DFS MMM Depth First Search

1. Specific source node দেয়া থাকবে
2. " " " " না থাকলে যেকোনো একটা node কে source node হিসেবে নিয়ে কাজ করতে হবে।
3. All time directed graph দেয়া থাকবে না।

Directed + Undirected both graphs can be Possible

4. DFS এ একটা node এ প্রথমে start mark এবং আরেকটা finishing mark দিতে হবে।

start / finishing

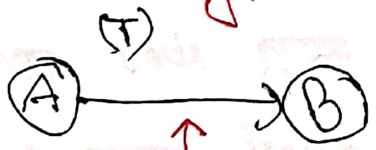
start mark → node টাকে first যখন discover করা হয়।

finish mark → node এর সব processing যখন শেষ হয়ে যাবে, সেটা finishing mark.

5. Edge গুলোকে 4 ধরনের classification করা যায়:

(Directed graph এর ক্ষেত্রে)

1. Tree Edge - New node discover edge



→ এখানে A node যে edge use করে নতুন একটা node কে discover করছে সেটা Tree Edge

2. Forward Edge: (F)

$$\boxed{\text{Ancestor} \xrightarrow[\text{Edge}]{} \text{Descendant}}$$

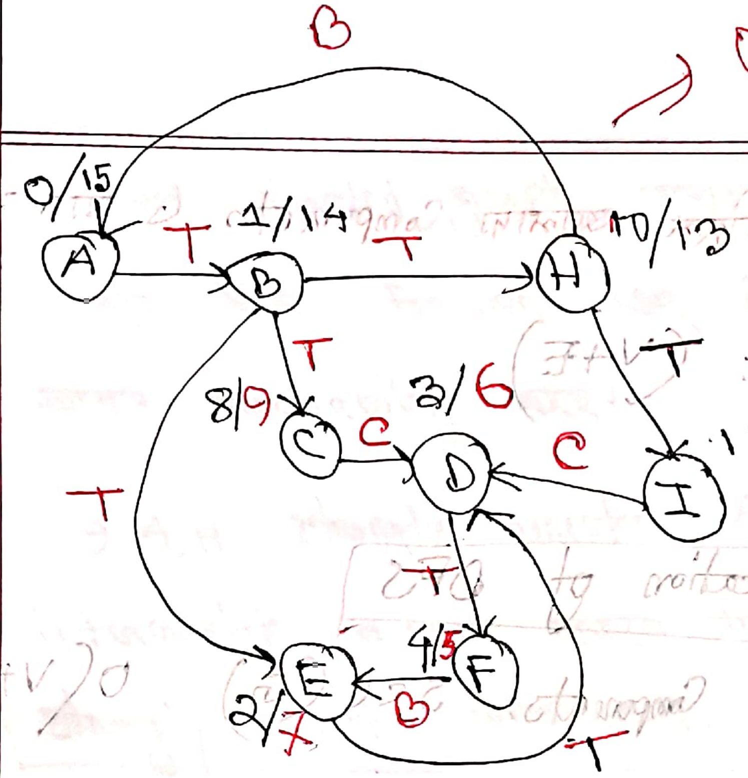
পূর্বপুরুষ " Edge " বংশধর

3. Backward Edge: (B)

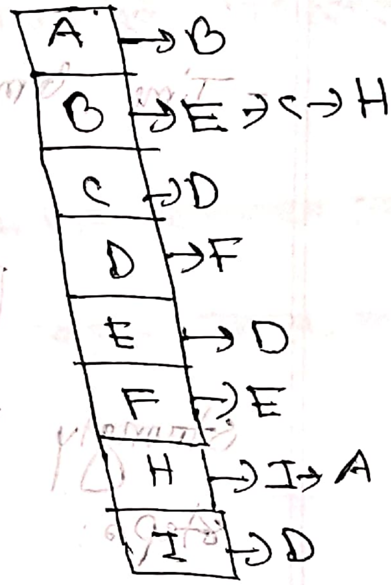
$$\boxed{\text{Descendant} \xrightarrow[\text{Edge}]{} \text{Ancestors}}$$

4. Cross Edge (C): উপরের ৩টা Edge এ classify করতে না পারলে সেটা Cross Edge.

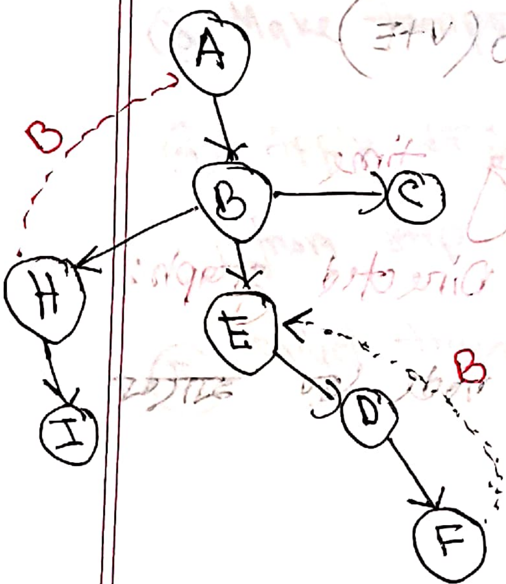
→ O(V+E)



Directed Unweighted



Adjacency List



8 Node

Time: 0–15 = 16

nodes – 1 Tree edge

→ In DFS:

① Parent node শেষ হবে তখনি, যখন তার child সব node গুলো এর finish হয়ে যাবে।

A যেহেতু Parent → A যেহেতু সবার শেষে শেষ হবে।

* Backward edge always produces cycle

- Cross Edge বাদ দিলে আলাদা Components এ ভাগ হবে।

- Time Complexity : (V+E)

$\boxed{\text{Application of DFS}}$

Strongly Connected Components: SCC(G)   O(V+E)

Steps:

1. Apply DFS upon G // O(V+E)
2. Make vertex order of finishing time

Strongly Connected Component for Directed graph:

→ এমন দুইটা node যে দুইটা node এর মাঝে freely move করা যাবে।

A → B → E → D → F

F → E → D → F   So, F → A ✗ Back করা যাচ্ছে

∴ F → A পৌঁছানো সম্ভব না, A, F Strongly connected না।

→ একটা directed graph তখনই strongly connected হতে পারবে, যারা Forward এবং Backward. দুই দিক থেকেই যাদের Reachability আছে।

→ A, H strongly connected. A এবং H এর মাঝখানে যেসব intermediate node গুলো traverse করার সময়, তারাও strongly connected component.

3. Make transpose Graph G^T

4. Apply DFS upon G^T based upon vertex order (from step 2)

5. collect trees of transpose graph.

AKO DFS

→ যে edge গুলো traversal এর জন্য কাজে লাগে, সেগুলো Tree Edge.

BFS + DFS Directed/Undirected both graph এই কাজ করে।

DFS: stack (Recursion এও stack use হয়)

→ Recursion এর ক্ষেত্রে stack use হয়

→ Recursive Algorithm implement করার জন্য already Recursive method লেখা যায়।

→ DFS এ stack avoid করা যায় যদি Recursion use করি

→ Just DFS দিয়ে connected component বের করা যায় না।

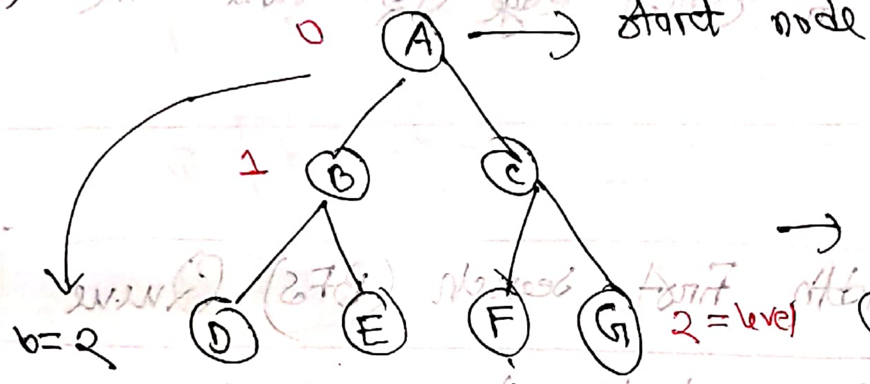
(Using modified DFS SCC বের করতে পারা যায়)

TR Uninformed Search 1

State space: কোনো একটি environment এর all possible configuration কে state space বলে।

→ কোনো state space কে Tree আকারে represent করা possible.

→ প্রত্যেকটা environment state কে একটা node ধরে:



→ state space tree (যদি কোনো loop না থাকে। Loop থাকলে graph হয়ে যায়)

↓ Unweighted tree

G → goal node

→ কোনো একটা particular parent এর under এ কতগুলো child থাকে সেটা Branching factors.

↓ Worst case এর জন্য যে parent এর most child থাকে

b - Branching factor

d - tree কত level পর্যন্ত গেছে, depth of the tree

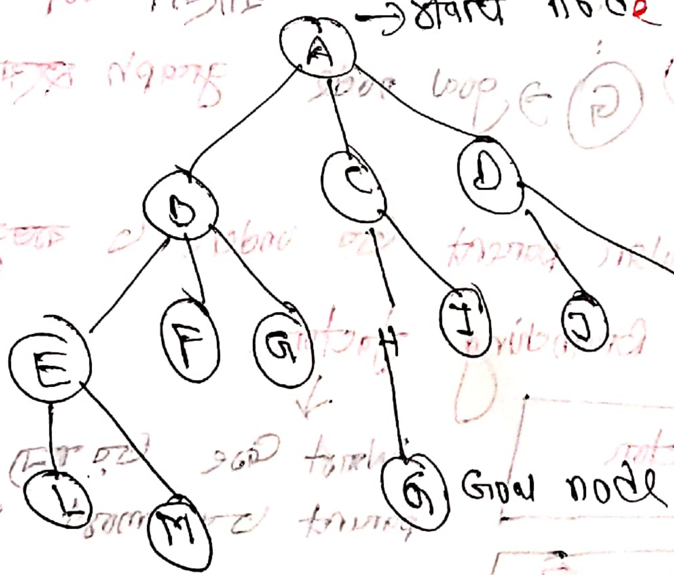
→ Unweighted tree / Graph → BFS, DFS works best

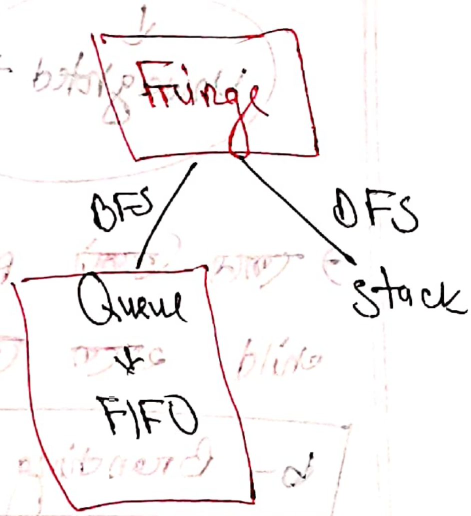
→ যেখানে goal node কোথায় থাকতে পারে সেটার কোনো idea থাকে না → BFS, DFS works best

→ Graph Weighted + ~~মোটামুটি~~ Idea থাকে কোথায় Goal node থাকতে পারে → BFS, DFS appropriate algo না।

→ Unweighted → যে ~~node~~ edge এ same value/ or কোনো edge এই value থাকে না।

Breadth First Search (BFS) Queue





A → start node থেকে traversal start

expand A → A | B C D

| C D E F G |

C | DEFGHI

D | EFGHIJK

E | FGHIJKLM

F | GHIJKLM

G | HIJKLM

H | IJKLMG → ওখানে goal node already push হয়েছে। ওখানে G just visit করা হয়েছে, expand করা হয়নি।

I | JKLMG

J | KLMG

K | LMG

L | MG

M | G

G | → G কে expand

→ searching Algorithm এর general Rule:

যতক্ষন Goal কে expand করা হবে না, ততক্ষন searching Algorithm চালিয়ে যেতে হবে।

How to create a path from start to goal node:

Null	A	A	A	B	B	B	
A	B	C	D	E	F	G	H

↑ parent list

→ কোনো Searching Algorithm এর ৪টা criteria থাকে:

1. Completeness: একটা searching Algorithm সবসময় goal find করার guarantee দিবে কিনা।

→ All possible tree scenario তে goal node exist করলে তা খুঁজে পেলে তা complete algorithm.

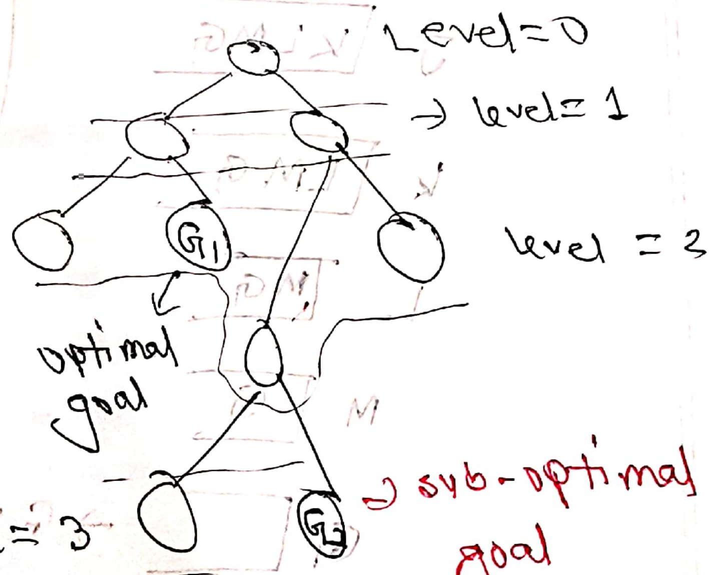
2. Optimal: যদি কোনো searching algorithm যদি multiple goal node এর মধ্যে best goal node টাকে খুঁজে পায়।

3. Time complexity → Worst case scenario

4. Space Complexity → " " "

BFS → 1. Complete if there are finite nodes

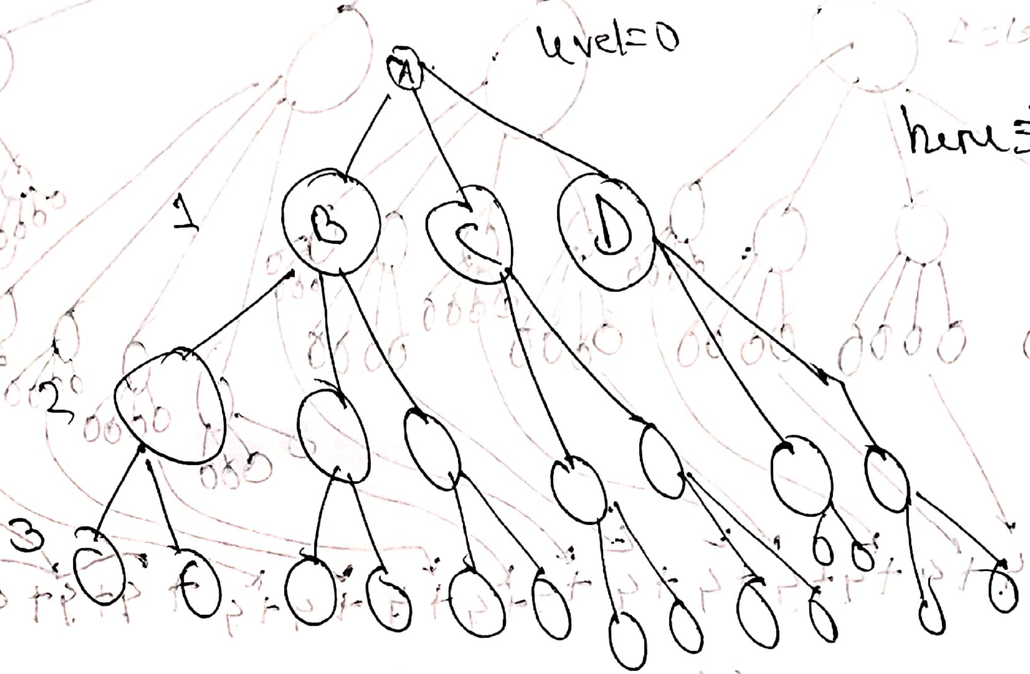
2. Optimal



$O(V+E) \rightarrow$ Algo (limited/finite graph)

3. Time Complexity : $O(b^d)$ / $O(b^{d+1}) \rightarrow$ AI

goal visit goal expand



here ⇒ $b=3$

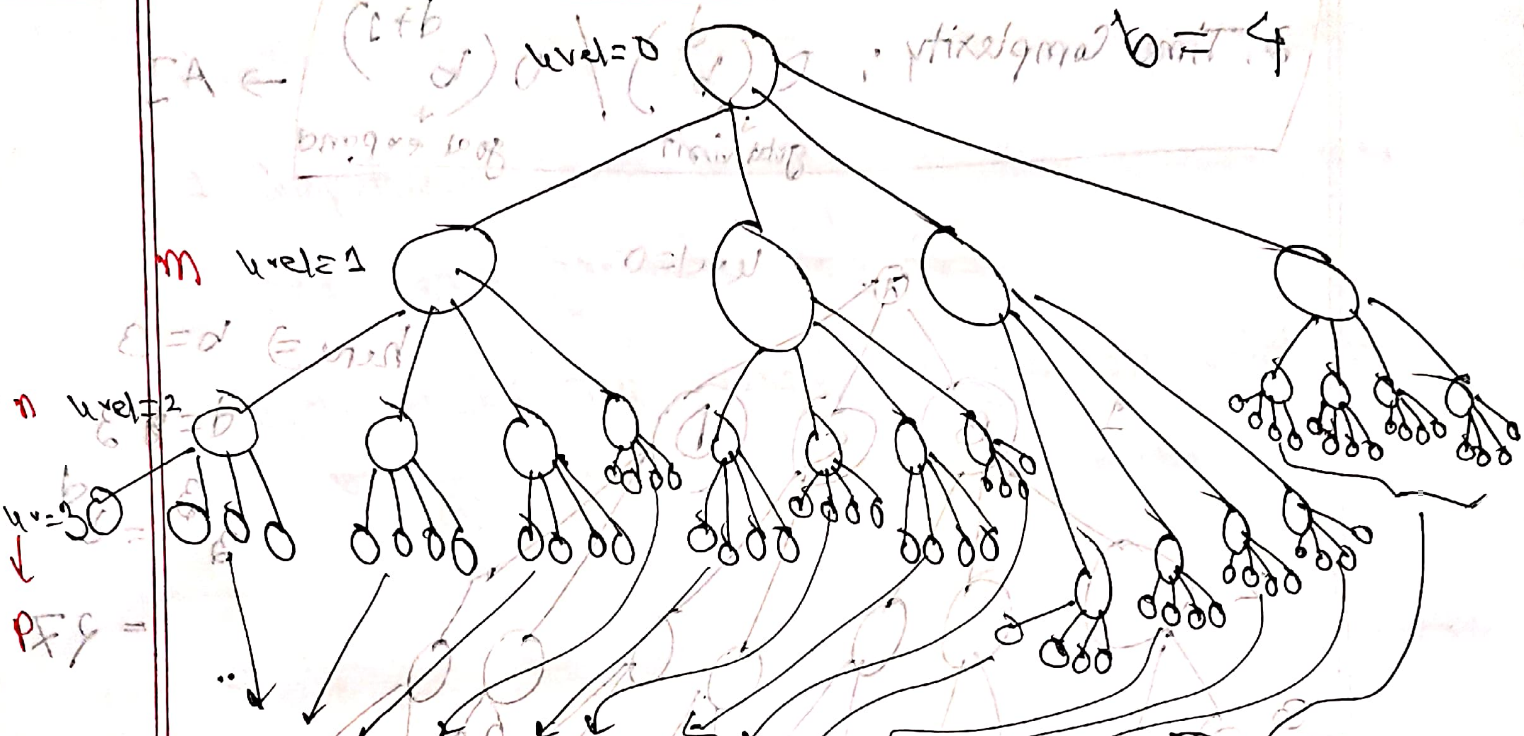
$d=3$

$3^3 = b^d$

$=27$

$2+2+2$

$(2+2+2)+(2+2)+(2+2)$ $=14 = $



$4+4+4+4+4+4+4+4+4+4+4+4+4+4+4+4$

$= 64$

$$\boxed{b^d = 4^3 = 64}$$

$m=4, \quad n=4, \quad p=4 \qquad m \times n \times p = 4\times4\times4 = 4^3 = 64$

AI → Rule Based

ML →

BFS Space complexity: b^{d+1}

DFS - Depth First Search

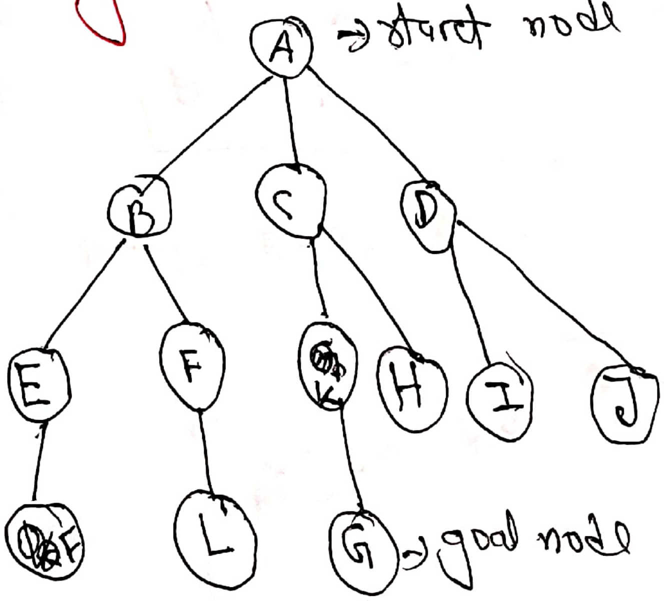
→ Space complexity issue resolve করার জন্য DFS use করা হয়।

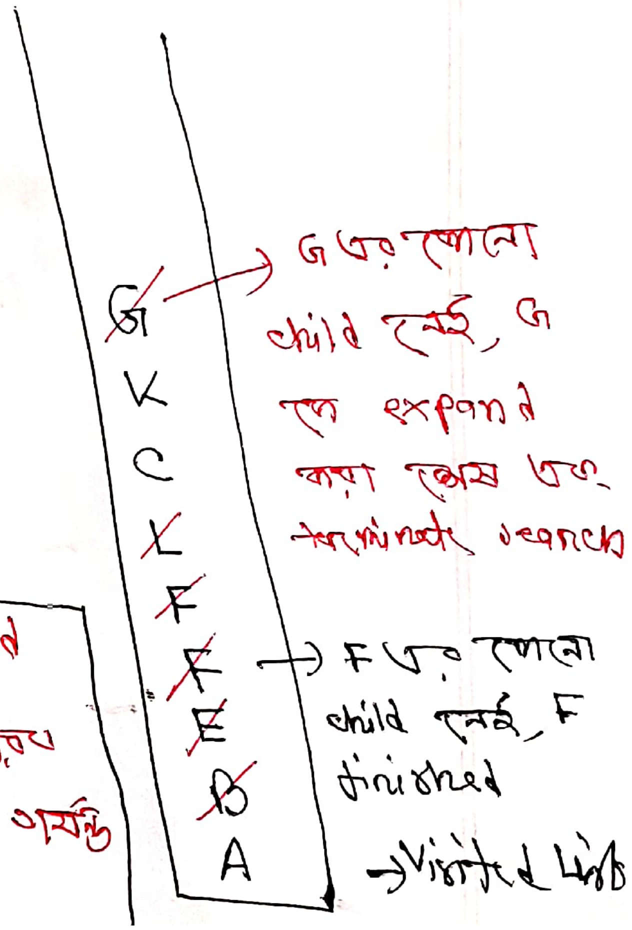
DFS দুইভাবে simulate করা যায়:

(i) Recursion

(ii) Stack

using stack:

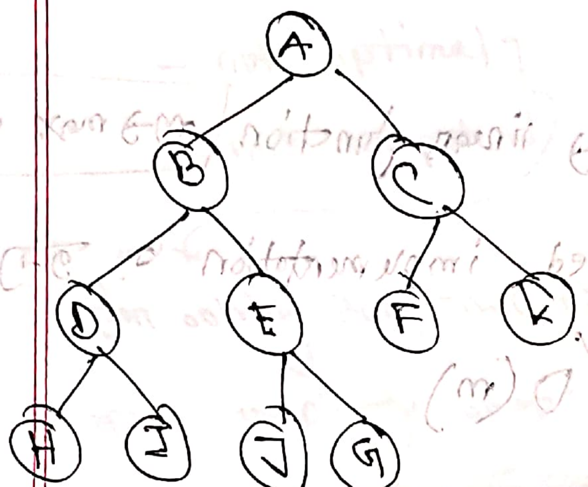




***

* যখন G (goal node) কে expand করে, তখন search terminate করে
* Goal Node কে expand করা পর্যন্ত DFS চালাতে হবে।

Using stack: (Another implementation) Fringe: Stack



A

LIFO
Last In First Out

	A
A	CB
B	CED
D	CEIH
H	CEI
I	CE
E	CGJ
J	CG
G	C

Using stack (normal):

~~G~~
~~J~~
E
~~I~~
~~H~~
~~D~~
B
A

DFS:

Time Complexity: $O(b^{d+1})$

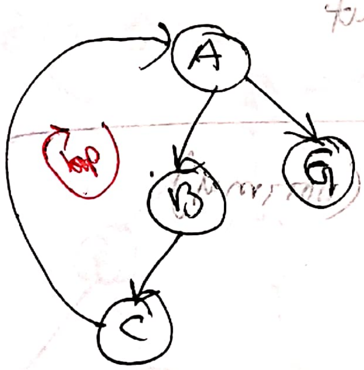
Space complexity: $O(bm)$ → linear function $\boxed{m \to \text{max depth}}$

$O(bd)$ ← ↓ stack based implementation এর জন্য

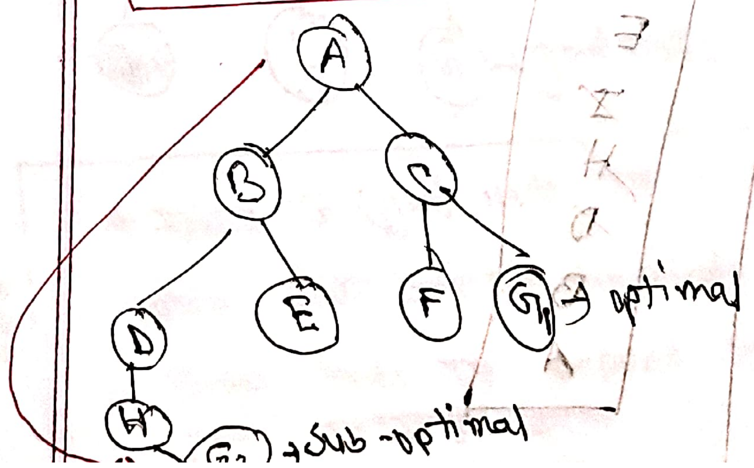
for Recursion: $O(d) / O(m)$

$\boxed{\text{Completeness: Not complete}}$

→ Tree এর বদলে graph এ DFS apply করলে goal খুঁজে নাও পেতে পারে।



$\boxed{\text{Optimal: Not optimal}}$



Uninformed search :

DFS: space complexity: O(bd)

- not optimal
- not complete

for solving these two problems keeping space complexity same:

① Depth limited search (DLS)

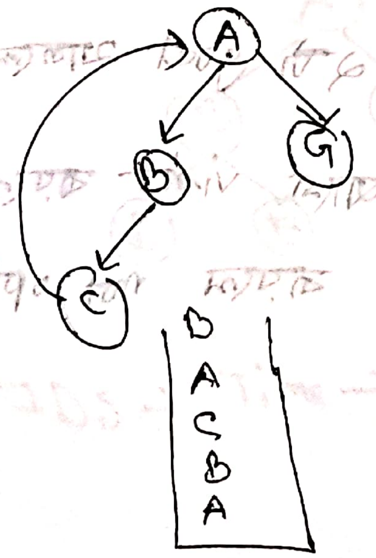
② Iterative Depending search (IDS)

① Depth Limited search (DLS):

→ We impose a limit

→ Depth = L (threshold)

limit = 4 , count = 0



A

AB count = 1

ABC count = 2

ABCA count = 3

ABCAB count = 4

now count == 4 == Limit

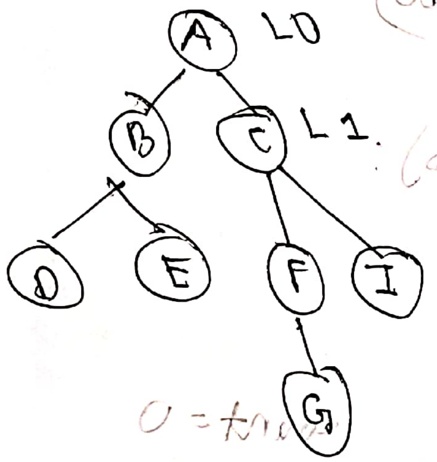
→ এখন জোর করে forced back track করবে।

DLS Pros:

① Prevents infinite loop

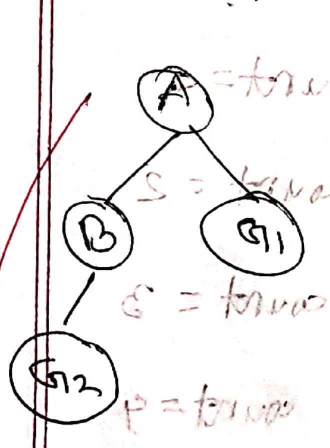
- But still incomplete and not optimal

goal beyond limit থাকলে goal reach possible না।



If limit = 2, goal node reach হবে না।

not optimal: limit এর মধ্যেই ২টা goal থাকলে 3 left most branch আগে visit করলে G2 আগে find করলে not optimal.



Complete

Optimal

Deepening

Better way: Iterative ~~Deepening~~ Search (IDS)

$\boxed{\text{DLS Go: better form}}$

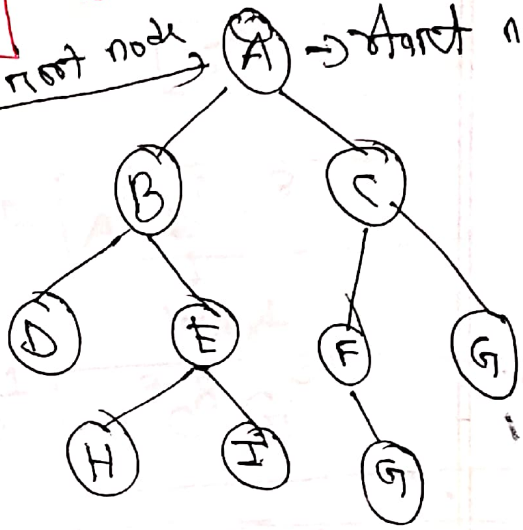
IDS ⇒ l = 0, run DLS

l = 1, run DLS

l = 2

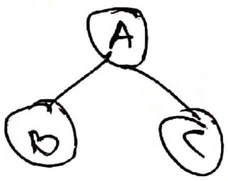
l = m

$\boxed{\text{Goal} \rightarrow m \text{ depth}}$

root node →  → start node

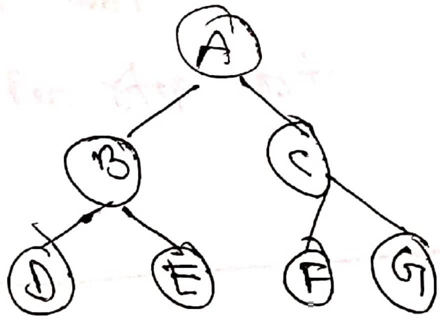
l = 0, 

l = 1, Again start from start node, Apply DLS



Basic DFS

l = 2, Again start from start node, Apply DLS



level

IDS → Time → $O(b^{d+1}) / O(b^{l+1})$

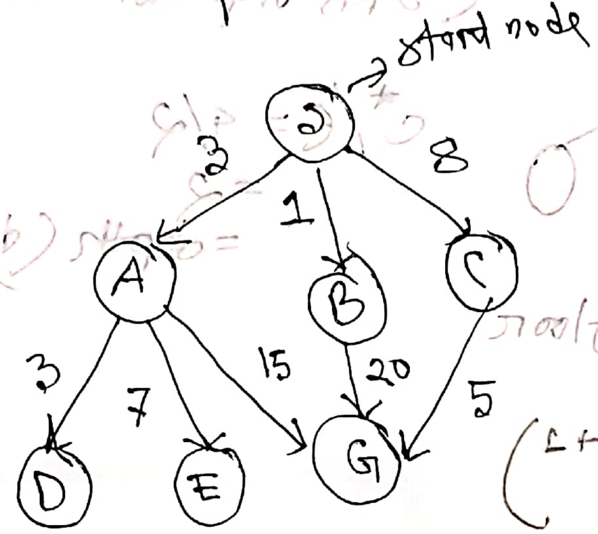
$\boxed{O(bd) / O(bl)}$

DFS, BFS, IDS, DLS → Unweighted graph

For Weighted Graphs

Uniform Cost Search (UCS)

→ Greedy Approach



S^0	S^0
S^0	A^3 B^1 C^8
	lowest
B^1	A^3 C^8 G^{21}
A^3	C^8 ~~G^{21}~~ D^6 E^{10} G^{18}
D^6	C^8 E^{10} G^{18}
C^8	E^{10} ~~G^{18}~~ G^{13}
E^{10}	G^{13}
G^{13}	

→ Optimal
→ Complete

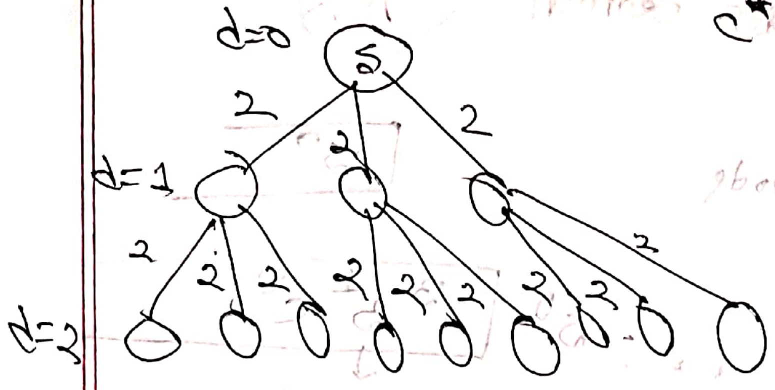
→ time complexity: $O(b^{[C^*/\epsilon]})$

C^* → root থেকে goal পর্যন্ত total cost

ϵ → Per step cost

first goal node expand হওয়া পর্যন্ত search চালু থাকবে।

→ Exception: Only BFS এ goal node visit করলেই search terminate করা যায়, যদি uninformed search এ expand করতে হয়।



$c^* \rightarrow$ source to Goal total cost $= 4$

$\epsilon \rightarrow$ per step cost $= 2$

$c^*/\epsilon = 4/2$

$= 2$

$=$ depth (d)

$\rightarrow$ floor

$O\left(b^{\lfloor c^*/\epsilon \rfloor + 1}\right)$

Bidirectional Search → BFS variant

Informed Search

FYS

Thursday
17/2/2022

Domain knowledge → Heuristic (Hint, আন্দাজ)

Session 6: Informed Search 1

TR 3

Informed Search

- Greedy Best First Search
- A*

Approximated

Criteria: Distance from the goal state → Heuristic (h)

8 Puzzle Heuristic function:

(i) $h_1 \Rightarrow$ No of misplaced tiles

(ii) $h_2 \rightarrow$ Manhattan distance (misplaced tile এর actual position থেকে যত ঘর দূরে আছে)

→ কোনো problem এর জন্য multiple heuristic function থাকতে পারে।

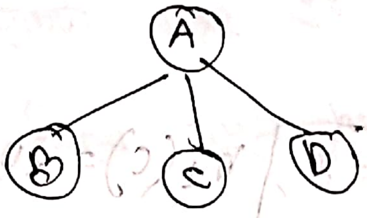
*** → Multiple heuristic function এর মধ্যে যেটা larger value দেয়, সেটাই choose করা হয়।

→ Manhattan distance এর value, no of misplaced tiles এর চেয়ে কখনোই ছোট হওয়া possible না।

→ Manhattan distance $(h_2) \geq h_1$

$\therefore h_2$ will be better

Heuristic: → Dominance
→ Admissibility
→ consistency



$h_1(A) = 12$, $h_1(B) = 15$, $h_1(C) = 20$, $h_1(D) = 9$

$h_2(A) = 7$, $h_2(B) = 9$, $h_2(C) = 17$, $h_2(D) = 8$

for all nodes, $h_1(\text{node}) > h_2(\text{node})$

Here h_1 will be better.

Dominance: কোনো একটা particular tree এর ক্ষেত্রে কোনো particular heuristic function এর সমস্ত node এর value আরেকটা heuristic function এর সমস্ত node এর value এর চেয়ে বড় হয়, তখন তাকে dominance বলে।

let, heuristic functions: h_1, h_2

if সমস্ত node value of $h_1 \geq$ সমস্ত node value of h_2

then h_1 dominates h_2

যখন mix থাকে?

$h_1(A) = 12$, $\boxed{h_1(B) = 8, \; h_2(B) = 9}$, $h_1(C) = 20$, $h_1(D) = 9$

$h_2(A) = 7$, $h_2(C) = 17$, $h_2(D) = 8$

→ এখানে $h_1(B) < h_2(B)$

→ তখন h_1, h_2 কে dominate করবে না।

→ একটা node এর জন্য condition break হলেও dominance হবে না।

if $h_1(\text{all nodes}) \geq h_2(\text{all node})$ then h_1 dominates h_2

→ তখন কিভাবে heuristic choose করব?

→ কেউ কারো চেয়ে বড় না, mix

2 ways to solve this:

1. কোন heuristic function এর most of the nodes greater than others, we choose that heuristic

২য় function এ: Best of both

২. প্রত্যেকটা node এর মাঝে compare করে যেটা বড় সেটা নিয়ে নতুন heuristic বানানো হয়।

$h_1(A) = 12,\ h_1(B) = 8,\ h_1(C) = 20,\ h_1(D) = 9$

$h_2(A) = 7,\ h_2(B) = 9,\ h_2(C) = 17,\ h_2(D) = 8$

$h_f(A) = 12,\ h_f(B) = 9,\ h_f(C) = 20,\ h_f(D) = 9$

Greedy Best First Search (Slide Problem)

Fringe = Priority Queue

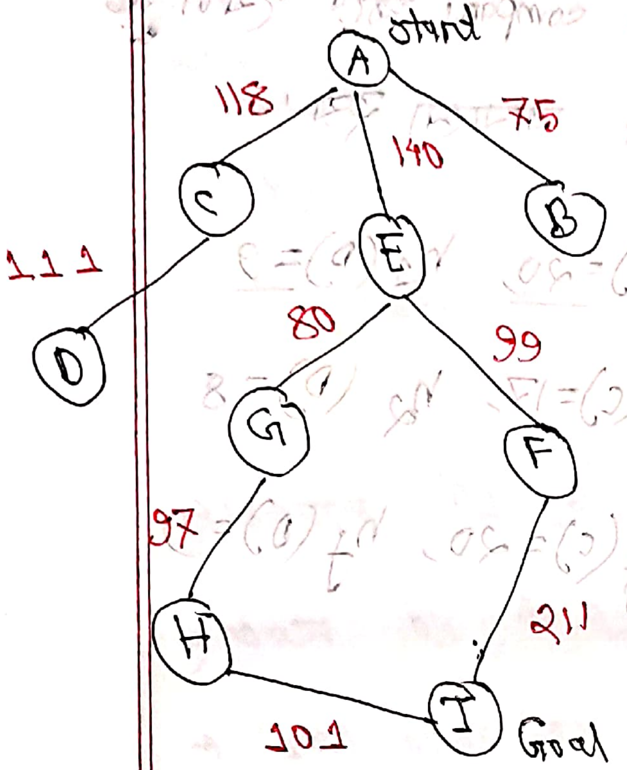
Initial: $\boxed{A^{366}}$

A^{366} $\boxed{C^{329}\ E^{253}\ B^{374}}$

then pop ⟶ A এর সব child কে PQ তে push করতে হবে।
like BFS

UCS → Edge cost নিয়ে

GBFS → Heuristic cost, Actual cost নিয়ে do not care



h-table

State	Heuristic : h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

$d(n) = h(n)$ = straight-line distance heuristic

Heuristic value

A^{366}

A | C^{329} E^{253} B^{374} → তখন C, E, B এর মধ্যে যার heuristic lowest তাকে pop/expand করবো। E^{253} প্রথমে expanded হবে।

E^{253} | C^{329} B^{374} G^{193} F^{178}

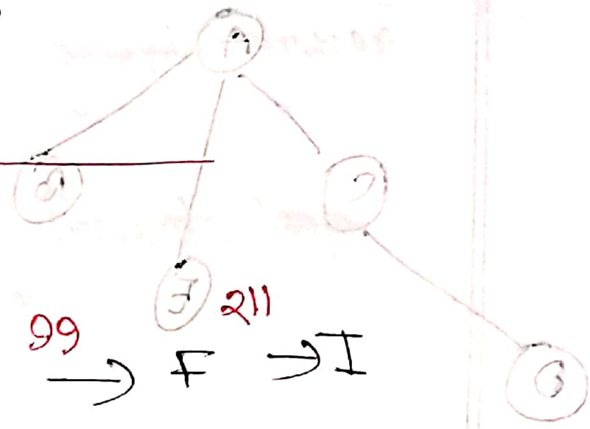
children of E^{253}

F^{178} | C^{329} B^{374} G^{193} I^{0}

Tree search version: কোনো parent already expand/visit করা হয়েছে কিনা ঐ track রাখে না, Graph search version এ রাখে

→ Any other search algorithm, যখন goal node expand হবে, terminate search

I^{0} | C^{329} B^{374} G^{193}



Total path: $A \xrightarrow{140} E \xrightarrow{99} F \xrightarrow{211} I$

Path cost = 450

→ Greedy BFS: (Best First)

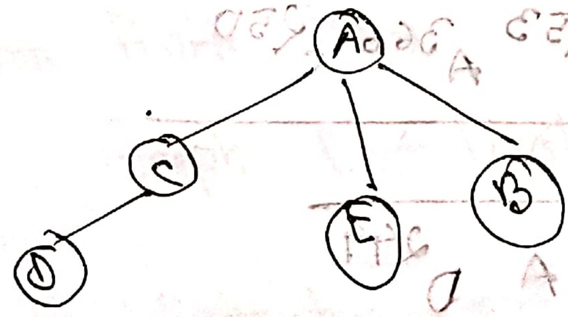
$\boxed{\text{① Not optimal}}$

(ii) complete?:

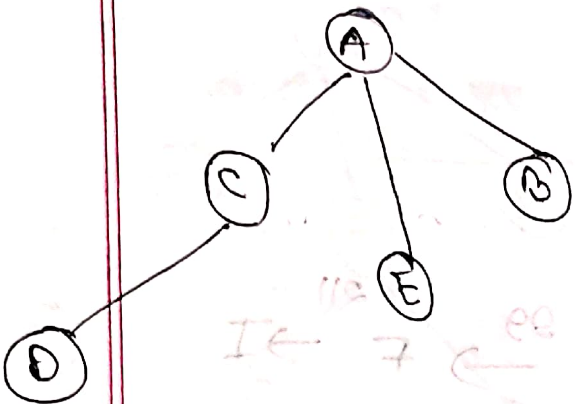
→ এ ধরনের search এর দুইটা version থাকে

① Tree version ② Graph version

① Considering Tree version:



$\boxed{A^{366}}$



State	Heuristic : h(n)
A	366
B	374
C	250 (new)
D	244
E	253

$\boxed{A^{366}}$

A^{366} [B^{374} C^{250} E^{253}]

C^{250} [B^{374} E^{253} A^{366} D^{244}]

visited কিনা সেটা track না রাখার জন্য Edge bi directional হওয়ার ~~জন্য~~ A, D এ C এর child হিসেবে এড হলো।

D^{244} [B^{374} E^{253} A^{366} C^{250}]

C^{250} [B E A D^{244}]

⋮

Infinite loop Problem

→ Not complete for tree ~~graph~~ version

For Graph search version: Prevents Loop

A^{366}

A^{366} | B^{374} C^{250} E^{253}

C^{250} | B^{374} E^{253} D^{244} → A^{366} আবার ঢুকবে না কারণ visited node track রাখা হয়।

D^{244} | B^{374} E^{253}

E^{253} | B^{374}

→ new branch

→ For finite tree →

Graph Best First search is complete

→ for infinite tree → not complete

Graph

Tree version → Incomplete

Graph version → finite → complete

Infinite → Incomplete

Completeness

Time and space Complexity : $O(b^m)$

b → branching factor

m → max depth

*** Time (Actual) → Depends on quality of heuristic

7 - Informed search 2

TRZ

Informed Search

Greedy Best First search | A*

GBFS

→ node যদি n হয়, তার heuristic h(n)

→ heuristic এর উপর base করে GBFS fringe থেকে কোনো node কে expand করে।

→ Expansion criteria function $f(n) = h(n)$

→ যেই particular node এর জন্য h(n) এর value lowest হবে সেই particular node টি আগে expand করা হয়।

In A* search | GBFS+UCS

$$\boxed{f(n) = h(n) + g(n)}$$

$f(n)$: expansion criteria function

$h(n)$: heuristic

$g(n)$: Path cost from start node to node n → Uniform Cost Search (UCS)

GBFS → - Entirely heuristic dependent,
- other criteria নিয়ে bothered না।

$$A^* \rightarrow f(n) = h(n) + g(n)$$

$\boxed{\text{if } h(n) = 0}$

$f(n) = g(n) \rightarrow$ A^* works like UCS

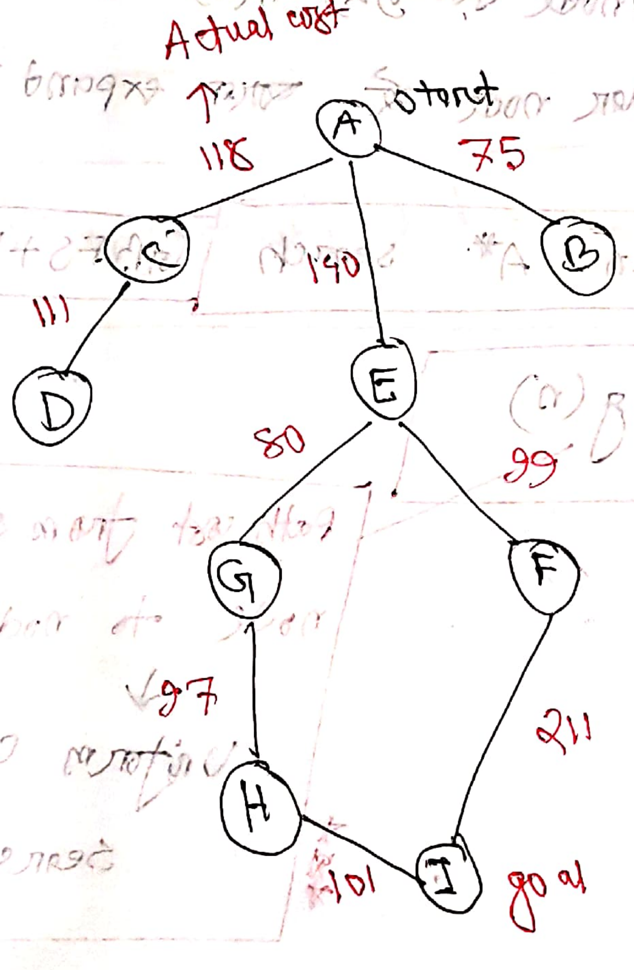
$\boxed{\text{if } g(n) = 0,}$

$f(n) = h(n) \rightarrow$ A^* works like GBFS

$\boxed{A^* \text{ search}}$

$\boxed{f(n) = h(n) + g(n)}$

$g(n)$ = কোনো একটা node এর জন্য root node হতে তার distance/ cost



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

total
Lowest cost → তাকে expand

→ start
→ root node থেকে A এর true cost

$\boxed{A^{366+0}}$ $f(A) = h(A) + g(A)$
$= 366 + 0 = 366$

A^{366} | C $329+118 = 447$ | E $253+140 = 393$ | B $374+75 = 449$

→ BFS এর মত
A^{366} এর মত
child কে push

E^{393} | C^{447} B^{449} | G $193+\overbrace{(140+80)}^{g(g)} = 413$ | F $178+(140+99) = 417$

G^{413} | C^{447} B^{449} F^{417} | H $98+(140+80+97) = 415$

$f(h) = h(h)+g(h)$
$= 98+(140+80+97)$
$= 415$

H^{415} | C^{447} B^{449} F^{417} | $I_{(H)}$ $0+(140+80+97+101) = 418$

F^{417} | C^{447} B^{449} $I_{(H)}^{418}$ | $I_{(F)}$ $0+(140+99+211)$

$I_{(H)}^{418}$ | C^{447} B^{449} $I_{(F)}^{450}$

goal node expanded — search complete

$A^{366} \rightarrow E^{393} \rightarrow G^{413} \rightarrow H^{415} \rightarrow I_{(H)}^{418}$

total cost 418

h ⇒ admissible → optimal

h not admissible → not optimal

1. Check Optimality:

→ A* সব scenario তে optimal কিনা তা depend করে heuristic এর value এর উপরে।

→ " nature এর উপরে।

***

→ A* এর optimality entirely heuristic এর Admissibility এর উপর depend করে।

Admissibility:

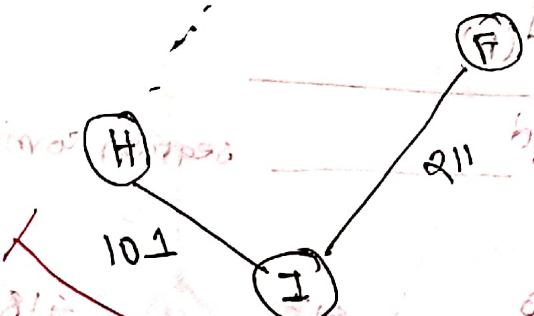
→ যদি graph/tree এর সব node এর;

$h \le$ Actual cost হয়, তাহলে তা Admissible

→ যদি $\boxed{h \le \text{Actual cost}}$ only একটা node এর জন্যও break করে, সেক্ষেত্রে graph টা Admissible হবে না।

(Previous 98)

$h(H) = 138$ (given)



Actual cost:

H থেকে final node এর total edge cost = 101

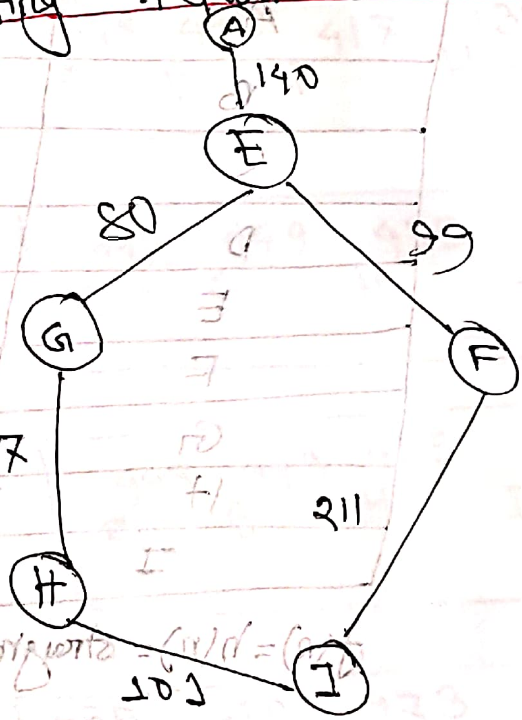
→ Calculate lowest Actual cost for a node having multiple paths.

Here, $h(H) = 138 >$ Actual cost

→ এই particular node এর জন্য h over-estimated

→ একটা single node এর জন্যও যদি h over-estimated হয়, সেই heuristic কে আর Admissible বলা যাবে না।

Calculating Actual cost and Admissibility:



→ E থেকে I তে যাওয়ার দুইটা true cost আছে।

for E → G → H → I,
Actual cost : $80 + 97 + 101 = 278$

for E → F → I, Actual cost $= 99 + 211 = 310$

→ $278 < 310$, always lowest actual cost calculate করে h এর সাথে compare করতে হবে।

$h(E) = 253 <$ Actual $(E) = 278$

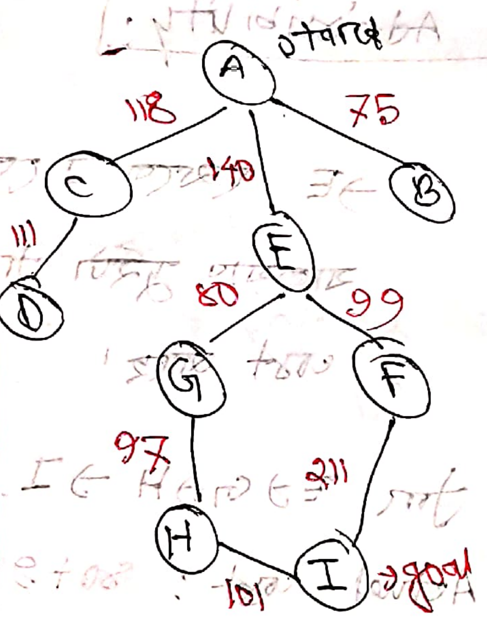
$\therefore h(E) =$ Admissible

* কোনো single node এর h over-estimated অর্থাৎ Admissibility

$$\boxed{h(\text{all nodes}) \le \text{Actual cost (all nodes)} = \text{admissible}}$$

heuristic admissible হবে না

break হলে, A* sub-optimal root return করতে পারে।



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	138
I	0

$f(n) = h(n) =$ straight-line distance heuristic

for $h(H) = 138 > H \xrightarrow[101]{\text{Actual cost}} \text{Goal, I}$

$\therefore$ H over-estimated.

$\therefore$ Admissibility break

Heuristic → (কোনো particular node হতে goal node (এর estimated cost

A^{366+0}

A^{366} | C $329+118 = 447$ | E $253+140 = 393$ | B $374+75 = 449$

E^{393} | C^{447} | B^{449} | G $193+(140+80) = 413$ | F $178+(140+99) = 417$

G^{413} | C^{447} | B^{449} | F^{417} | H $138+(140+80+97) = 455$

F^{417} | C^{447} | B^{449} | H^{455} | $I_{(F)}$ $0+(140+99+211) = 450$

C^{447}, $\cancel{I_{(F)}^{450}}$ | $\cancel{C^{447}}$ | B^{449} | H^{455} | $I_{(F)}^{450}$ | D $244+(118+111) = 473$

B^{449} | H^{455} | $I_{(F)}^{450}$ | D^{473}

$I_{(F)}^{450}$ | H^{455} | D^{473} — search end

sub optimal

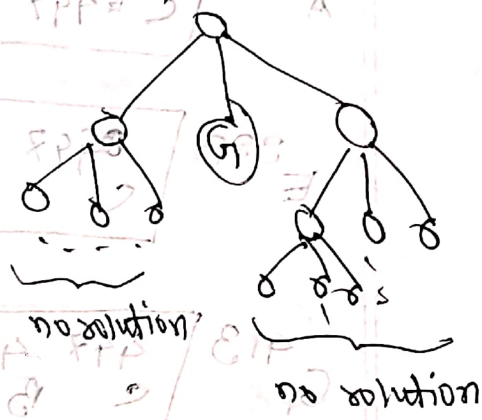
$A \xrightarrow{140} E \xrightarrow{99} F \xrightarrow{211} I_F$ 417

Total cost = 450

2. time and Space Complexity:
3.

$O(b^{\epsilon d})$ → যখন আসতে পারে depending on the quality of the heuristics.

$\boxed{\epsilon \rightarrow \text{quality of heuristic}}$

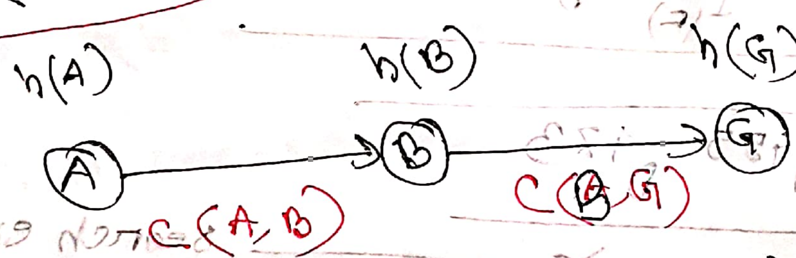


Heuristic ভালো করার উপায়:

→ Heuristic Consistency:

→ consistent না হলে A^* এর Actual implementation time এর space complexity বেশি লাগে। কারণ heuristic consistence না হলে, সে relevant branch এ না খুঁজে আগে irrelevant branch এ যাবে।

** → Heuristic function admissible but not consistence হতে পারে।



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

$h(n)$ → each node

$h(n')$ → child node এর heuristic cost

$h(n, n')$ → একই node থেকে child এ যাওয়ার True cost

$h(n, n')$ $\quad$ $h(n')$

for $h(A) \le c(A,B) + h(B)$

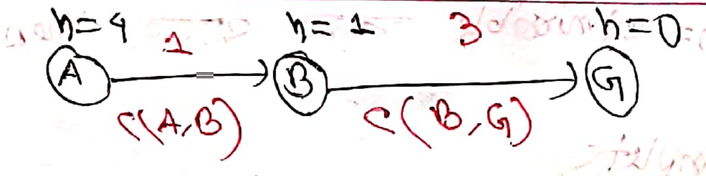
for $h(B) \le c(B,G) + \cancel{h(B} \; h(G) \rightarrow$ G এর heuristic cost

B থেকে G তে যাওয়ার path cost

Except the goal node, যদি সমস্ত node এর জন্য $h(n) \le h(n') + h(n,n')$ condition true হলে তাহলে সেটা Graph এর heuristic consistent

****

Heuristic admissible but not consistence:



admissible check: True

for A: $h(A)=4$, A to G

Actual cost $= 1+3=4$

$h(A)=4 \le 4$

for B: $h(B)=\cancel{3}\,1$, B to G Actual cost $= 3$, $h(B)=\cancel{3}\,1 \le 3$

for G: $h(G)=0$, G to G

Actual cost $=0$

Consistency check: False

for A: $h(A)=4 \le c(A,B)+h(B)$

$4 \le 1+1$

4 not less than equal to 2

4. Completeness:

For Graph search Version:

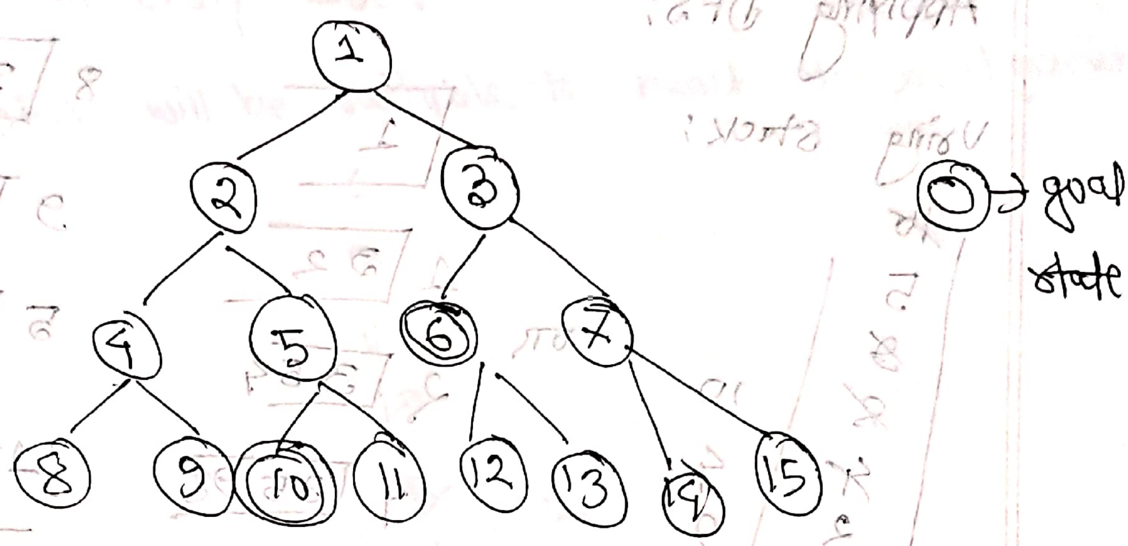
(i) heuristic ~~consistent~~ admissible হলেই তা complete

For tree search:

(i) h কে both admissible এবং consistent হতে হবে complete হওয়ার জন্য

$$\boxed{\rightarrow \text{যদি just } h = \text{admissible বলা থাকে, then complete}}$$

Q1.



Applying BFS: Fringe: Queue

1

1 | 2 3

2 | 3 4 5 ~~6~~

3 | 4 5 6 7

search complete (without expanding goal 6)

with expanding goal:

4 | 5 6 7 8 9

5 | 6 7 8 9 10 11

6 | 7 8 9 10 11 12 13

time complexity:

b = 2

goal (6) depth = 2

$O(2^2) = 4$

(11)

Applying DFS:

Using stack:

~~10~~
5
~~8~~
~~8~~
~~4~~
2
1

10
4
9
8

$\boxed{1}$

1 $\boxed{3\ 2}$

2 $\boxed{3\ 5\ 4}$

4 $\boxed{3\ 5\ 9\ 8}$

8 $\boxed{3\ 5\ 9}$

9 $\boxed{3\ 5}$

5 $\boxed{3\ 11\ 10}$

10 $\boxed{3\ 11}$

— x — finish

here, $b = 2$, $d/m = 3$ $\quad O(b^{m+1}) = 2^3 = 8$

So, DFS needs more time to return a goal (8>4)

But this is not the main case. BFS beats DFS on the question of optimality. Here we have two goals. 6 is located at level = 2 and another goal state 10 is located at level = 3. As BFS traverse level-wise, BFS will return the goal state which is located at shallowest/lowest depth. As we are traversing DFS in a left manner, it will return non optimal goal (10) first, so, DFS is not

(iii)

optimal in every case.

So, BFS will be suitable to reach the goal quickly.

Q2. (a)

6

3

~~4~~

~~10~~

~~5~~	2
~~9~~	5
~~8~~	11
~~4~~	10
~~2~~	9
1	8

Q. $\boxed{X^7}$

X^7 $\boxed{Y^4 \quad Z^7_{(x)}}$

Y^4 $\boxed{Z^7_{(x)} \quad M^5 \quad N^1 \quad O^4}$

Y^4

N^1 $\boxed{Z^7 \quad M^5 \quad O^4 \quad P^0}$

P^0 $\boxed{Z^7 \quad M^5 \quad O^4}$

using A^* :

$\boxed{X^{7+0}}$

X^7 $\boxed{Y^{4+4}_{=8} \quad Z^{7+2}_{=9}}$

Y^8 $\boxed{Z^9_{(x)} \quad M^{5+1}_{=6} \quad N^{1+3}_{=4} \quad O^{4+2}_{=6} \quad Z^{7+1}_{(Y8)}}$

N^4 $\boxed{Z^9_{(x)} \quad M^6 \quad O^6 \quad Z^8_{(Y8)} \quad P^{0+3}}$

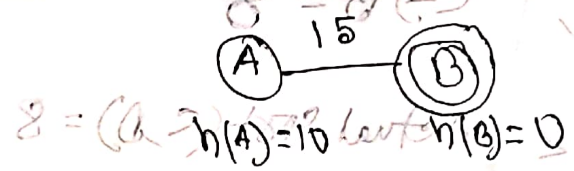
$P3$ $\boxed{Z^9_{(x)} \quad M^6 \quad O^6 \quad Z^8_{(Y8)}}$

Q5. (a) Formula for admissibility:

$$h(n) \leq \text{Actual cost}(n)$$

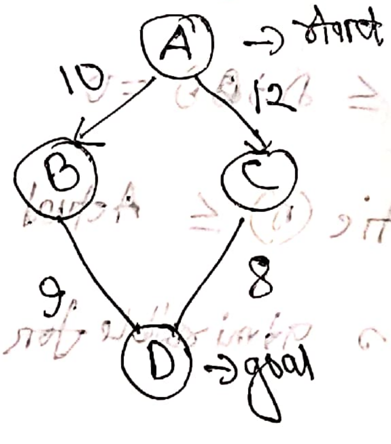
here n = each node of a tree/graph

When all the nodes of a graph/tree abide by the rule mentioned above then the heuristic be called admissible.



here, $h(A) = 10 \leq 15$, the actual cost from A node to goal node B.

(b)



state	heuristic: h(n)
A	10
B	8
C	7
D	0

$h(A) = 10$,

for A → B → D: Actual cost: $10 + 9 = 19$

$$h(A) = 10 < 19$$

for A → C → D: Actual cost: $12 + 8 = 20$

$$h(A) = 10 < 20$$

(10)

As there is two path from A to goal D,
we will take the path with lowest true cost.

taking $A \to B \to D = 19$, $h(A) = 10 \leq 19$

for B: $h(B) = 8$, $B \to D = 9$

$h(B) <$ Actual cost $(B, D) = 9$

for C: $h(C) = 7$, $C \to D = 8$

$h(C) = 7 <$ Actual cost $(C, D) = 8$

for D: $h(D) = 0$, $D \to D = 0$

$h(D) = 0 \leq D \to D = 0$

for all node, heuristic $(n) \leq$ Actual cost (n)

$\therefore$ The heuristic is admissible for my example.

1 CSE422

G.M. Arafat Rahman

ID: 19101498 Section: 11

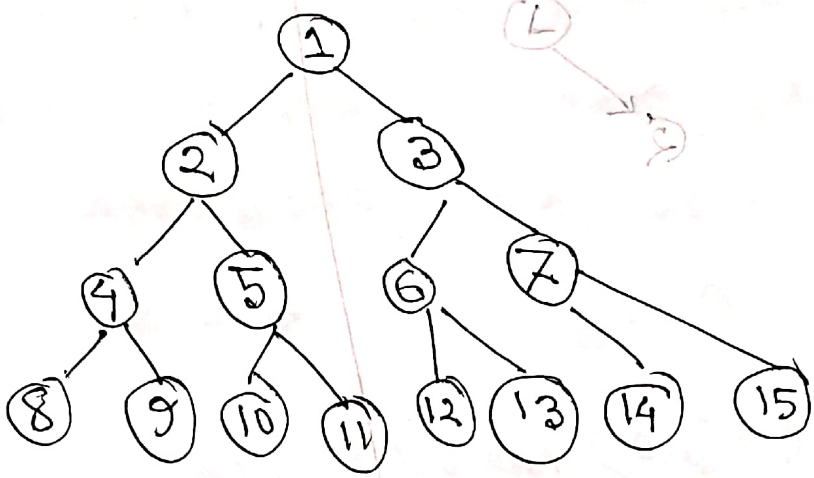
Assignment

Q1.

1 → 2

1

CSE422 G.M. Arafat Rahman ID: 19101498 section: 11 Assignment 1



Goal ~~state~~ = 10:

BFS: Fring: Queue

	1
1	2 3
2	3 4 5
3	4 5 6 7
4	5 6 7 8 9
5	6 7 8 9 10 11
6	7 8 9 10 11 12 13
7	8 9 10 11 12 13 14 15
8	9 10 11 12 13 14 15
9	10 11 12 13 14 15
10	11 12 13 14 15

Here BFS will expand: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10

DFS: fring stack

Stack	
10	
5	
~~9~~	
~~8~~	
~~4~~	4
2	9
1	8

or

[1]

1 [3 2]

2 [3 5 4]

4 [3 5 9 8]

8 [3 5 9]

9 [3 5]

5 [3 11 10]

10 [3 11]

Path that DFS will follow: 1 → 2 → 4 → 8 → 9 → 5 → 10

Here, DFS will give the result more quickly than BFS. So, DFS will be suitable for reaching the goal (10) quickly.

When goal state 6:

BFS:

[1]

1 [2 3]

2 [3 4 5]

3 [4 5 6 7]

4 [5 6 7 8 9]

5 [6 7 8 9 10 11]

6 [7 8 9 10 11 12 13]

goal expanded

(III)

nodes needed to expand for BFS:

1 → 2 → 3 → 4 → 5 → 6

Using DFS:

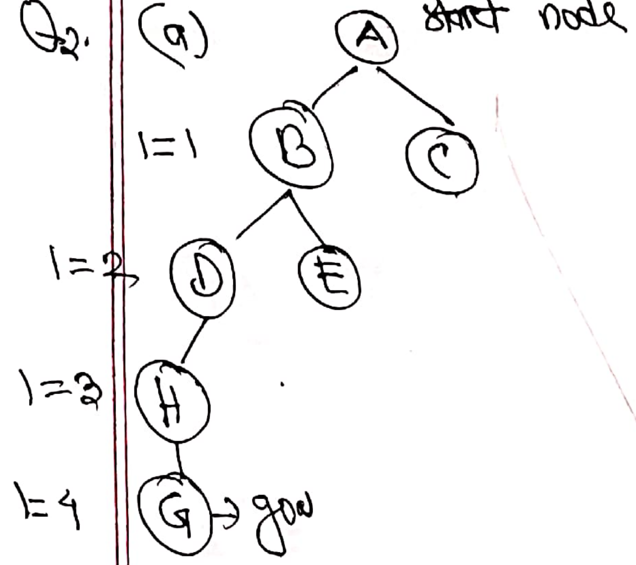
	1	10	3 11
1	3 2	11	3
2	3 5 4	3	7 6
4	3 5 9 8	6	7 13 12
8	3 5 9		
9	3 5		
5	3 11 10		

Here BFS outperforms DFS as a goal in shallowest depth is found by BFS first in all conditions. (when cost is ignored)

So, for goal 6, BFS will work better.

(IV)

Q2. (a)



Applying DFS:

$\boxed{A}$

A $\boxed{CB}$

B $\boxed{CED}$

D $\boxed{CEH}$

H $\boxed{CEG}$

G $\boxed{CE}$

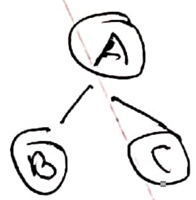
Path to goal: A → B → D → H → G

Using Iterative Deeping search:

l=0 → A

l=1 → $\boxed{A}$

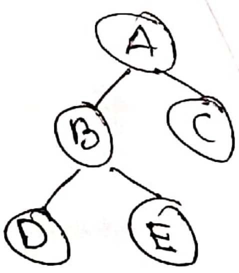
A $\boxed{CB}$



l=2 → $\boxed{A}$

A $\boxed{CB}$

B $\boxed{CED}$

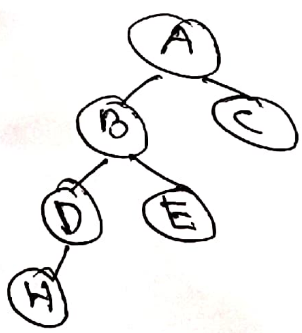


l=3 → $\boxed{A}$

A $\boxed{CB}$

B $\boxed{CED}$

D $\boxed{CEH}$



l=4, $\boxed{A}$ D $\boxed{CE\ H}$ IDS will take more time

A $\boxed{CB}$ H $\boxed{CE\ G}$ to find the goal. Here

B $\boxed{CED}$ G $\boxed{CE}$ DFS > IDS

(b) Complete: If there is a solution, guarantee of finding that goal. BFS traverse a tree/graph level wise. It checks all the nodes in a level then moves to next level. So if a step-cost is zero, it does not matter to BFS as it will go level by level. So BFS is complete even if zero-step costs are allowed.

Q3. UCS runs in greedy approach. It first expands the node which has lowest cost among the then visited nodes. On the other hand, plain BFS is entirely free from edge cost, it only check the levels. The shallowest level is considered ~~as~~ close to BFS where UCS thinks about the cost.

For, an example:

(7)

Q4. Using GBFS:

$\boxed{X^7}$

X^7 $\boxed{Y^4 \; Z^7}$

Y^4 $\boxed{Z^7 \; M^5 \; N^1 \; O^4 \; Z^7}$

N^1 $\boxed{Z^7 \; M^5 \; O^4 \; Z^7 \; M^5 \; P^0}$

P^0 $\boxed{Z^7 \; M^5 \; O^4 \; Z^7 \; M^5}$

Cost: $X^7 \xrightarrow{4} Y^4 \xrightarrow{3} N^1 \xrightarrow{3} P^0$

$4 + 3 + 3 = 10$

Using A^*:

$\boxed{X^{7+0}}$

X^7 $\boxed{Y^{4+4=8} \; Z^{7+3=9}}$

Y^8 $\boxed{Z^{9}_{(X)} \; M^{5+(4+1)}=10 \; N^{1+(4+3)}=8 \; O^{4+(4+2)}=10 \; Z_{(Y)}^{7+(4+1)}=12}$

N^8 $\boxed{M^{10} \; \cancel{N^{8}} \; O^{10} \; Z^{9}_{(X)} \; Z^{12}_{(Y)} \; M^{5+(4+3+4)}=16 \; P^{0+(4+3+3)}=10}$

$Z^9_{(X)}$ $\boxed{M^{10} \; O^{10} \; Z^{12}_{(Y)} \; M^{16}_{(N)} \; P^{10}_{(N)} \; O^{4+(2+3)}_{(Zx)}=9}$

8

$O^{9}_{(Zx)}$ $\quad M^{10}\ O^{10}\ Z^{12}_{(Y)}\ M^{16}_{(N)}\ P^{10}_{(N)}\ N^{1+(2+3+1)}_{(O)} = 7$

$N^{7}_{(O)}$ $\quad M^{10}\ O^{10}\ Z^{12}_{(Y)}\ M^{16}_{(N)}\ P^{10}_{(N)}\ P^{0+(2+3+1+3)}_{(N_O)} = 9$

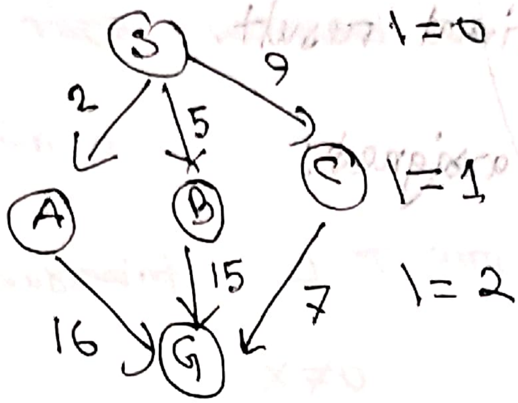
$P^{9}_{(N_O)}$ $\quad M^{10}\ O^{10}\ Z^{12}_{(Y)}\ M^{16}_{(N)}\ P^{10}_{(N)}$

∴ Path cost $P^{9} = 9$

$X^{7} \xrightarrow{2} Z^{9}_{(X)} \xrightarrow{3} O^{9}_{(Zx)} \xrightarrow{1} N^{7}_{(O)} \xrightarrow{3} P^{9}_{(N_O)}$: Goal path

Path cost = $2+3+1+3 = 9$

(VI)



Using BFS: $\boxed{S^0}$

S^0 $\boxed{A^2\, B^5\, C^9}$

A^2 $\boxed{B^5\, C^9\, G^{18}}$

B^5 $\boxed{C^9\, G^{18}_{(A)}\, G^{20}_{(B)}}$

C^9 $\boxed{G^{18}_{(A)}\, G^{20}_{(B)}\, G^{16}_{(C)}}$

$G^{18}_{(A)}$ $\boxed{G^{20}_{(B)}\, G^{16}_{(C)}}$

Using UCS:

$\boxed{S^0}$

S^0 $\boxed{A^2\, B^5\, C^9}$

A^2 $\boxed{B^5\, C^9\, G^{18}_{(A)}}$

B^5 $\boxed{C^9\, G^{18}_{(A)}\, G^{20}_{(B)}}$

C^9 $\boxed{G^{18}_{(A)}\, G^{20}_{(B)}\, G^{16}_{(C)}}$

$G^{16}_{(C)}$ $\boxed{G^{18}_{(A)}\, G^{20}_{(B)}}$

Here, G is at level 2. For BFS only level matters. That's why it return a sub-optimal goal, on the other hand, as UCS traverses on greedy basis, it expands on lowest cost. So it returned the optimal result.

returned an optimal result.

When cost is not assigned:

Constraint Satisfaction Problem Thursday
FYS - 03/03/22

→ Path নিয়ে concerned না

→ Solution " "

Unary constraint : 1 টা variable involved

$X \neq 0$

Binary " : 2 টা " " , $X \neq Y$

→ সব variable এ ~~সকলে~~ কোনো না কোনো value

থাকলে CSP complete

→ সব ~~cont~~ constraint মানতে হবে।

→ Node Consistency : শুধুমাত্র Unary constraint এ

fulfill হয়।

→ Arc Consistency : Depends on binary constraint

Genetic Algorithm

[A-Z_] ⟶ $\underbrace{\text{I\_LOVE\_PYTHON}}_{\text{length} = 13}$ → goal state

→ character set

step: 1:

(i) character set থেকে কিছু character নিয়ে population create করতে হবে। goal state এ যত সংখ্যক character আছে, ততো character নিতে হবে এবং, population বানাতে হবে

Assignment 2

(i) Transaction register বানাতে হবে

(ii) transaction register থেকে randomly population generate করতে হবে

⇗ Initial population

1 0 1 1 0 0 1

1 1 1 0 0 0 1

⋮

take as many as you want

Local search

TRZ Session 8

* Informed / Uninformed search search criteria:

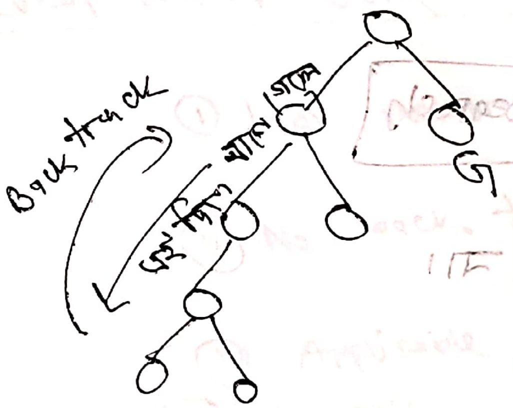
(i) Technical search

(ii) Path cost matters

(iii) Path fringe ও save করা লাগে।

(iv) Back Tracking for path detection. How it will go from start node to goal node.

(v) Back tracking for finding perfect goal.



- এইসব ক্ষেত্রে informed ও uninformed search necessary.

- কিন্তু AI এ অনেক বড় tree/graph নিয়ে কাজ করা লাগে। যেখানে informed ও uninformed search অনেক expensive (mostly). For instance, DFS has linear space complexity. DFS এ huge depth এ কোনো branch থাকলে, সেটার জন্য back tracking করলে অনেক

গুলো node store করা লাগে। এই stored node অনেক বেশি space নেয় (অনেক সময় এর memory তে অনেক stress put করে। এই সমস্যা কিছুটা mitigate করা যায় using informed search as they use heuristics. But heuristics এর quality খারাপ হলে search এ অনেক সময় লাগতে পারে।

- বারবার Back track করে different branch visit করলে time complexity অনেক বেড়ে যায়।

Local search

- Path নিয়ে concerned না।
- Only solution matters
- Image classifier who can identifies an animal as cat / dog
- Not concerned with perfect solution
- Just solution matter করায় Back tracking necessary না। কোনো branch এ sought solution না থাকলে back track করে different branch এ search করতে হয়।

Local Search:- Back tracking elimination
Path এর জন্য node store করা লাগে না।

যেহেতু local search এতে back tracking লাগলেও, যদি perfect solution নিয়ে concerned না থাকি তাহলে back track করাটা necessary না।

Path নিয়ে bothered না। তাই কোন কোন node visit করে solution পাওয়া গেছে সেটা memory তে store করা লাগে না। Memory complexity কমে আসে।

Local search Benefits:

(i) Less memory - কোনো node store করে না।

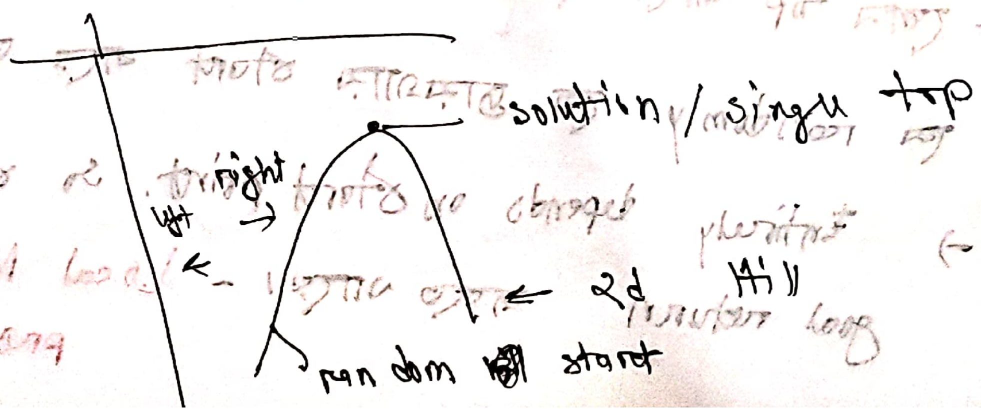
(ii) No back tracking, so take less time

(iii) Applicable for large size graphs

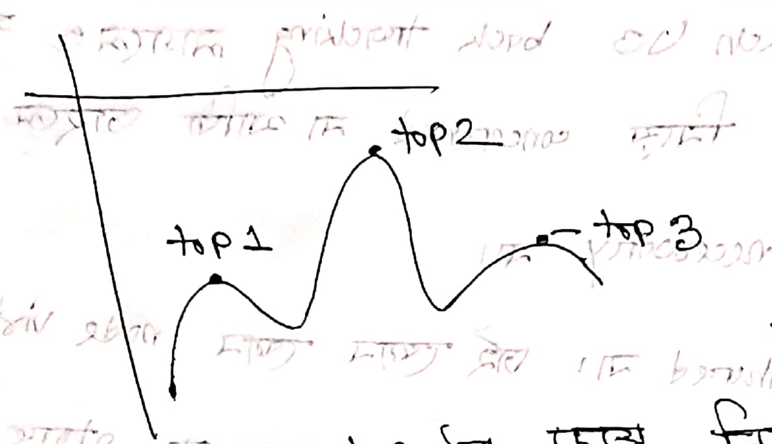
(iv) scalable

Types of Local search:

(i) Hill climb search → Can apply on Regression Analysis



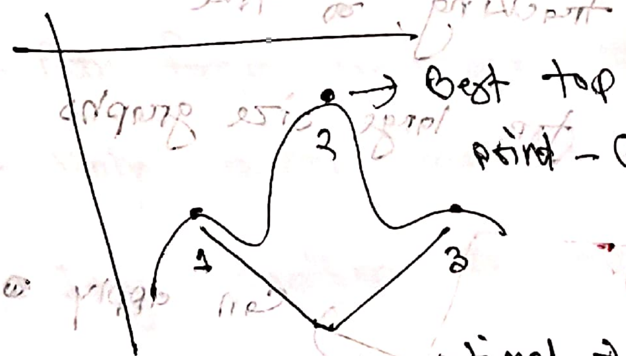
→ n সংখ্যক unknown, hill shape n+1 dimensional



top: যে particular point থেকে যে direction এই ~~সময়~~ যাওয়া হোক না কেন, সব direction এই top এর চেয়ে নিচে হবে।

Local search Disadvantages:

① multiple top points.



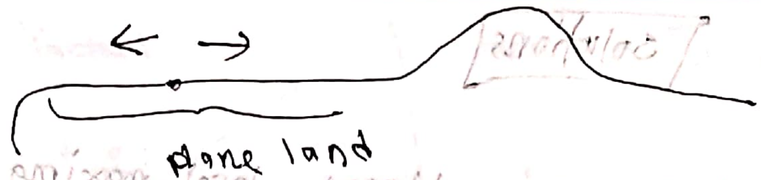
- যখনই কোনো top পাওয়া যায়, তখনই search terminate হয়ে যায়।
- কোন top পাবে সেটা entirely depend করে

যে randomly যে জায়গায় start করে তার উপর।

→ Entirely depends on start point. So sub-optimal goal return করতে পারে। → Local Maxima problem

⇒ Hill climb Iterative search

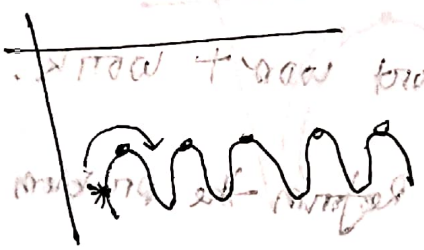
(ii)



- plane land হওয়ায় left এ গেলেও level same এবং right এ গেলেও level same থাকবে। তখন search algo বুঝতে পারে না কোনদিকে যেতে হবে। এই ধরনের সমস্যাকে Plateaus বলে।

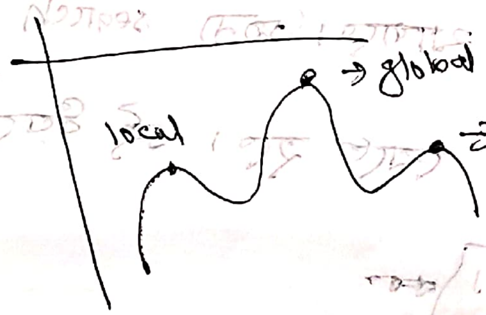
Plateaus: Hill এর plane land এ এমন অবস্থা যেখানে কোনো direction খুঁজে পাওয়া যায় না।

(iii) Ridges: যখন multiple local maxima একটা আরেকটা এর খুব কাছাকাছি থাকে।



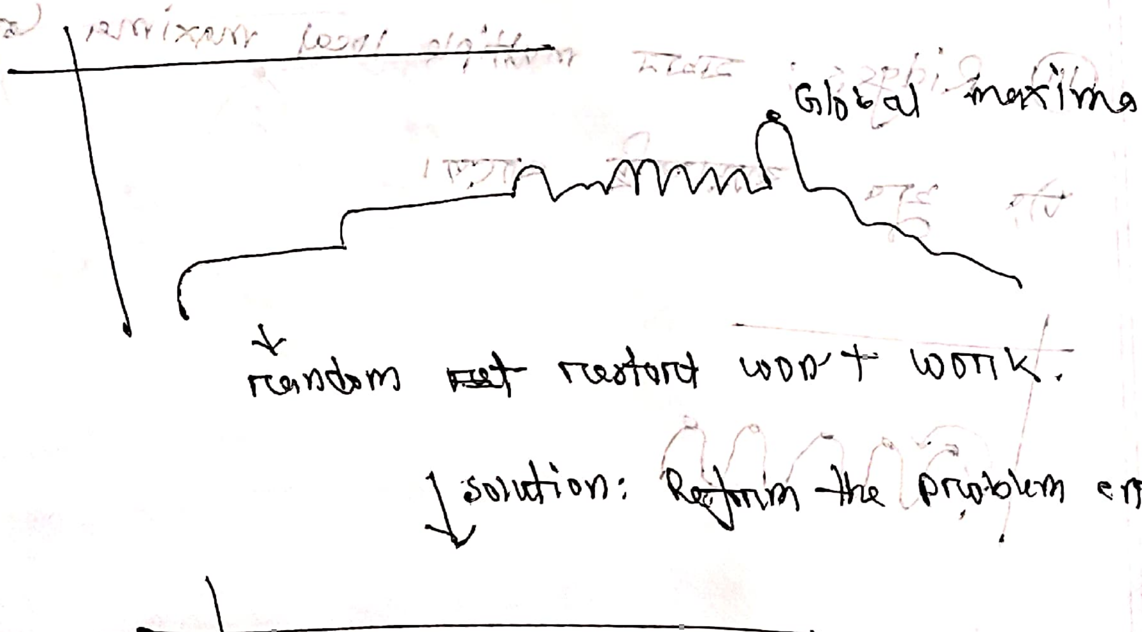
Solutions

① Random restart: Can address local maxima, ~~plateau~~



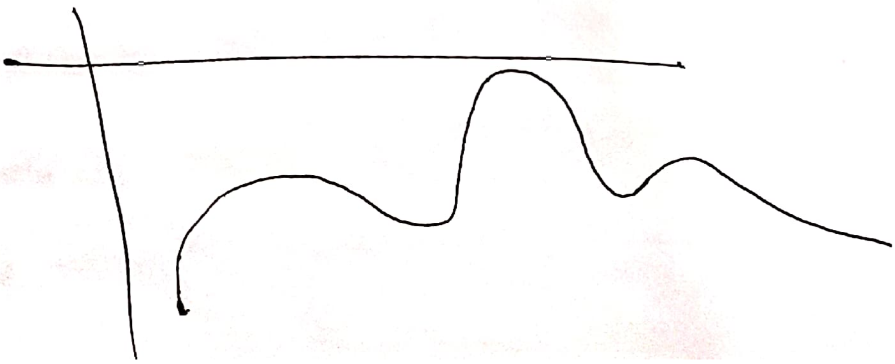
→ বারবার randomly starting position change ~~বারে~~ করে global maxima find করা।

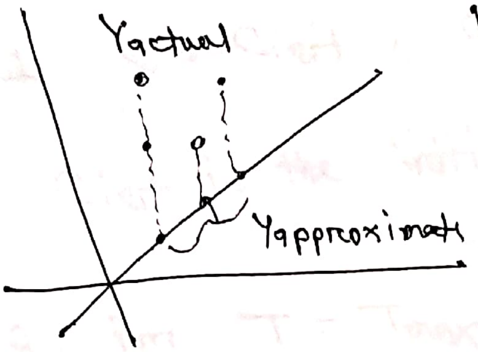
⑪ Problem reformulation: Hill কে shape change করা।



random ~~ret~~ restart won't work.

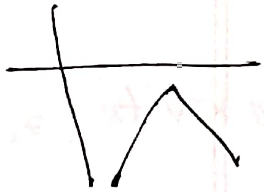
solution: Reform the problem entirely





$$error = \sum (Y_{actual} - Y_{approximate})^2$$

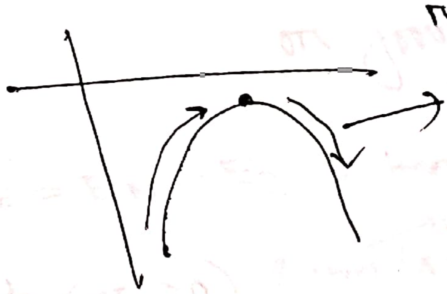
$$error = \sum |Y_{actual} - Y_{approximate}|$$ →



(iii) Simulated Annealing: Modified version of Hill climb

→ Initially there will be a temperature value
↳ initially temp high $\xrightarrow{\text{will get}}$ low

→ যখন temp → High, তখন কোনো state top এ reach করলেও যে নিচে নামা accept করবে।



→ temperature value high থেকে low হতে থাকে।

Simulated Annealing Algorithm:

1. $C = C_{init}$ // C is the current state and C_{init} is the initial state.

2. For $T = T_{max}$ to T_{min} // here, T is the control temperature for annealing

3. $E_C = E(C)$ // Here E_C is the Energy i.e. utility or goodness value of state C

4. $N = Next(C)$ // N is the next state of current state C

5. $E_N = E(N)$ // E_N is the energy i.e. utility or goodness value of state N

6. $\Delta E = E_N - E_C$ // Here, ΔE is the Energy difference

7. If $(\Delta E > 0)$ // child (N) এর energy value parent (C) এর থেকে বেশি যদি $\Delta E > 0$ হয়।

8. $C = N$ // Accept the child

Parent/initial state

C

E(C)

energy value of C

9. Else if ($e^{\Delta E/T} > rand(0,1)$) //

// suppose, $\Delta E = \cancel{-1}$, $T_{max} = 100$ and $T_{min} = 2$

// $e^{\Delta E/T} = 0.99$ for $T_{max} = 100$

// $e^{\Delta E/T} = 0.60$ for $T_{min} = 2$

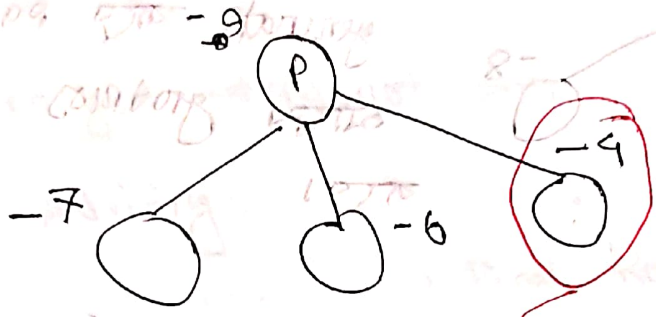
10. C = N

11. End

Hill climb variations: Not in slides

→ Steepest Hill climb

P = Parent



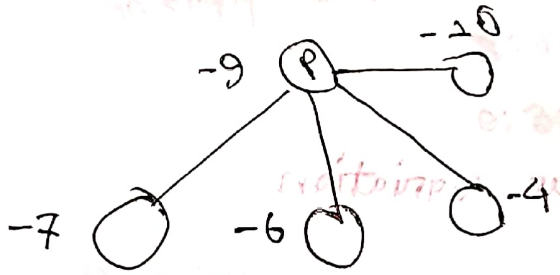
→ Parent এর চেয়ে ভালো child ই better কারণ Goodness value বেশি।

best child accept করবে।

→ Steepest hill climb very best child accept করে।

→ " " multiple direction থাকলে, যেই direction এর upward progress টা সবচেয়ে steep সেই direction accept করে।

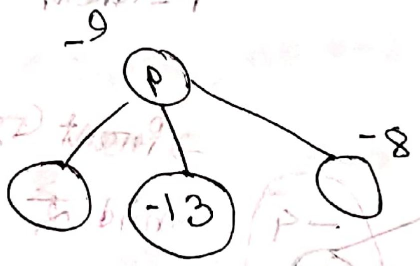
(ii) Stochastic Hill climb:



→ Instead of choosing the best, random child choose করবে। (best গুলার মধ্যে)

→ যদি কোনো child ~~worst~~ worst goodness value provide করে, সেটা considered হবে না।

(iii) First choice Hill climbing



→ Parent একটা একটা করে children generate করে parent এর সাথে goodness value check করে।

→ যখন কোনো child generate করে child এর goodness value যদি parent এর goodness value এর চেয়ে better হলে সেই child কেই accept করবে।

→ First যে child parent এর চেয়ে better হয় সেটাই choose করে।

→ এই Algo যখন অনেক available direction থাকে তখন helpful.

→ Faster compared to other variation

Session 9 - Constraint Satisfaction Problem (CSP)

→ 8 puzzle এ Agent এর goal state এর path জানা necessary.

→ কিছু Problem এ Path matter করে না।

→ End goal matters, not path

Pre-Advising:

↳ Routine → Final Routine matters

→ End result matters

Some Terminologies of CSP:

1. Variable: যেই slot গুলোতে value Assign করা হয়।

$\boxed{\text{value}}$ → Pre-Advising এ slot হলো বিভিন্ন Routine এর বিভিন্ন slot,

variable

an empty basket

8:00 - 9:20 am

9:30 - 10:50 am

2. Domain:

- Variable এ value assign করতে হয়।
- Variable এ যত possible values assign করা যায়, সেই value গুলোর set কে domain বলে।

Ex: Pre-Advising: Routine এর various slot গুলোতে বিভিন্ন course এর section assign করা হয়।

→ সমস্ত course এবং তার available section কে যেটাকে domain বলে।

Domain: { CSE422 [section 1-12], — }

- এই section গুলো empty slot এ assign করা যায়।

Domain: set of possible values যেগুলো variable এ assign করা যায়।

3. Constraints:

- No two courses in the same time slot
- No final clash
- Seat availability
- Preferred Instructor
- Pre-requisite

→ এই constraint গুলো fulfill করে Domain থেকে values নিয়ে variable এ assign করা হয়।

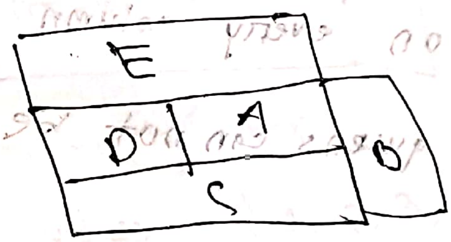
- Final Routine এ সব constraint satisfy করতে হয়।

End Result Matters → CSP
not path

4. Goal:
- সব constraint fulfill করে final end state.
- Goal can be multiple

Map Coloring

- 5 Regions



- প্রত্যেক grid এ RGB (এর মধ্যে) যেকোনো একটা color assign করা যাবে।

শর্ত / constraint: কোনো adjacent node এ same color থাকা যাবে না।

এখানে:
variable: সেই সব গুলোতে value assign করা হয়।
এখানে map position গুলো variable (A, B, C, D, E)

{A, B, C, D, E}

Domain: Variable গুলোতে যে possible values Assign করা যায়।
- RGB হচ্ছে values
Domain: {Red, Green, Blue}

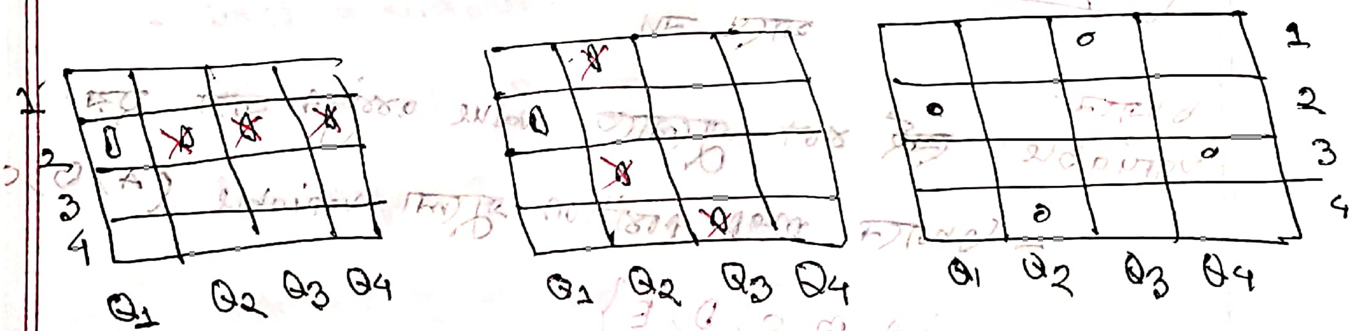
Constraint: { $A \neq C$, $A \neq D$, $A \neq E$, $A \neq B$, $D \neq E$, $B \neq C$ }

তাই [No same color in adjacent nodes]

Goal: Various color combination fulfilling all the constraints.

CSP more examples: ~~n-queen~~

- n×n board — 4×4 board
- n numbers of queens — 4 queens
- a queen on every column
- multiple queens can not be in same horizontal level
- " " " " " diagonal level.



Here:

Variables: যেসব empty slot এ value assign করা হয়।
- যেখানে queen গুলোকে বসানো হবে।
- column গুলো variable

$\{Q_1, Q_2, Q_3, Q_4\}$

Domain: variable গুলোতে যে possible value assign করা হয়।

- প্রত্যেকটা column এ queen কে row তে assign করতে হবে। {1, 2, 3, 4}

constraint: ① No horizontal / vertical / Diagonal ~~cla~~ clash

goal: end ~~state~~ fulfilling all the constraints.

Another CSP problem: sudoku ***

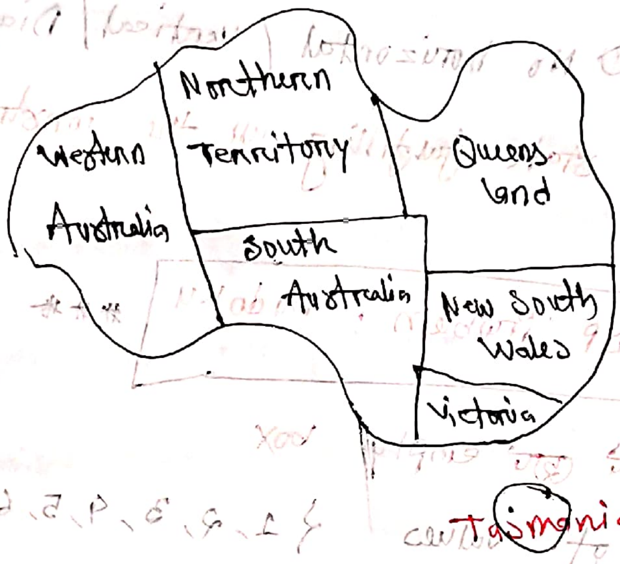
Variables: ছোট ছোট empty box

domain: set of values {1, 2, 3, 4, 5, 6, 7, 8, 9}

How to solve CSP Problems

Common Approach: Back-tracking ***

- Map coloring Problem



Variables: - ~~[illegible]~~ Region {WA, NT, SA, Q, V, T}

Domain: {R, G, B}

Constraints: No same color in adjacent regions.

goal: Using back-tracking method

→ কোনো CSP Problem এ Backtracking approach apply করার আগে সেই problem এর একটা Constraint graph form করা লাগে।

Graph: set of nodes connected with edges ***

In constraint Graph:

- Variable গুলোকে node হিসেবে রাখা হয়।
- প্রত্যেকটা node এর variable (here regions) এর মধ্যে যে constraint based যে connection সেটাকে edge দিয়ে define করা হয়।

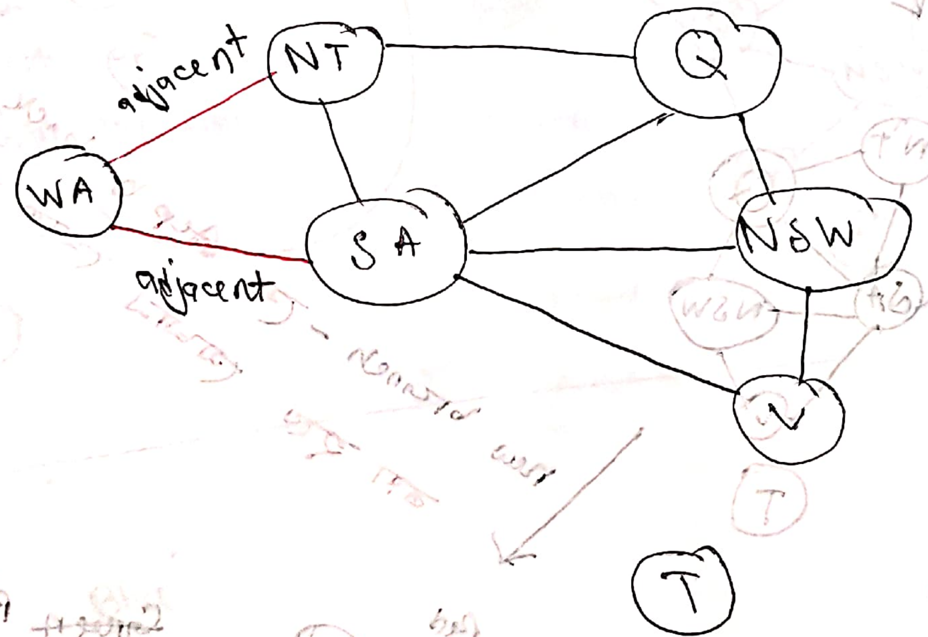
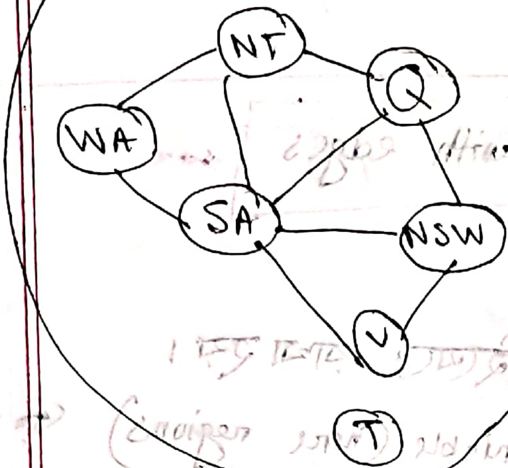


Fig: Constrainted Graph

Now Apply Back-Tracking on Constrained Graph.

multiple option যখন থাকবে

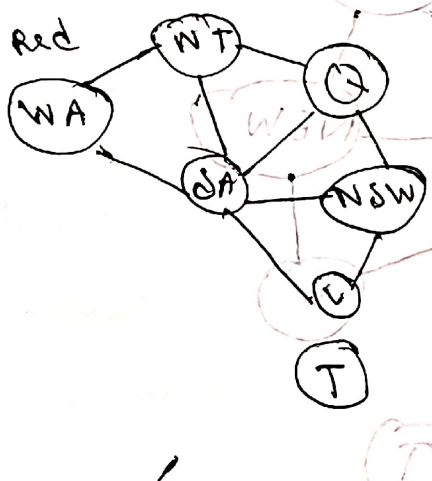
1. 

- Take variable (a region) and assign a color

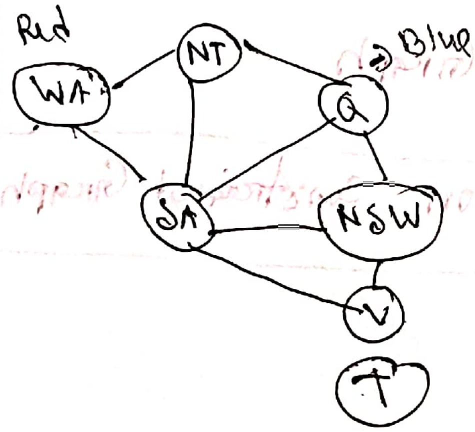
In purest back-tracking algorithm,

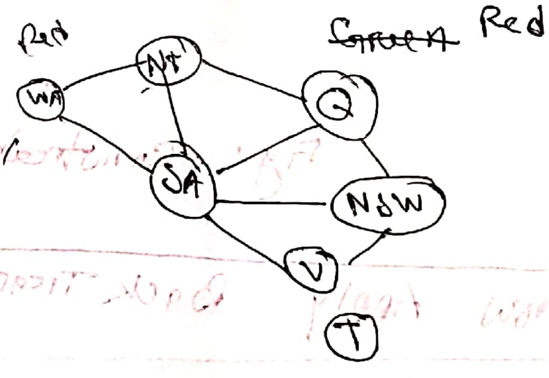
- Randomly যেকোনো variable pick
- Variable এ কোন color assigned হবে সেটাও Randomly selected হবে

- সব কাজ constraints গুলো satisfy করে করতে হবে।

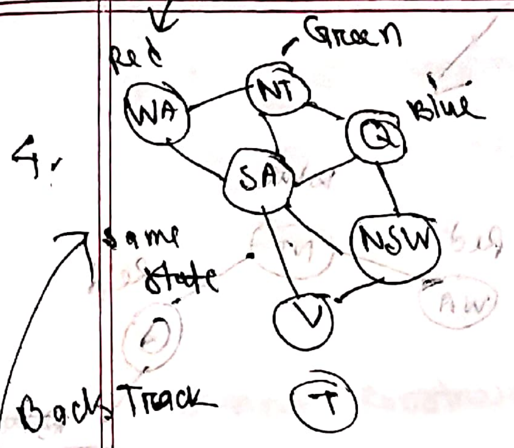
2. 

new branch - যে step এ back যেতে হবে সেখানে তৈরি করা হবে

3. 

3' 

→ এক step করে Back করতে হবে like DFS

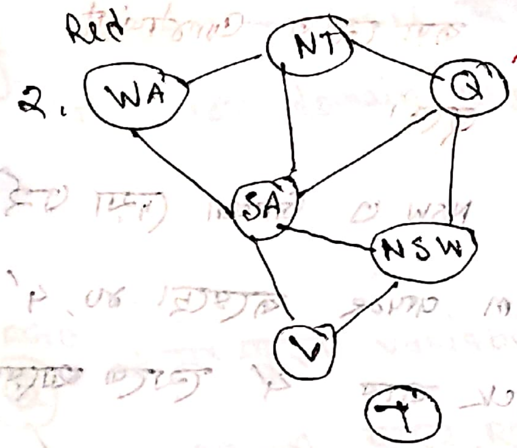


→ Backtrack to step 3 as step 4 is same as step 5.

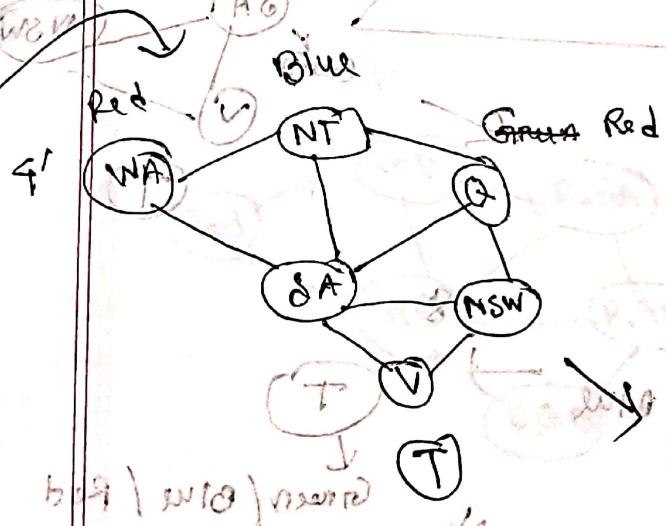
In step 4/3, NT এ green ছাড়া বসানোর মত color আর নেই। so, আরও এক step back-track করে step 2 এ যেতে হবে।

can not assign any color

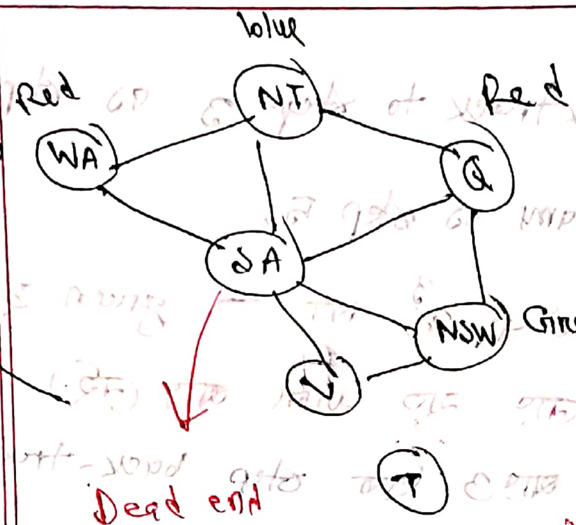
dead end → leaf nodes



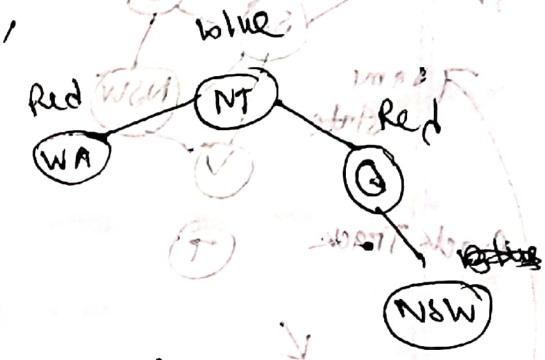
→ এখানে Blue বসানোতে step 5 এ problem হয়েছিল।

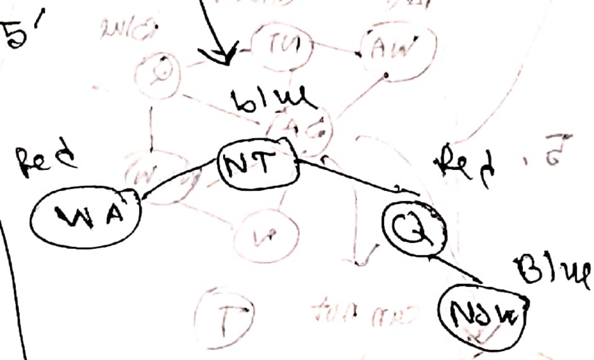


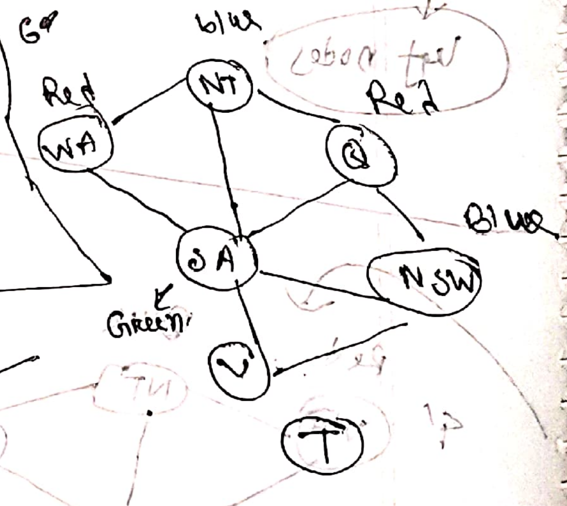
4'
to
Back
track

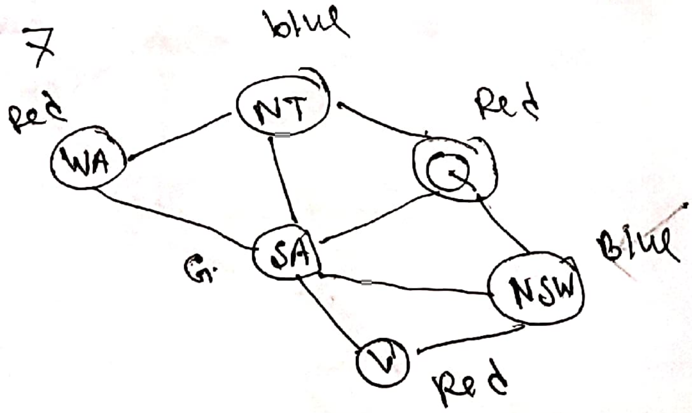


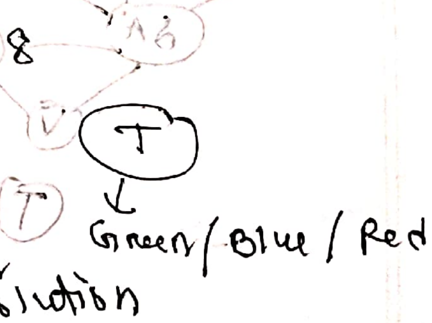
Dead end
since SA এ যে color ই
assign করা হোক constraint
break হবে।
4' এ NSW এ Green দেয়া তেই
Problem arise করেছে। so, 4'
এ back করে 4' থেকে আরেকটা
branch create করে NSW এ
জন্য possible value assign করতে
হবে।











→ Most Constrained Variable (Russia)

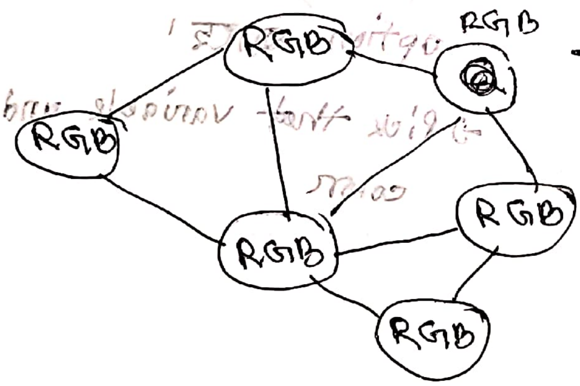
→ যে variable এর least available option আছে Domain থেকে value choose করার।

→ Most constraining variable (USA)

- যে variable সবচেয়ে বেশি সংখ্যক variable এর domain reduce করবে।

→ Least constraining value : যে value অন্যান্য variable এর least domain reduce করবে।

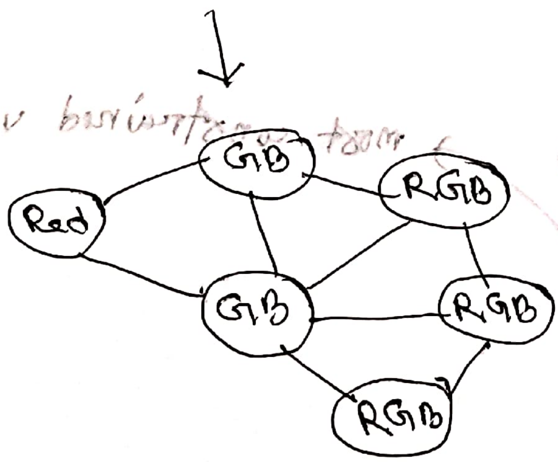
Most Constrained :



→ সব গুলো variable এর সমান option আছে, RGB

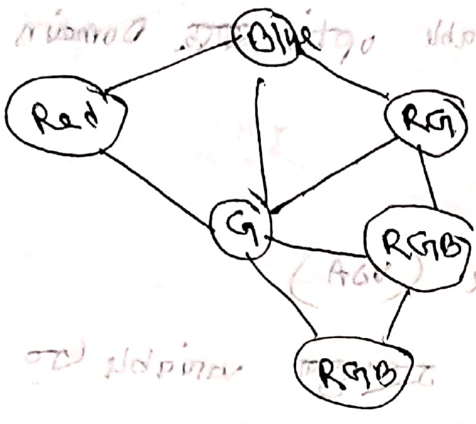
→ যেকোনো একটা variable pick করে value Assign করা যাবে।

↓



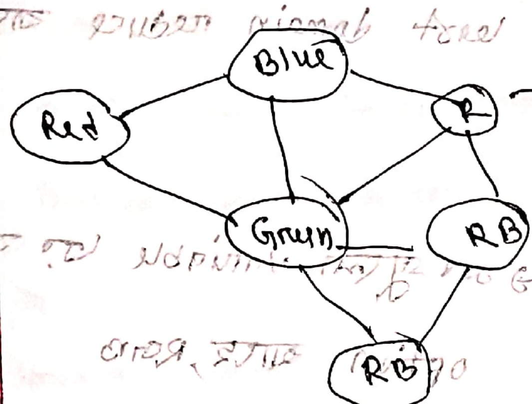
→ এখানে most constrained variable ২ টা (GB) আছে, যাদের হাতে option কম।

→ দুইটার মধ্যে যেকোনো একটা pick করে value assign করতে হবে।



এখানে least ~~constra~~ available option আছে G circled variable টার।

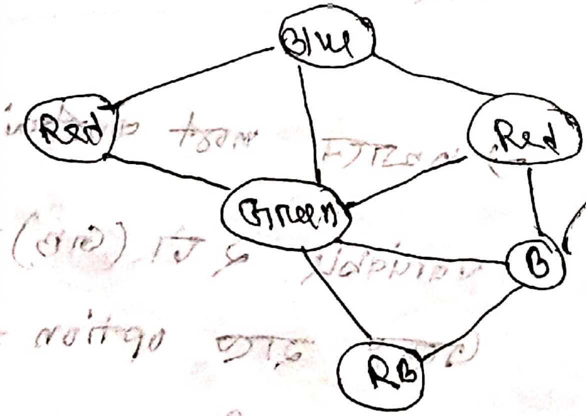
→ most constrained variable হিসেবে এটা picked হবে এবং তারে value Assign করা হবে।



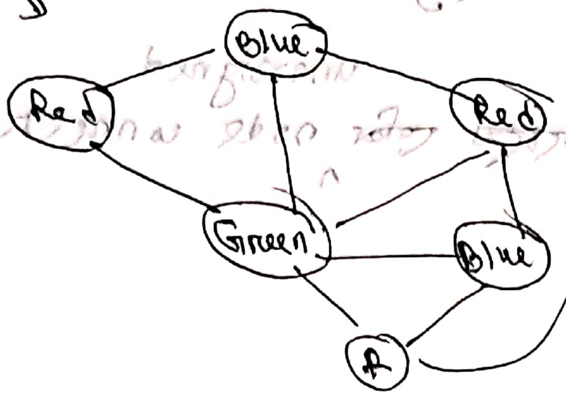
→ এটা most constrained variable

→ কেননা এটাতে least available option আছে।

→ Pick that variable and assign color

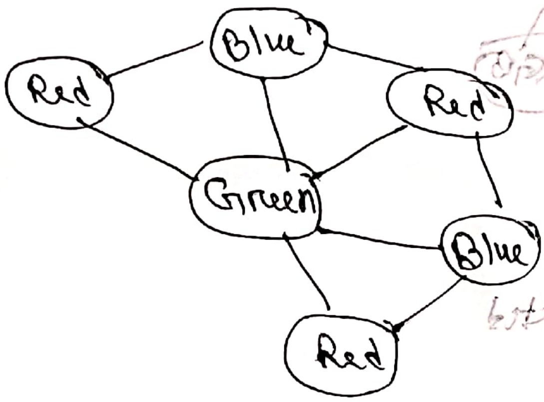


→ most constrained variable



→ most constrained ~~var~~ variable
কারণ এর হাতে option least,

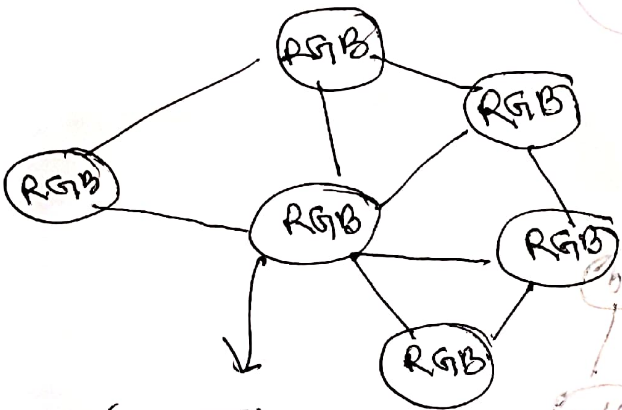
↓



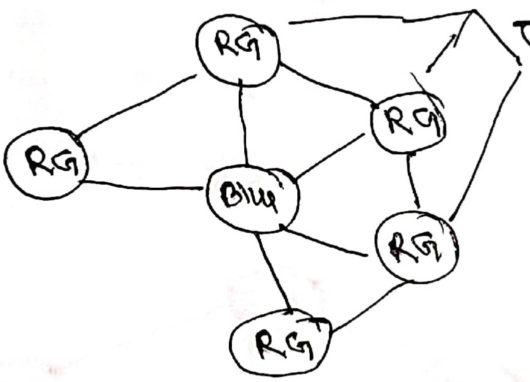
Most Constraining Variable

যে variable সবচেয়ে বেশি সংখ্যক variable এর domain reduce করে।

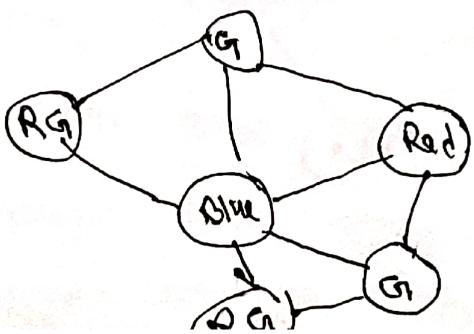
- যে node এর সাথে সবচেয়ে বেশি unassigned node connected



এর সাথে
সবচেয়ে বেশি node connected



যেকোনো একটা
pick করা যাবে।



Tie হলে   Degree Heuristic

Most Constrained variable > Most Constraining variable >

least constraining value

Session 10 – CSP 2

TRZ

→ Failure early detect করার mechanism কে Constraint Propagation বলে।

Constraint Propagation

1 step — (node consistency)

Multi step — (Arc Consistency)

1 step :

– দুইটা adjacent node এ same value (এই value বাদে অন্য value available থাকবে না) থাকলে ঐ সেই combination এ graph node consistence না।

Multi step: Node consistency এর জন্য 1 step চালাতে হবে

1st step: যেখানে একটা value বসানো যায় শুধু সেখানে সেই value বসিয়ে তার adjacent node গুলো থেকে ওই value বাদ দিতে হবে।

2nd step: কোন variable এ শুধুমাত্র একটা value দেয়া যায় তাকে pick করতে হবে এবং তাতে সেই value assign করতে হবে এবং তার adjacent node থেকে

সেই value বাদ দিতে হবে।

~~-- যেমন: যদি~~

3rd step:

Repeat step 2

- যদি কোনো node এ কোনো color/value assign না করা গেলে arc consistency break হবে।

Uninformed search

* যখন search algorithm এ Goal state কে expand (
(একটা node expand মানে তার সকল child কে visit করা)
করার পর search বন্ধ করতে হবে।

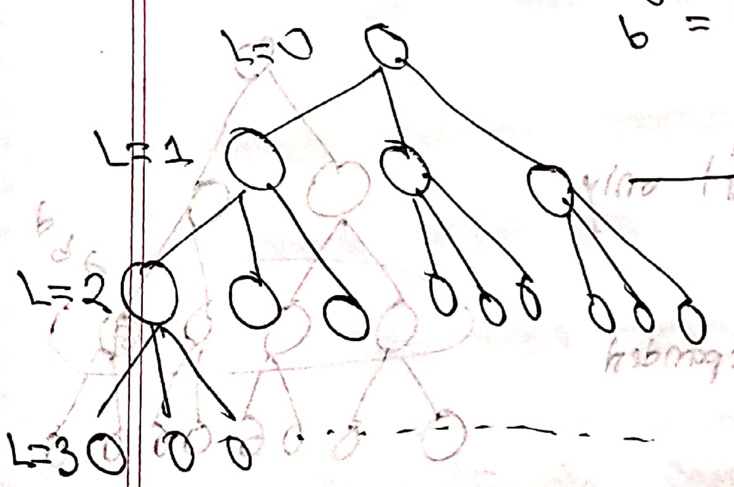
→ Only BFS এ goal node visit করে search বন্ধ করা যায়।

Queue তে push
↓
FIFO

BFS

1. Time complexity: $O(b^d)$

d = level of the goal node
= যে level এ goal node আছে সেটা d কে বলবে।



$b^0 = 3^0 = 1$ here, $b = 3$

$\rightarrow b^1 = 3^1 = 3$

$\rightarrow b^2 = 3^2 = 9$

$\rightarrow b^3 = 3^3 = 27$

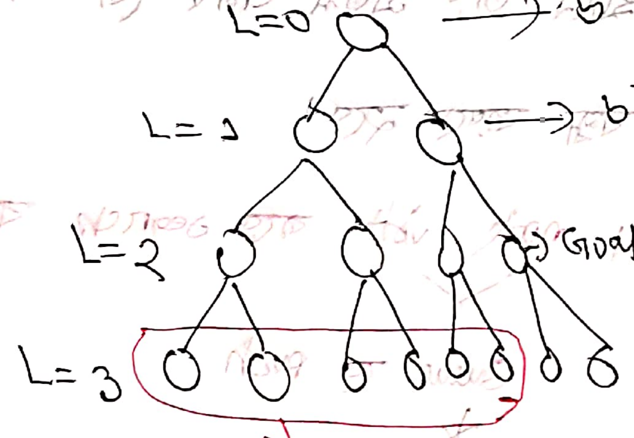
সব level এর node যোগ করে: $b^0 + b^1 + b^2 + b^3 \dots\dots \boxed{+ b^d}$

highest

→ goal expansion $O(b^d)$ → goal visit

$\to O(b^{d+1})$ when?

$d = 2$ = depth of the goal



$\to b^0$

$\to b^1$

Goal $\to b^2 \to O(b^{d=2})$

visit এর সময় terminate

$\to b^3 = b^{2+1} = b^{d+1}$

total node visited in L=3

Goal কে expand করলে এসব node extra visit করা লাগে।

$b^3 - 2$

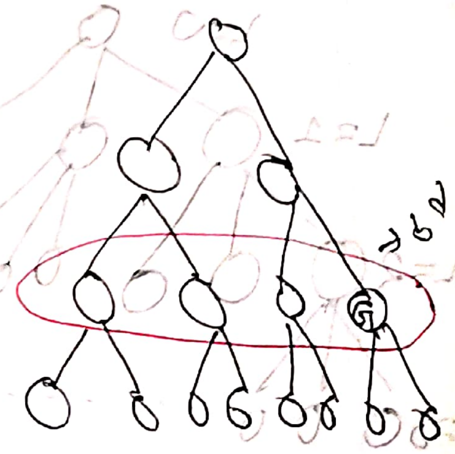
$b^{d+1} - b = b^{d+1}$

$$b^0 + b^1 + b^2 + \dots + b^d + (b^{d+1} - b) = b^{d+1}$$

Space Complexity:

$O(b^d) \to$ node visit only

$O(b^{d+1}) \to$ node expanded



BFS with stack → DFS

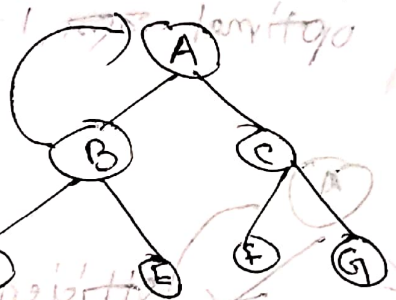
BFS

3. Completeness: Goal থাকলে always ensure করবে যে goal find করবে।

BFS → Complete

i. Tree এর সব node ই visit করে।

ii. loop থাকলেও complete



	A
A	BC
B	CADE
C	ADEFG
A	DEFGBC
D	EFGBC
E	FGBC
F	GBC
G	BC

Tree search:

- No track of already visited node
- কোনো node visit, expand হয়েছে কিনা সেটার track রাখা হয় না।
- DFS tree search not complete — loop এ পড়বে।

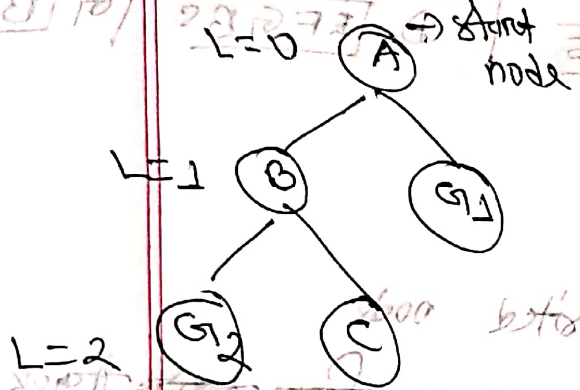
Graph search: - কোনো node already visited কিনা সেটার track রাখা হয়।

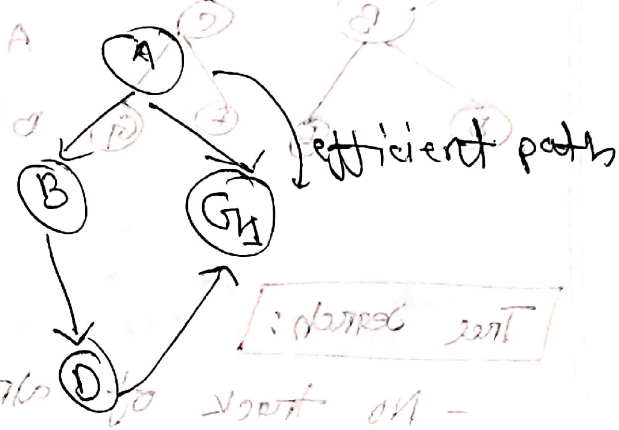
- loop avoid করার জন্য Graph search use হয়।

Tree / Graph search → BFS Complete

4. Optimality:

- যদি multiple goal node থাকে বা কোনো goal node এ reach করার multiple path থাকে, সেক্ষেত্রে most efficient goal / path find করার guarantee দিতে পারলে সেই search কে optimal বলে।



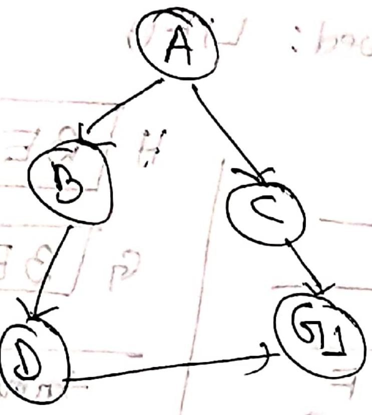


- All possible graph এর জন্য optimality এর guarantee দিতে হবে।

BFS → Optimal

→ multiple goal যদি থাকে, most optimal goal always shallowest depth এ থাকবে।

shallowest
↓
lowest



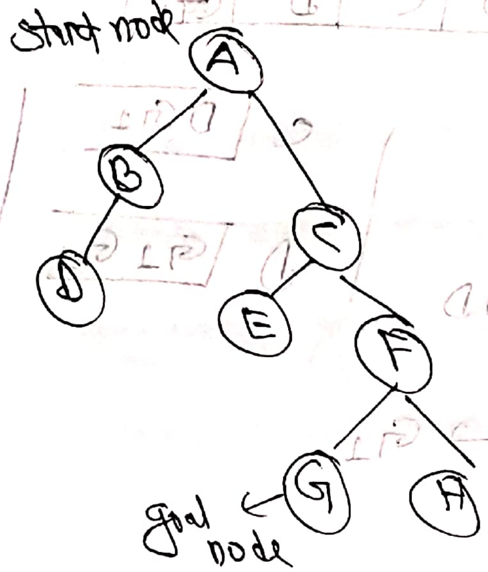
A	A	D	C	
A	B	C	D	G_1

	A
A	B C
B	C ~~B~~ D
C	D G_1
D	G_1 G_1
G_1	G_1

A → C → G_1

Depth First Search (DFS)

- Stack based
- Back tracking (Recursion)



Fringe :→

Stack based : LIFO

	A
A	B C
C	B E F
F	B E G H
H	B E G
G	B E

terminate

~~G~~
F
~~E~~
C
~~D~~
~~B~~
A

Backtracking :

A

~~G~~
F
~~E~~
C
~~D~~
~~B~~
A

G
E
B
D

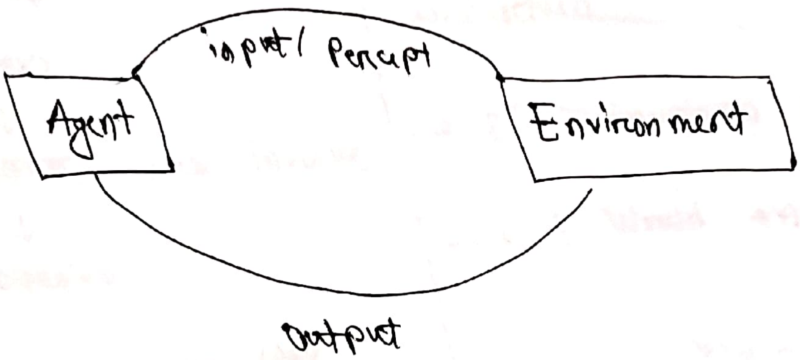
1. BFS, DFS:

m = max depth

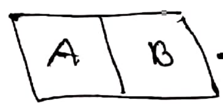
d = depth এর difference

session 3 Environments TR2

Properties of Environments



1. → Fully or Partially Observable



→ Partially Observable কারণ এইখানে একটা বাকি Tiles এর situation observe করতে পারে না।
↓ at a time

↓ যদি কোনো camera বা sensors দ্বারা পুরো environment observe করতে পারে, then fully observable
→ at a time

Fully Observable : chess

Partially " : Poker

2. single Agent or Multiagent :

↓ এই Agent এর আশেপাশে অনেক Agent থাকলেও তারা এর সাথে Related না। Lamppost Agent

↓ একটা Agent এর সাথে আরেকটা Agent Related. self driving car

3. Deterministic or Stochastic
Uncertain

Chess
→ যেখানে Luck involve
→ Sudoku

→ Opponent এর play consider করা যাবে না

→ Given agent এর action কে consider করতে হবে।

→ আমার নিজের Action এর কোনো uncertain outcome আছে কিনা, থাকলে Deterministic

→ Dies involved যেমন game ex. Ludo

→ Real World environment

→ Snake Ludo → single agent Stochastic

→ Action এর determined outcome থাকে এবং যদি আগে থেকে predict করা যায় তাহলে Deterministic.

→ " " multiple " " এবং যদি uncertain outcome থাকে then stochastic.

4. Episodic or sequencial:

environment এর প্রত্যেকটা state একটার সাথে আরেকটার connection নাই।

→ Image classification
- Brain MRI

chain / connected sequence

→ chess
→ most of the game
→ Real world environment

5. Discreate or Continuous

limited perception	unlimited perceptions → infinite
→ chess - 10^{120} perception	→ Real World
→ Poker (Depends)	→ Poker (depends)

6. Static or dynamic

chess

- certain change হয় থাকবে না।	- certain change থাকবে।
- " " সেই Agent এর perspective থেকে environment discuss করা হবে, সেই Agent Action নেয়ার সময় আসতে হবে।	

→ other agent

RL / learning Agent

7. Known or Unknown (depends on Agent)

Known:

G. M. Arafat Rahman

19101498 Section: 10

Assignment 1

1. Physical Address = Base Address $\times$ 10h + offset

$$\begin{array}{r} 33330 \\ -1111 \\ \hline 3221F \end{array}$$

3221F / 10h = Fraction

As this is a fraction value, it will not be valid segment number. As the offset is given, we can not increase to match the valid segment number.

As this is a non overlapping segmentation, for this the segment will be 30000h / segment 3.

∴ highest segment address =

$$\begin{array}{r} 30000h \\ FFFF \\ \hline 3FFFFh \end{array}$$

2. (a) Physical Address = 91ABh $\times$ 10h + 2340h

= 91AB0 + 2340h

= 93DFD

(b) Physical Address = AA10h $\times$ 10h + 18<3h

= AB963h

(c) FFFF8

1. FFFF0 → need an offset 0007

2. FFFE0 → need an offset 0017

3. FFFD0 → need an offset 0

(12)

(c) FFFF8

1. FFFF0 → need an offset 0008

2. FFFE0 → need an offset 0018

3. FFF D0 → need an offset 0028

3. ~~AX = EEEEh →~~ 1110 1110 1110 1110

BX = BBBBh → 1011 ~~1011~~ 1011 1011

1010 1010 1010 1101

(111) (11)

3. (a) AX = EEEEh ⇒ 1110 1110 1110 1110

BX = BBBBh ⇒ 1011 1011 1011 1011

1 1010 1010 1010 1001

~~AF = 1, there is a carry from lower nibble to higher nibble~~

ZF = 0, The output is not zero

PF = 1, LSB 8 bit has 4 = even numbers of 1

SF = 1, MSB bit is 1

CF = 1, there is a carry from 16th to 17th bit

OF = 0, because there is a carry from 15th to 16th bit and also 16th to 17th bit.

(b) AX = 8000h = 1000 0000 0000 0000

BX = 1234h = 0001 0010 0011 0100

1001 0010 0011 0100

3 1's

SF = 1, MSB bit is 1

PF = 0, 3 = odd number of 1's in lower Byte

ZF = 0, the result is not 0

CF = 0, there is no carry from 16^{th} to 17^{th} bit

OF = 0, there is no carry from 15^{th} bit or

no carry from 16^{th} to 17^{th} bit

4. ADD AX, [10h]

This is a direct addressing of addressing data. Here, the data address (offset) is directly given. So, it need to fetched using DS base register. To calculate the effective address location for the given instruction: DS × 10h + offset

AB12h × 10h + 10h = AB130h

5. (a) For smallest segment address: FFFEF - FFFF

= FFF0

EFFF : FFFF

2nd largest;

The largest segment address is

FFFE0

So the second largest segment address = FFFD0

FFFD0 : 001F

b. 2n

smallest = 00000

0000 : 0002

For 2nd largest:

We know lowest segment address is 16 byte.
16 slots

But 00000 + 16 ~~byte~~ slots will exceed the physical address given. So second largest segment address is not possible.

6. As 8086 has 20 bit address bus and address bus define the number of locations can be accessed using those 20 bits,

∴ Total slots using 20 bits: $2^{20} = 1048576$ slots

(vi)

Each slot can contain 1 byte

∴ 1048576 byte

1048576/1024 = 1024 KByte

1024/1024 = 1 MByte

7. 2^{28} = 268435456 slots

⇒ 268435456 Bytes

⇒ 268435456/1024 = 262144 KByte

⇒ 262144/1024 = 256 MByte

Initial physical address = 0000000h

last physical address = FFFFFFFh

8. 89 80 → First 2 Byte

1000 1001 1000 0000

$\underbrace{1000\ 100}_{\text{opcode for MOV}}$ 1 → D, 1 → W, 10 → MOD, 000 → REG, 000 → R/M

VII

(iv)

69 10h

$\underbrace{69}_{}$ $\underbrace{10}_{}$ —— high byte displacement in Byte 4

↓

Low Byte displacement in Byte 3

69 → 0110 1001

10 → 0001 0000

D = 0, so register used as source

W = 1, so 16 bit source register

MOD = 10 so memory with 16-bit displacement

REG = 000, W = 1, REG = 000, so AX

R/M = 000 corresponds to [BX] + [SI] + d16

∴ find assembly language:

MOV 1069h [BX+SI], AX

(VIII)

(b) MOV DI, [BP+42h]

100010	1	1	01	111	110
opcode	D	W	MOD	REG	R/M

DI 16 bit destination index register.

so, D = 1, W = 1

[BP+42h] → memory with 8 bit displacement, so

MOD 01

[BP+d8] → R/M → 110

$\underbrace{1000}_{8}$ $\underbrace{1011}_{B}$ $\underbrace{0111}_{7}$ $\underbrace{1110}_{E}$

Probability-1: FPS

Probability Theory for Inference

Random Variable: যে variable গুলো যাদের একটা Range of values থাকে এবং যে Range of values থেকে যেকোনো value কে নেওয়ার জন্য pick করতে পারে এবং that will be associated with a probability.

For example: Random variable = Today's weather

↓ Range of ~~variable~~ values

Sunny, Cloudy, Rainy

এই Random variable এর Range of values এর মধ্যে থেকে যেকোনো value এর probability যদি same হয়, তাহলে

$p(\text{today's weather} = \text{sunny}) = \frac{1}{3}$

$p(\text{today's weather} = \text{Rainy}) = \frac{1}{3}$

Random Variable

- Discrete (↑ most)
  - ↓ Possible values গুলো discrete
- continuous
  - ↓ Any real number between 0 and 1

~~Discrete~~

Discrete Random Variable

Boolean Random Variable

{true, false}

$p(\text{true}) = .7$

then $p(\text{false}) = 1 - .7$

$= .3$

Multi Random Variable

1. addition always 1
2. $0 \le \text{Possible} \le 1$

Joint Probability of a and b

Probability

$$p(a \vee b) = p(a) + p(b) - p(a \wedge b)$$

$$\& \quad p(a \cup b) = p(a) + p(b) - p(a \cap b)$$

intersection/and

set theory

Probability Theory

- Random variables : • Alarm, Burglary, Earthquake
- Domain - Boolean (like these), discrete, continuous

- Atomic event: Complete specification of state, কোনো particular ঘটনা ঘটার সম্ভাবনা। যেমন:

(ক) বাসায় alarm আছে; চোর এসে

(i) চোর আসে জন্য alarm বাজে
(ii) চোর না আসে জন্য alarm বাজে
(iii) চোর আসছে জন্য alarm বাজে নাই

প্রত্যেকটাই একটা করে Atomic event

→ যতগুলো Probable case হতে পারে, each of them are Atomic event.

Prior Probability: (→ আগে) কোনো situation দেখার আগে যদি আমি জানি যে আমার বিশ্বাস কতটুকু।

যেমন: আমার class এ পুরুষ student 46% জন একটা আমি যদি class শুরুর আগে বলি based on আমার অভিজ্ঞতা থেকে, তাহলে আমার class গুলোর জন্য

$P(\text{student} = \text{man}) = .46$ → Prior Probability

Joint Probability: দুইটা ঘটনা বা দুইটা Random variable এর দুইটা value একই সাথে হওয়ার Probability.

→ প্রতি ১০ বাসায় 1 বাসায় চুরি হয় (একটা এলাকায়:

$P(\text{চুরি}) = \frac{1}{10} = 0.1$ — Prior Probability আমার বাসায় চুরি হওয়া আমি দেখিনি কিন্তু আমার বাসায় চুরি হওয়ার Probability $= 1$ এর মধ্যেই আমার বাসায় কোন প্রমাণ না, যেহেতু আমার বাসার observation আমি জানি না।

দুইটা ~~জিনিস~~ একইসাথে ঘটার Probability

Joint Probability

Random variable → Boolean (both)

P(Alarm, Burglary) -

চোর আসছে তখন alarm বাজছে

চোর আসছে তখন alarm বাজে নাই

	alarm	¬ alarm
burglary	.09	.01
¬ burglary		

চোর আসে নাই তখন alarm বাজছে

চোর আসে নাই তখন alarm বাজে নাই

Computing Conditional Prob : Joint Probability

$$\boxed{P(\underbrace{a}_{\text{cause}} \mid \underbrace{b}_{\text{effect}}) = \frac{P(a \wedge b)}{P(b)}}$$

$\boxed{P(b) - \text{Normalizing constant}}$

→ Joint Probability দুইটা ঘটনা একইসাথে ঘটার সম্ভাবনা কিন্তু কেউ কারো উপর নির্ভরশীল না। একটা ঘটনা

Bayes' Theorem: Relation between Conditional and Joint Probability.

কারণ আরেকটা ঘটেনি, দুইটা ঘটনা Just একইসাথে ঘটে।

- কিন্তু conditional probability হচ্ছে একটা ঘটনা ঘটলে তাহলে আরেকটা ঘটার সম্ভাবনা

Ex: P(বৃষ্টি হবে | আকাশে মেঘ থাকলে)

বৃষ্টি হবে → effect

আকাশে মেঘ থাকলে → cause

Normalizing Constant:

P(b) → এটা কোনো observation এর উপর dependent না, এটা always constant থাকবে।

Prior belief

Chain Rule

Product Rule → কিভাবে conditional থেকে Joint এ যেতে হয়

Conditional Probability: $P(a|b) = \frac{P(a \wedge b)}{P(b)}$

Now, $\boxed{P(a \wedge b) = P(a|b)P(b)}$ → Product Rule

Conditional Probability থেকে Joint Probability

Marginalizing:

→ যখন কোনো Random variable এর total probability জানতে চাই।

marginal

Random variable → Capital letter → variable itself.

→ $P(B)$

Random variable এর value গুলো নিতে পারে।

small letter → possible values

$P(b)$

to the weather itself.

→ $P(W)$

$P(w)$ → can either be rainy, cloudy, sunny

small w

$P(B)$ হচ্ছে $P(B)$ এর প্রত্যেকটা value এর যোগফল

B = sunny, B = cloudy, B = snow

$$\rightarrow P(B) = \sum_a P(B, a)$$

given a (অন্যেকটি Random variable এর value)

$$\rightarrow P(B) = \sum_a P(B|a) P(a)$$

Bayes' Rule

Bayes' Rule: কিভাবে একটা conditional থেকে আরেকটা conditional Probability বের করা হয়।

$\rightarrow P(a|b)$ — a: effect, b: cause

তাহলে P(a given b) এর কিভাবে version এ লিখতে পারি।

cause থেকে → effect উল্টো

$$\underset{\text{Posterior}}{P(a|b)} = \frac{\overset{\text{likelyhood}}{P(b|a)} * \overset{\text{Priors}}{P(a)}}{\underset{\text{Normalization}}{P(b)}}$$

Priors — degree of belief, আমার নিজস্ব কোনো observation দেখার আগে effect এর Probability কত হতে পারে?

$$P(\overset{a}{\text{disease}} \mid \overset{b}{\text{diagnosis}}) = \frac{P(\text{diagnosis} \mid \text{disease}) * P(a)}{P(\text{diagnosis})}$$

/ P (disease)

আমি কোনো diagnosis result দেখার আগে in general population এ সেই disease হওয়ার সম্ভাবনা কত?

$\boxed{\text{Normalization:}}$ P(diagnosis) =

Likelyhood: আমার যদি disease টা হয় তাহলে diagnosis positive আসার সম্ভাবনা কত?

Posterior: Patient এর diagnosis result positive আসছে, তার actually সেই disease থাকার সম্ভাবনা কত?

$$\boxed{P(\text{Cause} \mid \text{Effect}) = \frac{P(\text{Effect} \mid \text{Cause}) * P(\text{Cause})}{P(\text{Effect})}}$$ ★★★

$\boxed{\text{Chain Rule}}$ → n সংখ্যক variable

→ Product Rule এর generalize version.

$$P(X_1, X_2, \ldots X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1,X_2) \ldots\ldots$$

↓

Joint Probability

$$= \prod_{i=1}^{n} P(X_i | X_1, \ldots, X_{i-1})$$

$\boxed{\text{Independence}}$ X and Y independent if and only if

$$\forall x,y : P(x,y) = P(x)P(y)$$

↓ For all

↓ Joint Probability of x and y

যদি for all x এবং all y, Joint Probability of x and y becomes equal to the multiplication of individual probability of x and individual probability of y, then variable x and y are independent to each other.

X and Y are conditionally independent given Z if and only if :

→ কোনো variable আরেকটা variable এর উপর পুরোপুরি independent না, relation আছে কিন্তু condition দিয়ে দিলে condition এর উপর base করে তারা independent.

$\boxed{\text{Conditional Independence}}$

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

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

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

Guilty or not?

$$P(\text{criminal} \mid \text{found guilty}) = \frac{P(\text{found guilty} \mid \text{criminal}) \times P(\text{criminal})}{P(\text{found guilty})}$$

$P(\text{criminal}) = .008$ → Prior Probability

$P(\text{found guilty} \mid \text{criminal}) = .98$

$P(\text{not guilty} \mid \text{not criminal}) = .97$

$$= \frac{.98 \times .008}{.0376} = 0.2085$$

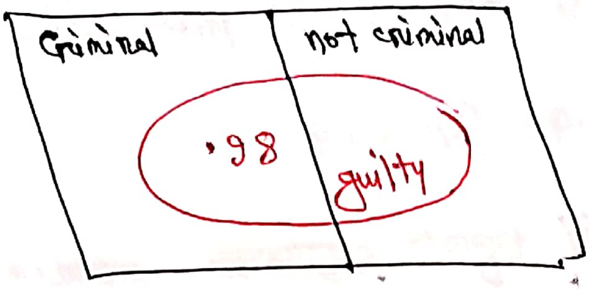
Prior Probability

$P(\text{not found guilty} \mid \cancel{\text{not}}\ \text{criminal}) = .02$

$P(\text{guilty} \mid \text{not criminal}) = .03$

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

$$= .03 + .98 = 1.01$$



$$P(\text{found guilty}) = P(\text{criminal} \cap \text{found guilty}) + P(\text{not criminal} \cap \text{found guilty})$$

$$= 0.0376$$

$$=$$

$$P(\text{criminal} \cap \text{found guilty}) = P(\text{found guilty} \mid \text{criminal}) * P(\text{found guilty})$$

$$= P(\text{criminal}$$

$$P(\text{found guilty} \mid \text{criminal}) = \frac{P(fg \cap c)}{P(\text{criminal})}$$

$$P(fg \cap c) = P(fg \mid \text{criminal}) * P(\text{criminal})$$

$$= .98 * .008$$

$$P(c) = .008$$

$$P(nc) = 1 - .008$$

$$= 0.992$$

$$P(nc \cap fg) = P(fg \mid nc) * P(nc)$$

$$= .03 * 0.992$$

$$=$$

Bayes' Theorem

Snigdho sir

২টি event independent হলে:

$$P(A\cap B) = P(A)\cdot P(B)$$

target hit করার probability

* একজন আর্চারের target hit করার ঘটনা A, $P(A) = .8$

* আরেকজন " " " " B, $P(B) = .5$

→ প্রথম জন ২য় জনের target hit করবে, এই ঘটনা দুটো পরস্পর ~~নির্ভর~~ dependent না, independent.

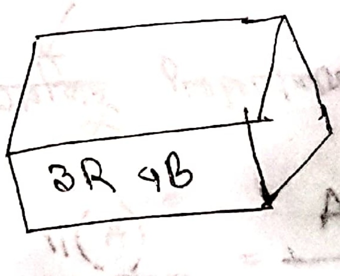
→ দুজনই target hit করতে পারবে তার probability

$$P(A\cap B) = P(A)\cdot P(B) = .8\times .5 = .4$$

A ঘটবে এবং B ও ঘটবে

Dependent Event:

→ এই Box থেকে 1 টার পর আরেকটা ২টা বল তুলতে হবে।



প্রথম বল Red, এই ঘটনা R_1

২য় " Red, " " R_2

$$P(R_1) = \frac{\text{Total no. of Red balls}}{\text{Total no. of balls}}$$

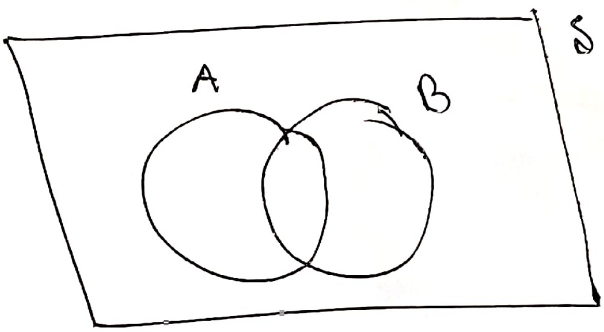
→ Event গুলো একটা set কে Represent করে।

→ যদি ঘটনা independent না হয়, then conditional probability এর concept ব্যবহার করি।

Conditional Probability:

$P(A|B)$ → A given that B occured. B ঘটনা যদি already ঘটে গিয়ে থাকে, তবে A ঘটার Probability কত?

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



S → Sample space

সবকিছু Proportional বা আনুপাতিক হারে থাকতে হবে।

$$\frac{n(A)}{n(B)} = \frac{Area(A)}{Area(B)}$$

বর্তমানে যেকোনো মানুষের কোভিড আক্রান্ত হওয়ার সম্ভাবনা

(#) $P(Covid) = .004$

কোভিড আক্রান্ত হয়ে থাকলে মারা যাওয়ার সম্ভাবনা,

$$P(D|C) = .05$$

যেই কোভিড আক্রান্ত হওয়ার ঘটনা

ঘটনা D কারোর death

$P(C \cap D)$ এখানে মানুষের শুধু মারা যাওয়ার chance এর ক্ষেত্রে কোভিড আক্রান্ত হয়ে মারা যাওয়ার chance

কোনো ব্যক্তি কোভিড আক্রান্ত হবে এবং মারা যাবে? ক্ষেত্রে

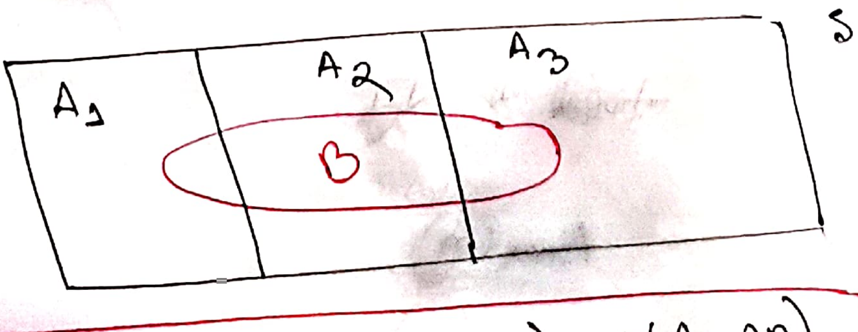
We know, $P(D|C) = \dfrac{P(D \cap C)}{P(C)}$

যেহেতু সুতরাং D এবং C পরস্পর dependent. তাই এখানে conditional এর concept আসবে।

$$\Rightarrow P(C \cap D) = P(D|C) * P(C)$$

$$= .05 * .004 = .00020 \text{ (Ans)}$$

Law of Total Probability



$$\boxed{P(B) = P(A_1 \cap B) + P(A_2 \cap B) + P(A_3 \cap B)}$$

Europe → E → White Hair → 80% educated

Black Hair ↓ 70% educated

Brown Hair ↓ 90% educated

একজন Random European কে নিলে তার educated হওয়ার probability কত? $P(\text{Educated}) = ?$

Black	Brown	White
70%	90% Educated	80%

তাদের সংখ্যা Equal না

$x \quad 2x \quad 3x$

Black : Brown : White $= 1 : 2 : 3$

Europe এ total মানুষ আছে $= x + 2x + 3x = 6x$ জন

$x \rightarrow 70\% \rightarrow .7x$ educated

$2x \rightarrow 90\% \rightarrow 2x \times .9 = 1.8x$ educated

$3x \rightarrow 80\% \rightarrow 3x \times .8 = 2.4x$ "

Europe এ total educated মানুষ আছে $= .7x + 1.8x + 2.4x$

$= 4.9x$

$$P(\text{educated}) = \frac{4.92}{6} = .82 = 82\%$$

Chain Rule / Product Rule

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

$$\Rightarrow P(A\cap B) = P(A|B)\cdot P(B)$$

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

same

$$\Rightarrow P(A\cap B) = P(B|A)\,P(A)$$

$$\boxed{\begin{aligned} P(A\cap B) &= P(B|A)\cdot P(A) \\ &= P(A|B)\cdot P(B) \end{aligned}}$$

- এই ২ভাবে প্রকাশ করা যায়

Rain T → Thunderstorm

(*) $P(R) = .002$

$P(R \cap T) = .0004$

$$P(T|R) = \frac{P(R \cap T)}{P(R)} = \frac{.0004}{.002} = \frac{1}{5}$$

(*) একটা Box এ, 5 Red ball, 10 Black ball

We draw two balls from the box without replacement (দুইটি বল তুলব তবে যে ball টা প্রথমে তুলব সেটা আর box এ রাখব না)

P (second ball drawn = Red) ? → $P(R_2)$?

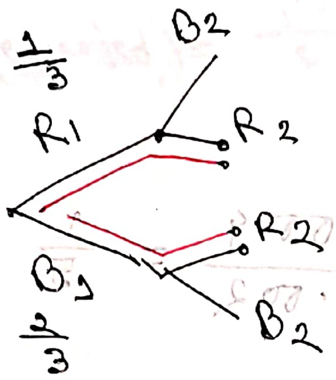
R_1 → প্রথম বল Red হওয়ার ঘটনা/event

B_1 → " " Black " " / "

R_2 → ২য় " Red " "

B_2 → " " Black " "

R_1	B_1
$R_1 \cap R_2$	$B_1 \cap R_2$



$$P(R_2) = P(R_1 \cap R_2) + P(B_1 \cap R_2)$$

$$P(B_1) = \frac{10}{15} = \frac{2}{3}$$

$$P(R_1) = \frac{5}{15} = \frac{1}{3}$$

↓ total ball

$$P(R_1 \cap R_2) = P(R_1|R_2)\,P(R_2)$$

$$= P(R_2|R_1)\,P(R_1) \rightarrow$$ এটা use করে

$$\frac{4}{14}$$

then, $P(R_1 \cap R_2) = P(R_2|R_1)\,P(R_1) = \frac{2}{7} \times \frac{1}{3} = \frac{2}{21}$

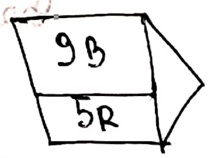
$$P(R_2|R_1) = \frac{4}{14} = \frac{2}{7}$$

Now, $P(B_1 \cap R_2) = P(B_1|R_2)\,P(R_2)$

$$= P(R_2|B_1)\,P(B_1)$$

$$= \frac{5}{14} \times \frac{2}{3}$$

$$= \frac{10}{42} = \frac{5}{21}$$



$$P(R_2|B_1) = \frac{5}{14}$$

$$P(R_2) = P(R_1 \cap R_2) + P(B_1 \cap R_2)$$

$$= \frac{2}{21} + \frac{5}{21} = \frac{7}{21} = \frac{1}{3} \quad \text{(Ans)}$$

* Approximately 1% of women have breast cancer. A woman with BC has 90% chance of a positive test result; while a woman without BC has 10% chance of false positive result. What is the probability that a woman has BC given that she just had a positive test?

Ans: ~~P(PT (Positive test) = .90~~

Breast Cancer = BC

$P(BC) = .01$

$P(PT \mid BC) = .90$

$P(\text{Not Positive test} \mid BC) = .10$

$P(\text{Positive Test} \mid \text{Not } BC) = .10$

$P(\text{Negative Test} \mid \text{Not } BC) = .90$

$\therefore P(B|T)\ ? = \dfrac{P(B \cap T)}{P(T)}$

$P(B\cap T) = P(B|T)\,P(T)$

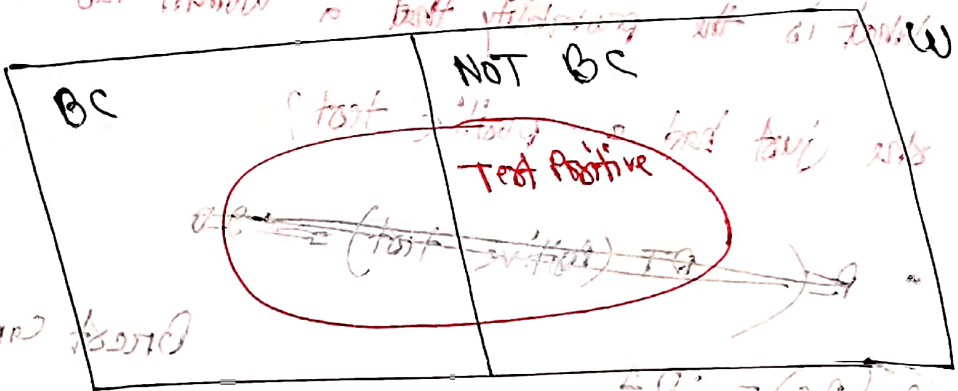
$= P(PT|BC)\,P(BC)$

$P(PT|BC) * P(BC) = .9 \times .01 = \cancel{P(BC\cap PT)}\ P(BC\cap PT)$

$= .009$

$\boxed{\text{Now } P(PT)\ ?}$

মহিলাদের



(Using Law of total Probability $P(PT) = P(BC\cap PT)$
$+ P(\text{Not } BC\cap PT)$)

$= .009 + .099 = .108$

$P(BC^{c}\cap T) = P(PT \mid \text{Not } BC) * P(\text{Not } BC)$

$= .10 * .99 = .099$

$P(BC) = .01$

$\therefore P(\text{Not } BC) = 1 - .01 = .99$

$$P(BC|PT) = \frac{.009}{.108} = \frac{1}{12} \; (And)$$

$$P(A|B \cap C \cap D) = \frac{P(A|B) * P(A|C) * P(A|D)}{\cancel{P(B) * P(C) * P(D)}}$$

$$P(A|B) * P(A|C) * P(A|D) = P(A \cap B) * P(A \cap C) * P(A \cap D)$$

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

Genetic Algorithm FYS

Target : Integently ভালো solution বের করা।

↓

Uninformed search

↓

Informed search

↓

Local search – Hill climbing

↓

Constraint satisfaction problem – Back-tracking Algo

Genetic Algorithm

→ Generalize algorithm based on biology.

Backtracking

→ For 4 Queen

→ 8/16 Queen হলে tree অনেক বড় হয়।

→ Result থাকলে solution দিবে

→ Exponentially tree বড় হবে।

→ 4 Queen থেকে 5 Queen এ যেতে হলে same tree again and again build করতে হবে।

GA

→ correctly solution দিতে পারে।

GA → 5 steps

→ begin with k randomly generated states (population)

→ each state (individual) is a string over some alphabet (chromosome)

→ fitness function (bigger number is better)

→ cross over

→ mutate (evolve)?

GA → John Holland thinking

→ অনেক গুলো random state আছে, এখান থেকে k randomly generate states নিতে হবে (population)

→ random state গুলোকে alphabet এর কোনো combination হিসেবে ব্যাখ্যা করতে হবে। It will look like a chromosome.

কোনো random state → string দিয়ে প্রকাশ → chromosome

→ random state গুলো কত ভাল সেটা বের করতে হবে।

Heuristics → এমন কোনো heuristic

মাপবে যেটা idea দিবে কোনো random state কত ভাল?

ওটা যথেষ্ট ভাল হলে ওটা থেকে আরও ভাল হওয়া সম্ভব হলে আরও ভাল করতে হবে এক একটা পর্যায়ে ভাল হতে হতে perfect হয়ে গেলে mutation লাগে যায়।

4. Crossover

5. Mutation – child এর genetic mapping এ কোন একটা single change যেখানে significant জিনিস সরিয়ে বদলে দেয়া হবে।

Mutation ও একটা heuristic এর উপর ভিত্তি করে করা হবে।

Step-1

→ k random states

→ একটা population থেকে sample নিতে হবে

→ sample ~~থেকে~~ random state

→ k = 2 হলে দুইটা randomly generated state নিতে হবে

একটা father এবং আরেকটা mother

Step-2

→ father এবং mother (random states, k = 2) কে individually string আকারে প্রকাশ করতে হবে। father এবং mother এর string representation গুলোকেই একটা বলে chromosome.

Step-3

→ এই state গুলো কতটুকু ভাল সেটা determine করার জন্য fitness function দরকার।

→ ৪টা Queen কে নিয়ে কয়টা unique জোড়া create করা যায়? → Unique pairs

$$\boxed{{}^8C_2 = ২৮ \text{ টা}}$$

→ ২৮ এখানে fitness function

→ fitness function এর value ২৮ হলে কোনো clash নাই।

→ fitness " = 0, all pairs have clashes.

Step-4: CrossOver

→ crossover এর জন্য প্রথমে crossover point determine করতে হবে।

→ যদি previous knowledge থাকে যে কোথায় crossover করলে ভাল হবে, সেই knowledge ব্যবহার করব। আর

তাহলে random crossover point select করতে হবে।

→ crossover point father ও mother, দুইজনেরই same point ও হতে হবে।

crossover এর আগে

Chromosome of father: 3 8 4 | 7 2 3 2 5

" " mother: 3 8 5 | 7 1 6 1 5

Crossover point = 3

2 more chromosomes/ sequences

After crossover

offspring 1: 3 8 4 | 7 1 6 1 5

offspring 2: 3 8 5 | 7 2 3 2 5

So 4 sequences are:

3 8 4 7 2 3 2 5
3 8 5 7 1 6 1 5
3 8 4 7 1 6 1 5
3 8 5 7 2 3 2 5

→ crossover করার পরে offsprings ভাল না খারাপ তা fitness function দিয়ে check করতে হবে।

↓
goodness

START

Generate the initial population

then each population কে string এ convert করতে হবে (chromosome).

compute fitness (প্রত্যেকের) (chromosome গুলো— যেমন father and mother)

REPEAT

selection (কোনগুলোর crossover করতে হবে)

crossover

Mutation (প্রয়োজন হলে)

compute fitness (offsprings এর)

UNTIL population has converged

STOP

যদি generated sequence এ কোনো obvious ঝামেলা থাকলে সেটাকে solve করতে হবে।

5. Mutation → Drastic idea

Before Mutation:

Offspring 1: 3 8 4 7 1 6 1 5

(Row 1 → column 5; Row 1 → column 7)

Offspring 2: 3 8 5 7 2 3 2 5

→ Offspring 1 এর column 5 এ, column 7 এ 1 মানে: দুইটা column এ same row তে row 1 এ Queen আছে। Queens থাকলে definitely there is a conflict.

→ So, সেটাকে change করতে হবে।

→ ঠিক করার জন্য অন্য কোনো row খালি আছে কিনা সেটা check করে খালি থাকলে একটা Row 1 কে change করে সেই row তে shift করতে পারি।

→ Offspring 1 এর ক্ষেত্রে row 2 unused. So 2 তে একটা row 1 এর Queen কে shift করলে obvious conflict এড়ানো যাবে।

→ ২ টা বসানোর কারণে নতুন conflict generate হলেও obvious problem রয়েছিল যেটা তাড়ানো যাচ্ছে।

After mutation:

Offspring 1 : 3 8 5 7 1 6 2 5

$\boxed{\text{Mutate করার পর fitness calculate করতে হবে}}$

GA Challenges

1. কোন ভাবেতে crossover করতে হবে সেটা Heuristically বের করা।

2. কোন " mutate করতে হবে সেটা Heuristically বের করতে হবে।

3. কোন fitness function ব্যবহার করে 1 বা 2 point কে judge করব সেটা বের করা।

Decision Tree

Information Gain:

* Dataset এর বিভিন্ন feature কে:

G.M. Arafat Rahman

CSE 422 Section: 11

Assignment 2

Question A

① If an item is selected will be denoted as 1.
and if not selected its will be denoted as 0.

∴ Encoding of the problem and initial of 4 different chromosomes are:

	A	B	C	D	E	F	G	H	
ch1 →	1	0	0	1	0	0	0	1	→ W=11
ch2 →	1		1	0	0	1	0	1	→ W=12
ch3 →	1			0	0	0	0	1	→ W=10
ch4 →	0	0	0	0	1	0	0	1	→ W=9

here: ch = chromosome

W = Weight

H is present in all 4 different chromosomes.

(11) (1)

(2) Appropriate fitness function would be the sum of collected items reward. But the total weight of the selected items must not cross 12.

∴ Fitness function:

if $12 - \text{weight} \geq 0$,

fitness score = sum(selected items reward)

else, fitness score = -sum(selected items reward)

fitness score of our chromosomes:

ch1 → 1 0 0 1 0 0 0 1 ; weight = 1+8+2 = 11 ≤ 12

∴ fitness score = 30+40+7 = 67 = fitness score 1

ch2 → 1 1 1 0 0 1 0 1 ; weight = 1+2+3+4+2 = 12 ≤ 12

fitness score = 20+5+10+25+7 = 67 = fitness score 2

ch3 → 1 1 0 0 0 0 1 1 ; weight = 1+2+5+2 = 10 ≤ 12

fitness score = 20+5+4+7 = 36 = fitness score 3

(iii)

ch4 → 0 0 0 0 1 0 0 1 ; weight = 7 + 2 = 9 ≤ 12

fitness score = 15 + 7 = 22 = fitness score 4

Natural selection will choose the 2 fittest chromosomes according to fitness score. so the selected chromosomes:

ch1 → 1 0 0 1 0 0 0 1 → fitness score 1 = 67

ch2 → 1 1 1 0 0 1 0 1 → fitness score 2 = 67

(3) Doing single point crossover, randomly selected point = 3 (c|D)

parent 1 = ch1 →	1 0 0	1 0 0 0 1
parent 2 = ch2 →	1 1 1	0 0 1 0 1

offspring 1, of 1 → 1 0 0 0 0 1 0 1

offspring 2, of 2 → 1 1 1 1 0 0 0 1

(iv)

(iii)

(i) Performing mutation randomly,

of 1 → 1 0 $\boxed{0}$ 0 0 1 0 1

of 1' = 1 0 1 0 0 1 0 1

Again of 2 → 1 $\boxed{1}$ 1 1 0 0 0 1

of 2' → 1 0 1 1 0 0 0 1

Here, while mutating, we have to exclude the position of H because it should be preserved.

Now, of 1' → 1 0 1 0 0 1 0 1, weight = 10 ≤ 12

fitness score, fs1' = 30 + 10 + 25 + 7 = 62

of 2' → 1 0 1 1 0 0 0 1, weight = 14 ≰ 12

fitness score fs2' = −(30 + 10 + 40 + 7)
= −77

fs1' = 62, fs2' = −77 (Ans)

2

Question B

① For visiting each city just once there will be 6 edges. I will encode the edges like this ABG which indicates A to B and B to G.

So following this encoding technique randomly generated population of 4 chromosomes:

ch 1 → A F B C D G H

ch 2 → A C F B E G D

ch 3 → E B F A D G C

ch 4 → D A F C G E B

② Here appropriate fitness function will be negative of the edge costs. By calculating fitness score difference of all the edges and cost of selected edges:

cost of all the edges:

$= (A \leftrightarrow B) + (A \leftrightarrow F) + (A \leftrightarrow C) + (A \leftrightarrow D) + (B \leftrightarrow C) + (B \leftrightarrow F) + (C \leftrightarrow D) + (C \leftrightarrow E) + (B \leftrightarrow E) + (C \leftrightarrow G) + (D \leftrightarrow E) + (D \leftrightarrow G) + (C \leftrightarrow F)$ +

$+ (E \leftrightarrow G)$

$= 15+10+14+7+3+6+8+8+3+5+2+17+9+11$

$= 118$

Now, ch 1 → A F B C D G H

cost of ch 1, $c_1 = 10+8+3+8+9+11 = 49$

fitness score, $f_1 = 118 - 49 = 69$

ch 2 → A C F B E G D

cost of ch 2, $c_2 = 14+5+8+6+11+9 = 53$

fitness score, $f_2 = 118 - 53 = 65$

ch 3 → E B F A D G C

cost of ch 3, $c_3 = 6+8+10+7+9+2 = 42$

fitness score, $f_3 = 118 - 42 = 76$

ch 4 → D A F C G E B

cost of ch 4, $c_4 = 7+10+5+2+11+6 = 41$

fitness score, $f_4 = 118 - 41 = 77$

(VII)

∴ So naturally selected two best fit chromosomes

Ch4 → D A F C G E B ($f_4 = 77$)

Ch3 → E B F A D G C ($f_3 = 76$)

(Ans)

(3) Performing single point cross over from third index_

parent 1 = Ch4 → D A F | C G E B

parent 2 = Ch3 → E B F | A D G C

offspring 1, of 1 → D A F A D G C

offspring 2, of 2 → E B F C G E B

Now if the offsprings are eligible as solution. The problem states, every city must be visited and cannot be visited more than once. In of 1 city B and E are not visited while A, D are visited twice.

(VIII) (12)

similarly in of 2, city A and D are not visited while city B and E are visited twice. For this reason, they are not eligible as a solution.

(9) Usual method of mutation will not work here. In our problem, every city must be visited only once. If we try to mutate any gene from our offsprings we will see that mutation can't make them eligible.

of 1 → D A F A $\boxed{D}$ G C

if we change D with any non-occurring city. B/E, there still exists one city A which appeared twice and either B or E will not be visited.

If we replace D with any occurring cities, this also keeps the chromosomes away from

(IX)

being eligible. If we replace by G,

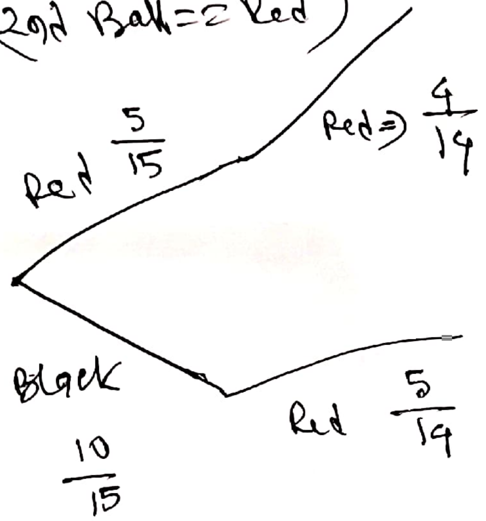
Of 1' → DAFAGGC,

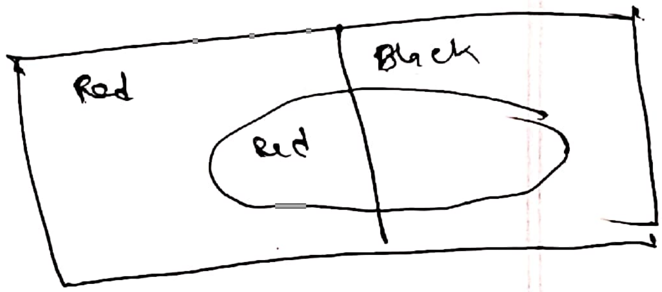
Here, now G appears twice.

If we randomly select a gene that appears only once to mutate, the chromosome will be far away from being eligible. For these reasons, usual method of mutation will not work here.

P(2nd Ball == Red)

$$\frac{5}{15} \times \frac{4}{14} + \frac{10}{15} \times \frac{5}{14}$$





$$P(\text{2nd Ball} == \text{Red}) = P(\text{Red} \cap \text{Red}) + P(B \cap R)$$

$$= \frac{5}{15} \times \frac{4}{14} + \frac{10}{15} \times \frac{5}{14}$$

20

=

mostly random

GA

heuristically

3 major Problems:

(i) কোন জায়গাতে cross over করতে হবে সেটা বের করা

(ii) কোন position এ mutate " " " " " "

(iii) কোন fitness function use করে এই ২টা task কে judge করব সেটা

GA uses:

(i) যেখানে Resource allocate করতে হয়, we have limited amount of resources

Pseudo-code of GA

```
START
    Generate the initial population
    compute fitness

    REPEAT
        selection
        cross over
        Mutation
        compute fitness
    UNTIL population has converged
STOP
```

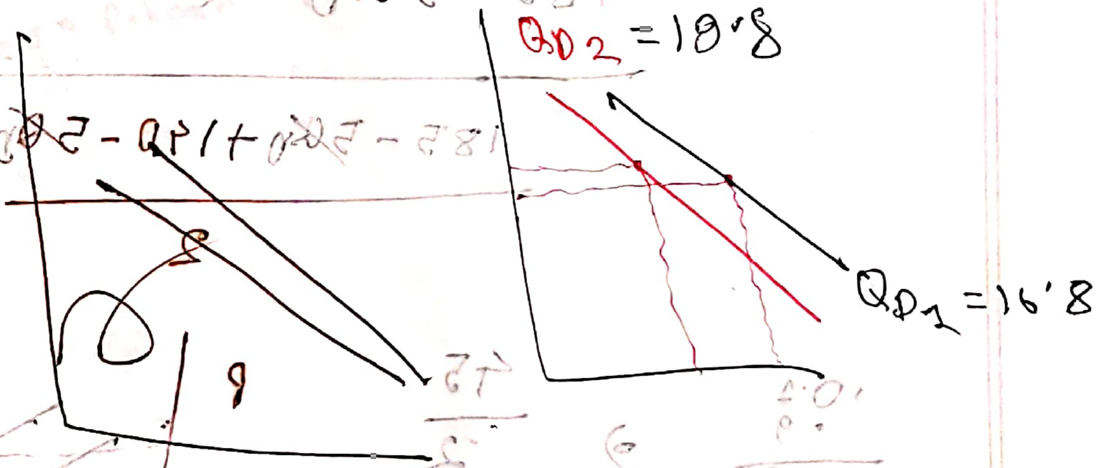
3. $P = 132 - 2Q_D$ $\quad 132 = 2Q_D =$

$82 = 132 - 2Q_D$

$\Rightarrow$

5412 Tk

1.



$P = $

$P_1,\ 56 = 140 - 5Q_D$

$Q_{D_1} = 16.8$

$P_2,\ Q_{D_2} = 25.8$

$\boxed{5.03}$

d_2 Prev: $P = 140 - 5Q_D$

new $\Rightarrow P = 185 - 5Q_D$ 56Tk to 86Tk per litre

$140 - 5Q_D = 185 - 5Q_D$

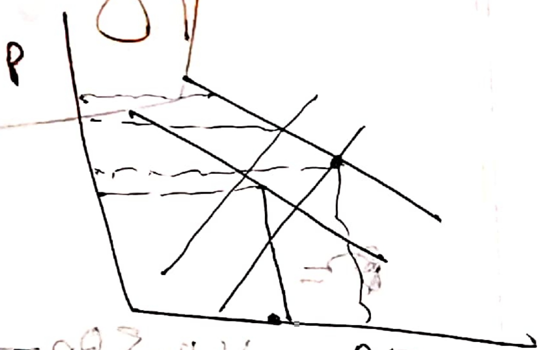
$\Rightarrow$

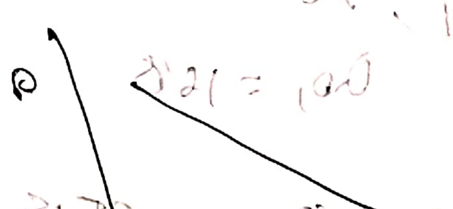
$$\frac{185 - 5Q_D - (140 - 5Q_D)}{\frac{185 - \cancel{5Q_D} + 140 - \cancel{5Q_D}}{2}}$$

$\Rightarrow \frac{45}{2}$

$= 3.30$

(ii)





So, what should we do?

সমাজ

→ CS হিসেবে আমরা society ঠিক করতে পারি না।

BIO → tagging

শেখ মুজিবুর Rahman n

11

কারণ

Mental health → bad thing → BD data

ML use হয়নি।

→ মুসলিম → Majority muslim country তো problem হবে না।

→ Alexa → Accent বুঝতে পারতো না

(110)

Bias from Research :

→ New scientist → Eng

English Rich → because there is a lot of works on English.

Worked on English

→ অন্য lang attention পায় না।

→ Bias → a lot of bad things happen

Career কেন

→ Overexposure, over amplication → ...

9

king → Ruler

woman → cooker

Pre-trained Model → nowadays friends
↓ Representation of words
BERT use thin

Bias from Models:

yellow banana | green banana

ML → color based
↓
it amplifies on what it learned

→ কোন time এ woman angry comment
সেটাকে ML শেখে learn করে।

⑧

→ অন্যটা ছিন্নিয়েই focus করে অন্যকে judge করা ঠিক না।

→ Bigger picture নিয়ে concerned হতে হবে।

Linguistic mismatch:

→ যে label করতেছে সে যদি সেই lang জানে না তাকে then সে ভুল translate করবে।

Ex: cats and dogs ⟶ Bangla

Bias from Rep: one of the most challenging

→ semantics → meaning of words

Vectors → a collection of numbers

→ 1D

→ cat and dog এর vectors close থাকা দরকার

data নিলে ML predict করবে Male

সবাই artistic.

→ Women were not allowed to do many things

Bias from Annotations:

→ tagging / labelling data

→ labelled data

Sampling:

confirmation bias: you get attracted ~~word~~ you know.

A person rides motorcycle $\xrightarrow[\text{you think}]{\text{Punk}}$

Insensitivity to sample:

correspondence: Restaurant এ এসে loud কথা বলা ~~বললে~~ ~~আমরা~~ তাকে rude বলি।

⑥

Speech Recognition Model:

→ If you only collect data from USA, UK, AU the model will not understand Indian English

② selection: Whoever collects the data, he is biasing for selecting data

③ Overgeneralization: ~~তোমাকে~~ দেখেই BD তে building বলা

④ Under-rep:

Voice recog

⑤ Stereo: ① Data collector এর ~~নিজেদের~~ biasness

⑥ Historical unfairness: 15-16 th century থেকে কোনো Female artist কখনো নাম করেনি, বলে না। এখন ML কে জানাও 15-16 th এর

164

③

ML learns → social media data, our daily talks,

Questions: got natural

- Reflections of our society, It shows human are bad.

Predictive Policing Model:

- data bias হলে more black person কে arrest করবে.

Sources of Bias in NLP:

Bias from Data:

Massive biased data → ML will learn wrongly,

→ Reporting: we collected data from only native english people

4

Effects of bias in NLP models:

(i) Gay → ~~so~~ much hate - ML identifies them as toxic

(ii) Podcast → Bad sentence

* Muslim → in toxic context
* Deaf → toxic comment कुछ कि बोला?
* mental illness → ~~joking~~, ~~attention~~ grabbing

We think these

ML learns from human data. It learned that mental illness as bad.

ML → learning wrong:

(i) ML model is wrong

(ii) We giving wrong data, society is wrong.

③

Gender Bias:

~~কোনো কিছু~~ gender centric হলে সেটা ML model bias ~~করে ফেলে~~।

A man can be a programmer

A woman " not be a "

Masculine → Toy

Doll → Girls

Racial Bias: ① no good stereotypes

ML —warned→ White leader → Black

Vernacular → offensive

Religious Bias:

ML Model: Chr → Neutral; Muslim → terrorist.

sentences →

Cultural integrity nosto করে।

1

Banana → Yellow / green

→ We identify what we know already from our experience, knowledge, media coverage

→ মডেলকে biasing training data দিলে সে হিসেবে identify করতে পারবে না। এখানেই NLP bias হবে।

→ We identify a "nurse" with their gender identity as female

→ "Female" Doctor, "Black" doctor

$\underbrace{\hspace{12em}}_{\text{Problematic}}$

→ "Islamic" → stereotypes as terrorist / ISIS

"Jews" → Philanthropic

→ male person can be a "nurse" too

→ these things do hamper in NLP / ML models

2

Male Jobs　　Female Jobs

Gender: Male vs Femal

ML

Amazon: AI was trained on a dataset because the tech data was mostly male, তাই female candidates reject করছিল। just they were female

Racial: Black doctor, ~~White~~ Doctor

প্রথম দুইজন white দেখে 3rd person কে black বলে identify করবে। ML model ও এরকম করবে।

Religion

Cultural Bias: বিয়ের ছবি।

→ Some bias are natural

→ then should remove