Unit - 2 Water Jug problem.

You are given 2 jugs, a 4-gallon one and a 3-gallon one. Neither has any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 gallons of water into 4-gallon jug?

Solution

The state space for this problem can be described as the set of ordered pairs of integers (x, y)

where

$X \rightarrow$ Quantity of water in the 4 gallon jug
$X = 0, 1, 2, 3, 4.$

$Y \rightarrow$ Quantity of water in 3-gallon jug
$Y = 0, 1, 2, 3$

Start State : $(0, 0)$
Goal State : $(2, 0)$

Production Rules

1) (x, y) if $x < 4$ $\rightarrow (4, y)$ Fill the 4-gallon jug

2) (x, y) if $y < 3$ $\rightarrow (x, 3)$ Fill the 3-gallon jug

3) (x, y) if $x > 0$ $\rightarrow (x - d, y)$ Pour some water out of the 4-gallon jug

4) (x, y) if $y > 0$ $\rightarrow (x, y - d)$ Pour some water out of the 3-gallon jug.

5) (x, y) if $x > 0$ $\rightarrow (0, y)$ Empty the 4 gallon jug on the ground

6) (x, y) if $y > 0$ $\rightarrow (x, 0)$ Empty the 3-gallon ~~jug~~ jug on the ground.

7) (x, y) if $(x + y \geq 4)$ and $y > 0$ $\rightarrow (4, y - (4 - x))$ Pour water from the 3-gallon jug into the 4-gallon jug until the 4 gallon jug is full.

8) (x, y) if $(x + y \geq 3$ and $x > 0$ $\rightarrow (x - (3 - y), 3)$ Pour water from the 4 gallon jug into the 3 gallon jug until ~~the~~ 3-gallon jug is full.

9) (x, y) if $x+y \leq 4$ and $y > 0$ → $(x+y, 0)$ Pour all the water from the 3-gallon jug into the 4-gallon jug

10) (x, y) if $x+y \leq 3$ and $x > 0$ → $(0, x+y)$ Pour all the water from the 4-gallon jug into the 3-gallon jug

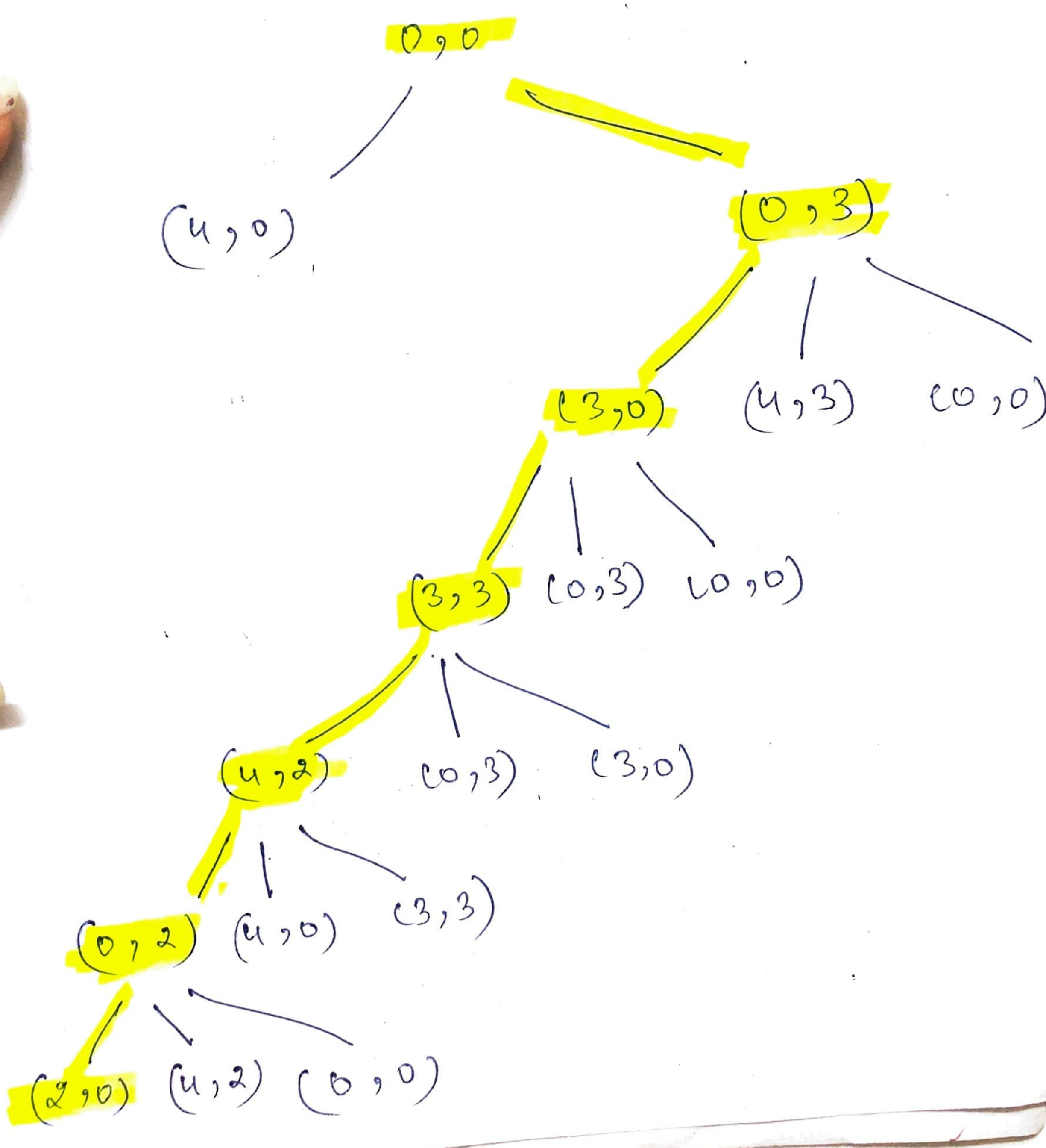
11) $(0, 2)$ → $(2, 0)$ Pour the 2 gallons from the 3 gallon jug in the 4 gallon jug.

12) $(2, y)$ → $(0, y)$ Empty the 2 gallons in the 4-gallon jug on the ground.

Solution to water jug problem

Gallons in 4-jug	Gallons in 3 jug	Rule Applied
0	0	
0	3	2
3	0	9
3	3	2
4	2	7
0	2	5 or 12
2	0	9 or 11

State Space tree for water jug problem



POORNIMA FOUNDATION Estd. 2000

POORNIMA FOUNDATION

DETAILED LECTURE NOTES

Campus: .... Course: ....

Name of Faculty: ....

Class/Section: ....

Name of Subject: ....

Date: ....

Code: ....

Production system in AI

- Helps in structuring AI Programs in a way that facilates describing and performing the search proces

Production system consists of:

1) Set of Rules
2) Knowledge Base
3) Control Strategy
4) Rule Applier.

Steps to solve the problem

i) First Reduce the problem so that it can be shown in a precised statement

ii) Problem can be solved by searching a path - through space. (start → goal)

iii) Solving process can be modelled through a production system

Advantages :

i) Structuring AI Problems - Excellent tool

ii) Highly modular - rules add, remove or change.
/ Highly flexible.

iii) Rules are expressed in Natural form.
- Easily understandable.

~~Production System~~

Characteristics of

~~Applicabi[illegible] of a [illegible]~~

Campus: .... Course: .... Class/Section: .... Date: ....
Name of Faculty: .... Name of Subject: .... Code: ....

Problems, Problem Spaces and Search

To build a system to solve a particular problem, we need to do 4 things

1) **Define the Problem Precisely** : The definition must include precise specifications of what the initial situations will be as well as what final situations constitute acceptable solns to the problem.

2) **Analyze the Problem** : Analyze all possible techniques to solve a particular problem.

3) Isolate & Represent the Task Knowledge that is Necessary to solve the problem.

4) Choose the Best problem solving technique(s) and apply it (/ them) to the particular problem.

Chess Problem

- To build a program that could "Play Chess", we would 1st have to specify the starting posⁿ of chess board, the rules that define the legal moves, and the board positions that represent a win for one side or the other.

- Our implicit goal is just not to play chess but also win this game.

Initial State: 8 X 8 Array where each position contains a symbol standing for the appropriate peice in the official chess opening position.

→ legal moves provide the way of getting from the initial state to goal state.

→ For this problem we may have large rules with roughly 10^{120} possible board positions.

→ Rules should be written in more general way as possible

→ No person could ever supply a complete set of such rules. It would take too long and could certainly not be done without mistakes.

→ No program can easily handle all those rules. Although a hashing scheme could be used to find the relevant rules for each move fairly quickly, just storing that many rules poses serious difficulties..

State Space Representation

Structure of problem solving in 2 ways.

1) Formal definition of problem as we need to convert some given situation into some desired solution using a set of permissible operations

Campus: .... Course: .... Class/Section: .... Date: ....
Name of Faculty: .... Name of Subject: .... Code: ....

→ No person could ever supply a complete set of such rules. It would take too long and could certainly not be done without mistakes.

→ No program can easily handle all those rules. Although a Hashing scheme could be used to find the relevant rules for each move fairly quickly, just storing that many rules poses serious difficulties..

State Space Representation

Structure of problem solving in 2 ways.

1) Formal definition of problem as we need to convert some given situation into some desired solution using a set of permissible operations

2) It permits us to define the process of solving a particular problem as combination of known techniques (each represented as rule defining a single step in a space) and search the general technique of exploring the space to try to find some path from the current state to a goal state.

Search is a very important process in the sol^n of hard problems for which no more direct techniques are available.

Important points:

→ For any problem

1) Define state space that contains all possible configurations.

2) Specify 1 or more states within the space that describe possible situations from which the problem solving process may start. These state are called Initial States

3) Specify 1 or more states that would be acceptable as solutions to the problem. They are known as Goal States.

POORNIMA FOUNDATION

DETAILED LECTURE NOTES

Campus: .... Course: .... Class/Section: .... Date: ....

Name of Faculty: .... Name of Subject: .... Code: ....

1) Specify a set of rules that describe the Actions (Operators) available.

Doing this will require giving thought to the following issues:

(1) What Unstated Assumptions are present in the Informal Problem description?

(2) How general should the Rules be?

(3) How much of the work Required to solve the problem should be precomputed and Represented in the Rules?

POORNIMA FOUNDATION

DETAILED LECTURE NOTES

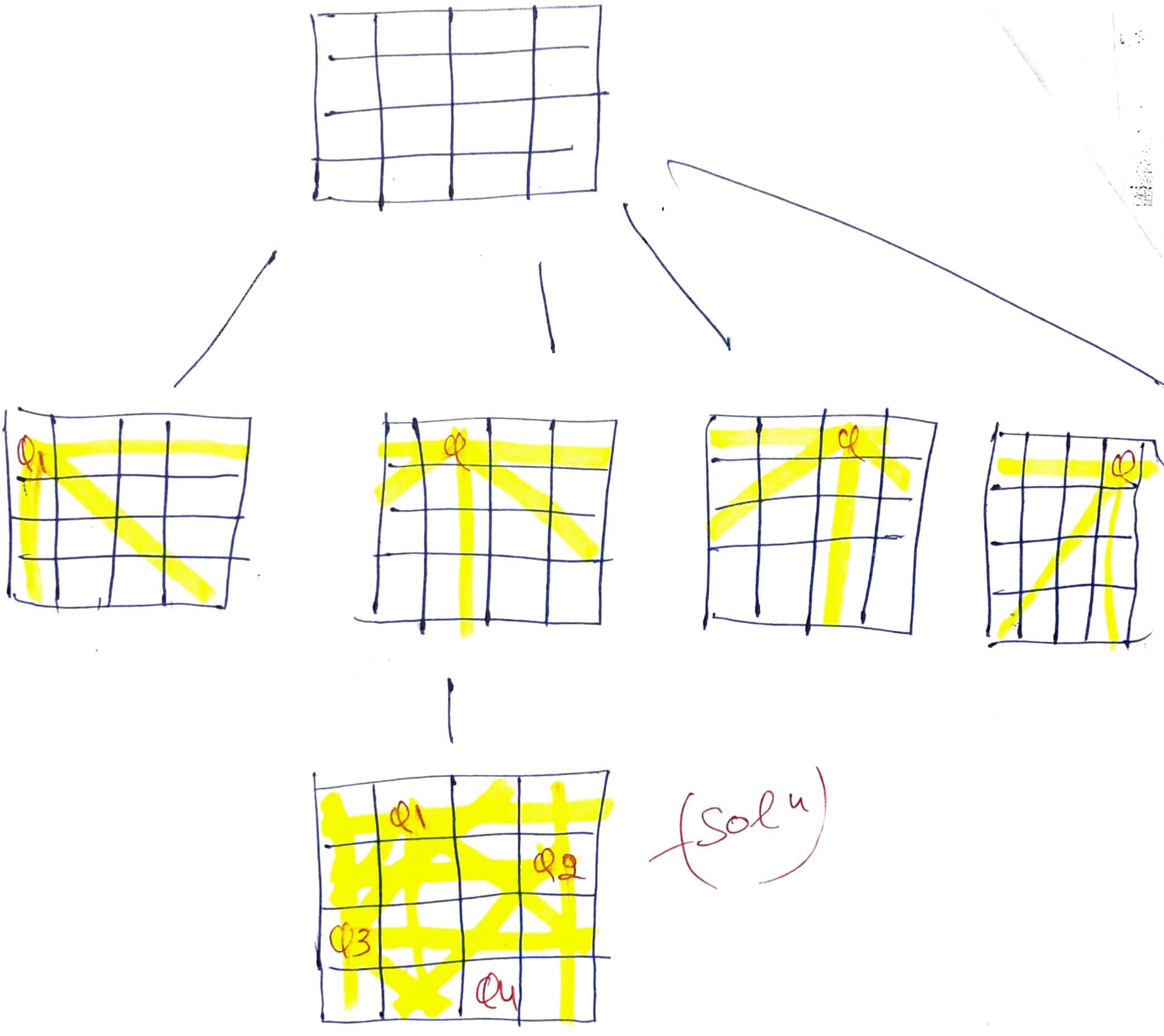
Campus: .... Course: .... Class/Section: .... Date: ....

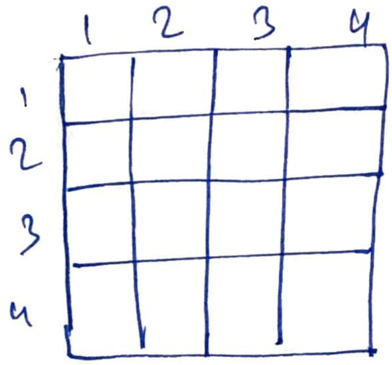
Name of Faculty: .... Name of Subject: AI Code: ....

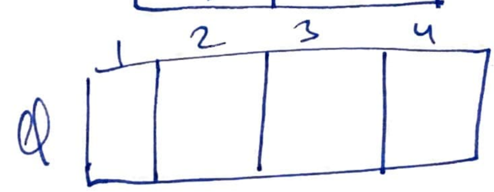
Unit -2 : Topic : Chess Problem

N- Queens Problem

- It is type of constraint satisfaction Problem.
- N - Queens problem is ~~to~~ to place n-Queens in such a manner on an nxn chessboard that no Queens attack each other by being in the same Roe, colmnn or diagonal.
- N - Queens are needed to placed on them.
- ~~Non~~
- there should be no - row / column without any Queen.


Q1
Q
Q
Q
Q1
Q2
Q3
Q4
(Sol^n)



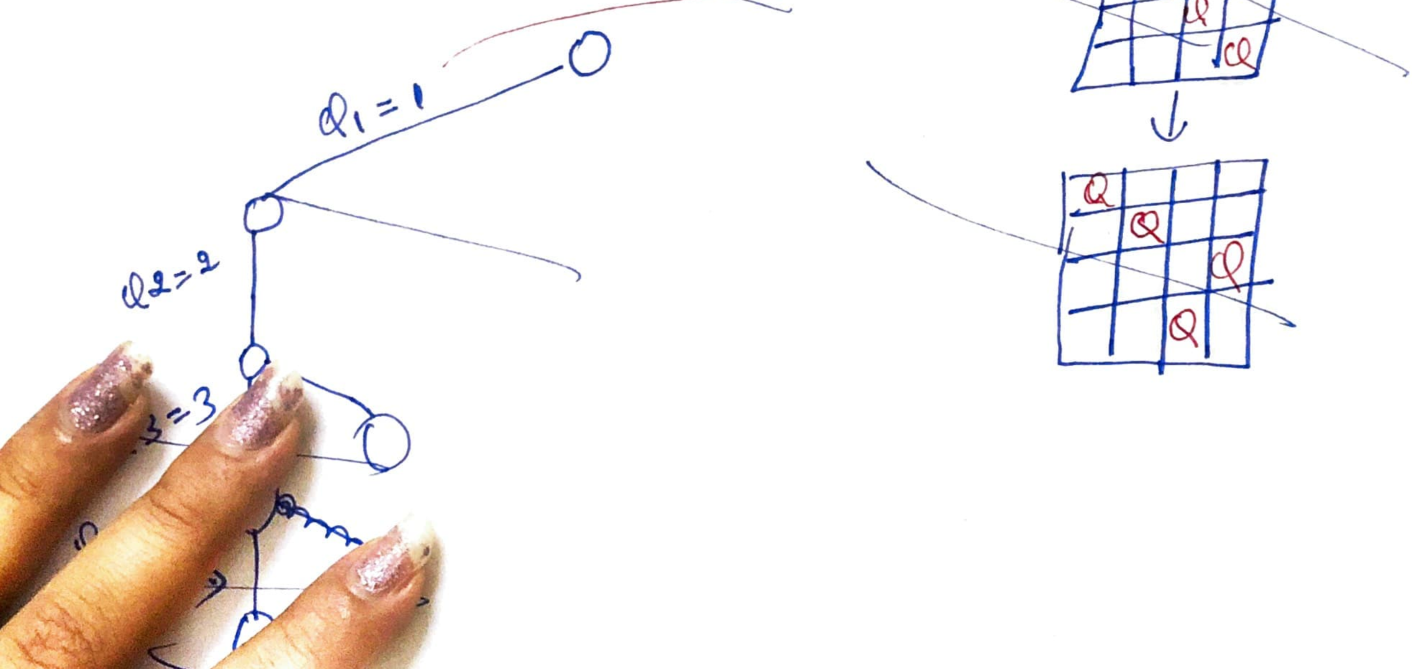


col. no.

- For this we can have many solutions, so here we will apply ~~Bla~~ Backtracking. because we want all these solutions.
- To get an optimal soln we will apply Dynamic Programming.
- I can place Queens (4) in $^{16}C_4$ ways. so it will become really bigger problem so we can work according to the rules that each Queen should be placed in unique rows. So no we don't need to decide the cell where the Queen needs to placed. Instead we will decide the column in which each Queen needs to be placed.

- Also we will try to avoid 2 Queen in the same column.
- So now we are directly taking care of the constraints as rows and columns.
- So, now we just need to check the Queens are in same diagonal or not.
- Now, we will try to solve it and prepare the solutions in a form of solution tree i.e State Space tree.

State Space Tree



POORNIMA FOUNDATION

DETAILED LECTURE NOTES

Campus: .... Course: ....

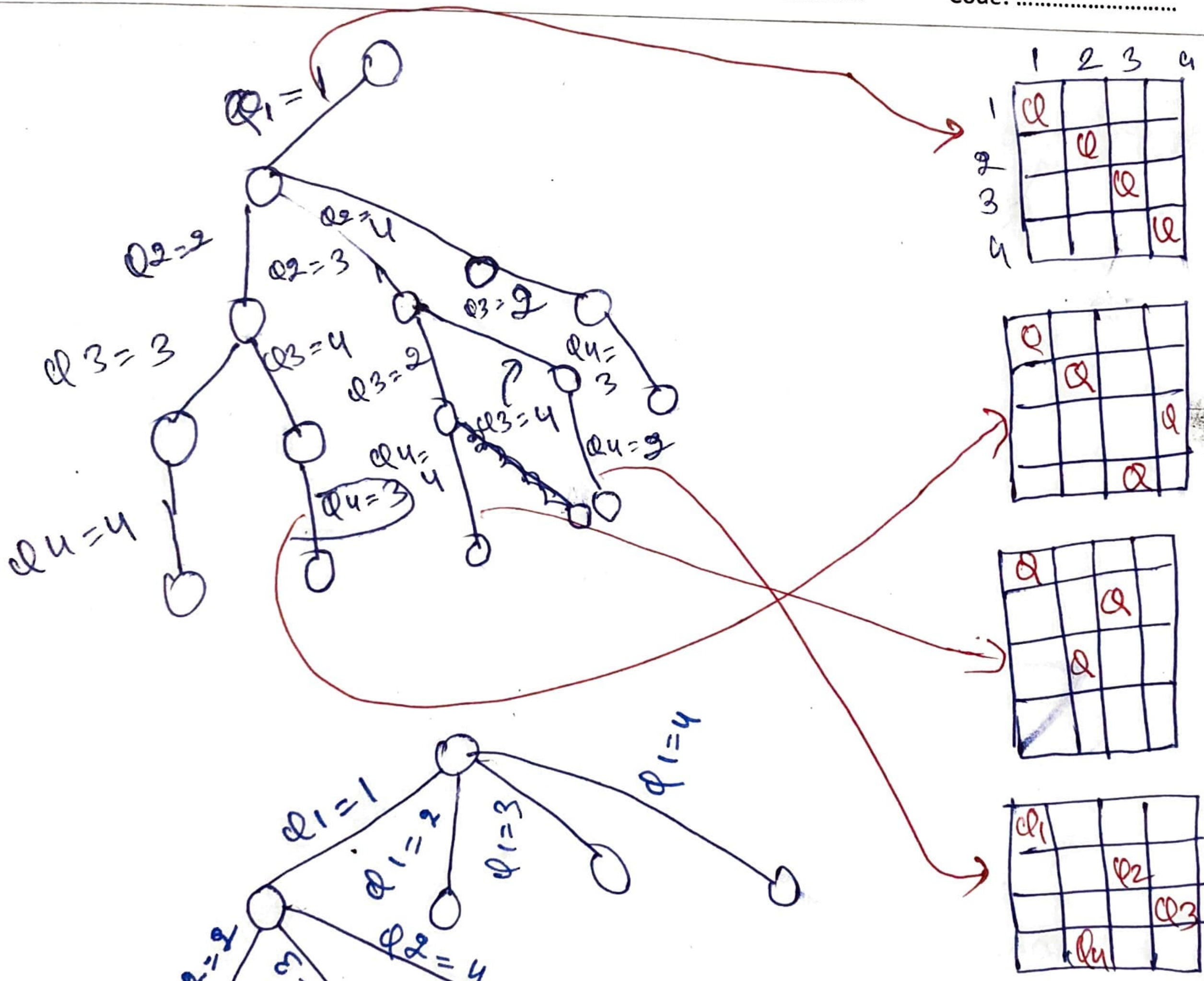
Class/Section: ....

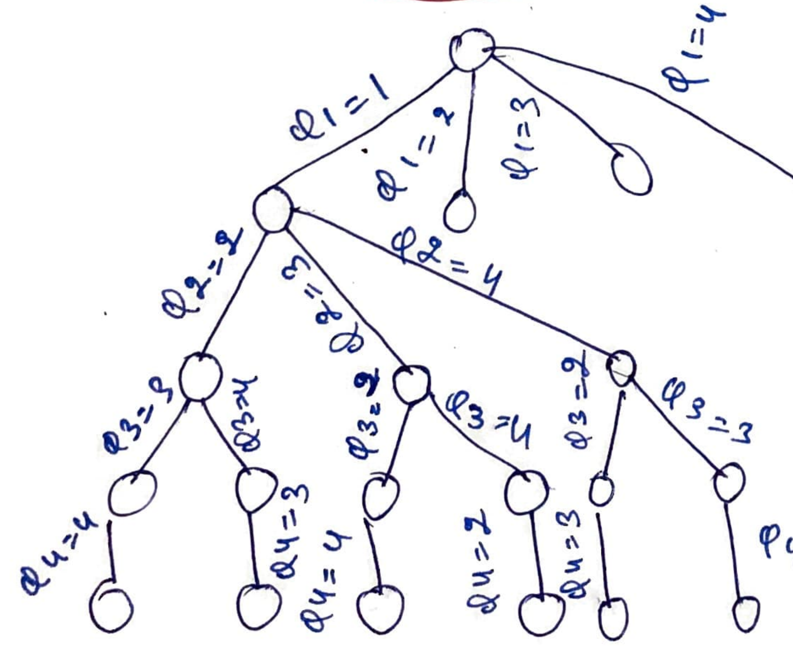
Date: ....

Name of Faculty: ....

Name of Subject: ....

Code: ....



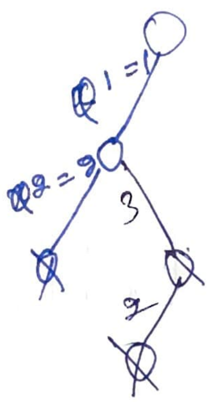


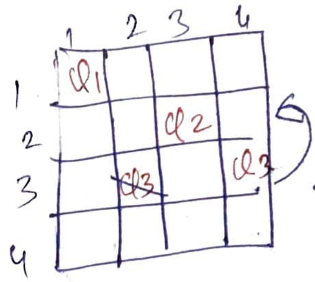
$$1+4+\overset{12}{(4\times3)}+\overset{24}{(4\times3\times2)}+\overset{24}{(4\times3\times2\times1)}=65$$

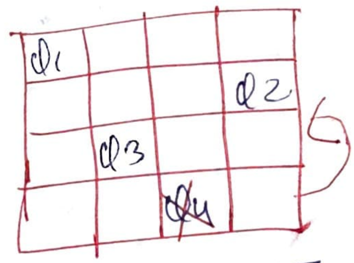
$$1+\sum_{i=0}^{3}\prod_{j=0}^{i}(\underset{N}{4}-j)$$

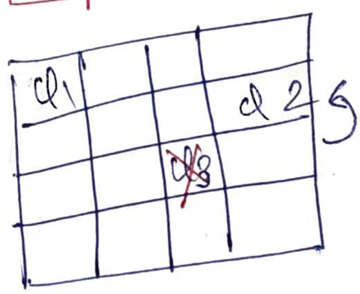
Bounding f^n (Condition / Constraint)

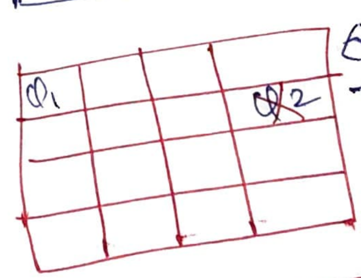
→ Not in same row, same column and same diagonal











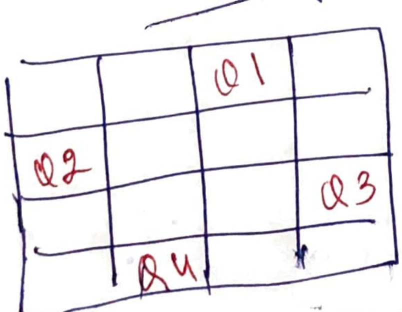
soln

Q	2	4	1	3
	1	2	3	4

col. no.

←

mirror image



Q	3	1	4	2

col no.

Unit - 2 8 - Puzzle Problem

The 8 Puzzle consists of 8 numbered, movable tiles set in a 3X3 frame. One cell of the frame is always empty thus making it possible to move an adjacent Numbered tile into the empty cell. Such a puzzle is illustrated in following diagram.

2	8	3
1	6	4
7		5

Initial

$\longrightarrow$

1	2	3
8		4
7	6	5

- To solve a problem using a production system, we must specify the global database the rules, and the control strategy.

- Important 3 Elements:

(1) States (2) Moves (3) Goals.

→ once the problem states have been conceptually identified, we must construct a computer representation, or description of them.

- This Description is then used as the database of Production system.

- Virtually any kind of data structure can be used to describe states.

- A move transforms one problem state into another state

Moves Move empty Box

↑ up ↓ down ← left → Right

- The Problem goal condition forms the basis for the termination condition of the Production system.
- The control strategy repeatedly applies rules to state description until a description of a goal state is produced

Solution

2	8	3
1	6	4
7		5

2	8	3
1		4
7	6	5

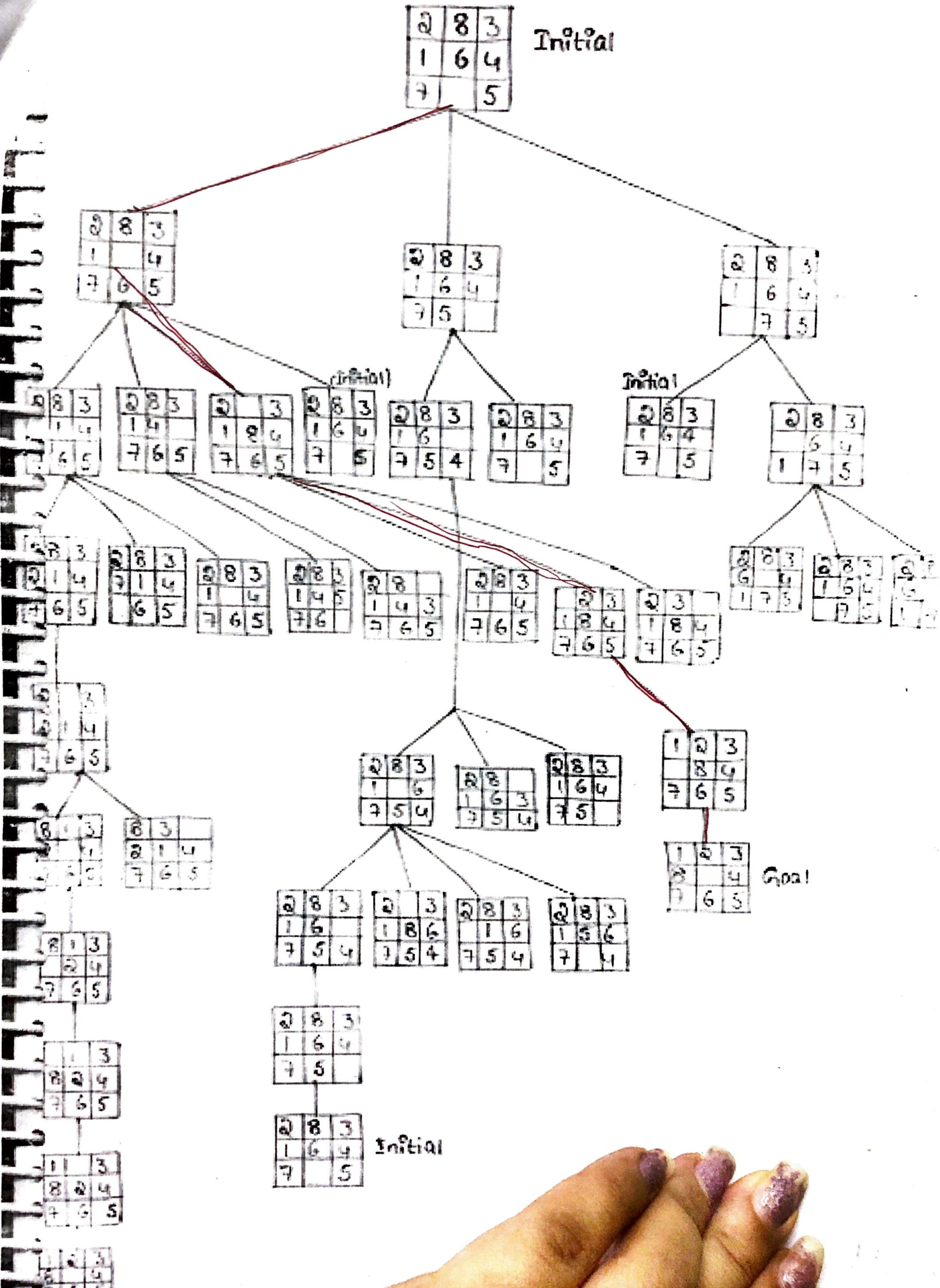
2		3
1	8	4
7	6	5

	2	3
1	8	4
7	6	5

1	2	3
	8	4
7	6	5

1	2	3
8		4
7	6	5

— goal state


Initial
(Initial)
Initial
Initial
Goal