# Bansilal Ramnath Agarwal Charitable Trust’s

Vishwakarma Institute of Technology Pune-37

*(An autonomous Institute of Savitribai Phule Pune University)*



**Department of Computer Engineering**

|  |  |
| --- | --- |
| **Division** | C |
| **Batch** | 2 |
| **GR-no** | 12111180 |
| **Rollno** | 17 |
| **Name** | Divya Mahajan |

# Title: Implementation of CSP Problem.

**Description:**

**Implementation of N-Queens problem:**

The N-Queens problem is a classic Constraint Satisfaction Problem (CSP) where you need to place N queens on an N×N chessboard in such a way that no two queens threaten each other. This means no two queens can be in the same row, column, or diagonal.

You can use various search algorithms to find a solution to the CSP, such as backtracking, constraint propagation, or local search. Backtracking is commonly used for the N-Queens problem.

Backtracking Algorithm:

1. Start with the first column (variable).
2. For each row in the domain of the current column:
   1. Check if placing a queen at this row violates any of the constraints. If it does, move to the next row.
   2. If a valid row is found, assign it to the current column and move to the next column.
   3. If you reach a column where you cannot place a queen without violating constraints, backtrack to the previous column and try the next row.
3. Continue this process until you've placed N queens on the board (solution) or determined that no solution exists.

**Implementation of Crypt-Arithmetic problem:**

Crypt-arithmetic puzzles are a type of Constraint Satisfaction Problem (CSP) where you need to assign digits to letters in such a way that a given arithmetic expression is valid. The goal is to find a digit-to-letter mapping that satisfies the equation.

Use a search algorithm to find a valid assignment of digits to letters that satisfies all constraints.

Backtracking is a common choice for solving cryptarithmetic puzzles:

1. Start with the leftmost letter and try each digit in its domain.
2. Move to the next letter and repeat the process.
3. If a digit assignment violates a constraint, backtrack and try the next digit for the previous letter.
4. Continue this process until a solution is found or it's determined that no solution exists.

**Code:**

* N-Queens Problem

#include <bits/stdc++.h>

using namespace std;

void printBoard(vector<vector<int>> &board)

{

    for (auto row : board)

    {

        for (auto i : row)

            cout << i << " ";

        cout << endl;

    }

    cout << endl;

}

bool isSafe(int row, int col, vector<vector<int>> &board, int n)

{

    int x = row;

    int y = col;

    while (y >= 0)

    {

        if (board[x][y] == 1)

            return false;

        y--;

    }

    x = row;

    y = col;

    while (x >= 0 && y >= 0)

    {

        if (board[x][y] == 1)

            return false;

        x--;

        y--;

    }

    x = row;

    y = col;

    while (x < n && y >= 0)

    {

        if (board[x][y] == 1)

            return false;

        x++;

        y--;

    }

    return true;

}

void nQueens(int col, vector<vector<int>> &board, int n, bool &flag)

{

    if (col == n)

    {

        flag = true;

        printBoard(board);

        return;

    }

    for (int row = 0; row < n; row++)

    {

        if (isSafe(row, col, board, n))

        {

            board[row][col] = 1;

            nQueens(col + 1, board, n, flag);

            board[row][col] = 0;

        }

    }

}

int main()

{

    int n;

    cin >> n;

    bool flag = false;

    vector<vector<int>> board(n, vector<int>(n, 0));

    nQueens(0, board, n, flag);

    if (!flag)

        cout << "No solution found" << endl;

    return 0;

}

* Crypt-Arithmetic Problem

#include <bits/stdc++.h>

using namespace std;

vector<int> use(10);

struct node

{

    char c;

    int v;

};

int check(node \*nodeArr, const int count, string s1,

          string s2, string s3)

{

    int val1 = 0, val2 = 0, val3 = 0, m = 1, j, i;

    for (i = s1.length() - 1; i >= 0; i--)

    {

        char ch = s1[i];

        for (j = 0; j < count; j++)

            if (nodeArr[j].c == ch)

                break;

        val1 += m \* nodeArr[j].v;

        m \*= 10;

    }

    m = 1;

    for (i = s2.length() - 1; i >= 0; i--)

    {

        char ch = s2[i];

        for (j = 0; j < count; j++)

            if (nodeArr[j].c == ch)

                break;

        val2 += m \* nodeArr[j].v;

        m \*= 10;

    }

    m = 1;

    for (i = s3.length() - 1; i >= 0; i--)

    {

        char ch = s3[i];

        for (j = 0; j < count; j++)

            if (nodeArr[j].c == ch)

                break;

        val3 += m \* nodeArr[j].v;

        m \*= 10;

    }

    if (val3 == (val1 + val2))

        return 1;

    return 0;

}

bool permutation(const int count, node \*nodeArr, int n,

                 string s1, string s2, string s3)

{

    if (n == count - 1)

    {

        for (int i = 0; i < 10; i++)

        {

            if (use[i] == 0)

            {

                nodeArr[n].v = i;

                if (check(nodeArr, count, s1, s2, s3) == 1)

                {

                    cout << "Solution found:" << endl;

                    for (int j = 0; j < count; j++)

                        cout << nodeArr[j].c << " = " << nodeArr[j].v << endl;

                    return true;

                }

            }

        }

        return false;

    }

    for (int i = 0; i < 10; i++)

    {

        if (use[i] == 0)

        {

            nodeArr[n].v = i;

            use[i] = 1;

            if (permutation(count, nodeArr, n + 1, s1, s2, s3))

                return true;

            use[i] = 0;

        }

    }

    return false;

}

bool solveCryptographic(string s1, string s2,

                        string s3)

{

    int count = 0;

    int l1 = s1.length();

    int l2 = s2.length();

    int l3 = s3.length();

    vector<int> freq(26);

    for (int i = 0; i < l1; i++)

        ++freq[s1[i] - 'A'];

    for (int i = 0; i < l2; i++)

        ++freq[s2[i] - 'A'];

    for (int i = 0; i < l3; i++)

        ++freq[s3[i] - 'A'];

    for (int i = 0; i < 26; i++)

        if (freq[i] > 0)

            count++;

    if (count > 10)

    {

        cout << "Invalid strings";

        return 0;

    }

    node nodeArr[count];

    for (int i = 0, j = 0; i < 26; i++)

    {

        if (freq[i] > 0)

        {

            nodeArr[j].c = char(i + 'A');

            j++;

        }

    }

    return permutation(count, nodeArr, 0, s1, s2, s3);

}

int main()

{

    string s1, s2, s3;

    cout << "Enter string 1 and string 2: " << endl;

    cin >> s1 >> s2;

    cout << "Enter string the sum string: " << endl;

    cin >> s3;

    if (solveCryptographic(s1, s2, s3) == false)

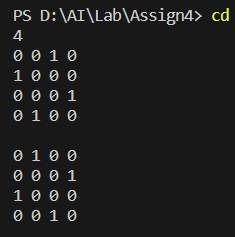
        cout << "No solution";

    return 0;

}

**Screenshots/Output:**

N-Queens:



Crypt-Arithmetic:

