Journal of Computer and Mathematical Sciences, *Vol.11(5),18-26 May 2020* *ISSN 0976-5727 (Print)*
(An International Research Journal), *www.compmath-journal.org* *ISSN 2319-8133 (Online)*

C++ Programme for Total Dominator Chromatic Number of Paths by Using Elementary Transformations

A.Vijayalekshmi[1*] and J. Virgin Alangara Sheeba[2]

[1]Associate Professor of Mathematics,
S.T. Hindu College, Nagercoil-629002, Tamil Nadu, INDIA.
[2]Research scholar, Reg no:11813
Department of Mathematics
S.T. Hindu College, Nagercoil-629002, Tamil Nadu, INDIA.
Affiliated to Manonmaniam Sundarnar University
Tirunelveli – 627 012, Tamil Nadu, India.
email:vijimath.a@gmail.com.

(Received on: May 17, 2020)

ABSTRACT

A total dominator coloring of a graph G=(V,E) without isolated vertices is a proper coloring together with each vertex in G properly dominates a color class. The total dominator chromatic number of G is a minimum number of color classes with additional condition that each vertex in G properly dominates a color class and is denoted by $\chi_{td}(G)$. In this paper, we find the total dominator chromatic number of paths by using elementary transformations through C++ programme.

2010 Mathematics subject classification code: 05C69, 68W25

Keywords: Total dominator coloring, Total dominator chromatic number.

1. INTRODUCTION

In this paper we only consider paths. Further details in graph theory can be found in[4].

Let G=(V,E) be a graph with minimum degree at least one. For two vertices v_0 and v_n of a graph G, a v_0 - v_n walk is an alternate sequence of vertices and edges

$v_0, e_1, v_1, \ldots, e_n, v_n$ such that the consecutive vertices and edges are incident. A path is a walk in which no vertex is repeated. A path with n vertices is denoted by P_n. A proper coloring of G is an assignment of colors to the vertices of G, such that adjacent vertices have different colors. The smallest number of colors for which there exists a proper coloring of G is called a chromatic number of G, and is denoted by $\chi(G)$. A total dominator coloring (td-coloring) of G is a proper coloring of G with extra property that every vertex in G properly dominates a color class. The total dominator chromatic number is denoted by $\chi_{td}(G)$ and is defined by the minimum number of colors needed in a total dominator coloring of G. This concept was introduced by A. Vijayalekshmi in[1]. This notion is also referred as a smarandachely k-dominator coloring of G, ($k \geq 1$) and was introduced by A.Vijayalekshmi in[2]. For an integer $k \geq 1$, a smarandachely k-dominator coloring of G is a proper coloring of G, such that every vertex in a graph G properly dominates a k color class. The smallest number of colors for which there exists a smarandachely k-dominator coloring of G is called the smarandachely k-dominator chromatic number of G and is denoted by $\chi_{td}^{s}(G)$.

In a proper coloring C of a graph G, a color class of C is a set consisting of all those vertices assigned the same color. Let C be a minimum td-coloring of G. We say that a color class is called a non-dominated color class (n-d color lass) if it is not dominated by any vertex of G and these color classes are also called repeated color classes.

In this paper we obtain a C++ programme to find the td-chromatic number of paths by using elementary transformations.

The total dominator chromatic number of paths was found in the following observation.

Theorem A [3]

Let G be P_n. Then

$$\chi_{td}(P_n) = \begin{cases} 2\left\lfloor \frac{n}{4} \right\rfloor + 2 & if\ n \equiv 0(mod4) \\ 2\left\lfloor \frac{n}{4} \right\rfloor + 3 & if\ n \equiv 1(mod4) \\ 2\left\lfloor \frac{n+2}{4} \right\rfloor + 2 & Otherwise \end{cases}$$

2. MAIN RESULT

We have to find the total dominator chromatic number of paths using C++ programme. The C++ programme is successfully compiled and run on C++ platform. The runtime test is included.

Program as follows

```
#include "stdafx.h"
#include <Windows.h>
#include <conio.h>
#include <iostream>
using namespace std;
        int main()
        {
        int inpt;
        cout << "Enter the Value of n" << endl;
        cin >> inpt;
     while (inpt >= 11)
        {
    // dimensions
        int N = inpt; // matrix row
        int M = inpt; // matrix column
        // dynamic allocation
        int** ary = new int*[N]; //logic matrix
        int** mat = new int*[N]; //adjacency matrix
        int** mat1 = new int*[N]; //adjacency matrix after removing 1
        for (int i = 0; i < N; ++i)
        {
        ary[i] = new int[M];
        mat[i] = new int[M];
        mat1[i] = new int[M];
        }
        // variables
        int k;
        int l;

        HANDLE p = GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |
FOREGROUND_INTENSITY);
        // fill ary
        for (int i = 0; i < N; ++i)
        for (int j = 0; j < M; ++j)
        ary[i][j] = i;
        cout << "\n";
        cout << "The Adjacency Matrix for P" << N << "\n" << "\n";
        // fill mat
        for (int i = 0; i < N; i++)
```

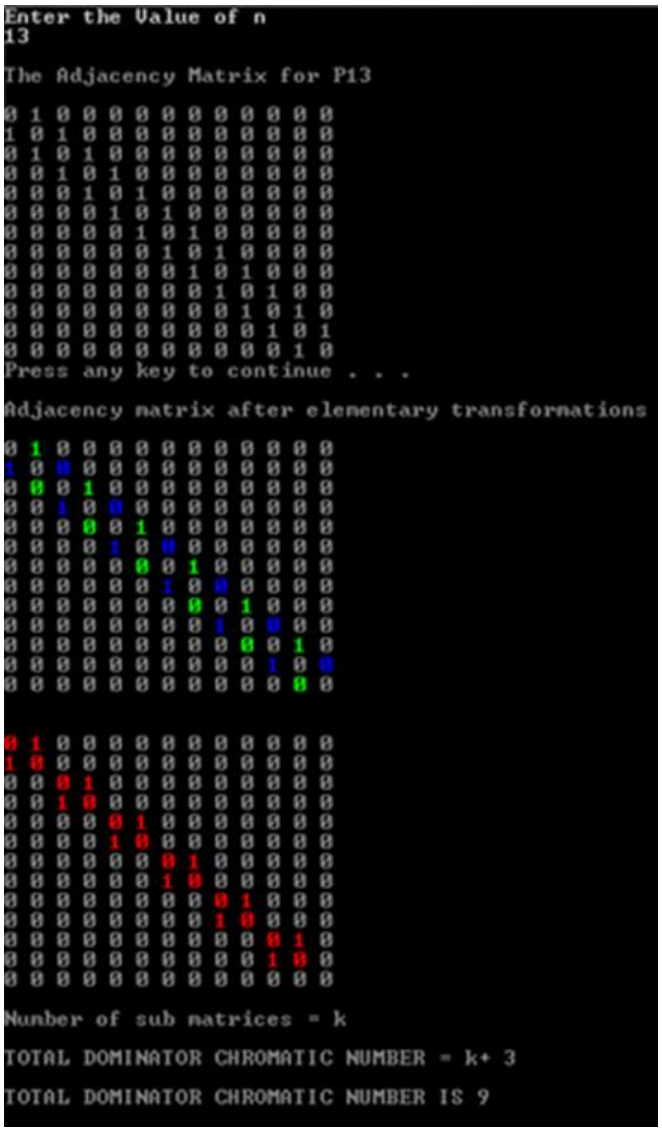
```
{
for (int j = 0; j < N; j++)
{
if (ary[j][i] == i + 1 | ary[j][i] == i - 1)
{
mat[i][j] = 1;
cout << mat[i][j] << " ";
}
else
{
mat[i][j] = 0;
cout << mat[i][j] << " ";
}
}
cout << "\n";
}
system ("pause");
cout << "\n";
// -----END LOGIC TO FORM MATRIX------
// ------ FORMING ADJACENCY MATRIX BY SUBSTRATING THE ROW AND
COLUMN VALUES
cout << "Adjacency matrix after elementary transformations" << "\n" <<"\n";
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
if (i >= i + 2 || mat[i][j] == 1 && mat[i][j - 2] == 1 && i % 2 != 0)
{
mat1[i][j] = mat[i][j] - mat[i][j - 2]; // R3 = R3-R1
SetConsoleTextAttribute(p, FOREGROUND_BLUE |
FOREGROUND_INTENSITY);
cout << mat1[i][j] << " ";
                    }
else if (i >= 2 && mat[i][j] == 1 && mat[i - 2][j] == 1 && i % 2 == 0)
{
mat1[i][j] = mat[i][j] - mat[i - 2][j];  // C3 = C3-C1
SetConsoleTextAttribute(p, FOREGROUND_GREEN |
FOREGROUND_INTENSITY);
cout << mat1[i][j] << " ";
}
else
```

```
        {
        if (mat[i][j] == 1 && i % 2 != 0 && i <= N - 2)
        {

        SetConsoleTextAttribute(p, FOREGROUND_BLUE |
FOREGROUND_INTENSITY);
        }
        else if (mat[i][j] == 1 && i % 2 == 0 && j <= N - 2)
        {

        SetConsoleTextAttribute(p, FOREGROUND_GREEN |
FOREGROUND_INTENSITY);
        }
        else
        {

        SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |
FOREGROUND_INTENSITY);
        }
        mat1[i][j] = mat[i][j];
        cout << mat1[i][j] << " ";
        }
        }
        cout << "\n";
        }

        SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |
FOREGROUND_INTENSITY);
        system("pause");
        cout << "\n";

        // --- END OF ADJACENCY MATRIX AFTER SUBSTRACTING THE ROW AND
COLUMN VALUES
        // ---- SHADING THE 2X2 MATRIX-------
        if (N % 2 != 0)
        {
        l = N - 1;
        }
        else
        {
        l = N;
        }
```

```
        for (int i = 0; i < N; i++)
        {
        for (int j = 0; j < N; j++)
        {
        if (j == i && j != l || mat1[i][j] == 1)
        {

        SetConsoleTextAttribute(p, FOREGROUND_RED |
FOREGROUND_INTENSITY);
        cout << mat1[i][j] << " ";
        }
        else
        {

        SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |
FOREGROUND_INTENSITY);
        cout << mat1[i][j] << " ";
        }
        }
        cout << "\n";
        }

        SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |
FOREGROUND_INTENSITY);
        // sub matrices
        {
    }
        if (N % 2 == 0)
        {
        k = N / 2;
        }
        else
        {
        k = (N-1) / 2;
        }
        // Final Formula
        if (N % 4 == 0)
        {
        cout << "\n";
    cout << "Number of sub matrices = k";
    cout << "\n";
```

```
            cout << "\n";
            cout << "TOTAL DOMINATOR CHROMATIC NUMBER = k + 2 ";
            cout << "\n";
            cout << "\n";

            cout << "TOTAL DOMINATOR CHROMATIC NUMBER is" << k + 2 << "\n";
            }
            else
            {

cout << "\n";
      cout << "Number of sub matrices = k";
      cout << "\n";
            cout << "\n";

            cout << "TOTAL DOMINATOR CHROMATIC NUMBER = k + 3 ";
cout << "\n";
            cout << "\n";
            cout << "TOTAL DOMINATOR CHROMATIC NUMBER IS " << k + 3 << "\n";
            }
            cout << "\n";
            system("Pause");
            return 0;

     // free ary and mat

            for (int i = 0; i < N; ++i)

            {

            delete[] ary[i];
            delete[] ary;
            delete[] mat1[i];
            delete[] mat1;
            delete[] mat[i];
            delete[] mat;

            }
            }
            return main();

            }
```

Runtime Test



REFERENCES

1. A.Vijayalekshmi, Total dominator colorings in paths, *International Journal of Mathematical Combinatorics*, Vol 2, p.89-95 (2012).
2. A.Vijayalekshmi, Total dominator colorings in cycles, *International Journal of Mathematical Combinatorics*, Vol 4, p.92-96 (2012).

3. A.Vijayalekshmi, J.Virgin Alangara Sheeba,Total dominator chromatic number of Paths, Cycles and Ladder graphs, *International Journal of Con temporary Mathematical Sciences*, Vol 13, no. 5,199-204 (2018).
4. F.Harrary , Graph theory,Addition-Wesley Reading Mass.(1969).
5. M.I.Jinnah and A.Vijayalekshmi, Total dominator colorings in graphs, Ph.D Thesis, University of Kerala. (2010).
6. Terasa W.Haynes, Stephen T.Hedetniemi,Peter J. Slater, Domination in Graphs, Marcel Dekker, New York, (1998).
7. Terasa W.Haynes, Stephen T.Hedetniemi,Peter J.Slater, Domination in Graphs - Advanced Topics, Marceel Dekker, NewYork, (1998).