INDEX

Name Shamanth · K · Murthy Sub AI Lab

Std.: Div. Roll No.

Telephone No. E-mail ID.

Blood Group. Birth Day

Sr.No.	Title	Page No.	Sign./Remarks
1	Tic Tac Toe Game.		SSM 29/9/24
2	Vacuum. Cleaner World.		
3	8 Puzzles Problem		SSM 12/11/24
4	Iterative Deepening DFS		
5	Hill Climbing		
6	Simulated Annealing		
7	Knowledge based using Propositional logic		
	Unification in first order logic	3	SSM 19/11/24
8	Forward Reasoning		
9	FOL to Resolution		
10	Alpha beta Pruning		

CS

24/09/24

SURYA Gold
Date Page

LAB - 1

TIC - TAC TOE GAME

Algorithm:

1) Initialize the Board:
   → Create 3x3 2D array with '-' for empty cells

2) Check if board empty function:
   → Iterate through each row & check if '-' exists, if yes return true

3) Check for winner function:
   → Check each row, column & diagonals for 3 matching marks & return the winning mark if exists.

4) Function to display the board:

5) Start_Game function:
   → Initialize the board & randomly select a player & take input.
   → Loop until game ends:
      ↳ Display the board
      ↳ take input & validate if it is a number & between 0 & 2 else print invalid.
      ↳ Update the board with current player's mark
      ↳ Check if player has won & display
      ↳ if board is full & no winner print tie
      ↳ Switch to next player.

6) End Game.

SSV
24/9/24

01/10/24

SURYA Gold
Date Page

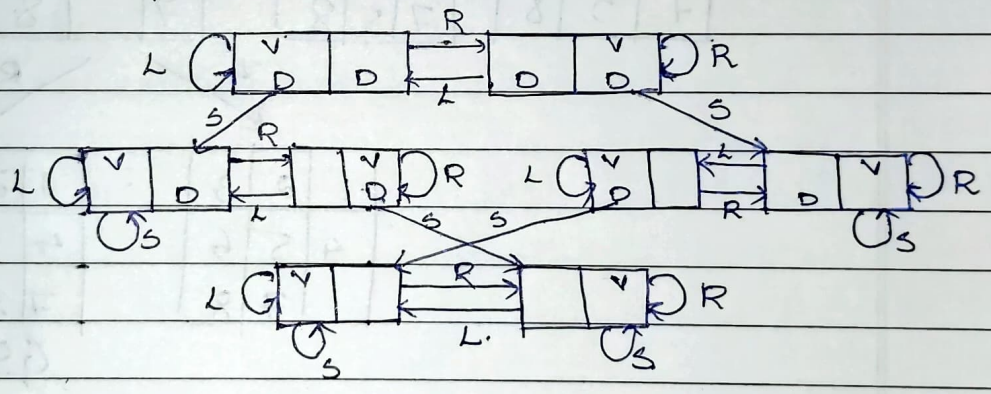
Lab-2
Vacuum Cleaner World

Pseudo code:

function REFLEX-VACUUM-AGENT ([location, status]) returns an action

if status = Dirty then return Suck
else if location = A then return Right
else if location = B then return left

State-Space diagram:



Algorithm.

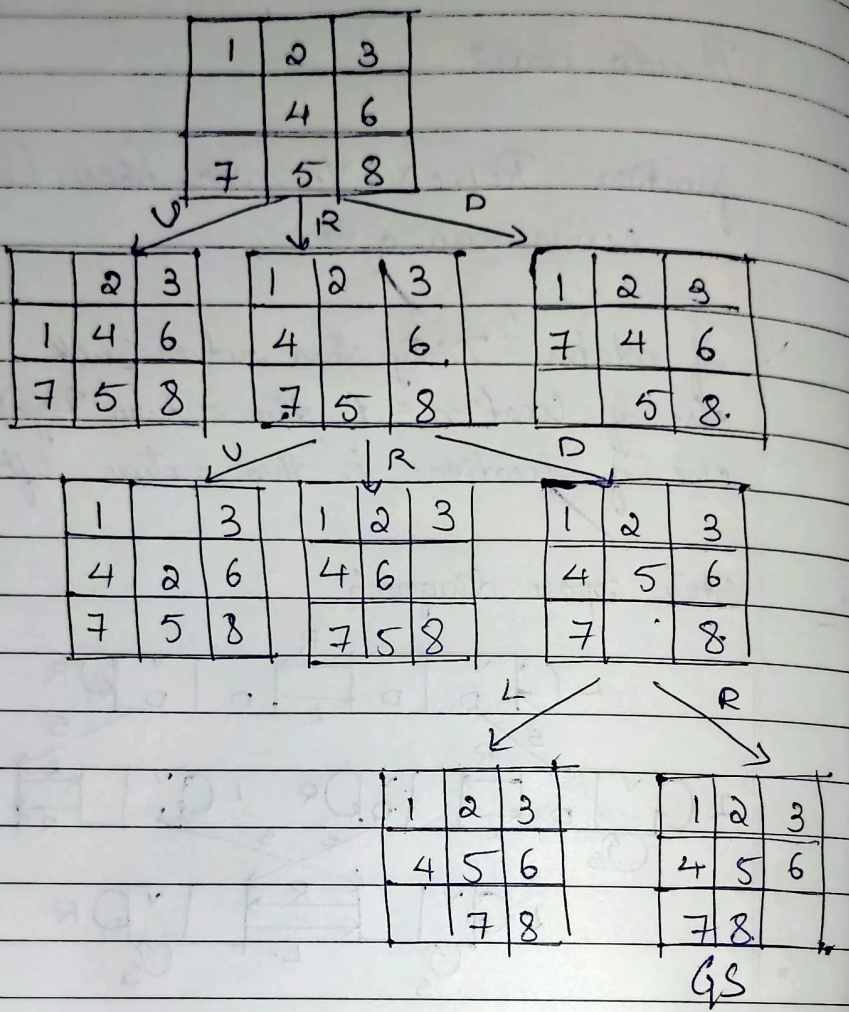
→ Initialize an array with status of both locations as dirty
→ In a loop, take input of location & status (0/1) from user & pass it to check function.
→ If status is dirty print dirty suck & increament cost else print location is clean & update the status in list
→ Repeat until user enters status as -1

1/10/24

07/10/24

SURYA Gold
Date _____ Page _____

8-Puzzle Problem Using BFS & DFS



Algorithm: BFS

→ Create a queue to store states & add the first state & mark as visited.

→ In Loop while queue is not empty:
Take first element in queue & check if it's target else try all possible moves & add them to queue & marks as visited.

→ If queue is exhausted without finding solution print No Solution

DFS Algorithm:

-> Create a stack to store current state & add the first state & mark as visited.

-> In loop while stack is not empty:
pop the top element & check if its target else mark it as visited & push it into the stack all possible states from that state.

-> If stack is exhausted print No solution

SSD
8/10/24

15/10/24

SURYA Gold
Date Page

Lab - 3

For 8-puzzle problem using A* implementation to calculate f(n) using :

a) g(n) = depth of node
h(n) = heuristic value
a) ↳ No of misplaced tiles
b) ↳ Manhattan distance

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

1) Step 1: Place the starting node in OPEN list
2) Check if open list is empty or not, if empty return failure.
3) Select the node with smallest f(n) value if n is goal return success
4) Generate all successors & put current node in closed list.
5) If it is not in open or closed compute f(n)
6) Return to step 2.

SURYA Gold
Date ____ Page ____

a) Give state space diagram for:

Initial state

2	3	3
1	6	4
7		5

$f(n) = g(n) + h(n)$
$= 0 + 4$
$= 4$

Goal state

1	2	3
8		4
7	6	5

L — UP — R

L: 2 8 3 / 1 6 4 / 7 7 5 (1+5 = 6)

UP:

2	8	3
1		4
7	6	5

(1+3 = 4)

R: 2 8 3 / 1 6 4 / 7 5 (1+5 = 6)

UP:

2		3
1	8	4
7	6	5

(2+3 = 5)

L: 2 8 3 / 1 4 / 7 6 5 (2+3 = 5)

R: 2 8 3 / 1 4 / 7 6 5 (2+4 = 6)

UP:

	2	3
1	8	4
7	6	5

(3+2 = 5)

R: 2 3 / 1 8 4 / 7 6 5 (3+4)

UP:

1	2	3
	8	4
7	6	5

(4+1 = 5)

UP:

1	2	3
8		4
7	6	5

(5+0=5)

Goal State

1 2 3 / 7 8 4 / 6 5

SURYA Gold
Date _____ Page _____

b)

2	8	3
1	6	4
7		5

2	8	3
1		4
7	6	5

(1+4 = 5)

2 8 3
1 6 4
7 5
(1+1+2+1+1=6)

2 8 3
1 6 4
7 5
(1+1+2+1+1) = 6

h(n)

2		3
1	8	4
7	6	5

1+1+1 = 3

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

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

	2	3
1	8	4
7	6	5

1+1 = 2

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

1	2	3
	8	4
7	6	5

1

1	2	3
8		4
7	6	5

0

1 2 3
7 8 4
6 5

1+1 = 2

Goal state.

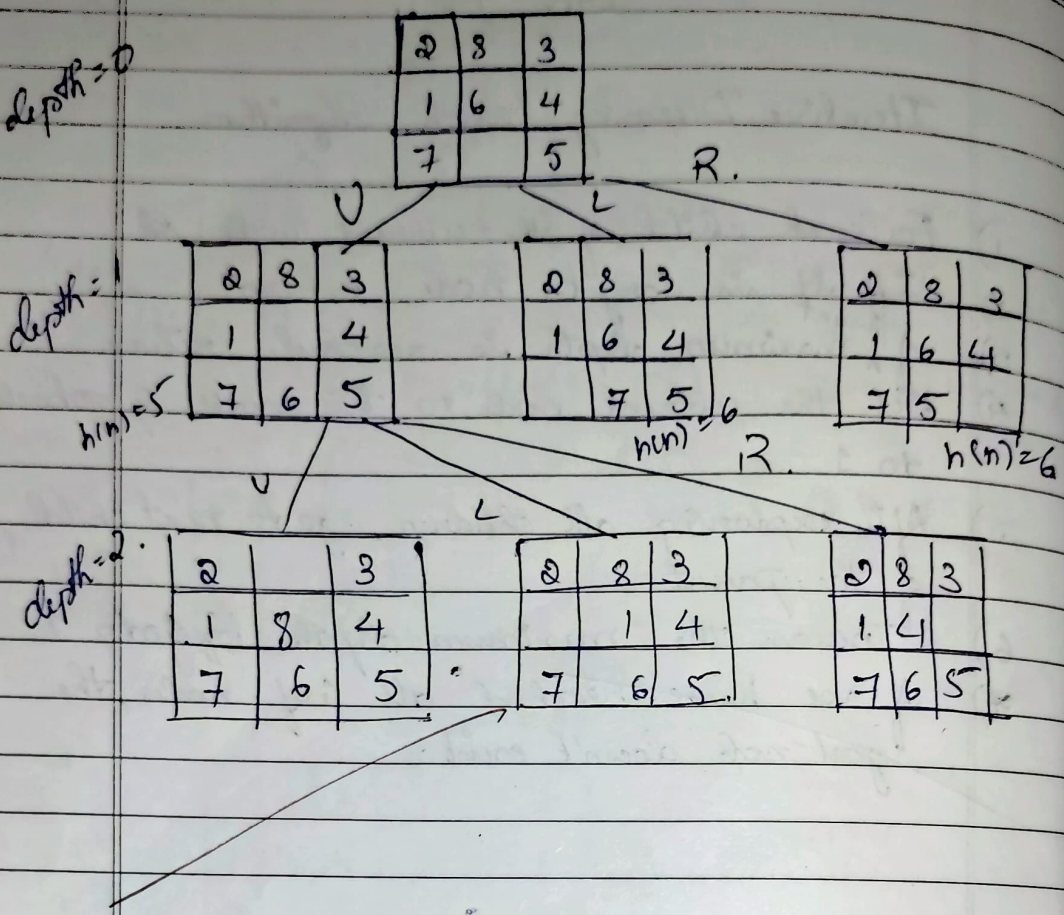
cost = 5

SURYA Gold
Date Page

LAB - 4

Iterative Deepning Search Algorithm

1) For each child of the current node
2) If it is target node, return
3) If current maximum depth is reached return
4) Set the current node to this node & goback to 1
5) After exploring all children, go to next child of the parent
6) Increase the maximum depth. Goto 1
7) If we have reached all of nodes, the goal node doesn't exist.


depth=0
depth=1
depth=2
U
L
R.
h(n)=5
h(n)=6
h(n)=6

22/10/24

SURYA Gold
Date Page

Lab - 4

* Implement Hill Climbing search algorithm to solve N-Queens Problem. N>4

function HILL-CLIMBING (problem) returns a state that is local maximum

current ← make-NODE (problem.INITIAL-STATE)

loop do

neighbour ← a highest-valued successor of current.

if neighbour.VALUE ≤ current.VALUE then return current.STATE

current ← neighbour.

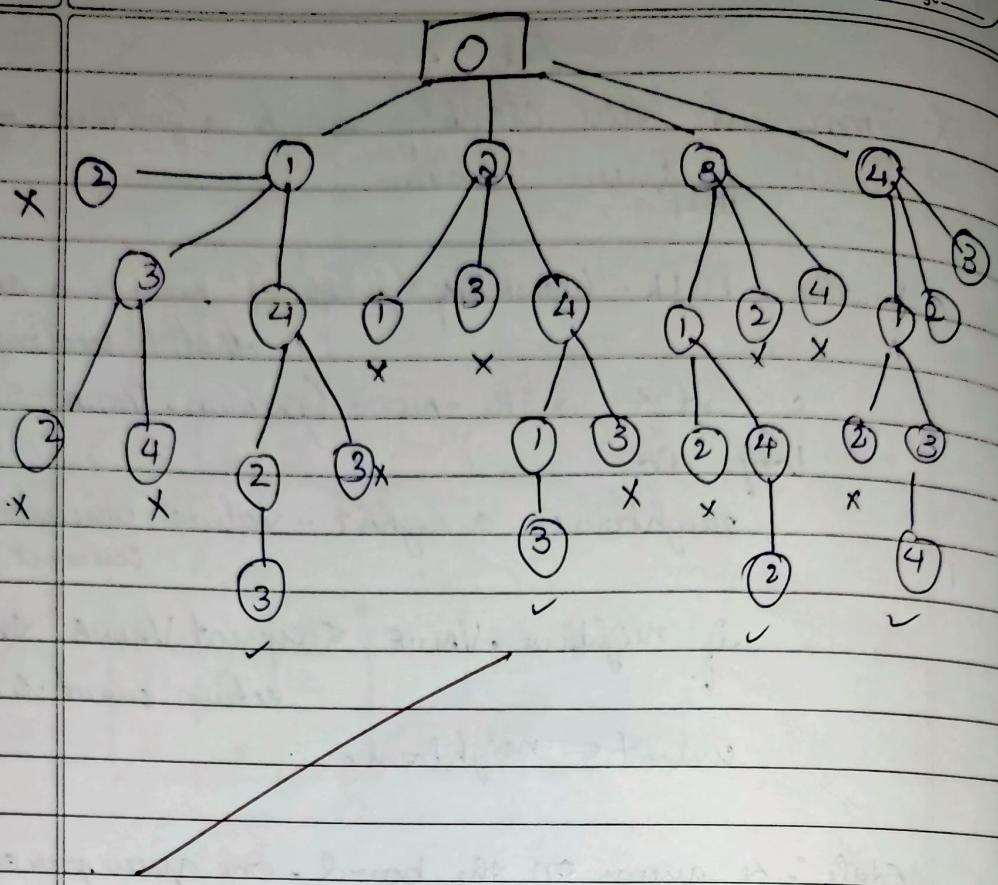
State: 4 queens on the board. one queen per column
- variables: x_0 x_1 x_2 x_3 where x_i is the row position of the queen in column i. ~~Assume~~ that there is one queen per column.
- Domain for each variable $x_i \in \{0, 1, 2, 3\}$ $\forall i$

Initial State: a random state

Goal state: 4 queens on the board. No pair of queens are attacking each other.

Neighbour relation: Swap row positions of 2 queens

Cost function: The number of pairs of queens attacking each other directly or indirectly.


0
1
2
3
4
1
3
4
1
3
4
1
2
4
1
2
3
2
4
2
3
1
3
2
4
2
1
2
3
3
4
2
3
4

09/10/24

SURYA Gold
Date Page

Lab - 5

Write a program to implement Simulated Annealing

function SIMULATED ANNEALING (problem, schedule)
current ← MAKE-NODE(problem.INITIAL-STATE)
for t = 1 to ∞ do
T ← schedule(t)
if T = 0 then return current
next ← a randomly selected successor of current
ΔE ← next.Value − current.Value
if ΔE > 0 then current ← next
else current ← next only with probability $e^{\Delta E/T}$

1) Start at a random point x.
2) Choose a new point x_j, on a neighbourhood N(x).
3) Decide whether or not to move to new point x_j
The decision will be made based on the probability

function P(x, x_j, T)

$$P(x, x_j, T) = \begin{cases} 1 \\ e^{\frac{F(x_i) - F(x)}{}} \end{cases}$$

For 8 Queens Problem:

1) Initialize: Start with arrangement of queens
   Set initial temp & cooling parameter
2) Evaluate fitness:
   Calculate no of conflicts between queens
   Lower conflicts = fewer fitness.
3) Generate Neighbour:
   Create a neighbour by swapping 2 queens.
   calculate its fitness
4) Accept or Reject Neighbour:
5) Gradually Reduce the temperature & repeat until the temp is very low.
6) Return the best arrangement found.

Output: Best State found [4 7 5 0 6 3 7 2]
Number of conflicts = 2.

SBB
29/10/24

12/11/24

SURYA Gold
Date Page

LAB - 6

Create a knowledge based using propositional logic & show that the given query entails knowledge base or not

Algorithm:

→ Generate all possible models: For n symbols there are 2^n combinations

→ For each model: evaluate each statement in KB, if all statements are true proceed to next test otherwise skip to next model.

→

→ Check the Query: If KB is true under the model, check if query is true.

→ If query is true in all cases when KB is true the KB entails the Query

Ex:- $KB = (A \vee C) \wedge (B \vee \neg C)$ $\alpha = A \vee B$

A	B	C	A ∨ C	B ∨ ¬C	KB	α
F	F	F	F	T	F	F
F	F	T	T	F	F	F
F	T	F	F	T	F	T
F	T	T	T	T	T	T
T	F	F	T	T	T	T
T	F	T	T	F	F	T
T	T	F	T	T	T	T
T	T	T	T	T	T	T

12/11/24

12/11/24

SURYA Gold
Date Page

LAB - 6

Create a knowledge based using propositional logic & show that the given query entails knowledge base or not

Algorithm:

→ Generate all possible models: For n symbols there are 2^n combinations

→ For each model: evaluate each statement in KB, if all statements are true proceed to next test otherwise skip to next model.

→

→ Check the Query: If KB is true under the model, check if query is true.

→ If query is true in all cases when KB is true the KB entails the Query

Ex:- KB = (A ∨ C) ∧ (B ∨ ¬C) α = A ∨ B

A	B	C	A ∨ C	B ∨ ¬C	KB	α
F	F	F	F	T	F	F
F	F	T	T	F	F	F
F	T	F	F	T	F	T
F	T	T	T	T	T	T
T	F	F	T	T	T	T
T	F	T	T	F	F	T
T	T	F	T	T	T	T
T	T	T	T	T	T	T

12/11/24

19/11/24

SURYA Gold
Date Page

Lab - 7

Implementation unification in first order logic

Algorithm : Unify (ψ_1, ψ_2)

Step 1 : If ψ_1 or ψ_2 is a variable or const:
a) If ψ_1 or ψ_2 are identical, then return NIL
b) Else if ψ_1 is a variable.
a. then if ψ_1 occurs in ψ_2 then return Failure.
b. else return $\{(\psi_2 / \psi_1)\}$
c) Else if ψ_2 is a variable :
a. If ψ_2 occurs in ψ_1 then return FAILURE
b. Else return $\{(\psi_1 / \psi_2)\}$
d) Else return FAILURE

Step 2 : If the initial predicate symbol in ψ_1 and ψ_2 are not same, then return FAILURE

Step 3 : If ψ_1 and ψ_2 have a different no of arguments, then return FAILURE

Step 4 : Set substitution set (SUBST) to NIL.

Step 5 : For i = 1 to the no of elements in ψ_1
a) Call Unify function with the ith element of ψ_1 and ith element of ψ_2 & put the result into S.
b) If S = failure then return Failure
c) If S ≠ NIL then do,
a. Apply S to the remainder of both L1, L2

b. SUBST = APPEND (S, SUBST)

Step 6: Return SUBST.

Q) $\psi_1 = P(f(a), g(Y))$
$\psi_2 = P(X, X)$

→ Predicate symbols are same
→ No of arguments are same.

→ For unification: We'll equate:
$f(a) = X \quad g(Y) = X$
$\Rightarrow f(a) = g(Y)$

→ but f & g are two different functions hence they cannot be unified.

Q) $\psi_1 = P(b, X, f(g(z)))$
$\psi_2 = P(z, f(y), f(y))$

$\theta_0 = b/z \quad \theta_1 = f(Y)/X \quad \theta_2 = g(b)/Y$

19/11/24

26/11/24

SURYA Gold
Date _____ Page _____

Lab - 8

Create a knowledge base consisting of first order logic statements & prove the given query using forward reasoning.

1) Input:
→ (condition, result)
↳ set of [facts that must be true
↳ fact inferred if condition true.
→ A set of initial facts known to be true
→ A query to determine if it can be inferred.
→ applied_rules = True.

2) while applied_rules is True:
set applied_rules = False
For each (condition, result) in rules:
check if condition ⊆ facts
if results ⊄ facts
add result to facts
set applied_rules = True.
if query ∈ facts return True
if no new rules applied_rules = False exit loop

3) Output:
if query ∈ facts return true (inferred)
else return false (query not inferred).

3/12/24

SURYA Gold
Date Page

Lab-9

Convert a given first order logic statement into resolution.

1. Convert to Claus Form (CNF):
   - → Eliminate implication (A→B becomes ¬A∨B).
   - → Move negations inward using De Morgan's Law.
   - → Standardize variables
   - → Drop universal quantifiers
   - → Convert to a conjuction of disjunctions (CNF form)

2) Negate the statement to be Proved:
   - → Negate the query & add it to knowledge base.

3) Apply Unification
   - → Identify & resolve two clauses that contain complimentary literals.

4) Derive new Clauses:
   - → Continue resolving clauses until either:
     - ↳ A contradiction is found
     - ↳ No further resolution is possible (query cannot be proved)

3/12/24

SURYA Gold
Date _____ Page _____

Lab - 10

Implement Alpha-beta pruning

1) Input:
-> A game tree with terminal node values
-> Depth d & whether maximizer or minimizer's turn

2) Steps:
-> Initialize $\alpha = -\infty$ & $\beta = +\infty$
-> Start at root & recursively evaluate child nodes
-> For each node:
  -> if maximizer:
    -> initialize to $-\infty$
    -> update α = max value
    -> if $\alpha \geq \beta$ -> stop
  -> if minimizer:
    -> initialize to $+\infty$
    -> update β = min value
    -> if $\alpha \geq \beta$ -> stop

-> Return optimal value of root