LAB-4

22/10/2024:

(#) Project title:-

Implement hill climbing search algorithm to solve N-Queens problem.

(##) Algorithm:-

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

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

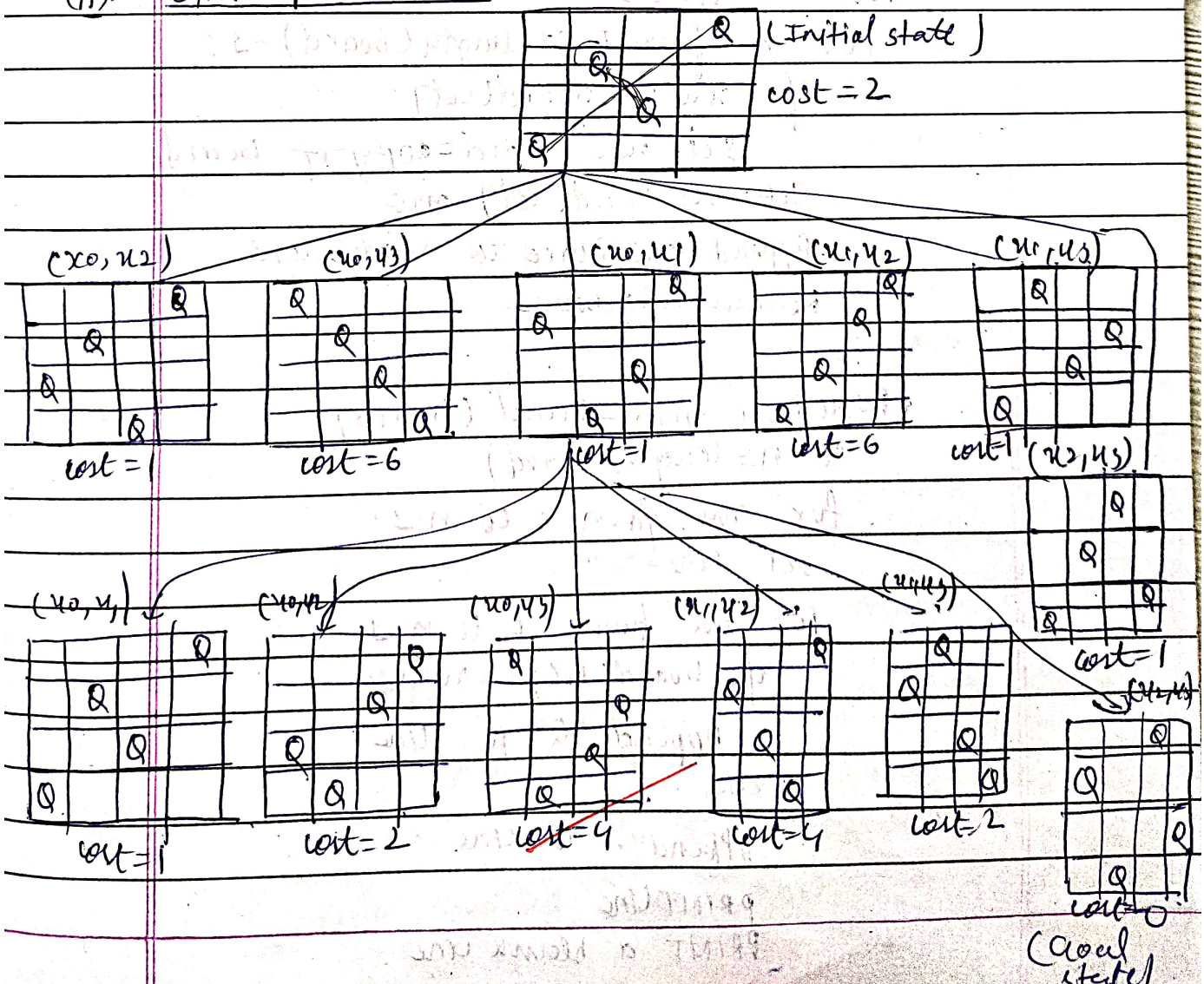
loop do

neighbour ← a highest-valued successor of current

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

current ← neighbour.

(##) State space tree:-



Mangal
Date
Page

(##) ~~State space tree:~~

(##) PSEUDOCODE:

```
FUNCTION calculate-conflicts(barad):
  Set n = length(board)
  Set conflicts = 0
  For i From 0 to n-1:
    For j from i+1 to n-1:
      if board[i] == board[j] OR
      ABS(board[i]-board[j]) == ABS(i-j):
        increment conflicts.
    Return conflicts
FUNCTION
generate_neighbours(board):
  SET neighbours = Empty list
  For col from 0 to length(board)-1:
  For row from 0 to length(board)-1:
    if row != board[col]:
      Set new_board = copy of board
    Set new_board[col] = row
    Append new_board to neighbours
    Return neighbours.

FUNCTION print_board(board):
  Set n = length(board)
  for row from 0 to n-1:
    set line = " "
    For col from 0 to n-1:
      if board[col] == row:
        Append "Q" to line
      else
        Append "." to line.
      PRINT line
      PRINT a blank line
```

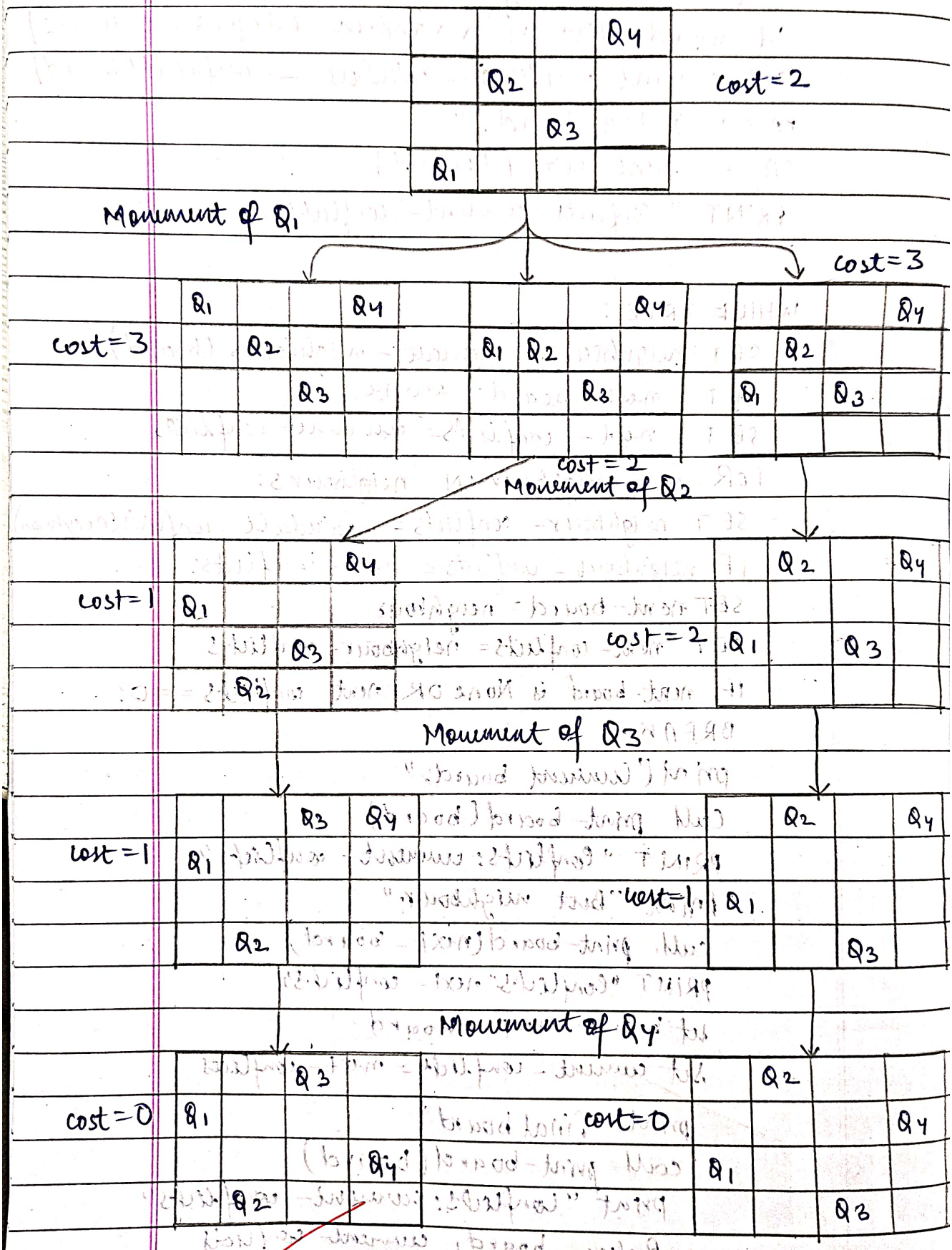
```
FUNCTION hill-climbing(n):
    Set board = list of n random integers (0 to n-1)
    set current-conflicts = calculate-conflicts(board)
    PRINT "Initial board:"
    CALL print-board(board)
    PRINT "conflicts: current-conflicts".

WHILE TRUE:
    SET neighbours = generate-neighbours(board)
    SET next-board = None
    SET next-conflicts = current-conflicts
    FOR each neighbour IN neighbours:
        SET neighbour-conflicts = calculate-conflicts(neighbour)
        IF neighbour-conflicts < next-conflicts:
            SET next-board = neighbour
            SET next-conflicts = neighbour-conflicts
    IF next-board is None OR next-conflicts == 0:
        BREAK
    print("Current board:"
    Call print-board(board)
    PRINT "conflicts: current-conflicts",
    Print "Best neighbour:"
    call print-board(next-board)
    PRINT "conflicts: next-conflicts"
    set board = next-board
    Set current-conflicts = next-conflicts
    print "final board"
    call print-board(board)
    print "conflicts: current-conflicts"
    Return board, current-conflicts

Set n=4
Set solution, conflicts = CALL
hill-climbing(n)
```

Mangal
Date
Page

(##) State Space tree:



Namrata. M.
29/10/2024