})$

LHS, $P(\text{smart} \wedge \neg \text{study}) = 0.16 + 0.16 = 0.32$

RHS, $P(\text{smart}) * P(\neg \text{study})$

$= (0.432 + 0.16 + 0.048 + 0.16)$

$* (0.16 + 0.16 + 0.008 + 0.72)$

$= 0.8 * 0.4$

$= 0.32$

$\therefore$ Smart & ¬study are independent

* Is smart independent of not prepared?

P(smart ^ ¬prepared) =

LHS, $P(\text{smart} \wedge \neg\text{prepared}) = 0.084 + 0.16 = \boxed{0.64}$

RHS, $P(\text{smart}) * P(\neg\text{prepared}) = \cancel{0.76}(0.432 + 0.16 + 0.084 + 0.008) * (0.048 + 0.16 + 0.036 + 0.072)$

$= 0.8 * \cancel{0.316} \; 0.316$

$= \boxed{0.2528}$

$0.64 \neq 0.2528$

Smart is not independent of not prepared.

The coin might be biased → condition introduce করা হয়েছে

→ ৪০ পার head এর outcome বেশি হয়, তাহলে tail পাওয়ার probability বাড়ে।

→ কোন দিকে biased সেটা বলে দেওয়া হয়নি

→ head & tail are not independent in this scenario.

→ They are dependent

— x — x —

→ কোন দিকে coin bias সেই info দিলে, তাহলে আমরা জানি যে কার উপর bias

→ independency থাকে

→ $P(H) = 0.8$
$P(T) = 0.2$

→ This is conditional independence

— x — x — x —

Conditional Independence

$P(A \cap B | C) = P(A|C) * P(B|C)$ → প্রত্যেক outcome এর আগে condition include করে দিই

To proof conditional independence, this formula is used.

↳ LHS = RHS ✓

* Is smart conditional independent of study, given not prepared?

$P(smart \wedge study \mid \neg prepared) = P(smart \mid \neg prepared) * P(study \mid \neg prepared)$

L.H.S. $= P(smart \wedge study \mid \neg prepared)$

$= \dfrac{P(smart \wedge study \wedge \neg prepared)}{P(\neg prepared)}$

$= \dfrac{0.048}{0.316}$

$= 0.01519$

↳ Take upto 3 dp (if not mentioned in Q)

$P(smart \mid \neg prepared) * P(study \mid \neg prepared)$

$= \dfrac{P(smart \wedge \neg prepared)}{P(\neg prepared)} * \dfrac{P(study \wedge \neg prepared)}{P(\neg prepared)}$

$= \dfrac{0.048 + 0.16}{0.316} * \dfrac{0.048 + 0.036}{0.316}$

$= 0.658 * 0.266$

$= 0.1750$

Smart is not conditionally independent of study, given not prepared.

LECTURE 14

THURSDAY DATE: 28/03/2024

BAYES THEOREM

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

$P(A \wedge B) = P(A|B) * P(B)$ —— (I)

$$P(B|A) = \frac{P(B \wedge A)}{P(A)}$$

$P(B \wedge A) = P(B|A) * P(A)$ —— (II)

* $P(A \wedge B)$ and $P(B \wedge A)$ represent the same thing.

$P(A \wedge B) = P(B \wedge A)$

$P(A|B) * P(B) = P(B|A) * P(A)$

$$\boxed{P(A|B) = \frac{P(B|A) * P(A)}{P(B)}} \longrightarrow \text{Bayes Theorem}$$

09 – PROBABILITY [Slide - 10]

Guilty or Not?

A person is put in front of a judge. The jury finds the defendant guilty in 98% of the cases in which the defendant has committed a crime, and it finds the defendant not guilty in only 97% of the cases in which the defendant has not committed a crime.

Furthermore, only 0.008 of the entire population has committed a crime.

If a random person is found guilty by the jury, what's more likely; criminal or not?

1st case:-
given → commited a crime

if commited a crime → criminal
if jury finds guilty → guilty

$P(\text{guilty} \mid \text{criminal}) = 0.98$

$\boxed{P(A|B) + P(\neg A|B) = 1}$

$P(\neg A|B) = 1 - P(A|B)$

$P(\text{not guilty} \mid \text{not criminal}) = 0.97$

$$P(\text{not guilty} \mid \text{criminal}) = 1 - P(\text{guilty} \mid \text{criminal})$$
$$= 1 - 0.98$$
$$= 0.02$$

$$P(\text{guilty} \mid \text{not criminal}) = 1 - P(\text{guilty} \mid \text{not criminal})$$
$$= 1 - 0.97$$
$$= 0.03$$

$P(\text{criminal}) = 0.008$ $\boxed{P(A) + P(\neg A) = 1}$

$P(\text{not criminal}) = 1 - P(\text{criminal})$
$= 1 - 0.008$
$= 0.992$

*Only marginal and conditional probabilities are given, so Bayes' Theorem is used.

$P(\text{criminal} \mid \text{guilty}) = ?$

$P(\text{not criminal} \mid \text{guilty}) = ?$

$$P(\overset{A}{\text{criminal}} \mid \overset{B}{\text{guilty}}) = \frac{P(\text{guilty} \mid \text{criminal}) * P(\text{criminal})}{\cancel{P(\text{guilty})}}$$

$= 0.98 * 0.008$
$= 0.00784$

$$P(\text{not criminal} \mid \text{guilty}) = \frac{P(\text{guilty} \mid \text{not criminal}) * P(\text{criminal})}{\cancel{P(\text{guilty})}}$$

$= 0.03 * 0.992$
$= 0.02976$

* If denominator is the same, we can eliminate the denominator to compare between the two conditional probabilities.

$P(\text{criminal} \mid \text{guilty}) < P(\text{not criminal} \mid \text{guilty})$

$0.00784 < 0.02976 \rightarrow$ The person not a criminal given guilty.

*Entire population should not be used to determine commited crime

* If a particular ~~e~~ probability ~~is est~~ for an event is extremely low, then we can use an extremely high ~~case~~ 2 to determine the correct probability accuracy → Prosecutor's Theorem

* If question asks for actual probability, we cannot eliminate the denominator as we did during comparison.

$$P(\text{criminal}|\text{guilty}) = \frac{P(\text{guilty}|\text{criminal}) * P(\text{criminal})}{P(\text{guilty})}$$

$$= \frac{P(\text{guilty}|\text{criminal}) * P(\text{criminal})}{P(\text{guilty} \wedge \text{criminal}) + P(\text{guilty} \wedge \neg \text{criminal})}$$

$$= \frac{P(\text{guilty}|\text{criminal}) * P(\text{criminal})}{P(\text{guilty}|\text{criminal}) * P(\text{criminal}) + P(\text{guilty}|\text{not criminal}) * P(\neg \text{criminal})}$$

$$= \frac{0.98 * 0.008}{(0.98 * 0.008) + (0.03 * 0.992)}$$

$= 0.2085$

$\approx 20.85\%$

	A	¬A
B	0.6	0.2
¬B	0.15	0.05

$P(A) = 0.6 + 0.15$
$= 0.75$
$P(A \wedge B) + P(A \wedge \neg B)$

$P(A) = P(A \wedge B) + P(A \wedge \neg B)$

$P(A|B) = \frac{P(A \wedge B)}{P(B)}$

$P(A \wedge B) = P(A|B) * P(B)$

$P(\neg \text{criminal}|\text{guilty}) = 1 - P(\text{criminal}|\text{guilty})$
$= 1 - 0.2085$
$= 0.7915$
$= 79.15\%$

LECTURE 15

SATURDAY DATE: 30/03/24

NAIVE BAYES

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \rightarrow \text{Bayes Theorem}$$

OR

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

feature → set of values trying to observe

label (variable)

	temp	play
W	70	~~Yes~~ No
W	60	No
C	35	No
C	23	Yes
W	55	Yes
W	65	No
W	72	No

* If a day is warm, calculate the probability of playing.

- warm days at temp above 55°
  ↳ threshold information is given. which can be used to convert continuous data to discrete data.

* discrete → যেসব variable যা under a specific amount of data থাকে

day is warm → warm rows is isolated from table.

$\rightarrow P(\text{play}|\text{warm}) = \dfrac{1}{5}$ [play = yes for 1 row of warm]

OR

$\rightarrow P(\text{play} = \text{yes}|W) = \dfrac{P(\text{yes} \wedge W)}{P(W)}$

$= \dfrac{1/7}{5/7}$

$1/7$ → 1 row contains yes ^ warm | 7 rows

$5/7$ → 5/7 rows contain warm

* Usually, we have to take a decision considering multiple features.

10 – NAIVE – BAYES

EXAMPLE

Play Tennis : training examples

Att

features				label
Outlook	Temperature	Humidity	Wind	Play Tennis
Overcast	Mild	High	Strong	Yes
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	~~Strong~~ Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	~~Yes~~ No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Type 1 – Calculate whether the player will play tennis or

Type 2 – Calculate the probability of playing tennis given outlook is overcast, Temperature is cool, humidity is high and wind is weak.

P(Yes | outlook ^ Temperature ^ humidity ^ wind)

P(Yes | outlook ^ Temperature ^ humidity ^ wind)

P(No | outlook ^ Temperature ^ humidity ^ wind)

Using Bayes Theorem,

$P(Yes \mid \text{overcast} \wedge \text{cool} \wedge \text{high} \wedge \text{weak})$

$$= \frac{P(\text{overcast} \wedge \text{cool} \wedge \text{high} \wedge \text{weak} \mid Yes)\, P(Yes)}{P(\text{overcast} \wedge \text{cool} \wedge \text{high} \wedge \text{weak})} \Rightarrow \left(\frac{0}{9}\right)\left(\frac{9}{14}\right) = 0$$

$$\frac{P(\text{overcast} \wedge \text{cool} \wedge \text{high} \wedge \text{weak} \mid No)\, P(No)}{P(\text{overcast} \wedge \text{cool} \wedge \text{high} \wedge \text{weak})} \Rightarrow \left(\frac{0}{5}\right)\left(\frac{5}{14}\right) = 0$$

* We are doing comparison, so same denominator is eliminated from both probabilities.

For this scenario with multiple feature, we are getting 0 for both Yes and No.

Joint of all multiple feature is rigid.

For conditional independence :—

$P(A \wedge B \mid C) = P(A \mid C) * P(B \mid C)$

$P(A \wedge B \wedge D \wedge E \mid C) = P(A \mid C) * P(B \mid C) * P(D \mid C) * P(E \mid C)$

- → every element is considered for given condition
- → we can force this upon joint of multiple feature
- → The forcing over of conditional probability onto the joint probability of multiple features is Naive's Theorem

$P(overcast|Yes)P(cool|Yes)P(high|Yes)P(weak|Yes)P(Yes)$

$= \left(\frac{4}{9}\right)\left(\frac{3}{9}\right)\left(\frac{3}{9}\right)\left(\frac{6}{9}\right)\left(\frac{9}{14}\right)$

> 0

$\therefore$ Probability of playing tennis is greater.

$P(overcast|No)P(cool|No)P(high|No)P(weak|No)P(No)$

$= \left(\frac{0}{5}\right)\left(\frac{1}{5}\right)\left(\frac{4}{5}\right)\left(\frac{2}{5}\right)\left(\frac{5}{14}\right)$

$= 0$

Question A হলে calculate করে দেখো। Otherwise লাগবে না

Learning phase → conditional probability of feature given label.

→ P(Sunny | Yes)

Outlook

	Yes	No
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature

	Yes	No
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity

	Yes	No
High	3/9	4/5
Normal	6/9	1/5

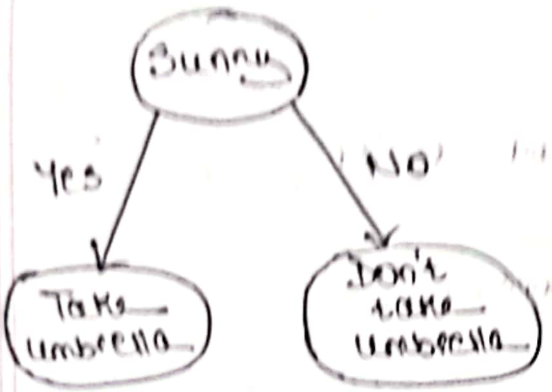
Wind

	Yes	No
Strong	3/9	3/5
Weak	6/9	2/5

LECTURE 16

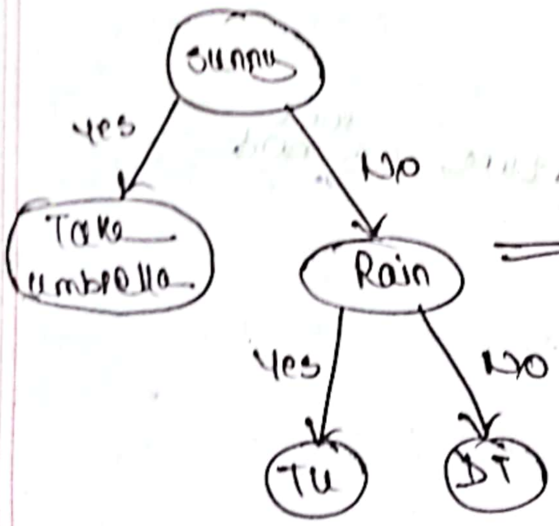
THURSDAY DATE: 13/04/23

Decision Tree



Leaf node → label and non-leaf nodes → features.

Here, there's 1 feature.



Features

Sunny	Rain	Decision
Yes	Yes	TU
Yes	No	TU
No	Yes	TU
No	No	DT.

Tabular Representation of Decision Tree

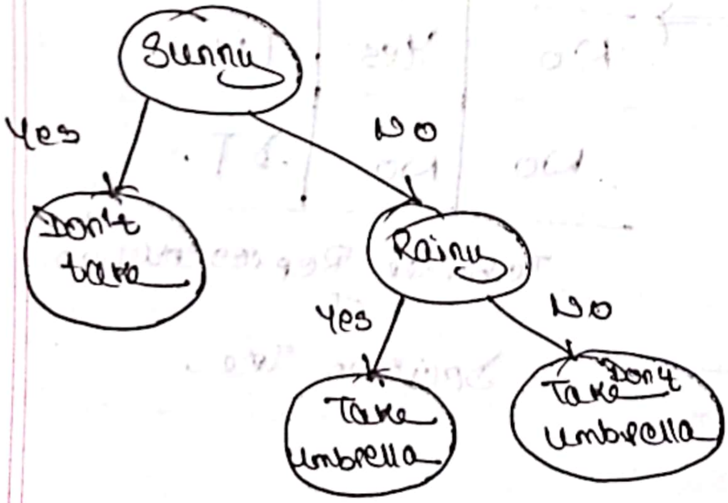
Here, 2 features are involved.

short & precise decision tree needs to be formed from a given ~~decision tree~~ table.

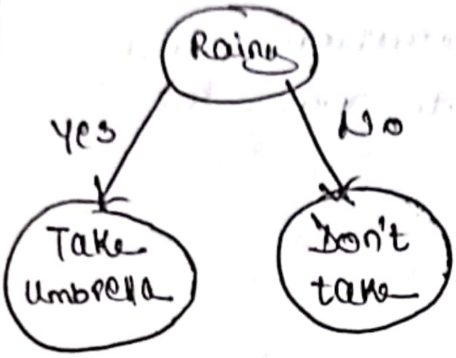
- → inference will be faster and easier
- → decision is more concise
- → information density is more.

Sunny	Rainy	Decision
Yes	No	Don't Take
No	Yes	Take Umbrella
No	No	Don't Take
~~No~~		

Randomly choosing one feature as root node :-

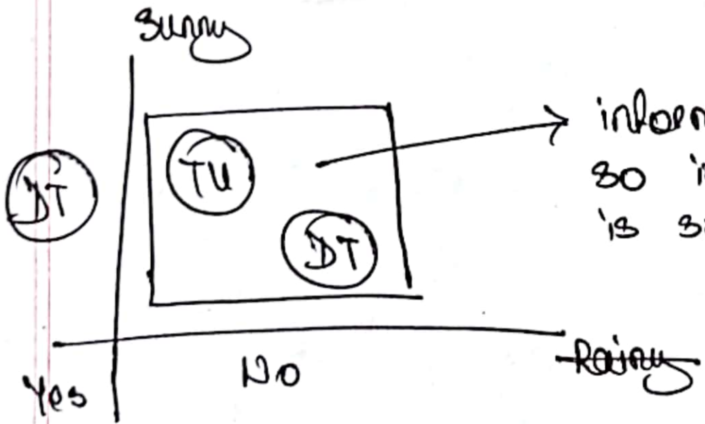


Considering ~~too~~ rainy as root node:-

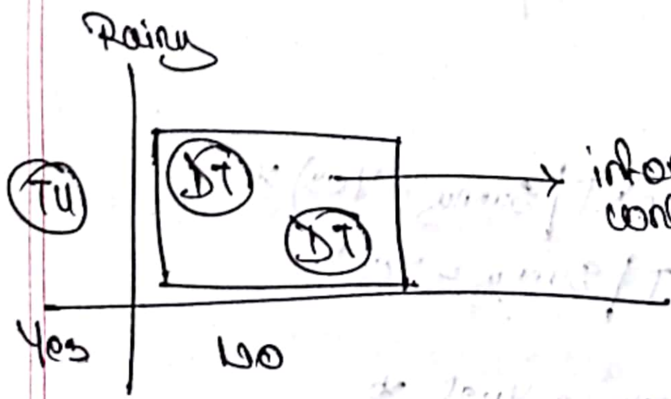


* Using rainy as root node, decision ~~is~~ tree is more concise.

We need to pick up a root node which is ~~not~~ can give short and concise decision tree.



information conflicts, so information density is small



information doesn't conflict.

ID3 — The algorithm which we can use to find the information purity to determine the root node for the decision tree.

ID3

→ entropy calculate করব

→ using this entropy, information gain for a particular feature is calculated.

* যত কম info, তত কম entropy and তত বেশি information gain

eg.

$$\begin{array}{l} E(S=Y) \\ E(S=N) \\ E(\text{Decision}) \end{array} \rightarrow IG(S)$$

~~E(S = Y) =~~

$E(\text{Sunny} = \text{Yes}) = -P(\cancel{\text{Decision}}\,DT \mid \text{Sunny} = \text{Yes}) * \log_2 P(DT \mid \text{Sunny} = \text{Yes})$

$- P(TU \mid \text{Sunny} = \text{Yes}) * \log_2 P(TU \mid \text{Sunny} = \text{Yes})$

$$E(\text{Sunny}=\text{Yes}) = -P(1/1)\log_2(1/1) - P(0/1)\log_2(0/1)$$
$$= 0$$

যত information purity কমবে, তত ~~~~ বেশি entropy.
যত information purity বেশি, তত কমবে entropy.

$$E(\text{Sunny}=\text{No}) = -P(DT \mid \text{Sunny}=\text{No})\log_2 P(DT \mid \text{Sunny}=\text{No})$$
$$- P(DTU \mid \text{Sunny}=\text{No})\log_2 P(DTU \mid \text{Sunny}=\text{No})$$
$$= -P(1/2)\log_2(1/2)$$
$$- P(1/2)\log_2(1/2)$$
$$= 1$$

- → there's a mixture of information
- → information purity কমবে
- → information conflict হবে

$$E(\text{Decision}) = -P(DT) * \log_2 P(DT)$$
$$- P(TU) * \log_2 P(TU)$$
$$= -P(2/3) * \log_2(2/3)$$
$$- P(1/3) * \log_2(1/3)$$
$$= 0.918$$

~~IG(Sunny) = E(Decision)~~

$$IG(Sunny) = E(Decision) - E(Sunny = Yes) * P(Sunny = Yes)$$
$$- E(Sunny = No) * P(Sunny = No)$$

$$= 0.918 - (0)\left(\frac{1}{3}\right) - (1)\left(\frac{2}{3}\right)$$

$$= 0.258$$

যত higher information আছে, তত information gain বেশি হবে।

Repeat the entire process for rainy to get another IG(Rainy) and compare the values of IG(Rainy) and IG(Sunny) to determine the root node between Rainy and Sunny.

$E(\text{Rainy} = \text{Yes})$

$E(\text{Rainy} = \text{NO})$ → $IG(\text{Rainy})$

$E(\text{Decision})$

$$E(\text{Rainy} = \text{Yes}) = -P(DT \mid \text{Rainy} = \text{Yes}) * \log_2 P(DT \mid \text{Rainy} = \text{Yes}) - P(TU \mid \text{Rainy} = \text{Yes}) * \log_2 P(TU \mid \text{Rainy} = \text{Yes})$$

$$= -P(1/1) \log_2 (1/1) - P(1/1) \log_2 (1/1)$$

$$= 0$$

$$P(\text{Rainy} = \text{NO}) = -P(DT \mid \text{Rainy} = \text{NO}) \log_2 P(DT \mid \text{Rainy} = \text{NO})$$

$$- P(TU \mid \text{Rainy} = \text{NO}) \log_2 P(TU \mid \text{Rainy} = \text{NO})$$

$$= -(2/2) \log_2 (2/2) - (0/2) \log_2 (0/2)$$

$$= 0$$

$$IG(\text{Rainy}) = E(\text{Decision}) - P(\text{Rainy} = \text{Yes}) * E(\text{Rainy} = \text{Yes})$$

$$- P(\text{Rainy} = \text{NO}) * E(\text{Rainy} = \text{NO})$$

$$= 0.918 - (1/3)(0) - (2/3)(0)$$

$$= 0.918$$

$IG(\text{Rainy}) > IG(\text{Sunny})$

$\Rightarrow 0.918 > 0.258$

$\therefore$ Rainy can be used as the root node

Rainy

$\swarrow \quad \searrow$

TU $\quad$ BT

Problem from slide :-

$E(\text{Outlook} = \text{Sunny})$

$= -P(\text{Yes} \mid O = S) \log_2 (\text{Yes} \mid O = S)$

$\quad - P(\text{No} \mid O = S) \log_2 (\text{No} \mid O = S)$

$= -(2/5) \log_2 (2/5) - (3/5) \log_2 (3/5)$

$= 0.97$

$E(\text{Outlook} = \cancel{\text{Rainy}}\ \text{Overcast})$

$= -P(\text{Yes} \mid O = O) \log_2 (\text{Yes} \mid O = O)$

$\quad - P(\text{No} \mid O = O) \log_2 (\text{No} \mid O = O)$

$= -(4/4) \log_2 (4/4) - 0$

$= 0$

$E(\text{Outlook} = \text{Rain})$

$= -P(\text{Yes} \mid O = R) \log_2 (\text{Yes} \mid O = R)$

$- P(\text{No} \mid O = R) \log_2 (\text{No} \mid O = R)$

$= -(3/5) \log_2 (3/5) - (2/5) \log_2 (2/5) = 0.97$

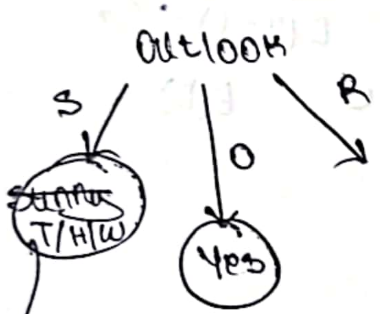
$E(\text{Decision}) = -(9/14) \log_2 (9/14) - (5/14) \log_2 (5/14)$

$= 0.94$

$IG(outlook) = E(Decision) - P(outlook = sunny) \times E(outlook = sunny)$

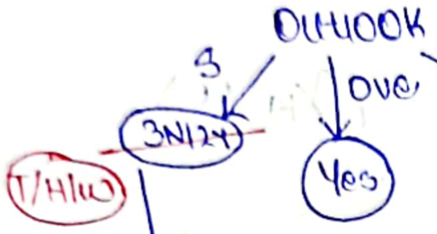
$= 0.94 - (5/14)(0.97) - (5/14)(0.97) - (4/14)(0)$

$= 0.247$



→ এইগুলোর IG বের করবো। তারপরে outlook = sunny rows isolate করবো, E(T), E(H) and E(W) and E(Decision) calculate করবো using the isolated rows and then calculate IG

SATURDAY DATE: 20/04/24



* sunny / rainy এর বাকি DT তৈরি করো। আগে → final DT will be same.

→ no. of volume কমিনা, so No ব্যাপার নিতে পারি. But, there are other features need to be utilised in the tree.

Outlook = sunny এর টা now isolate করব।

$$E(T=Hot) = -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$$

$$E(T=mild) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

$$E(T=cool) = -\frac{0}{1}\log_2\frac{0}{1} = 0$$

$$E(Decision) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.97$$

$E(T=H)$, $E(T=m)$, $E(T=c)$, $E(D)$ → $IG(S)$

~~IG(T) = E 0.97 −~~

$$IG(T) = E(Decision) - E(T=H) * P(T=H) - E(T=m) * P(T=m) - E(T=c) * P(T=c)$$

$$= 0.97 - \frac{2}{5} * 0 - \frac{2}{5} * 1 - \frac{1}{5} * 0$$

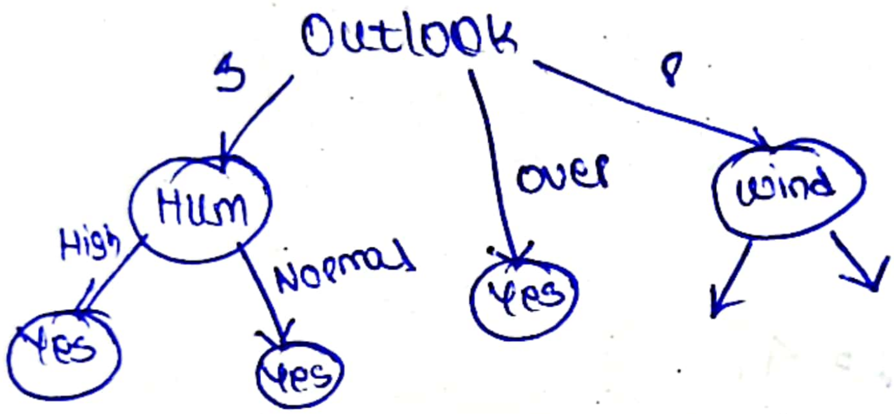
$$= 0.57$$

$E(H=H)$

$E(H=N)$ → $IG(H)$

$E(D)$

$IG(H) > IG(T)$



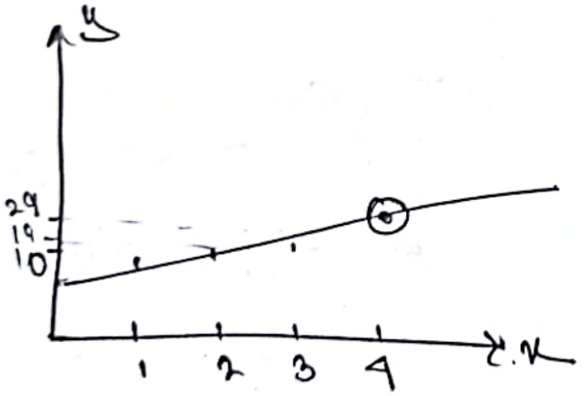
LECTURE 16

SATURDAY DATE: 04/09/21

Regression Analysis

x	y
1	10
2	19
3	29
4	?

* To calculate y, for $x = 4$, we need a function which fits all the above given datapoints.



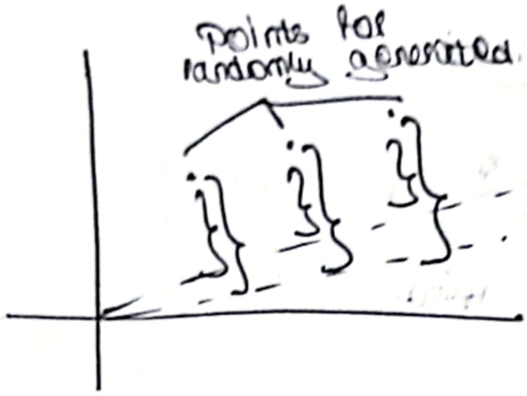
$y = mx + c$

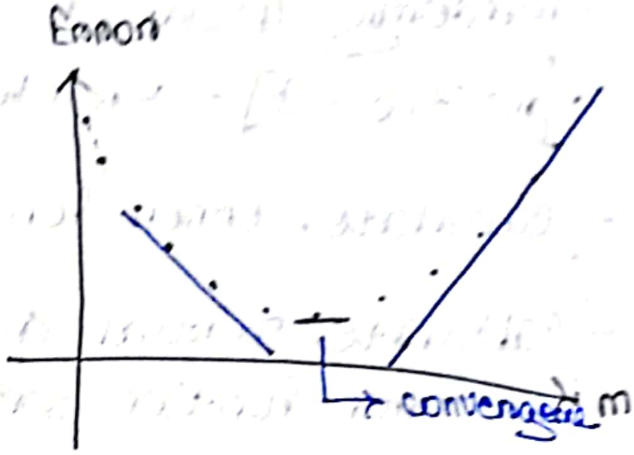
slope — m; intercept — c

if $c = 0$, $y = mx$

find m

* Randomly ~~m~~ value of m is guessed and the correct value of m is determined from all the randomly choosen values.





For this case,

m ↓, error ↑ → which is wrong

m ↑, error ↓ → correct approach

m এর value বেশি বাড়ালে, error অনেক বেশি বৃদ্ধি হয়

slope এর sign দেখে বুঝা যায়, slope কোন দিকে যাবে।

$\boxed{\text{derivative/slope sign determines the direction of slope.}}$

$m_{new} = m_{old} - \text{slope}$

$\boxed{\text{magnitude of slope determines closeness with convergence point.}}$ → m এর value কত দ্রুত update হবে

* complex hypothesis এর জন্য try and error function use করা suitable না

$y = mx + c$

→ c এর partial derivative দিয়ে update করব

→ m এর partial derivative দিয়ে update করব

Steps :—

- hypothesis selection $\Rightarrow y = mx + c$

- randomly assign m and c
  [m = 7, c = 3] → randomly assigned

- calculate error for m = 7, c = 3

- calculate partial derivative ~~with~~ for error function wrt c.
  Update c

- Calculate partial derivative for error function wrt m.
  Update m.

- Re-calculate error with the updated m and c

- Repeat steps until convergence
  ↳ error value is minimised [may not be 0]

Given data points

x	y
1	10
2	19
3	29
4	?

$y = mx + c$

① Hypothesis selection $\Rightarrow y = mx + c$

② Randomly assign m and c

$m = 7$

$c = 3$

③ Calculate error for $m = 7, c = 3$

$$\text{error} = \sum_{n=1}^{3} [y_{true} - y_{pred}]^2$$

$$= (y_{1true} - y_{1pred})^2 + (y_{2true} - y_{2pred})^2 + (y_{3true} - y_{3pred})^2$$

$$= (y_{t1} - (mx_1 + c))^2 + (y_{t2} - (mx_2 + c))^2 + (y_{t3} - (mx_3 + c))^2$$

$$= (10 - (7(1) + 3))^2 + (19 - (7(2) + 3))^2 + (29 - (7(3) + 3))^2$$

error = 29

learning rate বাড়ায় দিব if slow হয়

learning rate কমায় দিব if বেশি fluctuation হয়

---

(4) Calculate partial derivative of error function w.r.t. c.

error = $(y_{t1} - (mx_1 + c))^2 + (y_{t2} - (mx_2 + c))^2 + (y_{t3} - (mx_3 + c))^2$

$$\frac{d(\text{error})}{dc} = (-2)(y_{t1} - (mx_1 + c)) + (-2)(y_{t2} - (mx_2 + c)) + (-2)(y_{t3} - (mx_3 + c))$$

$$= (-2)(10 - (7(1) + 3)) + (-2)(19 - (7(2) + 3)) + (-2)(29 - (7(3) + 3))$$

$$\frac{d(\text{error})}{c} = -14$$

$$C_{new} = C_{old} - (lr)\frac{d(\text{error})}{c}$$

$\to$ learning rate $[10^{-1}$ to $10^{-5}]$

- convergence pt এর কাছাকাছি গেলে lr ~~বাড়ায় দিব~~ কমায় দিব
- otherwise ~~কমায়~~ বাড়ায় দিব

~~learn~~

$\Rightarrow C_{new} = 3 - (0.01)(-14) \Rightarrow C_{new} = 3.14$

(5) Calculate partial derievative of error function wrt m. Update m.

$$\frac{d(error)}{m} = (-2x_1)(y_{t1} - (mx_1 + c)) + (-2x_2)(y_{t2} - (mx_2 + c))$$

updated c এখন use করব না

$$+ (-2x_3)(y_{t3} - (mx_3 + c))$$

$$= (-2(1))(10 - (7(1) + 3)) + (-2(2))(19 - (7(2) + 3))$$

$$+ (-2(3))(29 - (7(3) + 3))$$

$$\frac{d(error)}{m} = -38$$

$$m_{new} = m_{old} - (lr) * \frac{d(error)}{m}$$

$$= 7 - (0.01)(-38)$$

$$m_{new} = 7.38$$

(6) Re-calculate error with updated m and c.

$$error = (10 - (7.38(1) + 3.14))^2 + (19 - (7.38(2) + 3.14))^2$$

$$+ (29 - (7.38(3) + 3.14))^2$$

$$= 15.32$$

initial error = 29

updated error = 15.32 → error এর amount কমেছে

* The way m and c are updated, this is a part of gradient descent.